**UNIVERSIDAD CENTRAL DE VENEZUELA**

**FACULTAD DE CIENCIAS**

**INTRODUCCION A LA INFORMATICA**



**PROYECTO PARTE I**

**Integrantes:**

Sebastián Valbuena CI. 31.745.088

Daniel Reyes C.I 32.303.416

Javier Ramírez C.I 31.707.800

Sesión P4

**Página web – BYNARY SHOP**

Ya entramos en la segunda fase del primer proyecto de introducción a la informática y nos piden que con los diagramas realizados en la anterior fase empecemos a desarrollar una página web utilizando los lenguajes de programación: HTML, CSS y JAVASCRIPT, teniendo toda esta información realizaremos nuestra pagina

Pero primero definamos para que se usa cada uno de estos programas:

**HTML**: en el podemos colocar casillas de texto, imágenes, símbolos, es básicamente como el esqueleto de nuestra página.

**CSS**: en este programa podemos modificar el tamaño, color, fuente, animación, fondo de nuestras casillas del HTML, aquí se hace el decorado básicamente.

**JAVASCRIPT**: en este lenguaje se realizan las funciones de la página, todo lo que tiene que ver con comprobación de información, acciones, eliminado, etc.

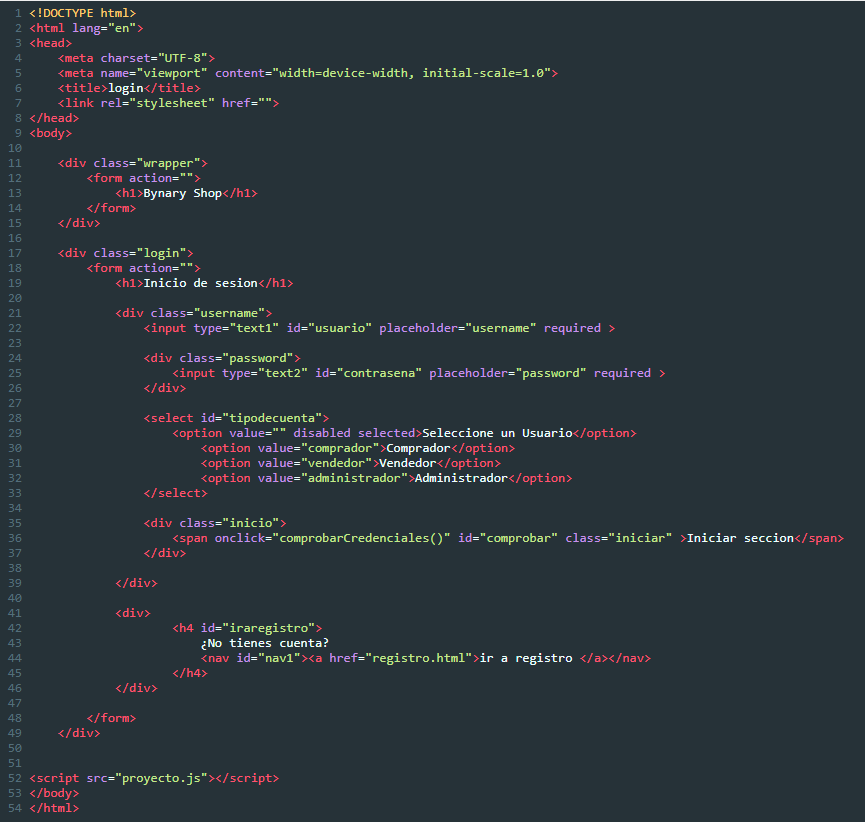
Ya con esto podemos empezar a crear nuestra página web BYNARY SHOP.

**INICIO**

En el inicio esta nuestro Loguin en donde los usuarios iniciaran sesión con su respectivo usuario / clave / y tipo de cuenta, a continuación, se presenta su **HTML** base:

Loguin **HTML** base:

Consta de un título con el nombre de la página y casillas de inicio de sesión con contraseña y el nombre del usuario junto con un selector del tipo de usuario, y en caso de no tener una cuenta hay una redirección a registro, esta es la base del Loguin

**Loguin** **HTML:**

Así se ve el **HTML** del Loguin en código, este **HTML** está conectado con un **CSS** llamado **Inicio y Registro.css** y también está conectado con un **JS** llamado **proyecto.js** en el que contiene sus funciones, cabe resaltar que tuvimos que ver varios videos para poder hacer el **HTML** y el **CSS**:



**Loguin CSS:**

Aquí está la página ya decorada con **CSS**, básicamente se le ha cambiado el fondo, el tamaño de los títulos, se le agregaron: **hovers** a los botones, se ordenaron los elementos del inicio de sesión.

Aquí está la vinculación del **HTML** con el **CSS**, y se vieron videos de igual manera para saber cómo manejar nuestras secciones y **div** del **HTML**

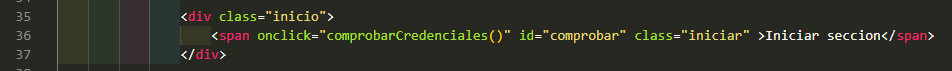
**Loguin Java Script:**

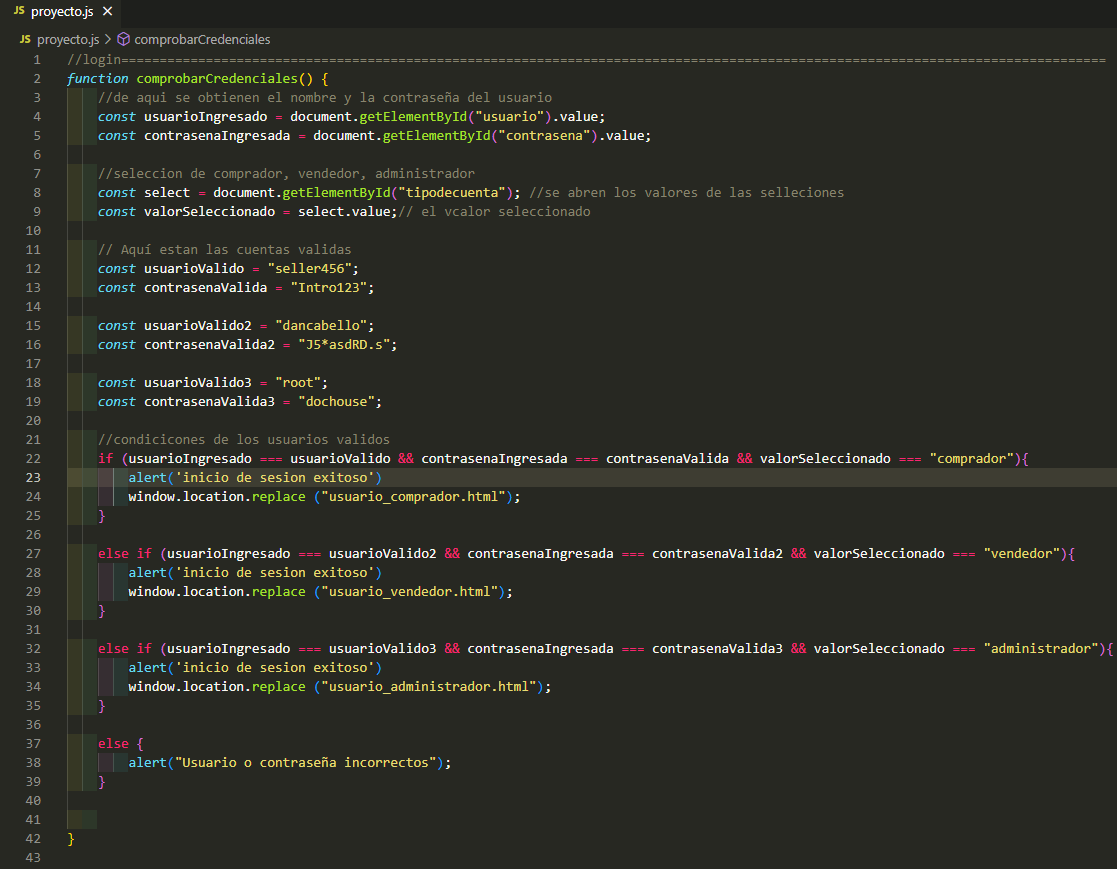
**Así se ve la conexión con el HTML Loguin y el Java Script:**

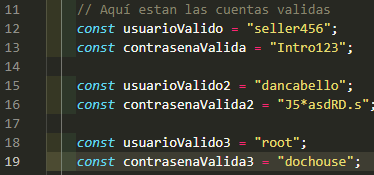


Aquí se presenta el Java script del Loguin, el Loguin está conectado con un java script llamado **proyecto.js** en el están también las funciones del comprador por eso este **js** es el más lago del proyecto.

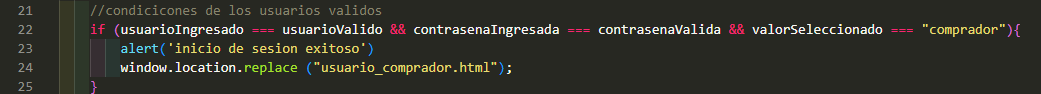
**Inicio de sesión:**

 básicamente cuando nosotros en la página presionamos iniciar sesión estamos presionando dentro del **HTML** un **span** de iniciar sección, este hace una acción llamada **“on click”**, esta acción llama a nuestra función **comprobarCredenciales().**

****

En esta función definimos variables **const** que almacenaran los datos que nuestro usuario halla almacenado previamente utilizando el código: **document.getElementById (“id”).value** este código nos ayuda a obtener dichos datos, y estos datos los comprobaremos con tres cuentas ya existentes dadas por el informe del proyecto:

Para comprobar los datos hacemos un **if** para cada usuario, y validamos si los datos ingresados coinciden, si es así se redirigirá a la interfaz que pertenece, y si no es así saldrá una alerta diciendo que los datos son inválidos.



Y aquí terminaríamos con el inicio de sesión.

**Registro**

En este **HTML** se llena una encuesta para crear un usuario, pero como no tenemos base de datos no se guardará los datos que ingresemos ni nada, tampoco tiene funciones, solo tiene **HTML** y **CSS** para la decoración. Que por cierto está conectado con el mismo **CSS** de inicio.

* **CSS**:

**Registro HTML – BASE:**

Así se ve el esqueleto del registro, se ha reciclado el **HTML** de inicio de sesión y se agregaron más casillas, también se le pregunta al usuario si ya tiene una cuenta junto una redirección de inicio

**Registro Con CSS:**

Como hemos reciclado el mismo **HTML** de inicio, nosotros solo le modificamos algunas cositas que si el tamaño creando nuevas clases en el **HTML** y remplazando, nos queda el inicio de sesión tal que así

Como el registro no tiene más nada lo dejamos por terminado

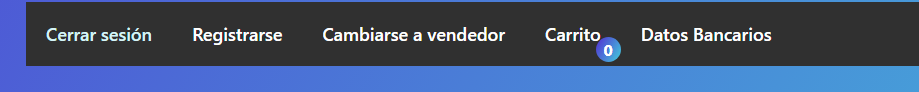
**Usuario Comprador**

Aquí se viene lo gordo del proyecto, como el **HTML** del usuario tiene muchas secciones explicaremos cada una de ellas, pero lo principal a definir son sus conexiones:

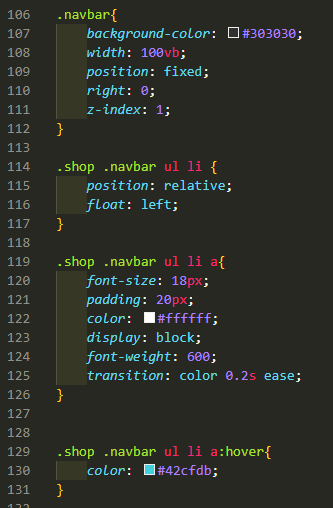
* **CSS**:



* **Java Script**:

**“BARRA DE NAV”:**

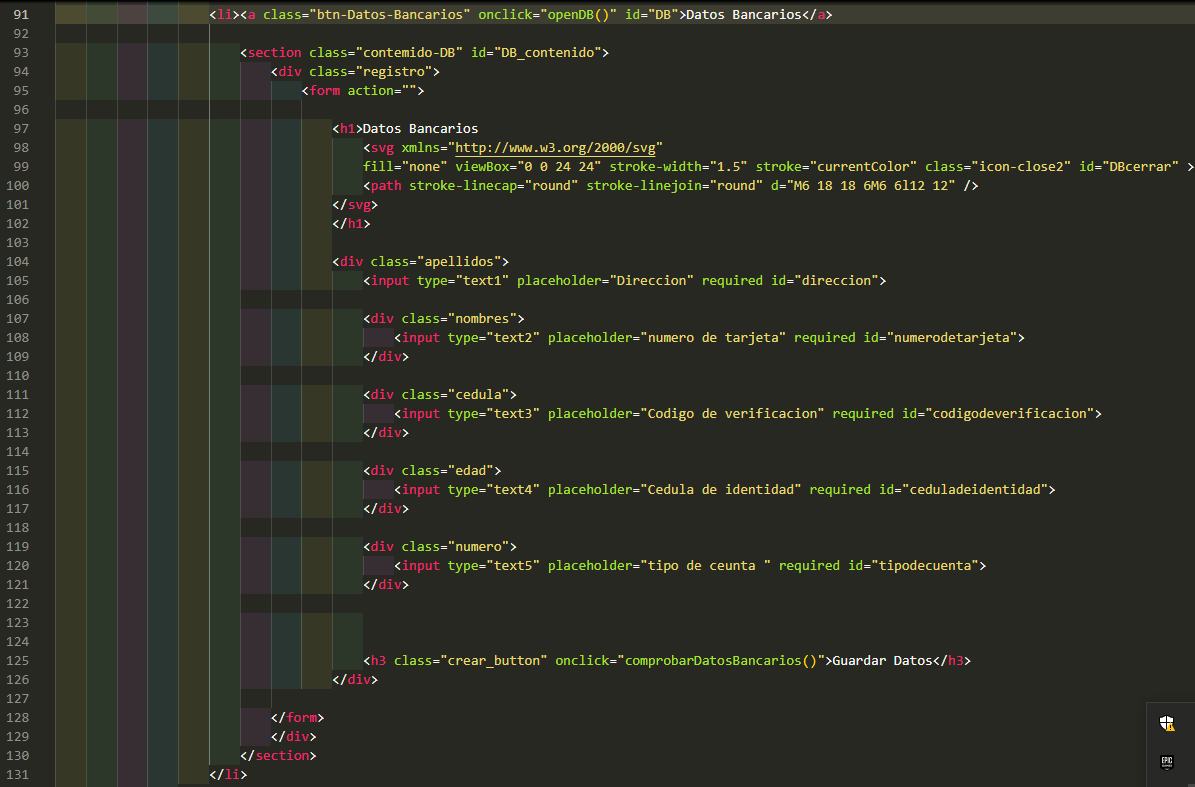


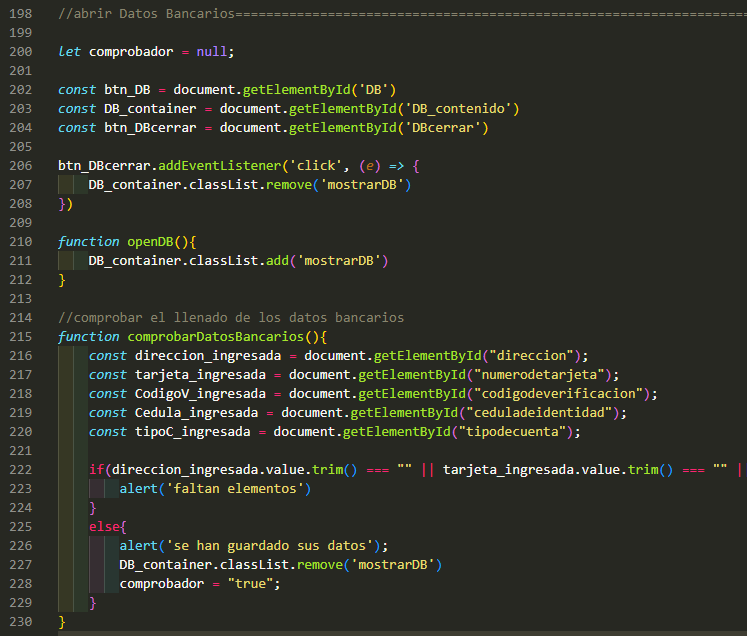
Se presenta el **HTML** de la barra de navegación junto en el CSS que le corresponde dentro de **pagina.css**, en ella esta:

* Cerrar sesión: nos devolverá al **inicio.html**.
* Registro: que nos redirigirá al **registro.html**.
* Cambiarse a Vendedor: una función del usuario comprador.
* Carrito: función del carrito al comprador.
* Datos Bancarios: el llenado de unos datos bancarios

En el **CSS** se manejó la barra de navegación para ponerla en la esquina de la página y hacer que cuando el usuario haga scroll está la acompañe, se le cambiaron el tamaño de las letras si se hicieron hovers para cuando seleccionemos los elementos de ella.

**“DATOS BANCARIOS”:**



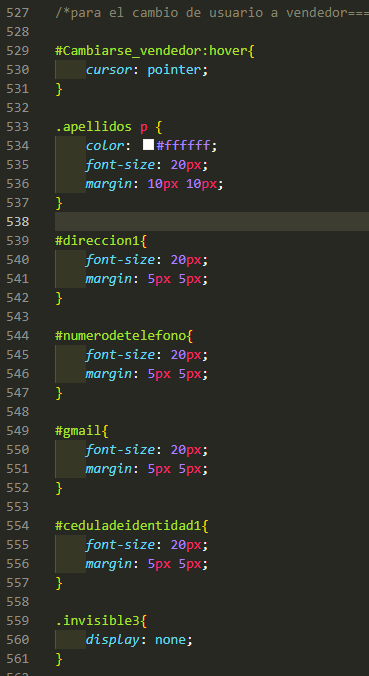
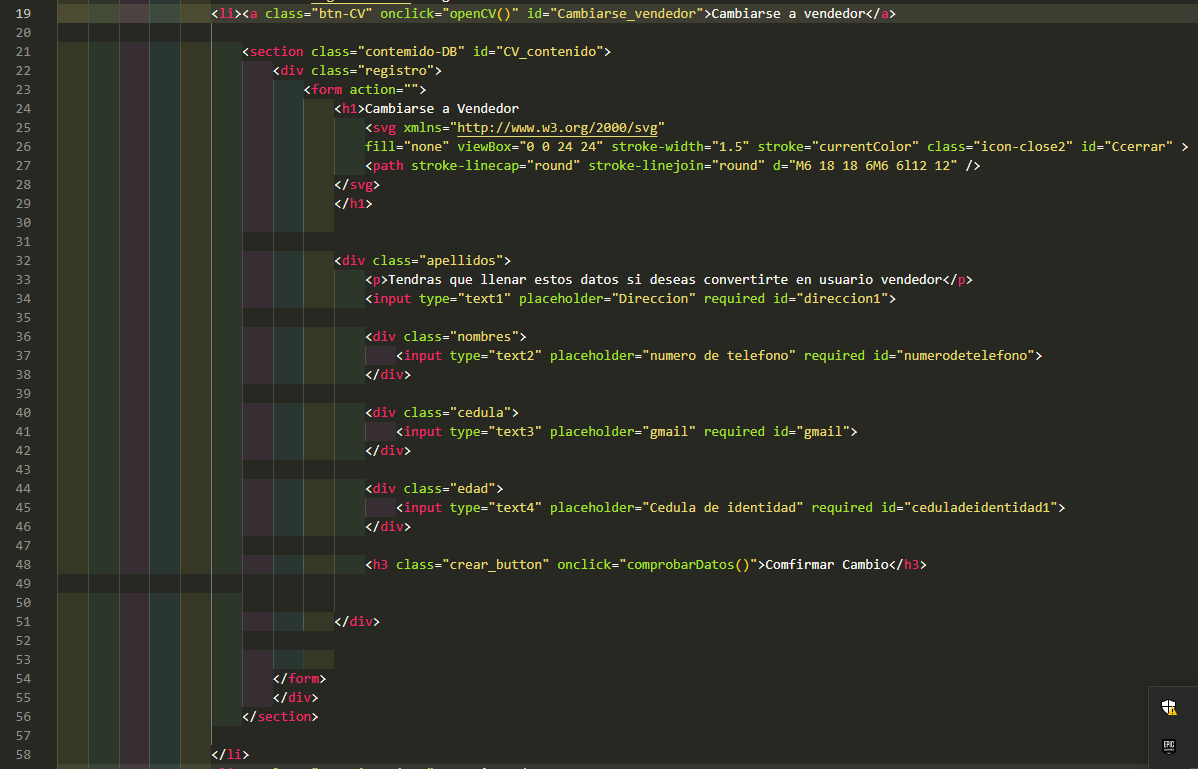


Se presenta el **HTML**, **CSS** y **Java Script** correspondientes de los datos bancarios, primero llenamos una ventana la cual estará invisible y solo se abre cuando presionamos el **“Datos Bancarios”,** dicha ventana tenemos que llenar unas casillas con información y los guardamos al comprobar que todos los datos se han llenado cerramos la ventana y validamos que los datos bancarios han sido llamados con un comprobador.

En el **CSS** se hizo que la ventana sea invisible y que siguiera al hacer **scroll**, junto con el decorado y la disposición, etc., lo importante es que tenemos una clase con un **display: none** que nos ayudara en JS para ocultarla.

En el **Java Script** se hace la comprobación de que los datos se hallan llenado correctamente guardando **const** con los datos ingresados y hacemos un condicional, si se cumple el condicional de que los datos no se hallan llenado completamente salta una alerta, si no se cierra la ventana y un comprobador definido anteriormente será verdadero, para abrir la ventana usamos un **classList.remove(‘mostrarDB’);** esta función remueve el **display:none** y así aparece la ventana, para desaparecerla se le pone **classList.add.**

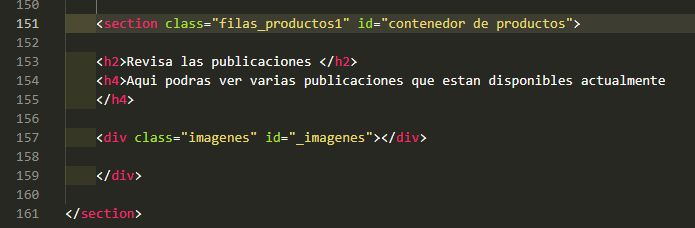
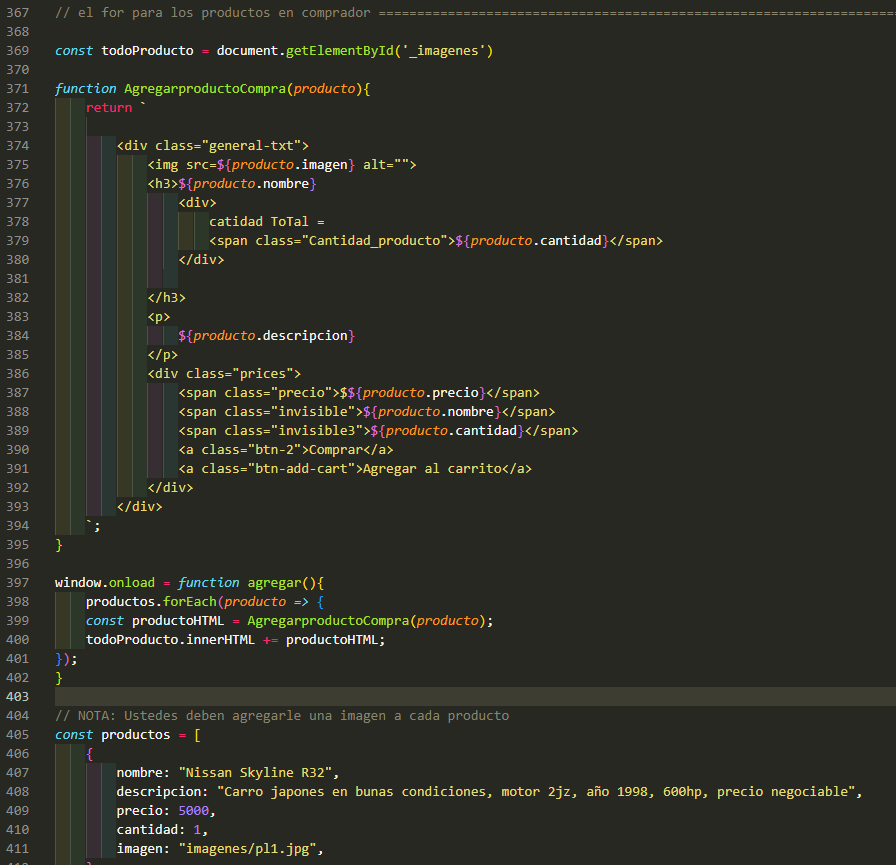
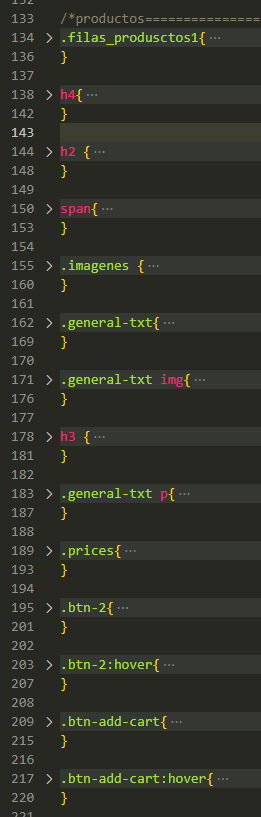
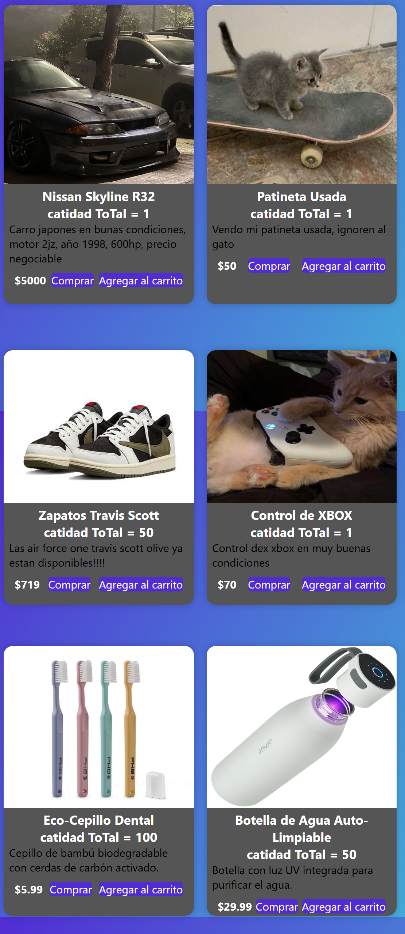
Este código es un reciclado del Loguin y también usamos el mismo código para la siguiente sección.

**“****CAMBIARSE A VENDEDOR”:**

Cambiarse a vendedor es un reciclado de Datos Bancarios por eso tenemos **class** que se repiten y casi el mismo código, al presionar la opción en la navbar se nos abrirá una ventana con el llenado de c información y cuando se halla llenado nos llevará al **usuario-vendedor.html.**

En **Css** se hizo lo mismo de datos bancarios con la diferencia de que como se le agregaron más casillas de llenado se tuvo que modificar el tamaño.

En el **Java Script** también se hizo la misma función de aparecer y desaparecer la ventana, y también se comprueban los datos, una vez llenados te sale una alerta de llenado exitoso y el HTML se redirigirá a usuario-vendedor

**“PRODUCTOS”:**

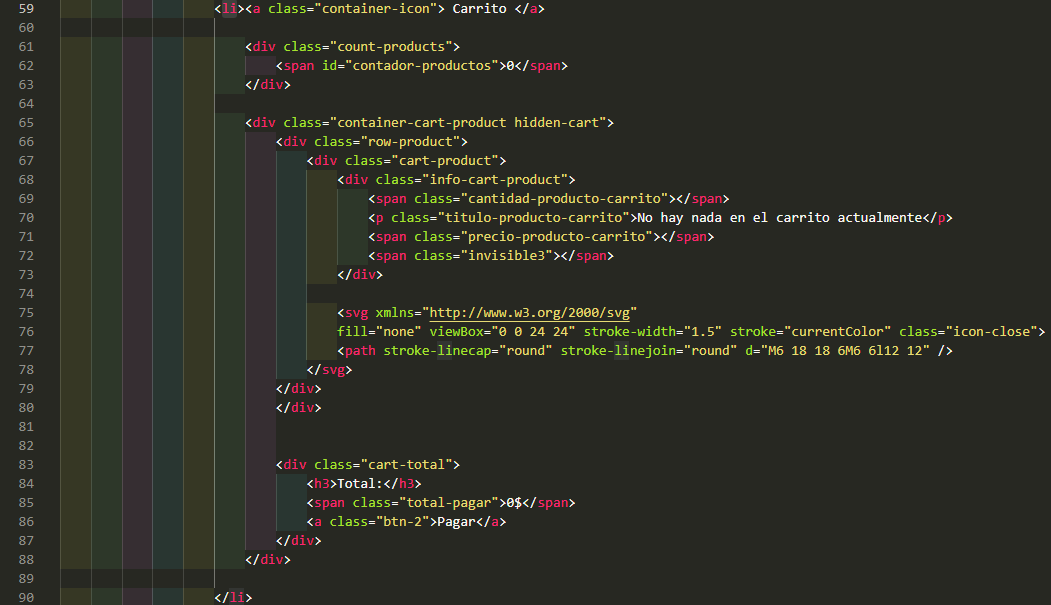
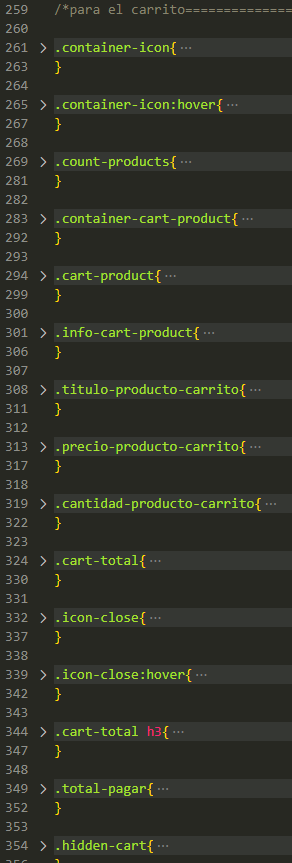
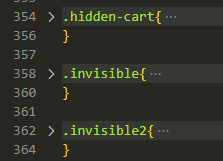
Cada producto tiene su nombre, precio, cantidad, descripción, etc, en el **HTML** se ha utilizado una clase general para todos los productos llamada **“general-txt”** y también tenemos una div con class “filas\_productos1”, en ella se pondrán los productos.

En el **CSS** de los productos se le dio el tamaño a las imágenes, se encerraron cada producto individualmente en una cajita, hovers para los botones, tamaños, etc, lo importante fue el grid-template-columns: repeat(3, 1fr); en la clase imágenes para que los productos se ordenaran en columnas de 3.

El **Java Script** va a hacer un bucle que por cada producto se genere su división **“general-txt”,** primero tenemos un arreglo de productos, cada posición del arreglo tiene un producto diferente, entonces en el bucle, por cada producto del arreglo se realiza la función de **AgregarProductoCompra(producto)** con el producto como parámetro, esta función va a retornar algo que está entre estas comillas **` `** llamadas plantillas literales, que nos ayudan a generar todo ese código para el HTML, gracias a esas comillas usamos el **${}** para adentro de el asignarle las especificaciones de cada producto, como el nombre, imagen, etc. Dentro de esas comillas se construye la estructura HTML de los productos básicamente, se le ponen los botones de compra, venta, cantidad, etc.

entonces, ya al crear estos div se le agregan a la división del HTML llamada **“\_images”** que la definimos como una const en el JS, este código se activa apenas se inicia la página, por eso usamos el **window.onload** para de una llamar a la función que me genera todos los productos.

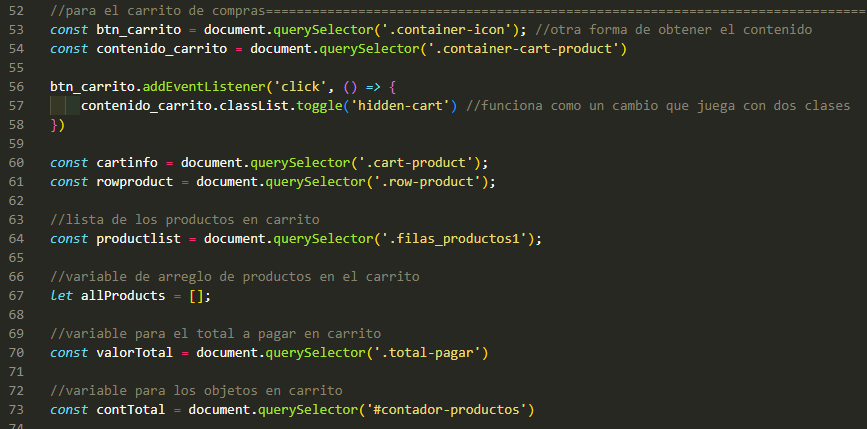
**“CARRITO”:**

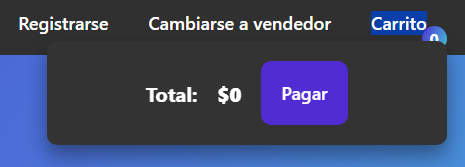
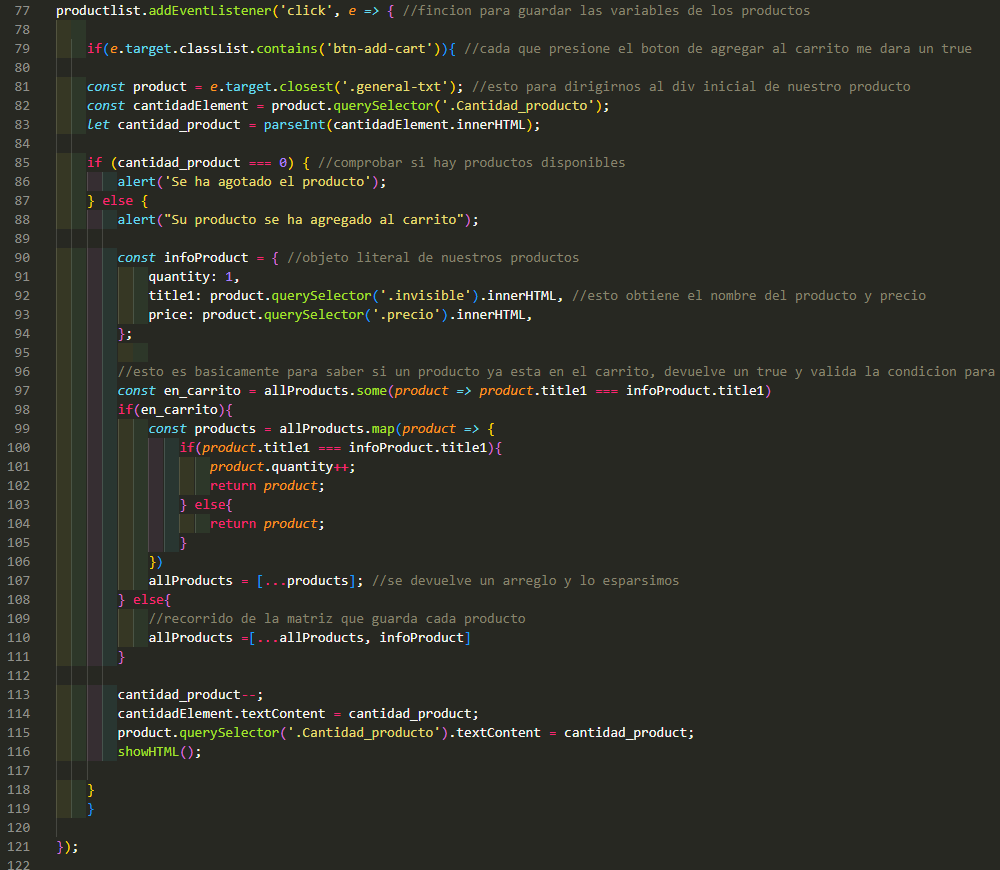
Esta es la función más extensa de nuestra página, por eso la vamos a explicar por partes para darles a entender como se ha logrado la función de compra

Primero tenemos que aclarar que nosotros nos guiamos de un video en YouTube de cómo hacer este código y lo hemos estudiado y modificado para que funcione en Bynary Shop

En **HTML** se definen el esqueleto que van a tener nuestros productos cuando sean agregados al carrito, se definen varias clases las cuales tendrán precio, cantidad en el carrito de productos, el nombre, la cantidad total a pagar, cantidad de productos en el carrito, etc.., debemos de resaltar la clase “container-cart-product” que en ella es que se guardaran los productos que hayamos agregado.

En el **CSS** se manejó la ventana de carrito, el tamaño, ubicación, botones, etc.., también de han definido clases invisibles, por eso te habrás dado cuenta de que los productos tienen ciertos spans con class “invisible” o “invisible2”, esto para que se nos facilite el manejo en el js.

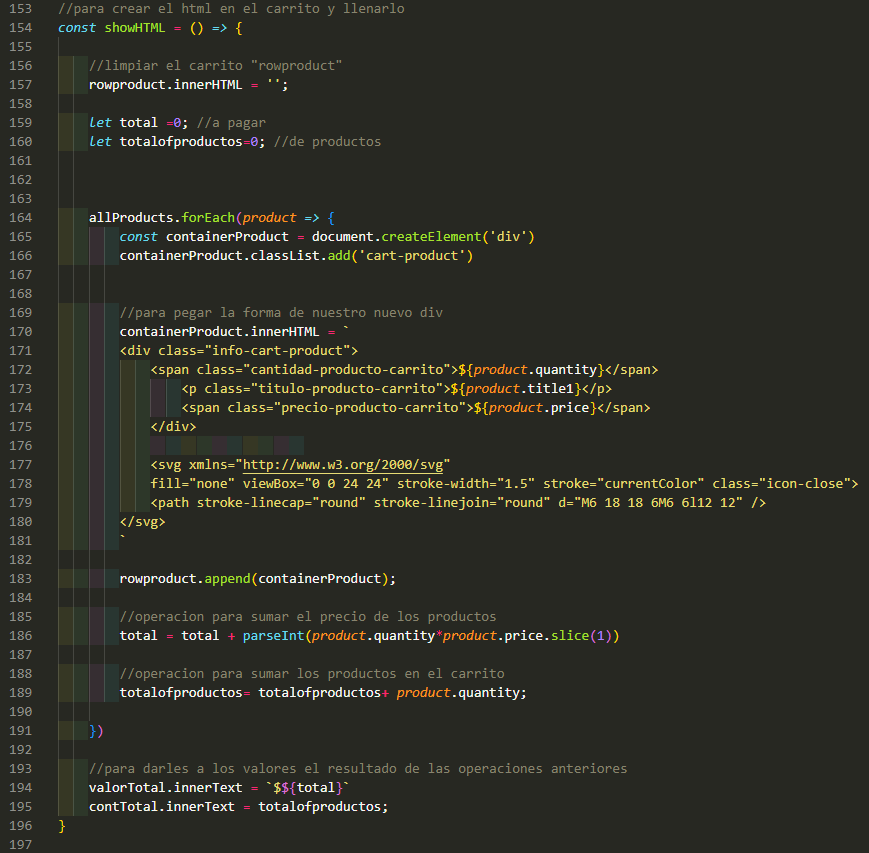
 En el **Java Script**

1. lo primero que se hace es una función de ventana emergente para cuando presionemos carrito se habrá la ventana de carrito, luego se definen varios const con querySelector (‘class’) para manejarlos, los importantes son el productlist donde se contiene la lista de todos los productos, con el objetivo de crear addEventListeners para hacer las función de agregado al carrito y compra, y la creación de un arreglo allProducts[] que contendrá los productos que le agreguemos al carrito y los muestre, se definen también los contadores de los productos tanto la cantidad total de productos en el carrito como la cantidad de un producto en el carrito.
2.  Este **eventlistener** comprueba que hayamos clicado el botón que tiene cada producto de la página “agregar al carrito” luego se dirigirá la general.txt, que es el **div** del producto que contiene toda la información de el cómo nombre, cantidad, etc , se comprueba de que la cantidad del producto no se haya agotado para que se agregue al carrito, (esta función la definiremos más adelante) si no se ha agotado se instancia un objeto llamado InfoProduct que tendrá **quantity**, **title1**, **price**, los atributos de nuestro producto, algo parecido vimos con el arreglo de productos.

Otra cosa que hace este código es que comprueba si un producto ya se ha agregado al carrito previamente, y en ese if hay otro if en el que se comprueba si ya hay un producto por el nombre, si es así se suma el **quantity** y se retorna **product**, si no es así simplemente se retorna **product**, por último, allProducts = [...products]; crea una copia del arreglo **products** y la asigna a allProducts. Si no se cumple ese if **en\_carrito**, se hace allProducts = [...allProducts, infoProduct]; que agrega un nuevo elemento **infoProduct** al array allProducts[] sin que se modifique el original, y así para sumar los productos dentro del carrito. Y al final hace algo del contador y llama a showHTML().

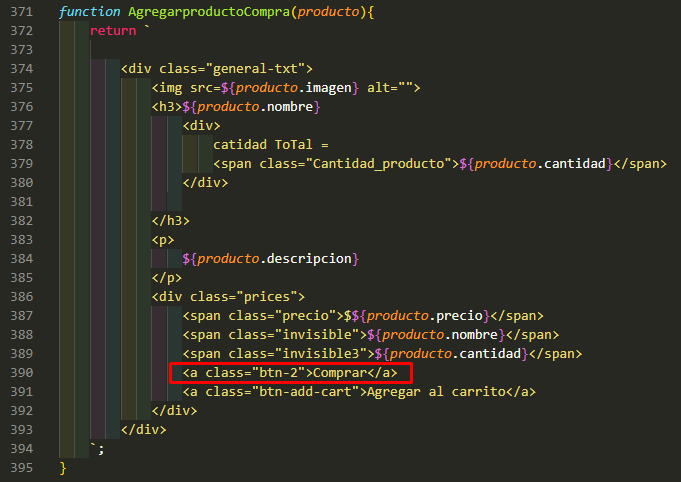
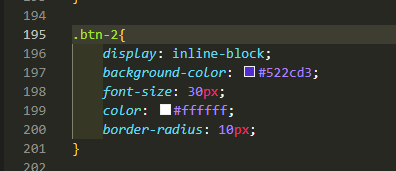
1. Este **eventlistener()** comprueba que se halla clicado un botón con la forma de **x** en el carrito para elimínalo se definen **const** con el nombre, cantidad, etc.

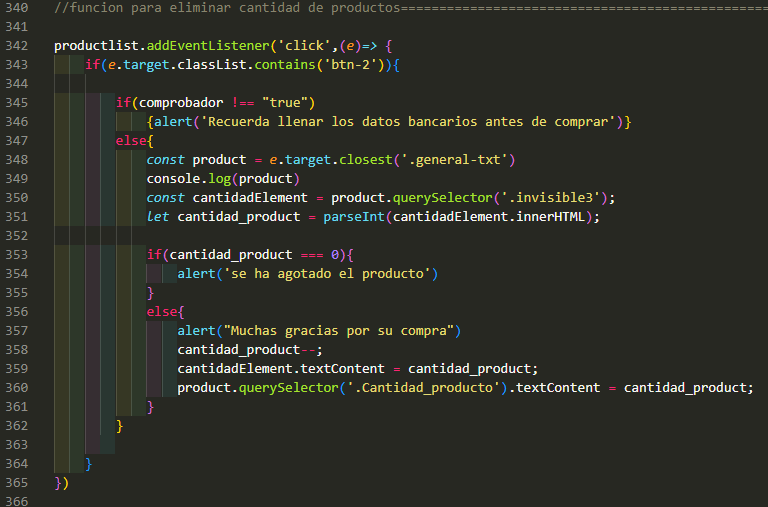
Se utiliza el .filtrer que es una función que comprueba de que no se repita un **título**, en caso de que se repita se elimina, luego se realiza una función del contador de productos y al salir de ese comprobador llama a la función showHTML().

1.  Esta es la funcion showHTML() que hemos mencionado anteriormente, aquí se limpia el contenido del carrito para que no se repita el producto, se definen dos contadores que son los del total en carrito y el total a pagar, y realiza un **for** por cada producto que tenga el arreglo allProducts[] se crea un **div** con las clases que personalizamos ya en el **CSS** sobre el carrito, en containerProduct.innerHTMl se pega la forma del **div** de los productos **HTML** en el carrito utilizando nuevamente las plantillas literales, y en cada **${}** se agrega la **cantidad**, **title**, **Price** del producto especifico.

El .append se utiliza para agregar el containerProduct que tiene el código **HTML** de los productos al **rowproduct,** que es el lugar en donde está el carrito.

Al final se realizan operaciones que suman el contador de productos en el carro y la cantidad total que el usuario debe de pagar.

**“COMPRAR Y CANTIDAD”:**

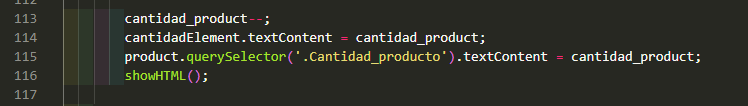


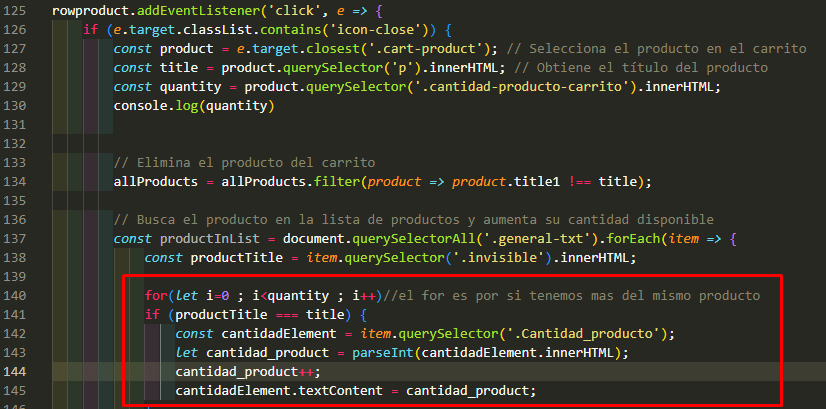
Definamos entonces la función de compra que está ligada con la cantidad de productos también, el **HTML** será el del producto que agregamos a la página con el for, allí tendremos una **class** llamada **“btn-2”,** también tenemos otra **class** llamada **“Cantidad\_Producto”** en donde estará la cantidad del producto.

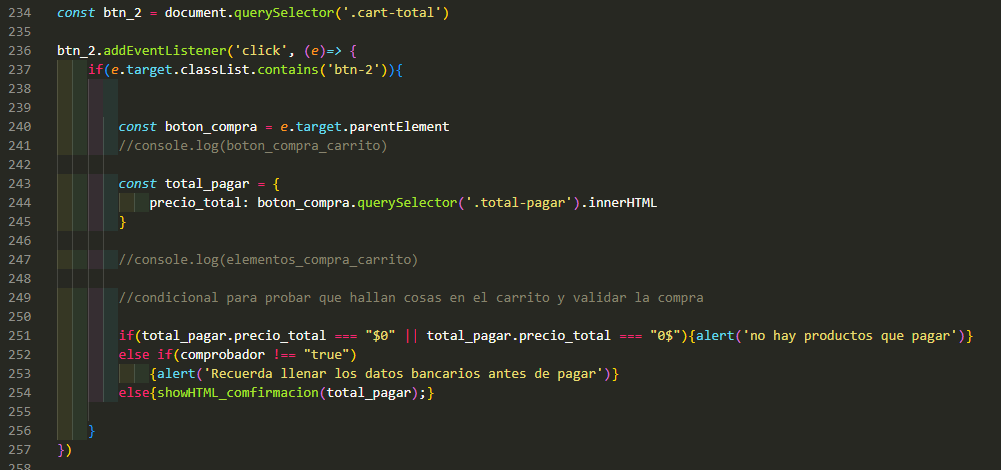
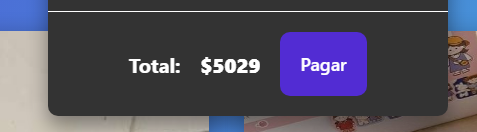
El **CSS** solo lleva al botón y su diseño junto con una ventana emergente que se utilizara cuando paguemos el total a pagar en carrito

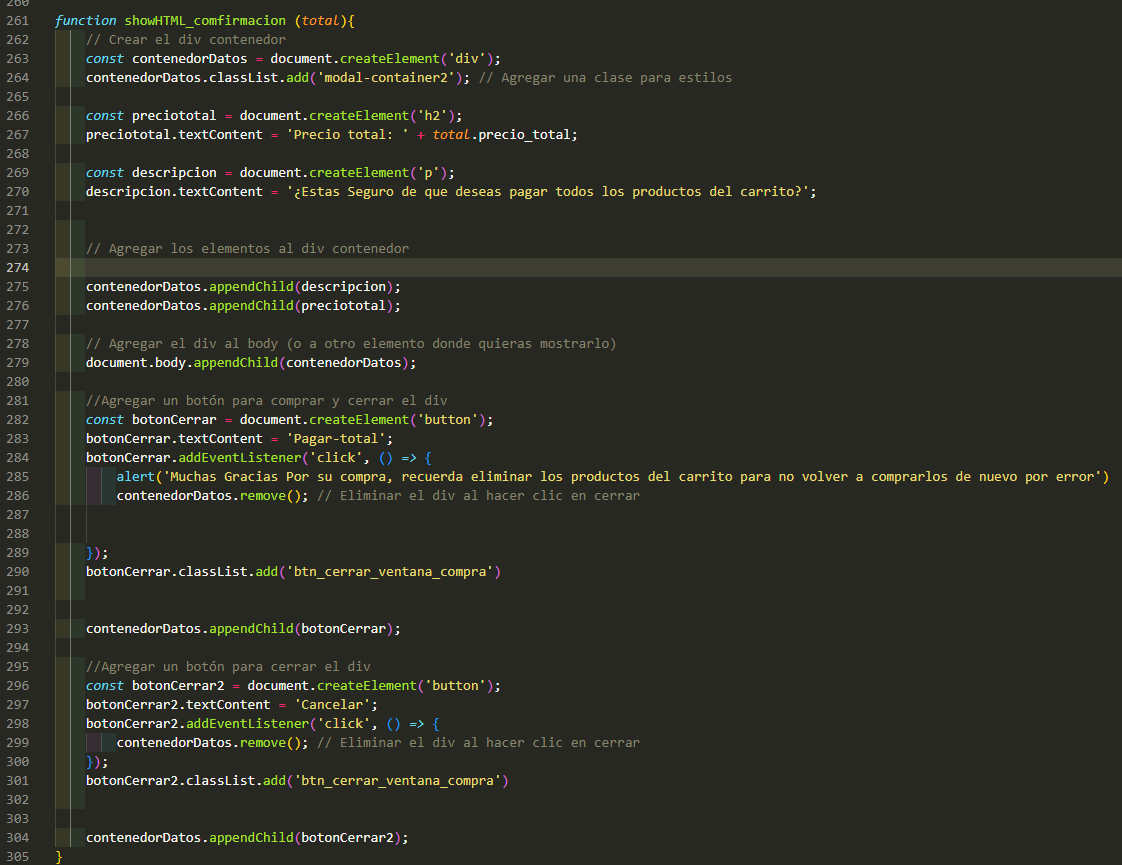
En el **Java Script** tenemos dos casos en donde la cantidad de producto se disminuye y una en la que aumenta:

1. Cuando presionamos el botón comprar tenemos un **Eventlistener()** con un **if** y un comprobador, este comprobador recordemos que lo definimos en Datos Bancarios, en este **if** se comprueba si se han llenado los datos bancarios para proceder con la compra, si no es así te pedirá llenar los datos bancarios, si ya los llenaste se le agradece al usuario por la compra y se resta la cantidad de producto.



1. Recordemos también que en el eventListener de **productList** tenemos casi a lo último, esta línea de código, que básicamente resta la cantidad del producto, ósea que cuando agreguemos un producto al carrito este también restara la cantidad del producto
2. En el tercer caso el contador aumenta, y es cuando eliminamos un producto del carrito, en este caso como puede haber más del mismo producto en el carrito se hace un **for** que itera hasta la **quantity** que es la cantidad de producto en el carrito, adentro de ese **for** se obtiene la cantidad del producto y se suma. Con este **for** aseguramos que se devuelvan la cantidad correctas de productos.

**“PAGAR CANTIDAD TOTAL”:**



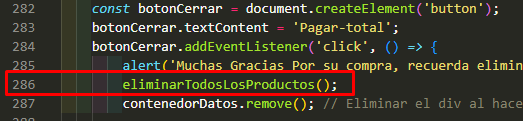
Y por ultimo en el **HTML** del comprador nos queda Hablar del boton pagar en la ventana del carrito.

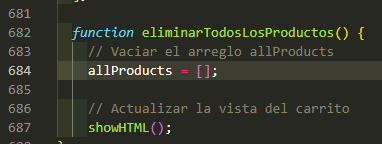
El **CSS** es el mismo que **“btn-2”.**

El **Java Sript** comprueba que hallamos clickeado el boton, y comprueba de que tengamos algo en el carrito, porque si no tnemos nada no podemos pagar, luego comprueba de que se hallan llenado los datos bancarios con el **if** del comprobador, si se cumplen las condiciones hara la funcion **showHTML\_comfirmacion().**

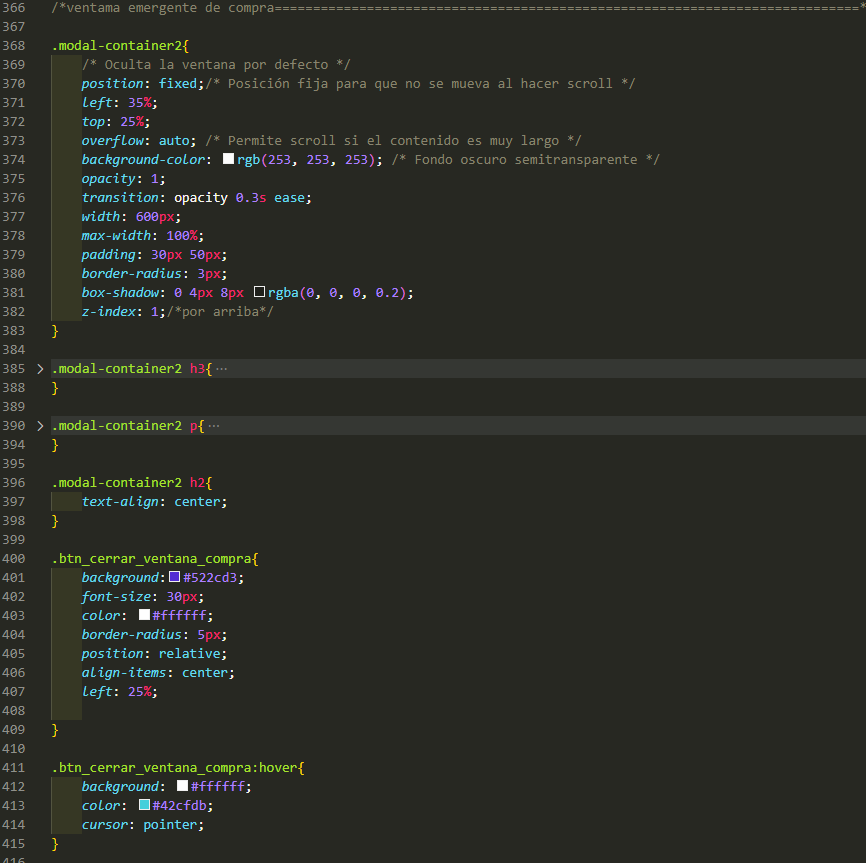
Esta funcion genera Una nueva **div**, una ventana emergente en el que aparece el precio total a pagar, en donde tiene dos botones, pagar y cancelar, en con ambos

saldras de esa ventana emergente, y por su puesto si pagas te agradece, desde este punto nos dimos cuenta de que no se eliminan los productos del carrito una vez pagados y implementamos esta funcion a lo ultimo del codigo para no arruinar el numero de las lineas de codigo:



Esta función ya me elimina todo lo que este en el carrito.

También he de mencionar que el **CSS** de la ventana emergente solo lo ordena y también sigue al hacer **scroll**.



El **CSS** de la ventana emergente

**Usuario Administrador**

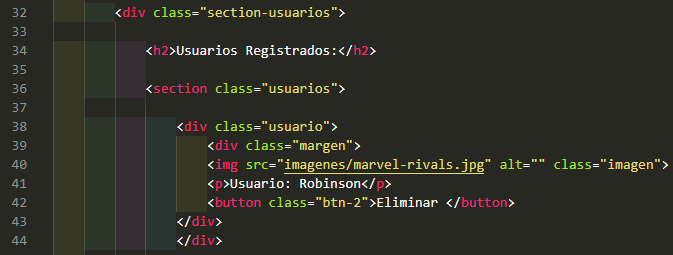
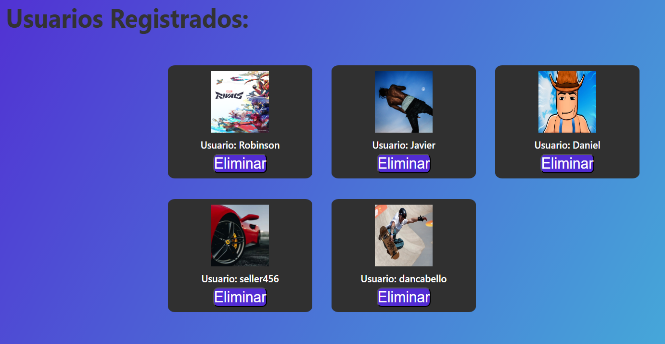
Ya con los códigos aprendidos en el usuario vendedor nos resulta más fácil las funciones del usuario Administrador que es solo eliminar, es un HTML diferente de nombre **usuario\_administrador.html** vinculado con el mismo **CSS** del **usuario\_comprador.html** pero el Java Script si es diferente para no sobrecargar más el **proyecto.js.**

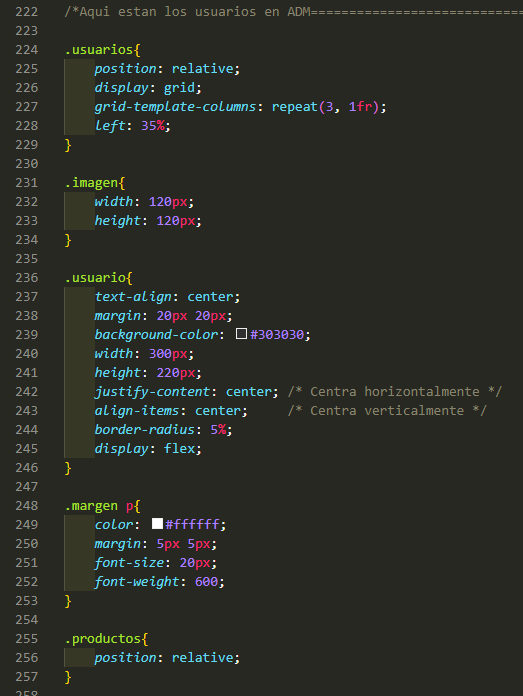
* **CSS:**



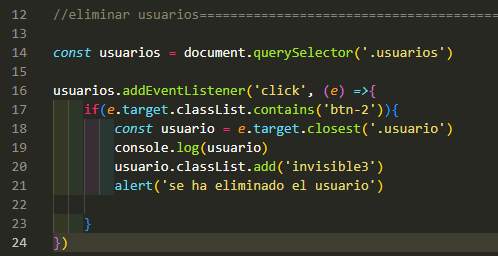
* **Java Script:**

El **HTML** es básicamente un reciclado del usuario comprador, quitando todas las opciones de la **navbar** dejando solo cerrar sesión, y se ha agregado una nueva sección de usuarios

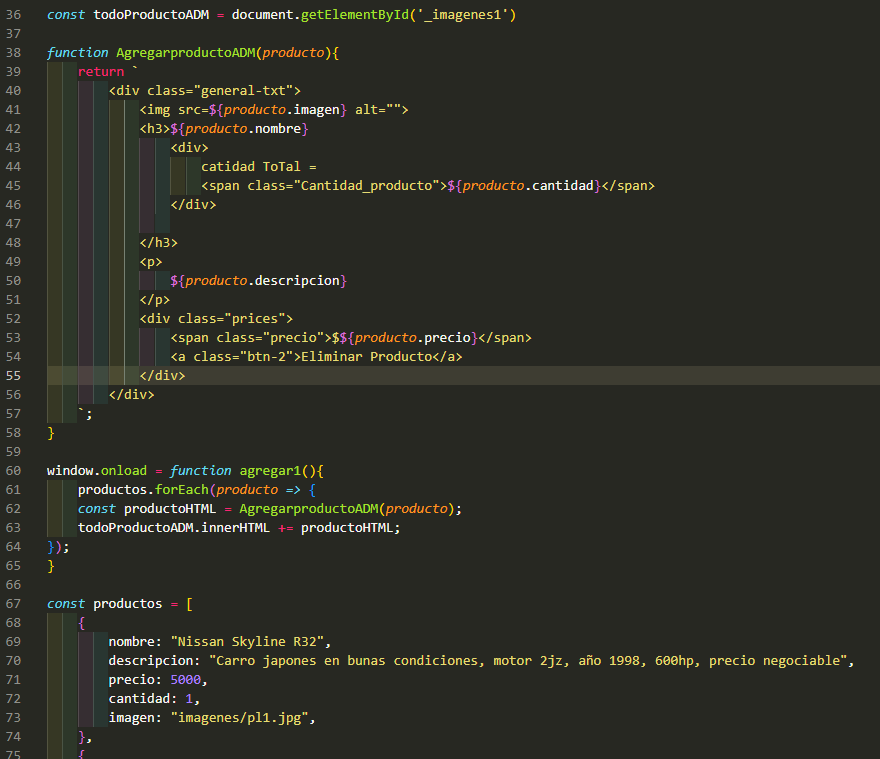
**“Usuarios y eliminado”:**

El **HTML** se tiene una **class** “usuario” que en ella contiene la información de cada usuario, esta **class** se repite 5 veces más para los usuarios visibles en la página.

En el **CSS** se ordenaron en columnas de tres y se les dio su posición, también se ha creado una **class** “invisible3” que nos servirá para eliminarlos con ayuda del **js**.

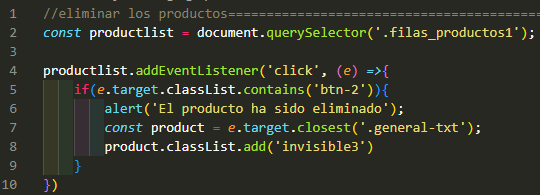
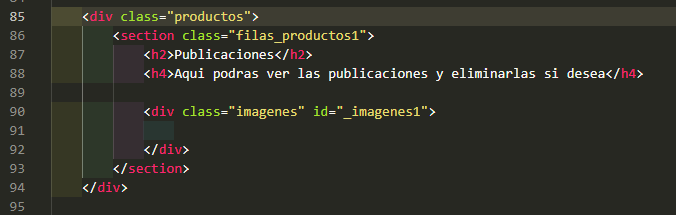
En el **Java Script** se le añade el EventListener para cuando se presione el botón eliminar, este añade la **class** “invisible3” que contiene un **display:none** haciendo que el usuario se elimine, esta misma lógica aplica para cuando eliminemos un producto

**“Productos y eliminado”:**

EL **HTML** es el mismo que el del usuario comprador con cambios en la **navbar** como hemos mencionado previamente junto con el agregado de los productos.

El **CSS** no cambia pues los productos son los mismos con cambio mínimo de los botones.

En el **Java Script** Tenemos un reciclado de función que utilizamos en el **proyecto.js** con la diferencia de que le hemos cambiado los botones de comprar y agregar al carrito por un botón eliminar.

Para eliminar los productos se agrega un **EventListener()** para comprobar que hallamos clicado el botón eliminar y este procederá a agregar la **class** “invisible3” que los elimina, es literalmente el mismo código que se utilizó en eliminado de usuario.

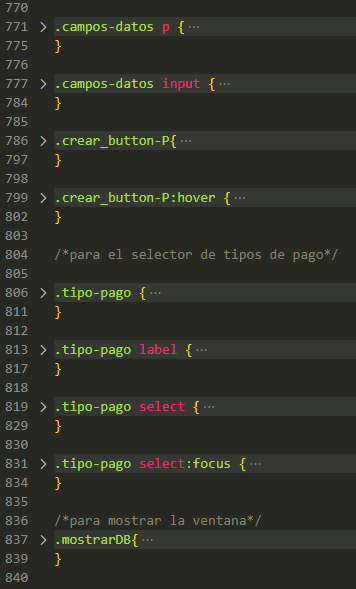
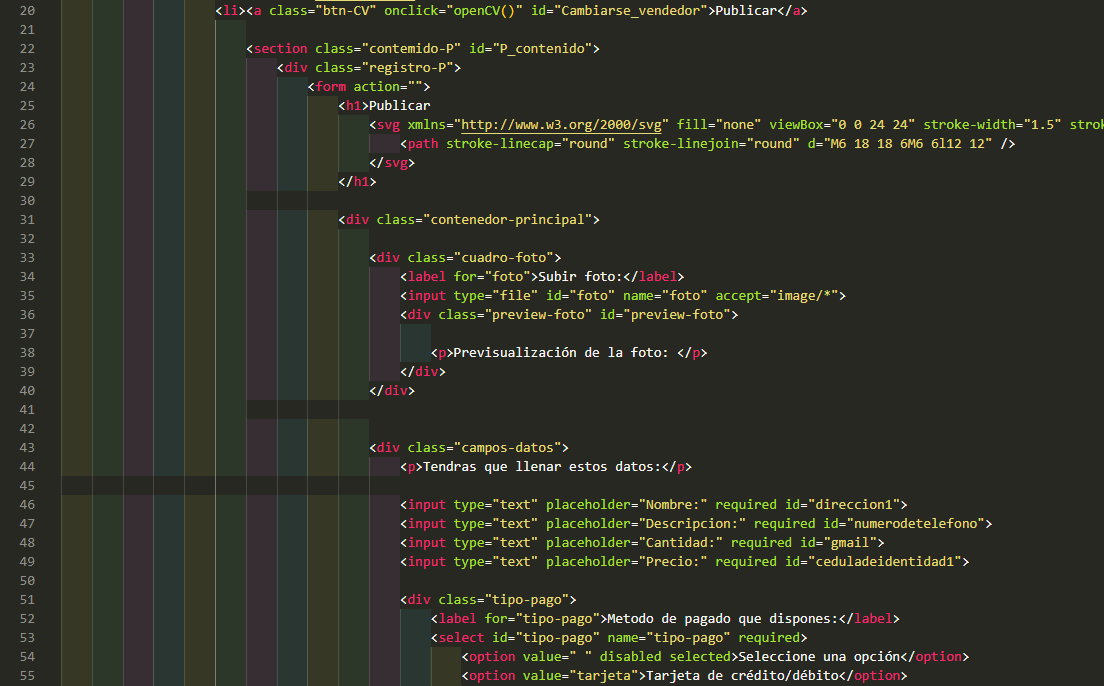
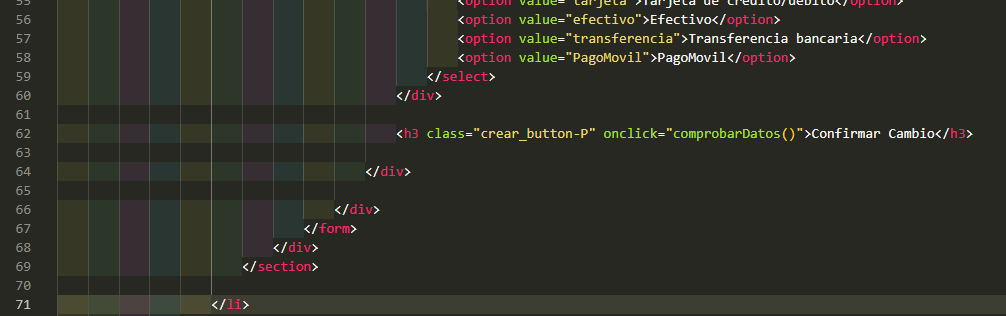
**Usuario Vendedor**

En este usuario también se han reciclado varias funciones, se le ha creado un **Java Script** nuevo para facilitar la comprensión y está vinculado al mismo **CSS** que el anterior código.

* **CSS**:



* **Java Script**:

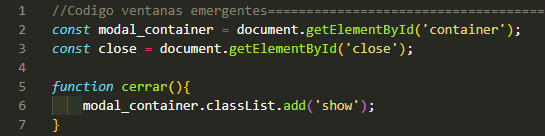
**“Publicado De Producto”:**

El **HTML** general de la página es el mismo, con la diferencia de que solo se podrán ver los productos y el **navbar** tiene cerrar sesión, registro y publicar, Cuando ciclemos comprar nos saltará una ventana emergente en el que nos pedirá información de nuestro producto a publicar junto con una sección en la que tenemos que publicar una foto, también debemos de elegir los métodos de pagos de los cuales disponemos.

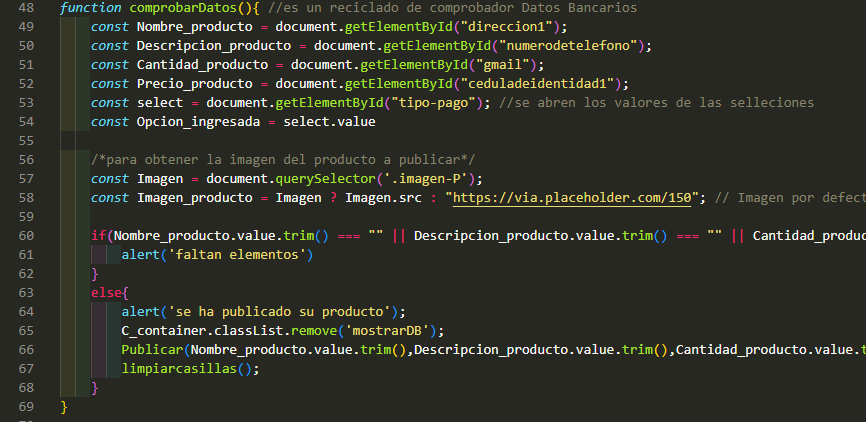
El **HTML** se divide en dos campos, el campo en donde se publica la foto y el otro en donde se llena la información, lo nuevo aquí es el **input** que se le pone en la sección de la foto, al presionar **“Subir Foto”** se abrirá el director para proceder a que publiquemos una imagen.

El **CSS** se ha manejado de tal manera que tengamos las dos secciones separadas bien ordenadas, también seguimos utilizando **display:none** para aparecer y desaparecer las ventanas.

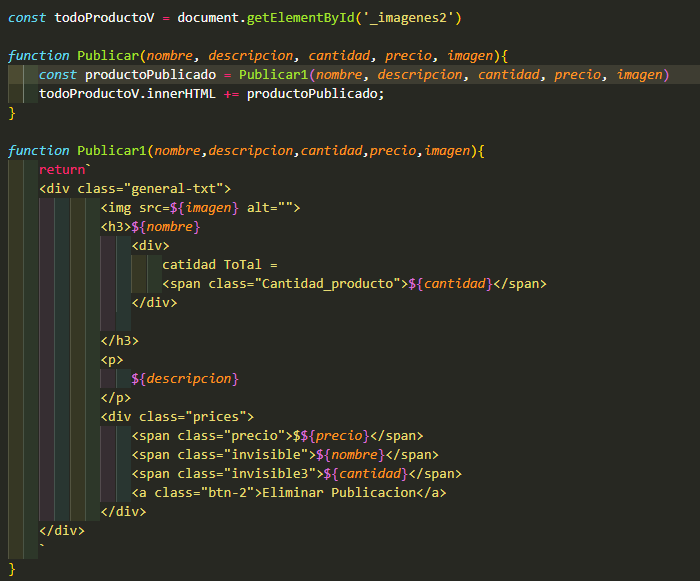
En el **Java Script** se ha agregado una función que me abre la ventana emergente, este también se divide en dos secciones, una que trabaja con la imagen que le hemos dado y comprobado de datos:

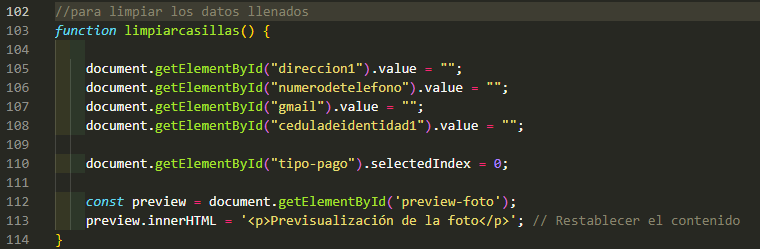
1. Principalmente obtenemos la foto publicada. El **EventListener change** funciona para que cuando nosotros seleccionemos una foto para publicar llame al evento, luego se genera un archivo de la foto seleccionada, se hace un **if** comprobando si se ha seleccionado un archivo, si no es así se quedara vacío el cuadro de previsualización, y si es así se ejecutara **fileReader(),** este lee el contenido del archivo que hemos compartido. Se cargara la foto que hemos compartido con el **.onload,** y **.readAsDataURL** nos ayudara a convertir la foto en dato URL que nos permitirá usar **src** para manejar mejor la foto, y el comprobador será **true**, con este código podemos comprobar que se haya publicado una foto y se verá la foto reflejada en previsualización. Y si en caso de que cerremos la ventana publicar y volvamos a abrirla se elimina la foto que hayamos publicado y queda como estaba originalmente.

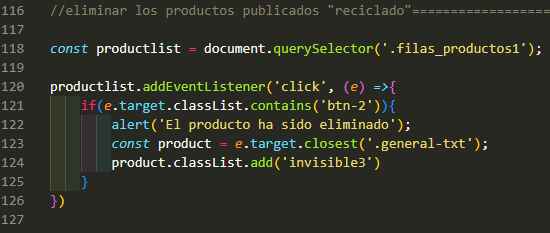




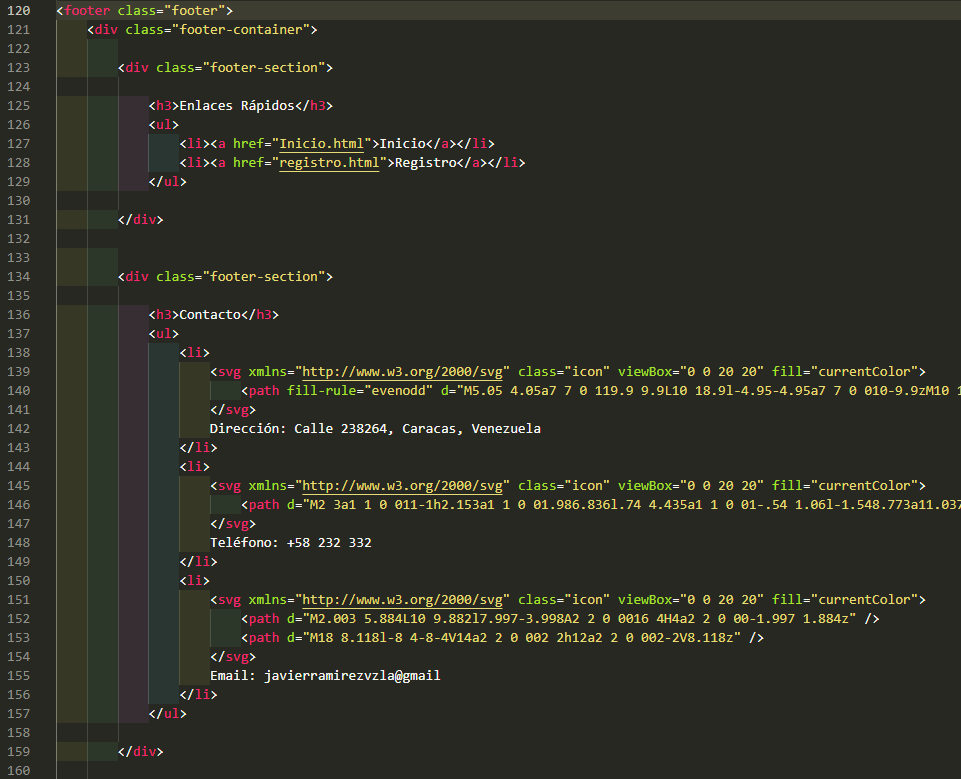
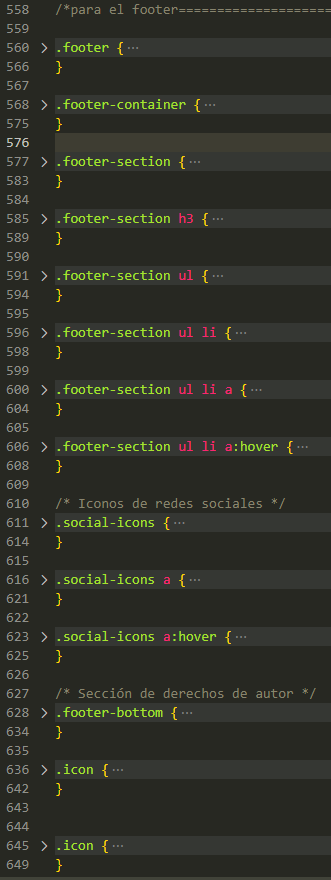
1. Ahora la siguiente sección del **Java Script** es el comprobado de datos, cuando ciclemos el botón de publicar producto llamara a la función **comprobarDatos(),** que es un reciclado de los comprobadores de llenado de datos bancarios, aquí se obtienen los datos ingresados junto con la foto ingresada también para usarla más adelante, se utilizó **Imagen.src** que nos ayuda a obtener el formato de imagen correcto **src**, luego se procede a realizar un condicional que comprueba que los datos se han llenado, si no se llenaron saltara la alerta de que no se han llenado, y si es así se notifica que se ha publicado el producto, se elimina la ventana emergente, y se llaman dos funciones **Publicar()** y **Limpiarcasillas().**

**Publicar()** con ayuda del **.value** la función recibe como parámetros el nombre del **producto**, **descripción**, **precio**, **cantidad** e **imagen**, en la función se definen el **const** del producto publicado que será igual al llamado de **Publicar1()** con los mismos parámetros, esta función retorna por sorpresa de todos el código “general.txt” de las publicaciones entre las plantillas literales y modificándose con **${}** se le agregan los datos de nuestro producto, y por último de la función **Publicar()** se le agrega el producto publicado a **todoProductoV**, que en él está el lugar en donde se guardaran nuestras publicaciones en el HTML.

**Limpiarcasillas().** Básicamente elimina los datos que hemos ingresados para que nos permita volver a publicar otro producto

Y por último también agregamos la función de eliminar publicación, no puede faltar.

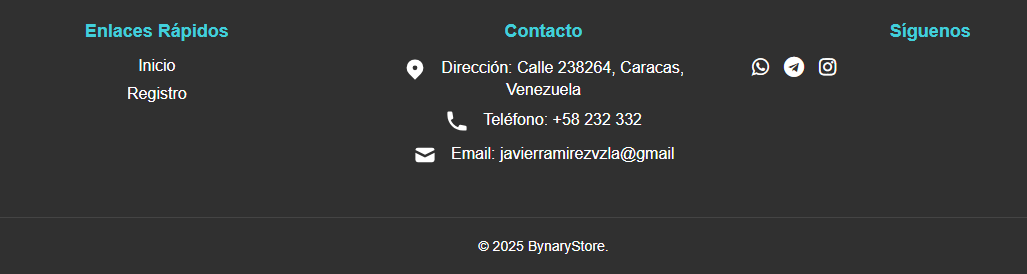
**El Footer**

No puede faltar tampoco el footer de nuestra página, solo es **HTML** y **CSS** agregado a las tres interfaces, de compra, venta y administración, no es más que decoración para finalizar nuestra página.



En el **HTML** fotter tiene cuatro divisiones, una para los enlaces, otra para el contacto, una para las redes sociales y una ultima para los derechos, se han utilizado inconos para representar cada informacion.

El **CSS** se ha manejado de tal manera que la informacion quede centrada y ordenada y se le dieron hovers y animaciones.



**Referencias e bases**

Solo falta mencionar que para la creación de la página web se han analizado varios videos de YouTube de los cuales enseñaban **HTML**, **CSS** y **Java script**, también se utilizaron páginas webs que nos ayudaron a decorar con símbolos lo que tiene que ver el footer y la x de las ventanas emergentes, también se ha utilizado inteligencias artificiales mínimamente para poder entender ciertas funciones y elementos que nos ayudaron a crear las funciones y decorado del proyecto.

Videos:

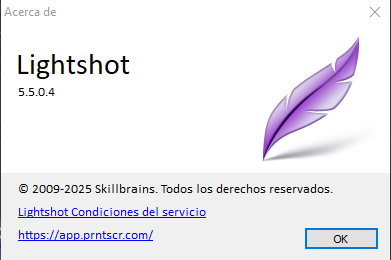
* <https://youtu.be/3BtyzOBLRcg?si=DHpGmFGg2D2gRM7N> Para el carrito.
* <https://youtu.be/3L_Wt6rfBvQ?si=DyGJg16FyWEE_n_M> Para también el carrito.
* <https://youtu.be/I394Leiw5nU?si=mCeEvG1q9CWwiRJM> Para manejar los **HTMLS** y **CSS** de la página general.
* <https://youtu.be/5c8NLiKW5aI?si=VE5L-zmxPugY2kJG> Para las ventanas emergentes.
* <https://youtu.be/9IlagqeePYs?si=fEjwWsR8zet-02cY> Para el Loguin.

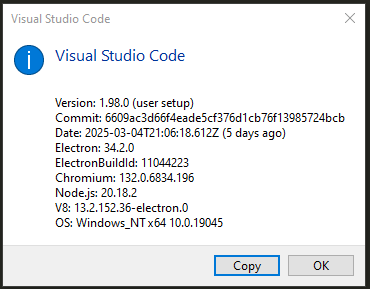
Páginas webs:

* <https://heroicons.com/> Para iconos del **CSS**;
* <https://www.w3schools.com/> Para iconos también y ciertas funciones del **Java Script**
* <https://github.com/RoaRobinson97/productos_ejemplo/blame/main/productos.js> para la lista de productos

IA:

* <https://chat.deepseek.com/> Se utilizo para entender cómo funcionan las Plantillas literales y a como se hacen los **.addEventListener** del **java sript**, el **querySelector**, etc.
* <https://gemini.google.com/app/a8a7b1030bed154d> Para la comprensión de código igualmente.

Aplicaciones:

* Se ha utilizado para las capturas de pantallas del código y de la página web:
* Programa con el que se ha programado: