



Trabajo de investigación

Carrito de Compras en JavaScript con Facturación

- Diseño y Programación de Software Multiplataforma DPS941 G01T -

Docente:

Ing. Alexander Alberto Siguenza Campos

Integrantes:

- William Alberto García Gómez - GG212522
- Christian Yahir López Hernández - LH212531
- Marco Rodrigo Funes Bonilla - FB200456
- Edwin Walberto Palacios Mejía - PM140373
- Alem Isai Vasquez Antillon - VA223253

Fecha:

31 de agosto de 2024


Índice

| | |
|----------------------------------|----|
| Partes del Sitio Web..... | 3 |
| Página principal..... | 3 |
| Productos.js..... | 4 |
| index.js..... | 5 |
| carrito.js..... | 9 |
| cart.js..... | 11 |
| Carrito..... | 12 |
| GitHub..... | 17 |
| ¿Cómo ejecutar el proyecto?..... | 17 |

Partes del Sitio Web

Página principal


Nuestros productos



Lentes

Precio: \$10.00
Disponibles: 5


[Agregar al carrito](#)



Pantalon

Precio: \$25.00
Disponibles: 12


[Agregar al carrito](#)



Camiseta

Precio: \$10.00
Disponibles: 6


[Agregar al carrito](#)



Gorra

Precio: \$20.00
Disponibles: 8


[Agregar al carrito](#)



Zapatos

Precio: \$50.00
Disponibles: 7


[Agregar al carrito](#)



Short

Precio: \$15.00
Disponibles: 9


[Agregar al carrito](#)



Lentes

Precio: \$10.00
Disponibles: 5


[Agregar al carrito](#)



Pantalon

Precio: \$25.00
Disponibles: 12


[Agregar al carrito](#)



Camiseta

Precio: \$10.00
Disponibles: 6


[Agregar al carrito](#)



Gorra

Precio: \$20.00
Disponibles: 8


[Agregar al carrito](#)



Zapatos

Precio: \$50.00
Disponibles: 7

[Agregar al carrito](#)



Short

Precio: \$15.00
Disponibles: 9

[Agregar al carrito](#)

Productos.js

En productos.js se define un Array de objetos products, donde cada objeto representa un producto con sus propiedades como id, nombre, precio, y cantidadDisponible.

```
JS productos.js X
js > Carrito-JS-main > js > JS productos.js > ...
1  const products = [
2      {
3          id:1,
4          nombre: "Lentes",
5          precio: 10,
6          cantidadDisponible: 5,
7      },
8      {
9          id:2,
10         nombre: "Pantalon",
11         precio: 25,
12         cantidadDisponible: 12,
13     },
14     {
15         id:3,
16         nombre: "Camiseta",
17         precio: 10,
18         cantidadDisponible: 6,
19     },
20     {
21         id:4,
22         nombre: "Gorra",
23         precio: 20,
24         cantidadDisponible: 8,
25     },
26     {
27         id:5,
28         nombre: "Zapatos",
29         precio: 50,
30         cantidadDisponible: 7,
31     },
32     {
33         id:6,
34         nombre: "Short",
35         precio: 15,
36         cantidadDisponible: 9,
37     },
38 ]
39
```

index.js

En este archivo se maneja la creación y visualización de los productos en la página principal, así como la lógica para agregar productos al carrito.

```
const cardContainer = document.getElementById("producto-container");

function crearProductos(products) {
  const memoria = JSON.parse(localStorage.getItem("products")) || [];

  products.forEach(producto => {
    const productoEnCarrito = memoria.find(product => product.id === producto.id);
    if (productoEnCarrito) {
      producto.cantidadDisponible -= productoEnCarrito.cantidad;
    }

    const nuevoProducto = document.createElement("div");
    nuevoProducto.classList = "producto-card";
    nuevoProducto.innerHTML = `
      
      <h4>${producto.nombre}</h4>
      <div class="producto-content">
        <div class="producto-info">

          <p>Precio: ${producto.precio}.00</p>
          <p>Disponibles: ${producto.cantidadDisponible}</p>
          <input type="number" min="1" max="${producto.cantidadDisponible}" value="1" class="cantidad-input">
        </div>
        <div class="popup-main">

          <a href="#popup" class="carrito-popup"><button class="add-carrito">Agregar al carrito</button></a>
        </div>
      </div>
    `;

    cardContainer.appendChild(nuevoProducto);

    nuevoProducto.getElementsByTagName("button")[0].addEventListener("click", () => {
      const cantidadInput = nuevoProducto.querySelector('.cantidad-input');
      const cantidadDeseada = parseInt(cantidadInput.value);

      if (cantidadDeseada > 0 ) {
        if(cantidadDeseada <= producto.cantidadDisponible){
          agregarProducto(producto, cantidadDeseada);
          producto.cantidadDisponible -= cantidadDeseada;
          nuevoProducto.querySelector('.producto-info p:last-of-type').innerText = `Disponibles: ${producto.cantidadDisponible}`;
        }else{
          if(producto.cantidadDisponible == 0){
            alert('Ya no hay más de este producto, intenta agregar uno distinto' );
          }else{
            alert(`¡Ingresaste muchos! Solamente hay '+ producto.cantidadDisponible + ' en la tienda' );
          }
        }
      } else {
        alert('Debe ingresar una cantidad válida');
      }
    });
  });
}

crearProductos(products);
```

```
function crearProductos(products) {
  const memoria = JSON.parse(localStorage.getItem("products")) || [];
```

function crearProductos(products): recibe array de productos como argumento para crear el HTML necesario para mostrarlo en la página (index.html).

Se verifica si existen productos almacenados previamente en el carrito, de lo contrario se utiliza un array vacío.

```
products.forEach(producto => {
  const productoEnCarrito = memoria.find(product => product.id === producto.id);
  if (productoEnCarrito) {
    producto.cantidadDisponible -= productoEnCarrito.cantidad;
  }
});
```

Luego se recorre cada producto en el array products, y se busca en el carrito un producto con el mismo id. Si el producto ya está en el carrito (productoEnCarrito), se reduce la cantidad disponible de ese producto para reflejar las unidades que ya están en el carrito.

```
const nuevoProducto = document.createElement("div");
nuevoProducto.classList = "producto-card";
nuevoProducto.innerHTML = `
  
  <h4>${producto.nombre}</h4>
  <div class="producto-content">
    <div class="producto-info">
      <p>Precio: ${producto.precio}.00</p>
      <p>Disponibles: ${producto.cantidadDisponible}</p>
      <input type="number" min="1" max="${producto.cantidadDisponible}" value="1" class="cantidad-input">
    </div>
    <div class="popup-main">
      <a href="#popup" class="carrito-popup"><button class="add-carrito">Agregar al carrito</button></a>
    </div>
  </div>
`;
```

document.createElement("div") crea un nuevo elemento div y se asigna la clase producto-card, luego se configura su contenido HTML con innerHTML.

Dentro del HTML, se incluyen los datos del producto: imagen, nombre, precio, cantidad disponible, y un input para ingresar la cantidad.

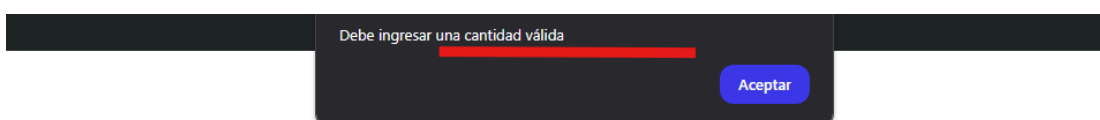
```

nuevoProducto.getElementsByTagName("button")[0].addEventListener("click", () => {
  const cantidadInput = nuevoProducto.querySelector('.cantidad-input');
  const cantidadDeseada = parseInt(cantidadInput.value);

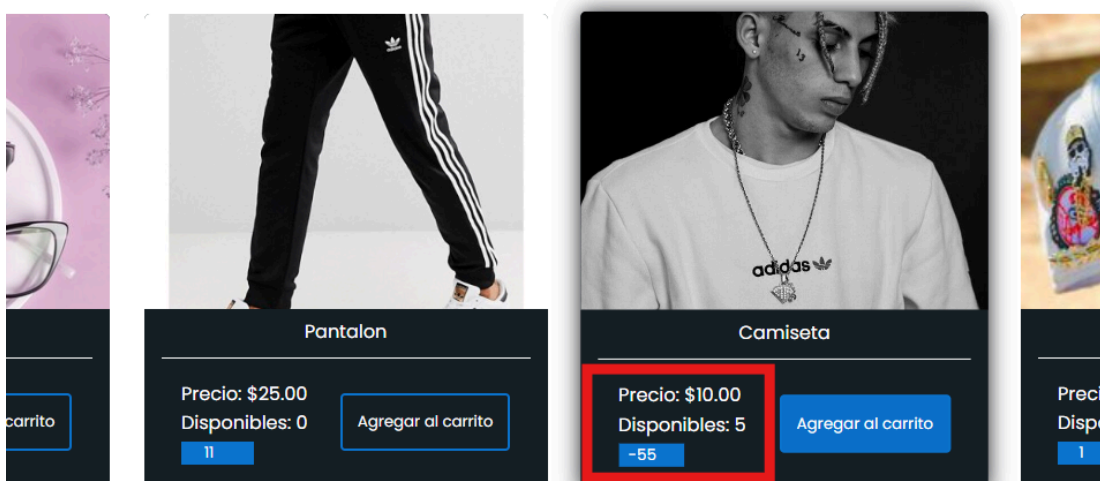
  if (cantidadDeseada > 0 ) {
    if(cantidadDeseada <= producto.cantidadDisponible){
      agregarProducto(producto, cantidadDeseada);
      producto.cantidadDisponible -= cantidadDeseada;
      nuevoProducto.querySelector('.producto-info p:last-of-type').innerText = `Disponibles: ${producto.cantidadDisponible}`;
    }else{
      if(producto.cantidadDisponible == 0){
        alert('Ya no hay más de este producto, intenta agregar uno distinto' );
      }else{
        alert(`¡Ingresaste muchos! Solamente hay '+ producto.cantidadDisponible + " en la tienda" );
      }
    }
  } else {
    alert('Debe ingresar una cantidad válida');
  }
});
});
}
crearProductos(products);

```

Se agrega un event listener al botón "Agregar al carrito". Cuando se hace clic, se verifica la cantidad deseada. Si la cantidad es válida y está disponible, se llama a la función agregarProducto para añadir el producto al carrito, se actualiza la cantidad disponible, y se refleja este cambio en el HTML. Si la cantidad deseada no es válida o no hay productos disponibles, se muestra el mensaje de alerta respectivo..



Nuestros productos



¡Ingresaste muchos! Solamente hay 11 en la tienda

Aceptar

Nuestros productos




Pantalon

Precio: \$25.00
Disponibles: 11

12

Agregar al carrito



Co

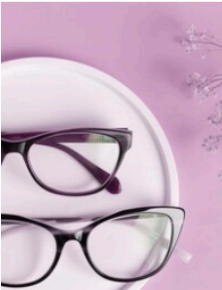
Precio: \$10.00
Disponibles: 5

1

Ya no hay más de este producto, intenta agregar uno distinto

Aceptar

Nuestros productos



entes

Agregar al carrito



Pantalon

Precio: \$25.00
Disponibles: 0

11

Agregar al carrito



Camiseta

Precio: \$10.00
Disponibles: 5

-55

Agregar al carrito



Precio: \$10.00
Disponibles: 5

1

carrito.js

En este archivo se maneja la lógica para agregar y quitar productos del carrito de compras.

```
function agregarProducto(producto, cantidad){
  const memoria = JSON.parse(localStorage.getItem("products"));
  console.log(memoria);
  if(!memoria){
    const nuevoProducto = getNuevoProductoMemoria(producto, cantidad);
    localStorage.setItem("products",JSON.stringify([nuevoProducto]));
  }else {
    const indiceProducto = memoria.findIndex(product => product.id === producto.id);
    console.log(indiceProducto);

    if (indiceProducto === -1) {
      const nuevoProducto = getNuevoProductoMemoria(producto, cantidad);
      memoria.push(nuevoProducto);
    } else {
      memoria[indiceProducto].cantidad += cantidad;
    }

    localStorage.setItem("products", JSON.stringify(memoria));
  }
}
```

Se obtiene el carrito de compras almacenado en localStorage. Si el carrito (memoria) está vacío (!memoria), se crea un nuevo producto utilizando la función getNuevoProductoMemoria y se almacena en localStorage. En caso de que el carrito existe, se verifica que el producto a agregar esté en el carrito. Si el producto no está (indiceProducto === -1, donde findIndex devuelve -1), se crea un nuevo producto y se agrega al carrito. En caso contrario, se incrementa su cantidad y por último se actualiza el carrito en localStorage:

```
    localStorage.setItem("products", JSON.stringify(memoria));
  }
```

```

function getNuevoProductoMemoria(producto, cantidad) {
  const nuevoProducto = { ...producto };
  nuevoProducto.cantidad = cantidad;
  return nuevoProducto;
}
function quitarProducto(producto){
  const memoria = JSON.parse(localStorage.getItem("products"));
  const indiceProducto = memoria.findIndex(product => product.id === producto.id);
  if(memoria[indiceProducto].cantidad === 1){
    memoria.splice(indiceProducto,1);
  }else{
    memoria[indiceProducto].cantidad--;
  }
  localStorage.setItem("products",JSON.stringify(memoria));
}

```

getNuevoProductoMemoria(producto)

Se crea una copia del objeto producto usando el operador de propagación (...), y se agrega la propiedad cantidad al nuevo producto.

quitarProducto(producto)

Al igual que en agregarProducto, se obtiene el carrito actual de localStorage y se busca el índice del producto en el carrito usando findIndex.

```

const memoria = JSON.parse(localStorage.getItem("products"));
const indiceProducto = memoria.findIndex(product => product.id === producto.id);
if(memoria[indiceProducto].cantidad === 1){
  memoria.splice(indiceProducto,1);
}
else{
  memoria[indiceProducto].cantidad--;
}

```

Si el producto tiene una sola unidad en el carrito, se elimina por completo del array "memoria" utilizando splice y se indica que debe eliminarse solo 1 elemento (indiceProducto, 1). Si el producto tiene más de una unidad, la cantidad se reduce en uno. Finalmente, se actualiza el carrito en localStorage con la nueva cantidad o la eliminación del producto.

cart.js

Interfaz de carrito de compra y proceso de compra y facturación

```
function crearProductos() {
  cardContainer.innerHTML = "";
  const products = JSON.parse(localStorage.getItem("products"));
  console.log(products)
  if (products && products.length > 0) {
    carritoVacio.style.display = "none";
    productosHeader.style.display = "block";
    productosTotal.style.display = "block";

    products.forEach((producto) => {
      const nuevoProducto = document.createElement("div");
      nuevoProducto.classList = "carrito-card";
      nuevoProducto.innerHTML = `
        
        <div class="carrito-info-text">
          <h4>${producto.nombre}</h4>
          <p>Precio: ${producto.precio}.00</p>
        </div>
        <div class="carrito-add">
          <button class="quitar">-</button>
          <span class="cantidad">${producto.cantidad}</span>
          <button class="añadir">+</button>
        </div>
      `;

      cardContainer.appendChild(nuevoProducto);
      nuevoProducto.getElementsByTagName("button")[1].addEventListener("click", (e) => {

        if (producto.cantidad < producto.cantidadDisponible) {
          agregarProducto(producto, 1);
          crearProductos();
          totalProductos();
        } else {
          alert(`No se puede agregar más. Solo hay ${producto.cantidadDisponible} unidades disponibles.`);
        }
      });
      nuevoProducto.getElementsByTagName("button")[0].addEventListener("click", (e) => {
        quitarProducto(producto);
        crearProductos();
        totalProductos();
      });
    });
  } else {
    carritoVacio.style.display = "block";
    cardContainer.style.display = "none";
    productosHeader.style.display = "none";
    productosTotal.style.display = "none";
  }
}
```

Crea y muestra los productos en el carrito. Si hay productos en el carrito, actualiza la vista y permite modificar la cantidad de cada producto. Si el carrito está vacío, muestra un mensaje apropiado y oculta los detalles del carrito.

Al intentar agregar más productos de los disponibles, se muestra un mensaje de error y no permite añadir más cantidades.

Carrito

Sin productos

No hay productos en el carrito

Seguir comprando

Producto añadido

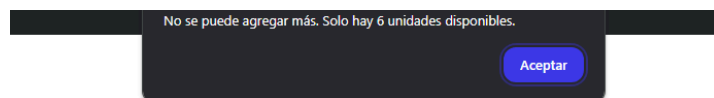
Tus productos



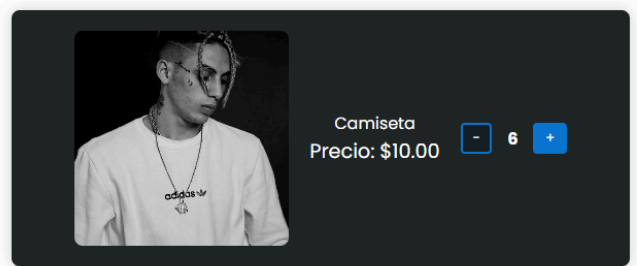
Total unidades: 3
Total precio: \$30

Comprar Eliminar todo

Error al ingresar más productos de la cantidad disponible



Tus productos



Total unidades: 6
Total precio: \$60

Calcula y muestra la cantidad total de productos y el precio total en el carrito modificando el HTML.

```
// Función para actualizar la cantidad total de productos y precios
function totalProductos(){
  const products = JSON.parse(localStorage.getItem("products"));
  let cantidadProductos = 0;
  let precio = 0
  if(products && products.length > 0){
    products.forEach(producto => {
      cantidadProductos += producto.cantidad;
      precio += producto.precio * producto.cantidad;
    });
    cantidadProduct.innerHTML = cantidadProductos;
    precioProduct.innerHTML = precio;
  }
}
```

```
// Función para llenar dinámicamente los selectores de mes y año
function populateExpiryDateOptions() {
  for (let i = 1; i <= 12; i++) {
    const option = document.createElement("option");
    option.value = i < 10 ? '0' + i : i;
    option.text = i < 10 ? '0' + i : i;
    expiryMonthSelect.add(option);
  }
  const currentYear = new Date().getFullYear();
  for (let i = currentYear; i <= currentYear + 6; i++) {
    const option = document.createElement("option");
    option.value = i;
    option.text = i;
    expiryYearSelect.add(option);
  }
}

populateExpiryDateOptions(); // Llamada para llenar las opciones al cargar el script

document.getElementById("comprar").addEventListener("click", () => {
  const products = JSON.parse(localStorage.getItem("products"));
  if (products && products.length > 0) {
    cardContainer.style.display = "none";
    carritoVacio.style.display = "none";
    productosHeader.style.display = "none";
    productosTotal.style.display = "none";
    paymentFormContainer.style.display = "flex"; // Asegura que el formulario esté centrado
    paymentForm.addEventListener("submit", handlePaymentFormSubmit);
  } else {
    alert("No hay productos en el carrito.");
  }
});
```

populateExpiryDateOptions() Llena los selectores de mes y año de expiración con opciones válidas para tarjetas de crédito.

```
function handlePaymentFormSubmit(e) {
    e.preventDefault(); // Prevenir el comportamiento por defecto del formulario

    // Obtener y limpiar valores
    const cardNumber = document.getElementById("cardNumber").value.replace(/\s/g, ''); // Remover espacios
    const expiryMonth = document.getElementById("expiryMonth").value;
    const expiryYear = document.getElementById("expiryYear").value;
    const cvv = document.getElementById("cvv").value;
    const cardHolder = document.getElementById("cardHolder").value;

    let isValid = true;

    // Validar número de tarjeta
    if (!validateCardNumber(cardNumber)) {
        cardNumberError.style.display = "block";
        isValid = false;
    } else {
        cardNumberError.style.display = "none";
    }

    // Solo mostrar el mensaje de confirmación si todos los datos son válidos
    if (isValid) {
        if (confirm("¿Deseas confirmar la compra?")) {
            const products = JSON.parse(localStorage.getItem("products"));
            generarFactura(products, cardHolder, cardNumber);
            localStorage.removeItem("products");
            alert("¡Gracias por comprar con nosotros!");
            setTimeout(() => {
                window.location.href = "index.html";
            }, 3000); // 3 segundos de retraso
        }
    }
}

function validateCardNumber(cardNumber) {
    const cardNumberRegex = /^[0-9]{13,16}$/;
    return cardNumberRegex.test(cardNumber);
}

document.getElementById("cardNumber").addEventListener("input", function (e) {
    let value = e.target.value.replace(/\D/g, '');
    value = value.replace(/(.{4})/g, '$1 ').trim();
    e.target.value = value;
});
```

handlePaymentFormSubmit(e) Maneja la validación del formulario de pago y genera una factura PDF si los datos son válidos. Luego, limpia los datos del carrito después de la compra y redirige al usuario a la página principal después de un retraso de 3 segundos.

validateCardNumber(cardNumber)

Valida el número de tarjeta de crédito utilizando una expresión regular.

```

function generarFactura(products, cardHolder, cardNumber) {
  const doc = new jsPDF();

  // Fecha y hora de la compra
  const date = new Date();
  const dateString = date.toLocaleDateString() + " " + date.toLocaleTimeString();

  // Últimos 4 dígitos del número de tarjeta
  const lastFourDigits = cardNumber.slice(-4);

  // Encabezado de la factura
  doc.setFontSize(18);
  doc.text("Factura Electrónica", 105, 20, { align: "center" });
  doc.setFontSize(12);
  doc.text(`Fecha y Hora: ${dateString}`, 20, 30);
  doc.text(`Titular de la Tarjeta: ${cardHolder}`, 20, 40);
  doc.text(`Método de Pago: Tarjeta terminada en ${lastFourDigits}`, 20, 50);

  // Tabla de productos
  doc.setFontSize(14);
  doc.text("Productos", 20, 70);
  doc.setFontSize(12);
  let positionY = 80;
  products.forEach(producto => {
    doc.text(`- ${producto.nombre}: ${producto.cantidad} x ${producto.precio}.00`, 20, positionY);
    positionY += 10;
  });

  // Calcular total
  const total = products.reduce((sum, product) => sum + product.precio * product.cantidad, 0);

  doc.text(`Total: ${total.toFixed(2)}`, 20, positionY + 10);

  // Estilo y descarga de la factura
  doc.save("factura-electronica.pdf");
}

```

En esta función se genera una factura en formato PDF utilizando la biblioteca jsPDF. Se incluyen detalles como la fecha, el titular de la tarjeta, método de pago, productos, cantidad comprada, precio unitario y precio total.

Eliminamos todos los productos del carrito al confirmar la decisión

```

limpiarCarrito.addEventListener("click", limpiarProduct);
function limpiarProduct(){
  const eliminarConfirm = confirm("¿Estás seguro de que deseas eliminar todos los productos del carrito?");

  if (eliminarConfirm) {
    localStorage.removeItem("products");
    totalProductos();
    crearProductos();
  }
}

```

¿Estás seguro de que deseas eliminar todos los productos del carrito?

Aceptar

Cancelar



Lentes
Precio: \$10.00



2



Camiseta
Precio: \$10.00



2



Total unidades: 6

Total precio: \$90

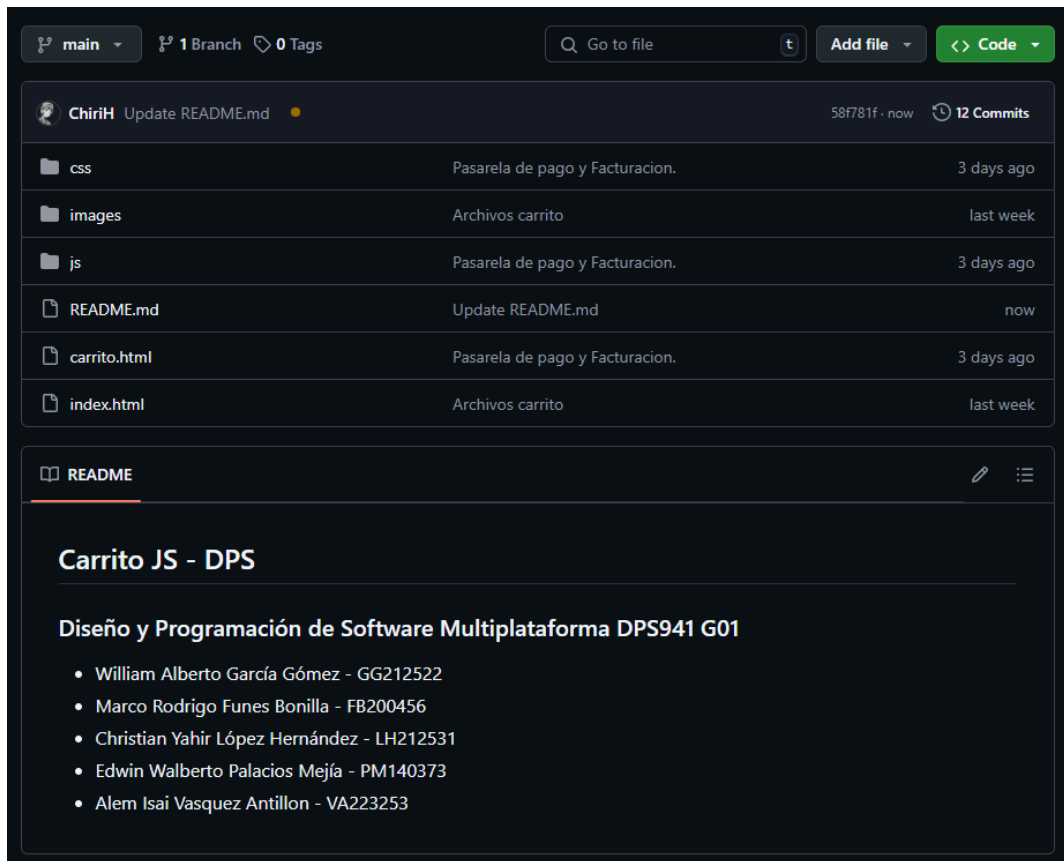
Comprar

Eliminar todo

GitHub

Repositorio: <https://github.com/ChiriH/Carrito-JS>

GitHub pages: <https://chirih.github.io/Carrito-JS/>



¿Cómo ejecutar el proyecto?

Se puede acceder al repositorio y en Deployments o descargar como archivo zip. Una vez descargado, abrir index.html

