

Universidad ORT Uruguay
Facultad de Ingeniería
Escuela de Tecnología

OBLIGATORIO PROGRAMACION 1

DOCUMENTO DE ANÁLISIS



Pablo Larnaudie – 340181



Natalia Rebella – 327283

Grupo: N1F

Docente: Mauro Nascimento

Analista en Tecnologías de la Información

Fecha de entrega del documento (20-06-2024)

Índice

1.	Descripción general del problema a resolver	5
1.1.	Tipos de usuario del sistema	5
1.2.	Listado de funcionalidades	5
2.	Detalle de Funcionalidades.....	6
2.1	F01 – Registro como comprador	6
2.1.1	Acceso	6
2.1.2	Descripción.....	6
2.1.3	Interfaz de usuario	6
2.1.4	Validaciones	7
2.2	F02 – Ingreso al sistema	7
2.2.1	Acceso	7
2.2.2	Descripción.....	7
2.2.3	Interfaz de usuario	8
2.2.4	Validaciones	8
2.3	F03 – Compra de productos.....	8
2.3.1	Acceso	8
2.3.2	Descripción.....	9
2.3.3	Interfaz de usuario	9
2.3.4	Validaciones	9
2.4	F04 – Visualización del listado de compras	10
2.4.1	Acceso	10
2.4.2	Descripción.....	10
2.4.3	Interfaz de usuario	10

2.4.4	Validaciones	11
2.5	F05– Visualización de productos en oferta.....	11
2.5.1	Acceso	11
2.5.2	Descripción.....	11
2.5.3	Interfaz de usuario	11
2.5.4	Validaciones	12
2.6	F06– Listado y aprobación de la compra de los usuarios	12
2.6.1	Acceso	12
2.6.2	Descripción.....	12
2.6.3	Interfaz de usuario	12
2.6.4	Validaciones	13
2.7	F07– Creación de productos	13
2.7.1	Acceso	13
2.7.2	Descripción.....	13
2.7.3	Interfaz de usuario	14
2.7.4	Validaciones	14
2.8	F08– Administración de productos	14
2.8.1	Acceso	14
2.8.2	Descripción.....	14
2.8.3	Interfaz de usuario	15
2.8.4	Validaciones	15
2.9	F09– Visualización de informes de ganancia	15
2.9.1	Acceso	15
2.9.2	Descripción.....	16
2.9.3	Interfaz de usuario	16
2.9.4	Validaciones	16

3.	Datos precargados	17
4.	Código completo comentado	19
4.1	HTML.....	19
4.2	CÓDIGO.JS:.....	25
4.3	SISTEMA.JS:	47
4.4	CLASES.JS:	54

1. Descripción general del problema a resolver

Se desea desarrollar un sistema de e-commerce para un negocio de artículos deportivos, mediante el cual, los usuarios puedan realizar compras online.

1.1. Tipos de usuario del sistema

- Administrador (usuarios que administrarán los productos y las compras)
- Comprador (usuarios que podrán navegar por el sitio y realizar compras de productos)

1.2. Listado de funcionalidades

Comprador

- F01 – Registro como comprador – Usuario/s: Comprador
- F02 – Ingreso al sistema – Usuario/s: Comprador
- F03 – Compra de productos – Usuario/s: Comprador
- F04 – Visualización del listado de compras – Usuario/s: Comprador
- F05 – Visualización de productos en oferta – Usuario/s: Comprador

Administrador

- F06 – Listado y aprobación de las compras de los usuarios – Usuario/s: Administrador
- F07 – Creación de productos – Usuario/s: Administrador
- F08 – Administración de productos – Usuario/s: Administrador
- F09 – Visualización de informes de ganancias – Usuario/s: Administrador

2. Detalle de Funcionalidades

A continuación, se presenta el detalle de cada una de las funcionalidades a resolver en el obligatorio. Las funcionalidades están ordenadas por tipo de usuario.

2.1 F01 – Registro como comprador

2.1.1 Acceso

Puede acceder el usuario: Comprador

2.1.2 Descripción

Los usuarios pueden registrarse en el sistema, completando los campos presentes (nombre, apellido, nombre de usuario, contraseña, número de tarjeta de crédito y CVC) y los diversos requisitos, logrando así un registro exitoso.

2.1.3 Interfaz de usuario



The screenshot displays a web interface for user registration. At the top left, there is a small icon of a shopping cart. The main heading is "REGISTRO" in green. Below it, there are four input fields with labels: "NOMBRE DE USUARIO", "CONTRASEÑA (AL MENOS 5 CARACTERES, UNA MAYUSCULA, UNA MINUSCULA Y UN NÚMERO)", "NUMERO DE TARJETA DE CREDITO" (with a placeholder "WWW-XXXX-YYYY-ZZZZ"), and "CVC" (with a placeholder "123"). At the bottom, there are two buttons: "REGISTRARSE" and "VOLVER AL LOGIN".

2.1.4 Validaciones

- Todos los campos son obligatorios.
- Validar que el nombre de usuario sea único en el sistema y case insensitive.
- La contraseña debe ser un String y contener al menos una mayúscula, una minúscula y un número.
- Validar que el número de tarjeta de crédito tenga el formato correcto, siendo este 4 grupos de 4 números separados por guiones.
- Validar el número de la tarjeta de crédito contra el algoritmo de Luhn.
- El CVC debe respetar el formato numérico de 3 dígitos.

2.2F02 – Ingreso al sistema

2.2.1 Acceso

Puede acceder el usuario: Comprador y Administrador.

2.2.2 Descripción

Esta funcionalidad dará acceso al sistema. Los usuarios una vez registrados podrán ingresar al sistema utilizando su usuario y contraseña para realizar el login.

2.2.3 Interfaz de usuario



LOGIN

TIPO DE USUARIO

Seleccione una opcion... ▼

USUARIO

CONTRASEÑA

INGRESAR

REGISTRATE

2.2.4 Validaciones

- Verificar que el usuario exista y la contraseña ingresada sea la correcta.

2.3F03 – Compra de productos

2.3.1 Acceso








Puede acceder el usuario: Comprador.

2.3.2 Descripción

El usuario podrá visualizar los productos disponibles para comprar. Los mimos detallarán su nombre, descripción, precio y condición (oferta o no). A su vez se acompañarán de una imagen ilustrativa.

También podrán efectuar compras, que quedarán en estado “pendiente” hasta ser confirmadas por el administrador.

2.3.3 Interfaz de usuario

						
BIENVENIDA: NATALIA PRODUCTOS PRODUCTOS EN OFERTA 3000 MIS COMPRAS						
SALIR						
ID	Nombre	Producto IMG	Descripción	Precio	Condición	Cantidad de Unidades
PROD_ID_1	Botella		Botella de plástico duro	100	Precio Regular	<input type="text" value="1"/> <button>COMPRAR</button>
PROD_ID_2	Inflador		Un inflador de pelota con múltiples punteros	450	Oferta	<input type="text" value="1"/> <button>COMPRAR</button>
PROD_ID_3	Mesa de Juego		Mesa de Tejo para tu diversion	200	Precio Regular	<input type="text" value="1"/> <button>COMPRAR</button>
PROD_ID_4	Bolsa Grande		Bolso de tela impermeable	100	Precio Regular	<input type="text" value="1"/> <button>COMPRAR</button>
PROD_ID_5	Kit de Entrenamiento		Todo lo necesario para tu entrenamiento	300	Oferta	<input type="text" value="1"/> <button>COMPRAR</button>
PROD_ID_6	Pack de Ciclismo		Articulos para ciclismo	250	Precio Regular	<input type="text" value="1"/> <button>COMPRAR</button>

2.3.4 Validaciones

- Verificar que el producto esté disponible para la compra (stock mayor a 0 y estado “activo”)
- Solamente se podrá comprar un producto por orden de compra.

2.4F04 – Visualización del listado de compras

2.4.1 Acceso


Puede acceder el usuario: Comprador.

2.4.2 Descripción

Esta funcionalidad permite al usuario visualizar un listado con sus compras, detallando nombre del producto, cantidad de unidades compradas y monto total. Las mismas podrán ser filtradas a partir de las opciones: todas las compras, aprobadas, canceladas o pendientes.

Así mismo, se permite al usuario cancelar aquellas compras que aún no hayan sido confirmadas por el administrador y por tanto se encuentren en estado “pendiente”.

2.4.3 Interfaz de usuario



ID	Nombre Producto	Nombre Cliente	Cantidad de Unidades	Monto total	Estado	
1	PROD_ID_1	1	2	100	Aprobado	CANCELAR
2	PROD_ID_2	2	2	450	Aprobado	CANCELAR
3	PROD_ID_3	3	2	200	Aprobado	CANCELAR
4	PROD_ID_3	3	2	200	Aprobado	CANCELAR
5	PROD_ID_2	2	2	450	Aprobado	CANCELAR
6	PROD_ID_4	4	2	100	Pendiente	CANCELAR
5	PROD_ID_5	4	2	300	Cancelada	CANCELAR

2.4.4 Validaciones

- Validar que el botón “cancelar compra” solo se encuentre disponible en aquellas compras que se encuentran en estado “pendiente”.
- En el párrafo final mostrar el monto de todas las compras realizadas que se encuentran en estado “aprobadas”.

2.5 F05– Visualización de productos en oferta

2.5.1 Acceso

Puede acceder el usuario: Comprador.

2.5.2 Descripción

El usuario contará con una vista en la que se muestre y pueda comprar únicamente los productos en oferta.

2.5.3 Interfaz de usuario



ID	Nombre	Producto IMG	Descripción	Precio	Condición	Cantidad de Unidades	
PROD_ID_2	Inflador		Un inflador de petola con múltiples punteros	450	Oferta	<input type="text" value="1"/>	COMPRAR
PROD_ID_5	Kit de Entrenamiento		Todo lo necesario para tu entrenamiento	300	Oferta	<input type="text" value="1"/>	COMPRAR
PROD_ID_8	Kit de Pelotas		Pelotas para practicar distintos deportes	368	Oferta	<input type="text" value="1"/>	COMPRAR

2.5.4 Validaciones

- Validar que se muestre al usuario solo productos en oferta.

2.6 F06– Listado y aprobación de la compra de los usuarios


2.6.1 Acceso

Puede acceder el usuario: Administrador

2.6.2 Descripción

El administrador contará con una vista que le permita visualizar, operar y modificar el estado de las compras de los usuarios.

2.6.3 Interfaz de usuario



Listado de Compras

Compras Aprobadas

IDCompra: 1	IDProducto: PROD_ID_1	IDCliente: 1	Cantidades: 2	Estado: Aprobado	Precio: 100
IDCompra: 2	IDProducto: PROD_ID_2	IDCliente: 2	Cantidades: 2	Estado: Aprobado	Precio: 450
IDCompra: 3	IDProducto: PROD_ID_3	IDCliente: 3	Cantidades: 2	Estado: Aprobado	Precio: 200
IDCompra: 4	IDProducto: PROD_ID_3	IDCliente: 3	Cantidades: 2	Estado: Aprobado	Precio: 200
IDCompra: 5	IDProducto: PROD_ID_2	IDCliente: 2	Cantidades: 2	Estado: Aprobado	Precio: 450

Compras Pendientes

IDCompra: 6	IDProducto: PROD_ID_4	IDCliente: 4	Cantidades: 2	Estado: Pendiente	Precio: 100	APROBAR
-------------	-----------------------	--------------	---------------	-------------------	-------------	---------

Compras Canceladas

IDCompra: 5	IDProducto: PROD_ID_5	IDCliente: 4	Cantidades: 2	Estado: Cancelada	Precio: 300
-------------	-----------------------	--------------	---------------	-------------------	-------------

2.6.4 Validaciones

- Deberá contener 3 listas (compras pendientes, canceladas y aprobadas) y un botón que permita al administrador aprobar la compra.
- Para poder aprobar una compra, el saldo del usuario debe ser suficiente para efectuar la misma, así como también el stock. Además debe estar en estado “activo”.
- Para compras aprobadas validar el cambio de estado a “aprobada”, el descuento del stock correspondiente y la actualización del saldo del comprador.
- En los casos en los que, luego de aprobada la compra el stock quede en 0, se actualizará el estado a “pausado”.
- Si por no cumplirse alguna de las condiciones anteriores, la compra no se puede efectuar, la misma cambiará su estado a “cancelada”. Esto no alterará el stock ni el saldo del comprador.
- Refrescar la lista automáticamente luego de cada compra o cancelación.

2.7 F07– Creación de productos

2.7.1 Acceso

Puede acceder el usuario: Administrador

2.7.2 Descripción

Por medio de esta funcionalidad, el usuario administrador podrá crear nuevos productos detallando nombre del mismo, precio, descripción, URL de la imagen y cantidad de stock disponible.

2.7.3 Interfaz de usuario

BIENVENIDA: ADMIN LISTADO Y APROBACION CREACION DE PRODUCTOS

ADMINISTRAR PRODUCTOS INFORMES DE GANANCIA SALIR

Nombre del Producto	Precio	Descripción	Producto URL	Cantidad de stock Disponible
Nombre:	Precio:	Descripción:	FOTO Seleccionar archivo Sin archivos seleccionados	Cantidad de stock:

CREAR

2.7.4 Validaciones

- Validar que se ingresen los datos correspondientes en todos los campos, que el precio y el stock sean valores numéricos y mayores a 0.
- El producto creado obtendrá un ID y se encontrará en estado “activo” automáticamente.
- Los productos creados no pueden estar en oferta.

2.8F08– Administración de productos

2.8.1 Acceso







Puede acceder el usuario: Administrador

2.8.2 Descripción

El usuario contará con una vista para la administración de los productos creados, pudiendo: modificar el stock, modificar el estado, asignarlo como oferta.

2.8.3 Interfaz de usuario



ID	Producto IMG	Nombre	Stock	Actualizar Stock	Estado	Condición	
PROD_ID_1		Botella	30	<input type="text" value="1"/>	ACTIVO ...	PRECIO REGULAR ...	ENVIAR
PROD_ID_2		Inflador	35	<input type="text" value="1"/>	ACTIVO ...	OFERTA ...	ENVIAR
PROD_ID_3		Mesa de Juego	25	<input type="text" value="1"/>	INACTIVO ...	PRECIO REGULAR ...	ENVIAR
PROD_ID_4		Bolsa Grande	30	<input type="text" value="1"/>	ACTIVO ...	PRECIO REGULAR ...	ENVIAR
PROD_ID_5		Kit de Entrenamiento	35	<input type="text" value="1"/>	ACTIVO ...	OFERTA ...	ENVIAR
PROD_ID_6		Pack de Ciclismo	25	<input type="text" value="1"/>	INACTIVO ...	PRECIO REGULAR ...	ENVIAR

2.8.4 Validaciones

- Mientras el stock sea 0, se deberá pausar la publicación.

2.9F09– Visualización de informes de ganancia

2.9.1 Acceso

Puede acceder el usuario: Administrador

2.9.2 Descripción

Esta funcionalidad permite al administrador llevar un control de la cantidad total de unidades vendidas para cada producto, así como también la ganancia total.

2.9.3 Interfaz de usuario



Producto ID	Precio Unitario	Unidades Vendidas
PROD_ID_1	100	2
PROD_ID_2	450	4
PROD_ID_3	200	4

Ganancias Totales: \$2800

2.9.4 Validaciones

- Esta funcionalidad no requiere validación.

3. Datos precargados

DATOS PRECARGADOS - USUARIO ADMINISTRADOR		
ID	NOMBRE USUARIO	CONTRASEÑA
0	"admin"	"admin"
1	"a"	"a"
2	"admin1"	"admin1"
3	"admin2"	"admin2"
4	"admin3"	"admin3"

DATOS PRECARGADOS - USUARIO COMPRADOR					
ID	NOMBRE USUARIO	CONTRASEÑA	SALDO INICIAL	NRO. TARJETA	NRO. CVC
1	"Pablo"	"123456aS"	3000	"1111-1111-1111-1111"	123 
2	"Natalia"	"123456As"	3000	"1111-1111-1111-1111"	123
3	"Mauro"	"222222Fd"	3000	"1111-1111-1111-1111"	123
4	"Santiago"	"111111Gg"	3000	"1111-1111-1111-1111"	123
5	"Shirley"	"333333sH"	3000	"1111-1111-1111-1111"	123

DATOS PRECARGADOS - USUARIO ADMINISTRADOR		
ID	NOMBRE USUARIO	CONTRASEÑA
0	"admin"	"admin"
1	"a"	"a"
2	"admin1"	"admin1"
3	"admin2"	"admin2"
4	"admin3"	"admin3"

DATOS PRECARGADOS - PRODUCTOS							
ID	NOMBRE	FOTO	DESCRIPCIÓN	PRECIO	CONDICIÓN	STOCK	ESTADO
"PROD_ID_1"	"Botella"		Botella de plástico duro	100	Precio Regular	30	Activo
"PROD_ID_2"	"Inflador"		Un inflador de pelota con múltiples punteros	450	Oferta	35	Activo
"PROD_ID_3"	"Mesa de Juego"		Mesa de tejo para tu diversión	200	Precio Regular	25	Inactivo
"PROD_ID_4"	"Bolsa Grande"		Bolso de tela impermeable	100	Precio Regular	30	Activo
"PROD_ID_5"	"Kit de Entrenamiento"		Todo lo necesario para tu entrenamiento	300	Oferta	35	Activo

DATOS PRECARGADOS - MIS COMPRAS					
ID	ID_PROD	ID_CLIENTE	CANTIDAD	MONTO TOTAL	ESTADO COMPRA
1	"PROD_ID_1"	1	2	200	Aprobado
2	"PROD_ID_2"	2	2	900	Aprobado
3	"PROD_ID_3"	3	2	400	Aprobado
4	"PROD_ID_3"	3	2	400	Aprobado
5	"PROD_ID_2"	2	2	900	Aprobado
6	"PROD_ID_4"	5	2	200	Pendiente
7	"PROD_ID_5"	4	2	600	Cancelada

4. Código completo comentado

4.1 HTML

```
<!DOCTYPE html>
<html lang="en">

<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Shop E-Commerce</title>
  <script src="js/clases.js"></script>
  <script src="js/sistema.js"></script>
  <script src="js/codigo.js"></script>
  <link rel="stylesheet" href="css/style.css">
</head>

<body>

  <div id="contenedor">
    <header>
      
    <nav>
      <ul id="navPrincipal">
        <li id="aux">
          <a id="nombreUsuarioLogeado"></a>
        </li>
        <li id="btnSeccionProductos" class="btnSeccion user admin">
          <a>Productos</a>
        </li>
        <li id="btnSeccionOfertas" class="btnSeccion user">
          <a>Productos en Oferta</a>
        </li>
        <li id="btnSeccionSaldo" class="user">
          <a id="pvalorSaldo"></a>
        </li>
        <li id="btnSeccionCompras" class="btnSeccion user">
          <a>Mis compras</a>
        </li>
        <li id="btnSeccionListadoAprobacion" class="btnSeccion admin">
          <a>Listado y Aprobacion</a>
        </li>
      </ul>
    </nav>
  </div>
</body>
</html>
```

```

        <li id="btnSeccionCreacionProductoss" class="btnSeccion admin">
            <a>Creacion de Productos</a>
        </li>
        <li id="btnSeccionAdministrarProductos" class="btnSeccion
admin">
            <a>Administrar Productos</a>
        </li>
        <li id="btnSeccionInformesGanancia" class="btnSeccion admin">
            <a>Informes de Ganancia</a>
        </li>
        <li id="menuSalir">
            <a>Salir</a>
        </li>
    </ul>
</nav>

</header>
<!-- FIN CABEZAL -->
<main>

    <!-- SECCION PARA HACER EL LOGIN -->
    <section id="seccionLogin" class="seccion">
        <h2>Login</h2>

        <label for="slcTipoUsuario">Tipo de Usuario</label>
        <select id="slcTipoUsuario" required>
            <option value="1">Seleccione una opcion...</option>
            <option value="user">Comprador</option>
            <option value="admin">Administrador</option>
        </select><br>

        <label for="txtUsuario">Usuario</label>
        <input type="text" id="txtUsuario" required><br>

        <label for="txtClave">Contraseña</label>
        <input type="password" id="txtClave" required><br>

        <input type="button" value="Ingresar" id="btnLogin"><br>
        <input type="button" value="Registrar"
id="btnMostrarRegistro"><br>

    </section>

    <!-- SECCION PARA HACER EL REGISTRO -->
    <section id="seccionRegistro" class="seccion">
        <h2>Registro</h2>

```

```

<label for="txtUsuarioRegistro">Nombre de Usuario</label>
<input type="text" id="txtUsuarioRegistro" required><br>

<label for="txtContraseña">Contraseña (al menos 5 caracteres, una
mayuscula, una minuscula y un número)</label>
<input type="password" id="txtContraseña" required><br>

<label for="txtTarjetaCredito">Numero de tarjeta de
Credito</label>
<input type="tel" id="txtTarjetaCredito" name="tarjeta"
pattern="[0-9]{4}-[0-9]{4}-[0-9]{4}-[0-9]{4}"
placeholder="WWW-XXXX-YYYY-ZZZZ" required><br>

<label for="txtCVC">CVC</label>
<input type="tel" id="txtCVC" name="CVC" pattern="[0-9]{3}"
placeholder="123" required><br>

<input type="button" value="Registrarse" id="btnRegistrar"><br>
<input type="button" value="Volver al Login"
id="btnOcultarRegistrar"><br>
</section>

<!-- AGREGAR ITEMS PARA LA VENTA -->
<section id="seccionAgregar" class="seccion">

</section>

<!-- TABLA CON ITEMS DE VENTA ITEMS PARA LA VENTA -->
<section id="seccionProductos" class="seccion user">
<table border="1">
  <thead>
    <tr>
      <th>ID</th>
      <th>Nombre</th>
      <th>Producto IMG</th>
      <th>Descripción</th>
      <th>Precio</th>
      <th>Condición</th>
      <th>Cantidad de Unidades</th>
      <th></th>
    </tr>
  </thead>
  <!-- RELLENAR ID -->
  <tbody id="tblProductos"></tbody>
</table><br><br>

```

```

</section>

<section id="seccionCompras" class="seccion">
  <table border="1">
    <thead>
      <tr>
        <th>ID</th>
        <th>Nombre Producto</th>
        <th>Nombre Cliente</th>
        <th>Cantidad de Unidades</th>
        <th>Monto total</th>
        <th>Estado</th>
      </tr>
    </thead>
    <!-- RELLENAR ID -->
    <tbody id="tblMisCompras"></tbody>
  </table><br><br>
</section>

<section id="seccionOfertas" class="seccion">
  <table border="1">
    <thead>
      <tr>
        <th>ID</th>
        <th>Nombre</th>
        <th>Producto IMG</th>
        <th>Descripción</th>
        <th>Precio</th>
        <th>Condición</th>
        <th>Cantidad de Unidades</th>
        <th></th>
      </tr>
    </thead>
    <!-- RELLENAR ID -->
    <tbody id="tblOfertas"></tbody>
  </table><br><br>
</section>

<section id="seccionListadoAprobacion" class="seccion">
  <h1>Listado de Compras</h1>
  <table border="1">
    <thead>
      <tr>
        <th>Compras Aprobadas</th>
      </tr>
    </thead>

```

```

        <!-- RELLENAR ID -->
        <tbody id="tblListadoAprobadas">
        </tbody>
    </table><br><br><br>

    <table border="1">
        <thead>
            <tr>
                <th>Compras Pendientes</th>
            </tr>
        </thead>
        <!-- RELLENAR ID -->
        <tbody id="tblListadoPendientes">
        </tbody>
    </table><br><br><br>

    <table border="1">
        <thead>
            <tr>
                <th>Compras Canceladas</th>
            </tr>
        </thead>
        <!-- RELLENAR ID -->
        <tbody id="tblListadoCanceladas">
        </tbody>
    </table><br><br><br>
</section>

<section id="seccionAdministrarProductos" class="seccion">
    <table border="1">
        <thead>
            <tr>
                <th>ID</th>
                <th>Producto IMG</th>
                <th>Nombre</th>
                <th>Stock</th>
                <th>Actualizar Stock</th>
                <th>Estado</th>
                <th>Condición</th>
                <th></th>
            </tr>
        </thead>
        <!-- RELLENAR ID -->
        <tbody id="tblAdministrarProductos"></tbody>
    </table><br><br>

```

```

</section>

<section id="seccionCreacionProductoss" class="seccion">
  <table border="1">
    <thead>
      <tr>
        <th>Nombre del Producto</th>
        <th>Precio</th>
        <th>Descripcion</th>
        <th>Producto URL</th>
        <th>Cantidad de stock Disponible</th>
        <th></th>
      </tr>
    </thead>
    <!-- RELLENAR ID -->
    <tbody id="tblCreacionProductos"></tbody>
  </table><br><br>
</section>


<section id="seccionInformesGanancia" class="seccion">

  <h1>INFORMES DE GANANCIA</h1>
  <table border="1">
    <thead>
      <tr>
        <th>Producto ID</th>
        <th>Precio Unitario</th>
        <th>Unidades Vendidas</th>
      </tr>
    </thead>
    <tbody id="tablaInforme">
    </tbody>
  </table>

  <p id="gananciasTotales"></p>
</section>


<!-- BUSCAR ITEMS PARA LA VENTA -->
<section id="seccionBuscar" class="seccion">
  <label for="txtBusqueda">Busqueda:</label>
  <input type="text" id="txtBusqueda"><br>

  <input type="button" value="Buscar" id="btnBuscar">

  <p id="pResultado"></p>

```



```

</section>

<section id="seccionX" class="seccion">
  <p id="pResultado">X</p>
</section>

</main>
<!-- FIN CONTENIDO -->
<footer>
  <a href="https://www.ort.edu.uy">Footer</a>
</footer>
<!-- FIN PIE -->
</div>
<!-- FIN CONTENEDOR -->

</body>

</html>

```

4.2 CÓDIGO.JS:

```

window.addEventListener("load", inicio);

function inicio() {
  /**Cargamos todos los escuchadores de eventos */
  //Ejecucion de la funcion registrarUsuario al cliquear en el boton de
  Registro
  document.querySelector("#btnRegistrar").addEventListener("click",
  registrarUsuario);
  //Ejecucion de la funcion hacerLogin al cliquear en el boton de
  Logearse
  document.querySelector("#btnLogin").addEventListener("click",
  hacerLogin);
  //Ejecucion de la funcion mostrarTabla al cliquear en el boton de
  Productos en el nav
  document.querySelector("#btnLogin").addEventListener("click",
  mostrarTabla);
  //Ejecucion de la funcion salir al cliquear en el boton de salir en
  el nav
  document.querySelector("#menuSalir").addEventListener("click",
  salir);
}

```

```

    //Ejecucion de la funcion mostrarCompras al clicar en el boton Mis
    Compras en el nav
    document.querySelector("#btnSeccionCompras").addEventListener("click"
, mostrarCompras);
    //Ejecucion de la funcion mostrarOfertas al clicar en el boton
    Productos en Oferta en el nav
    document.querySelector("#btnSeccionOfertas").addEventListener("click"
, mostrarOfertas);
    //Ejecucion de la funcion mostrarAdminstracion al clicar en el
    boton Administrar Productos en Oferta en el nav COMO ADMIN
    document.querySelector("#btnSeccionAdministrarProductos").addEventLis
tener("click", mostrarAdminstracion);
    //Ejecucion de la funcion mostrarListado al clicar en el boton
    Listado y Aprobacion en el nav COMO ADMIN
    document.querySelector("#btnSeccionListadoAprobacion").addEventListen
er("click", mostrarListado);
    //Ejecucion de la funcion creacionProductos al clicar en el boton
    Creacion de Productos en el nav COMO ADMIN
    document.querySelector("#btnSeccionCreacionProductoss").addEventListe
ner("click", creacionProductos);
    //Ejecucion de la funcion generarInformeGanancias al clicar en el
    boton Informes de Ganancia en el nav COMO ADMIN
    document.querySelector("#btnSeccionInformesGanancia").addEventListener("click", generarInformeGanancias);
    //Ejecucion de la funcion mostrarSaldo al clicar en el boton
    Loguearse, guardando el saldo del usuarioLogueado
    document.querySelector("#btnLogin").addEventListener("click",
mostrarSaldo);

    /*Por defecto el programa arranca ocultando todas las secciones
    primero. */
    ocultarSecciones();

    /*Toma las clases de la tabla del NAV y al escuchar que se hace click
    en una de ellas muestra esa seccion. */
    let botones = document.querySelectorAll(".btnSeccion");
    for (let i = 0; i < botones.length; i++) {
        botones[i].addEventListener("click", mostrarSeccion);
    }

    /*Mostramos la seccion del login con id seccionLogin y ocultamos la
    barra de navegacion para impedir que vayan directo al carrito u otros
    menus*/
    document.querySelector("#seccionLogin").style.display = "block";
    document.querySelector("#navPrincipal").style.display = "none";

    /*Si se hace click en Registrarse, se muestra la seccion Registro. */

```

```

        document.querySelector("#btnMostrarRegistro").addEventListener("click",
        mostrarSeccionRegistro);
        /*Si se hace click en Vovler al Login, se oculta la seccion Registro.
        */
        document.querySelector("#btnOcultarRegistrar").addEventListener("click",
        ocultarSeccionRegistro);
        //Al cliquear el boton Productos se ejecuta el mostrarTabla para
        mostrar los productos.
        document.querySelector("#btnSeccionProductos").addEventListener("click",
        mostrarTabla);

    }
    let sistema = new Sistema();
    //-----FUNCIONES DE NAVEGACION-----
    -----

    /*Muestra la seccion de la tabla*/
    function mostrarSeccion() {
        //primero oculta todo.
        ocultarSecciones();

        let idBtn = this.getAttribute("id");//"btnSeccionAgregar"
        let idSeccion = idBtn.charAt(3).toLowerCase() +
        idBtn.substring(4);//seccionAgregar
        (idBtn, idSeccion);
        document.querySelector("#" + idSeccion).style.display = "block"
    }
    //Aqui mostramos la seccion del login y ocultamos la seccion de
    registrarse
    function mostrarSeccionRegistro() {
        document.querySelector("#seccionRegistro").style.display = "block"
        document.querySelector("#seccionLogin").style.display = "none";
    }

    //Aqui mostramos la seccion del login y ocultamos la seccion de
    registrarse
    function ocultarSeccionRegistro() {
        document.querySelector("#seccionRegistro").style.display = "none"
        document.querySelector("#seccionLogin").style.display = "block";
    }

    //Aqui ocultamos todas las secciones que tengan la clase seccion
    function ocultarSecciones() {
        let secciones = document.querySelectorAll(".seccion");
        for (let i = 0; i < secciones.length; i++) {

```

```

        secciones[i].style.display = "none"
    }
}

//mostrarBotones acepta un tipo de usuario, oculta todos los botones y
muestra solo los que tienen la clase user o admin.
function mostrarBotones(tipo) {
    ocultarBotones();

    let botonesMostrar = document.querySelectorAll("." + tipo);
    for (let i = 0; i < botonesMostrar.length; i++) {
        botonesMostrar[i].style.display = "block";
    }
}

//ocultarBotones oculta todas las clases que tengan como valor btnSeccion
function ocultarBotones() {
    let botonesOcultar = document.querySelectorAll(".btnSeccion");
    for (let i = 0; i < botonesOcultar.length; i++) {
        botonesOcultar[i].style.display = "none";
    }
}

//----- FUNCIONES DE ACCESO Y REGISTRO -----
//-----

/*REGISTRAR USUARIO*/
let idUsuario = 5;
//Creamos la funcion registrar un usuario, comenzamos del 108-119 creando
variables para guardar cada valor de cada campo.
function registrarUsuario() {
    let nombre = document.querySelector("#txtUsuarioRegistro").value;
    let clave = document.querySelector("#txtContraseña").value;
    let numeroTarjeta =
document.querySelector("#txtTarjetaCredito").value;
    let numeroCVC = document.querySelector("#txtCVC").value;

    let camposCompletos = sistema.validarCamposVaciosRegistro(nombre,
clave);
    let formatoContrasenaValido =
sistema.verificarFormatoContrasena(clave);
    let formatoTarjetaValido =
sistema.verificarTarjetaCredito(numeroTarjeta);
    let formatoCVCValido = sistema.verificarCVC(numeroCVC);
    let existeUsuario = sistema.buscarNombre(sistema.usuarios, "nombre",
nombre);

```

```

    let verificarTajeta =
sistema.verificarTarjetaCreditoLuhn(numeroTarjeta);

    //Comprobamos que todos los campos tengan sus valores validos
    if (camposCompleto === true && formatoContraseñaValido === true &&
existeUsuario === false && formatoTarjetaValido === true &&
formatoCVCValido === true && verificarTajeta === true) {
        let usuario = new Usuario(idUsuario, nombre, clave, 3000,
numeroTarjeta, numeroCVC);
        //al crear un nuevo usuario incrementa el numero del id
        idUsuario++;
        //sistema se encarga mediante su funcion agregarUsuario de
agregarlo efectivamente a la lista de existentes.
        sistema.agregarUsuario(usuario);
        alert("Registro exitoso!")
    } else {
        alert("Error en Registro: Los campos son todos obligatorios, la
pass es minimo de 5 caracteres o ya existe un usuario con ese nombre");
    }
    //limpiamos los valores de los campos una vez que se haya ejecutado
la funcion registrarUsuario.
    document.querySelector("#txtUsuarioRegistro").value = "";
    document.querySelector("#txtContraseña").value = "";
}

//creamos variables globales usuarioLogeado, con propiedades null o vacio
por defecto.
let usuarioLogeado = null;
let usuarioLogeadoAdmin = null;
let tipoUsuario = "";

//Aqui se implementa el login, se extraen los valores de cada campo.
function hacerLogin() {
    let nombre = document.querySelector("#txtUsuario").value;
    let clave = document.querySelector("#txtClave").value;
    tipoUsuario = document.querySelector("#slcTipoUsuario").value;

    //Se verifica mediante la funcion en sistema verificarLogin que
exista ese usuario, y que los campos esten correctamente colocados.
    let login = sistema.verificarLogin(nombre, clave, tipoUsuario);

    //si retorna true el login entonces se guarda el usuarioLogeado en
esa variable para luego acceder a sus propiedades.
    if (login) {
        usuarioLogeado = sistema.obtenerObjeto(sistema.usuarios,
"nombre", nombre);
    }
}

```

```

        //Si el usuarioLogeado es un admin entonces se guardara en otra
variable distinta.
        usuarioLogeadoAdmin =
sistema.obtenerObjeto(sistema.usuariosAdmin, "nombre", nombre);
        //Aqui Ocultamos los campos de login y registro.
        mostrarMenuOcultandoLoginYRegistro();

        //Si el tipoUsuario es admin entonces ocultamos lo que no debe de
ver el admin.
        if (tipoUsuario === "admin") {
            document.querySelector("#btnSeccionProductos").style.display
= "none";
            document.querySelector("#btnSeccionOfertas").style.display =
"none";
            document.querySelector("#btnSeccionSaldo").style.display =
"none";
            document.querySelector("#btnSeccionCompras").style.display =
"none";
        }

        } else {
            alert("Usuario y/o contraseña incorrectos");
        }

        /**Una vez logeado, restaura los datos a vacios*/
        document.querySelector("#txtUsuario").value = "";
        document.querySelector("#txtClave").value = "";
    }
//En esta funcion, se encarga de ocultar los campos relacionados al login
y al registro.
function mostrarMenuOcultandoLoginYRegistro() {
    mostrarBotones(tipoUsuario);
    document.querySelector("#seccionLogin").style.display = "none";
    document.querySelector("#seccionRegistro").style.display = "none";
    document.querySelector("#navPrincipal").style.display = "block";
    /*Vuelve a mostrar el texto, sino al apretar salir, queda oculto
cuando se reingresa */
    document.querySelector("#aux").style.display = "block";

    //validamos que valor tiene el tipo de usuario para saber que mensaje
dar de bienvenida.
    if (tipoUsuario === "user") {
        //Para imprimir el nombre, accedemos al usuarioLogeado o
usuarioLogeadoAdmin a su propiedad nombre.
        document.querySelector("#nombreUsuarioLogeado").innerHTML =
`Bienvenido/a: ${usuarioLogeado.nombre}`
    }
}

```

```

        if (tipoUsuario === "admin") {
            document.querySelector("#nombreUsuarioLogeado").innerHTML =
`Bienvenido/a: ${usuarioLogeadoAdmin.nombre}`
        }
    }

    //Esta funcion, me permite acceder al saldo del usuarioLogeado e
    independizar el saldo de cada usuario distinto.
    function mostrarSaldo() {
        if (tipoUsuario === "user") {

            document.querySelector("#pvalorSaldo").innerHTML =
usuarioLogeado.saldo;
        }
    }

    //-----FUNCIONALIDADES ADMINISTRADOR -----
    //En esta funcion, empezamos a crear la tabla dinamica para admin.
    function mostrarAdminstracion() {
        //Primero creamos la variable tipoUsuario para almacenar que es, si
        user o admin.
        let tipoUsuario = document.querySelector("#slcTipoUsuario").value
        //si efectivamente es admin entonces procedemos a hacer un for para
        crear celda a celda la tabla en el body seleccionado mediante su id.
        if (tipoUsuario === "admin") {
            //Aca limpiamos la tabla para que los valores no se repitan.
            document.querySelector("#tblAdministrarProductos").innerHTML =
"";
            for (let i = 0; i < sistema.productos.length; i++) {
                const unProducto = sistema.productos[i];
                document.querySelector("#tblAdministrarProductos").innerHTML
+= `
                <tr>
                    <td>
                        ${unProducto.id}
                    </td>

                    <td>
                        
                    </td>

                    <td>
                        ${unProducto.nombre}
                    </td>

                    <td id="stockEditado">
                        ${unProducto.stock}

```

```

        </td>

        <td>
            <input type="number"
id="cantidadUnidades${unProducto.id}" value="1">
        </td>

        <td>
            <label
for="slcEstado${unProducto.id}">${unProducto.estado}</label>
            <select id="slcEstado${unProducto.id}">
                <option value="0">...</option>
                <option value="Activo">Activo</option>
                <option value="Inactivo">Inactivo</option>
            </select>
        </td>

        <td>
            <label
for="slcCondicion${unProducto.id}">${unProducto.condicion}</label>
            <select id="slcCondicion${unProducto.id}">
                <option value="0">...</option>
                <option value="Oferta">Oferta</option>
                <option value="Precio regular">Regular</option>
            </select>
        </td>

        <td>
            <input type="button" value="Enviar" id="btnEnviar${i}"
class="claseBoton" data-id="${unProducto.id}">
        </td>
    </tr>`;
    }

    //creamos una funcionalidad para obtener todos los botones que
    tengan como clase claseBoton.
    let botones = document.querySelectorAll(".claseBoton");
    //los recorremos y en el que se escuche que se hizo click se
    ejecuta la funcion modificarProducto, mas adelante.
    for (let i = 0; i < botones.length; i++) {
        botones[i].addEventListener("click", modificarProducto);
    }
}

//modificar producto lo que hará sera...
function modificarProducto() {

```



```

    //primero, obtener el producto del id que se escuchó que se hizo
    click,
    let idProducto = this.getAttribute("data-id");//ej;
    "btnSeccionAgregar"
    //con ese idProducto podemos buscarlo mediante la funcion de sistema
    obtenerObjeto, que requiere 3 parametros un array, un nombre propiedad y
    un valor de busqueda.
    let objProducto = sistema.obtenerObjeto(sistema.productos, "id",
    idProducto);
    //almaceno todos los datos que me sirven para luego modificar mi
    producto, estos datos son obtenidos del html.
    let estadoEditado = document.querySelector("#slcEstado" +
    idProducto).value;//Activo
    let cantidadEditada =
    Number(document.querySelector("#cantidadUnidades" +
    idProducto).value);//5
    let condicionEditada = document.querySelector("#slcCondicion" +
    idProducto).value;

    //Si la condicion es oferta entonces se quedara su valor con el
    string Oferta, de lo contrario si tiene Regular será regular.
    if (condicionEditada === "Oferta") {
        condicionEditada = "Oferta"
    } else if (condicionEditada === "Regular") {
        condicionEditada = "Regular"
    }

    //Esta es la parte donde al objeto le decimos que sus propeidades
    ahora serán las que se extrajeron del html.
    objProducto.estado = estadoEditado;
    objProducto.stock = cantidadEditada;
    objProducto.condicion = condicionEditada;
    //por ultimo vamos a recargar la tabla con mostrarAdministracion
    mostrarAdminstracion()
}

function mostrarListado() {
    //primero vemos que tipo de usuario es
    let tipoUsuario = document.querySelector("#slcTipoUsuario").value
    //si es admin, entonces ejecutamos la sentencia if, limpiando todas
    las tablas para que no se creen muchas por cada click en id
    #btnSeccionListadoAprobacion (línea: 20)
    if (tipoUsuario === "admin") {
        document.querySelector("#tblListadoAprobadas").innerHTML = "";
        document.querySelector("#tblListadoPendientes").innerHTML = "";
        document.querySelector("#tblListadoCanceladas").innerHTML = "";
    }
}

```

```

        //Procedemos a cargar las tablas mediante sus respectivas
funciones explicadas mas adelante.
        mostrarAprobadas();
        mostrarPendientes();
        mostrarCanceladas();
    }
}

function mostrarAprobadas() {
    //Buscamos todos los objetos que tengan estado aprobado
    let objProducto = sistema.obtenerObjetoEnArray(sistema.misCompras,
"estadoCompra", "Aprobado");
    //limpiamos tabal para que no se creen uno atras de otro y no quede
larguisimo.
    document.querySelector("#tblListadoAprobadas").innerHTML = "";
    //este for recorre la totalidad de objProcuto encontrados con
estadoCompra Aprobado.
    for (let i = 0; i < objProducto.length; i++) {
        //Esta variable inmutable const, nos permite recorrer uno a uno
sus propiedades para poder imprimirlas en la tabla dinamica, aunque se
peude hacer directo del objProducto
        const unProducto = objProducto[i];
        document.querySelector("#tblListadoAprobadas").innerHTML += `

        <tr><td><strong>IDCompra:</strong> <br>
${unProducto.id}<br><br></td>
        <td><strong>IDProducto:</strong>
<br> ${unProducto.idProductoCompra}<br><br></td>
        <td><strong>IDCliente:</strong>
<br> ${unProducto.idCliente}<br><br></td>
        <td><strong>Cantidades:</strong>
<br> ${unProducto.cantidad}<br><br></td>
        <td><strong>Estado:</strong> <br>
${unProducto.estadoCompra}<br><br></td>
        <td><strong>Precio:</strong>
<br> ${unProducto.monto}<br><br></td> </tr>`
    }
}

let objProductoPendiente = []

function mostrarPendientes() {
    //Aqui actualizamos el objeto con todos los que encuentre con estado
Pendiente.
    objProductoPendiente =
sistema.obtenerObjetoEnArray(sistema.misCompras, "estadoCompra",
"Pendiente");

```

```

    //limpiamos tabal para que no se creen uno atras de otro y no quede
    larguísimo.
    document.querySelector("#tblListadoPendientes").innerHTML = "";
    //este for recorre la totalidad de objProcuto encontrados con
    estadoCompra Pendiente.
    for (let i = 0; i < objProductoPendiente.length; i++) {
        //En esta oportunidad recorrimos directamente el
        objProductoPendiente pero podriamos haber heho un const unProducto como
        el anterior
        document.querySelector("#tblListadoPendientes").innerHTML += `

            <tr><td><strong>IDCompra:</strong> <br>
            ${objProductoPendiente[i].id}<br><br></td>
            <td><strong>IDProducto:</strong>
            <br> ${objProductoPendiente[i].idProductoCompra}<br><br></td>
            <td><strong>IDCliente:</strong>
            <br> ${objProductoPendiente[i].idCliente}<br><br></td>
            <td><strong>Cantidades:</strong>
            <br> ${objProductoPendiente[i].cantidad}<br><br></td>
            <td><strong>Estado:</strong> <br>
            ${objProductoPendiente[i].estadoCompra}<br><br></td>
            <td><strong>Precio:</strong>
            <br> ${objProductoPendiente[i].monto}<br><br></td>
            <td> <input type="button" value="APROBAR" id="btnAprobacion${i}"
            class="btnAprobarCompras" data-id="${objProductoPendiente[i].id}">
            </td></tr>`
        }

        //esta funcionalidad almacena todos las clases con valor
        btnAprobarCompras
        let btnAprobacion = document.querySelectorAll(".btnAprobarCompras");
        //Una vez que las recorra y escuche algun click en algunas de ellas
        ejecuta aprobarCompras funcion.
        for (let i = 0; i < btnAprobacion.length; i++) {
            btnAprobacion[i].addEventListener("click", aprobarCompra);
        }
    }

function mostrarCanceladas() {
    //Buscamos todos los objetos que tengan estado aprobado
    let objProducto = sistema.obtenerObjetoEnArray(sistema.misCompras,
    "estadoCompra", "Cancelada");
    //limpiamos tabal para que no se creen uno atras de otro y no quede
    larguísimo.
    document.querySelector("#tblListadoCanceladas").innerHTML = "";
    //este for recorre la totalidad de objProcuto encontrados con
    estadoCompra Aprobado.

```

```

    for (let i = 0; i < objProducto.length; i++) {
        //Esta variable inmutable const, nos permite recorrer uno a uno
        sus propiedades para poder imprimirlas en la tabla dinamica, aunque se
        puede hacer directo del objProducto
        const unProducto = objProducto[i];
        document.querySelector("#tblListadoCanceladas").innerHTML += `

            <tr>
                <td><strong>IDCompra:</strong> <br> ${unProducto.id}<br><br></td>
                <td><strong>IDProducto:</strong>
            <br> ${unProducto.idProductoCompra}<br><br></td>
                <td><strong>IDCliente:</strong>
            <br> ${unProducto.idCliente}<br><br></td>
                <td><strong>Cantidades:</strong>
            <br> ${unProducto.cantidad}<br><br></td>
                <td><strong>Estado:</strong> <br>
            ${unProducto.estadoCompra}<br><br></td>
                <td><strong>Precio:</strong>
            <br> ${unProducto.monto}<br><br></td>
            </tr>`
        }
    }

    function aprobarCompra() {
        //Esta funcion lo que nos hace es obtener primero el id de compra
        para poder encontrar el objeto del cual se escucho el click y tomar sus
        propiedades.

        let idCompra = Number(this.getAttribute("data-id")); //me devuelve 1
        //buscamos el objeto
        let compraParaAprobar = sistema.obtenerObjeto(objProductoPendiente,
        "id", idCompra);
        //Empezamos a obtener las propiedades idProductoCompra y idCliente
        del objeto obtenido.
        let traerIdProducto = compraParaAprobar.idProductoCompra;
        let traerIdCliente = compraParaAprobar.idCliente;
        //Ahora buscamos un objeto que tenga el idProductoCompra para obtener
        sus propiedades
        let devolverInfoProducto = sistema.obtenerObjeto(sistema.productos,
        "id", traerIdProducto);
        //Comenzamos a almacenar el valor de las propiedades obtenidos del
        objeto, en distintas variables.
        let traerStockProducto = devolverInfoProducto.stock;
        let traerEstadoProducto = devolverInfoProducto.estado;
        let traerPrecioProducto = devolverInfoProducto.precio;
        //Aquí hacemos la cuenta del montoTotal considerando las diferentes
        propiedades de los distintos objetos.
        let montoTotal = traerPrecioProducto * compraParaAprobar.cantidad;
    }

```

```

    //Vamos a buscar ahora un objeto que tenga el id del cliente, para
    modificar su saldo luego de aprobada la compra
    let devolverInfoUsuario = sistema.obtenerObjeto(sistema.usuarios,
    "id", traerIdCliente);
    //almacenamos en una variable el saldo obtenido del objeto.
    let traerSaldoCliente = devolverInfoUsuario.saldo;

    //seguimos con una verificacion de cada propiedad obtenida para ver
    si cumple con los requisitos, en base a eso, es si se actualizara el
    estado a Aprobado o Cancelada.
    if (traerSaldoCliente >= montoTotal && traerStockProducto >=
    compraParaAprobar.cantidad && traerEstadoProducto === "Activo") {

        //Si todo se cumple el estado pasa a ser Aprobado, aqui lo
        actualizamos
        compraParaAprobar.estadoCompra = "Aprobado";

        //guardamos los datos del cliente actualizados en esta cuenta.
        let actualizarSaldoCliente = traerSaldoCliente - montoTotal;
        //Aca actualizamos la propiedad del saldo del cliente,
        actualizando su saldo como le queda al final.
        devolverInfoUsuario.saldo = actualizarSaldoCliente

        //aqui almacenamos el stock del producto, en base al resultado de
        la cuenta del stock del producto menos la cantidad que se aprobo.
        let actualizarStock = traerStockProducto -
        compraParaAprobar.cantidad;
        //aca lo actualizamos.
        devolverInfoProducto.stock = actualizarStock;

        //Aqui comprobamos la condicion del stock, si el stock esta en 0,
        entonces el estado del producto será inactivo, ya que no queda stock.
        if (actualizarStock === 0) {
            devolverInfoProducto.estado = "Inactivo";
        }
        //luego recargamos la tabla
        mostrarListado();

    } else {
        //Si nada se cumple, el estado pasa a ser cancelado con un avio
        de uno de los motivos.
        compraParaAprobar.estadoCompra = "Cancelada";
        alert("Saldo insuficiente o Stock insuficiente o producto
        Inactivo")

        //recargamos la tabla.
        mostrarListado();
    }

```

```

    }
}

//En esta funcion implementamos la creacion de productos
function creacionProductos() {
    //obtenemos el tipo de usuario y lo almacenamos
    let tipoUsuario = document.querySelector("#slcTipoUsuario").value
    //si es admin, entonces puede crear un producto
    if (tipoUsuario === "admin") {
        //limpiamos la tabla y creamos las celdas correspondientes.
        document.querySelector("#tblCreacionProductos").innerHTML = "";
        document.querySelector("#tblCreacionProductos").innerHTML += `

                <td><input type="text" id="crearNombre"
placeholder="Nombre:"><br></td>
                <td><input type="text" id="creaPrecio"
placeholder="Precio:"></td>
                <td><input type="text" id="creaDescripcion"
placeholder="Descripcion:"></td>
                <td><label for="imgProducto">Foto</label>
                <input type="file" id="imgProducto"></td>
                <td><input type="text" id="crearStock" placeholder="Cantidad
de stock: "></td>

                <tr>
                <td><input type="button" id="crearProductoNuevo"
value="Crear"></td>
                </tr>`
    }
    //Cuando se escuche click en el boton id crearProducto nuevo,
    entonces se ejecuta agregarProducto
    document.querySelector("#crearProductoNuevo").addEventListener("click
", agregarProducto);
}
//En esta linea, designamos que el idProducto final, es el 10, asi
empieza a jugar con este valor e incrementarlo.
let idProducto = "PROD_ID_10";

//Esta funcion se encarga de incrementar el id
function aumentarIdProducto(string) {
    //obtiene el ultimo objeto de productos, especificamente su id.
    const valorPropiedad = sistema.productos[sistema.productos.length -
1].id; //PROD_ID_3
    //guarda el numero
    let numero =
Number(valorPropiedad.substring(valorPropiedad.lastIndexOf("_") + 1))/3

```

```

    //le aumenta 1
    numero++//4
    let texto = valorPropiedad.substring(0, 8);//PROD_ID_
    return texto + numero//PROD_ID_4
}

//Esta funcion, me permite aumentar el numero del idCompra.
function aumentarIdCompra(numero) {//1
    numero++//2
    return numero//2
}

//Esta funcion es para agregar un producto
function agregarProducto() {
    //vamos a guardar en variables la informacion obtenida del html.
    let crearNombre = document.querySelector("#crearNombre").value;
    let creaPrecio = Number(document.querySelector("#creaPrecio").value);
    let creaDescripcion =
document.querySelector("#creaDescripcion").value;
    let crearImagen = document.querySelector("#imgProducto").value;
    let nombreImagen =
crearImagen.substring(crearImagen.lastIndexOf("\\") + 1);
    let crearStock = Number(document.querySelector("#crearStock").value);
    let unaCantidad = 1;

    //Si cumple los requisitos entonces pasa a la siguiente validacion
    if (crearNombre && creaPrecio && creaDescripcion && nombreImagen &&
crearStock !== "") {
        //Si todo cumple con los tipos de datos y con los valores que
debe tener cada campo, una condicion pasa a ser Precio Regular y unEstado
a Activo como valores por default
        if (!isNaN(creaPrecio) && !isNaN(crearStock) && creaPrecio > 0 &&
crearStock > 0) {

            unaCondicion = "Precio Regular";
            unEstado = "Activo";
            //Ahora, creamos un nuevo objeto, una nueva instancia de la
clase Producto me permite pasar todas estas variables como propiedades
del objeto.
            let producto = new Producto(aumentarIdProducto(idProducto),
crearNombre, nombreImagen, creaDescripcion, creaPrecio, unaCondicion,
crearStock, unEstado, unaCantidad);
            //sistema se encargara mediante su funcion agregarProducto de
pushear el producto guardado en la variable producto
            sistema.agregarProducto(producto);
            alert("Registro exitoso!")
        } else {

```

```

        alert("Error: El precio y el stock deben ser valores
numéricos y mayores a 0");
    }
    } else {
        alert("Error: Todos los campos son obligatorio");
    }
    //luego de registrar un producto todos los campos quedaran vacios
    //prontos para volver a colcoar otro producto.
    document.querySelector("#crearNombre").value = "";
    document.querySelector("#creaPrecio").value = "";
    document.querySelector("#creaDescripcion").value = "";
    document.querySelector("#imgProducto").value = "";
    document.querySelector("#crearStock").value = "";
}

function generarInformeGanancias() {
    //Primero limpiamos la tabla para que cada vez que hagamos click no
    //se multiplique la informacion una abajo de otra.
    document.querySelector("#tablaInforme").innerHTML = "";
    //recorremos con un for of del array productos donde cada producto lo
    //almacenamos en la variable constante; producto
    for (const producto of sistema.productos) {
        //luego imprimimos la tabla, donde en la primer celda va el
        //nombre del producto, en la segunda va a ir el precio unitario y en la
        //ultima el acumulativo de cantidad.
        document.querySelector("#tablaInforme").innerHTML +=`<tr>
<td>${producto.nombre}</td>
<td>${producto.precio}</td>
<td>${sistema.obtenerCantidadProductos(producto.id)}</td>
</tr>`

    }
    //Por ultimo generamos la variable ganancias, que la funcion dentro
    //de sistemas es responsable de hacer esta operacion y se guardara su
    //resultado en ganancias.
    let ganancias = sistema.obtenerGananciasTotales();
    //Luego seleccionamos el elemento html que queremos que se muestre la
    //ganancia.
    document.querySelector("#gananciasTotales").innerHTML = ganancias;
}

//-----FUNCIONALIDADES USUARIO-----
//Al hacer click en Productos o al loguearse, vamos a ejecutar esta
//funcion.
function mostrarTabla() {

```



```

    //Cuando se ejecuta la funcion mostrarTabla, lo primero que hacemos
    es limpiar la tabla
    document.querySelector("#tblOfertas").innerHTML = "";
    //Luego obtenemos el valor del tipoUsuario
    let tipoUsuario = document.querySelector("#slcTipoUsuario").value
    //si es user, entonces esta habilitado a ver la tabla.
    if (tipoUsuario === "user") {
        //antes que nada, limpiamos la tabla productos
        document.querySelector("#tblProductos").innerHTML = "";
        //con un for, cargaremos dinamicamente la tabla
        for (let i = 0; i < sistema.productos.length; i++) {
            //Vamos almacenando los objetos recorridos en la variable
            const unProducto asi accedemos a sus propiedades mas adelante en la
            tabla..

            const unProducto = sistema.productos[i];
            document.querySelector("#tblProductos").innerHTML += `
            <tr>
                <td>${unProducto.id}</td>
                <td>${unProducto.nombre}</td>
                <td></td>
                <td>${unProducto.descripcion}</td>
                <td>${unProducto.precio}</td>
                <td>${unProducto.condicion}</td>
                <td><input type="number"
id="CantidadUnidades${unProducto.id}" value="1"></td>
                <td><input type="button" value="Comprar"
class="btnComprar${i} btn" data-id="${unProducto.id}" disabled></td>
            </tr> `;

            //Si el estado del producto es inactivo, no sucede nada, pero
            si es activo, se le remueve el atributo disabled para que se pueda
            comprar.

            if (unProducto.estado === "Inactivo") {
            } else if (unProducto.estado === "Activo") {
                document.querySelector(`.btnComprar${i}
`).removeAttribute(`disabled`, `disabled`);
            }
            //Luego almacenamos todos los botones con clase .btn, en la
            variable btn.
            let btn = document.querySelectorAll(".btn")

            //los recorremos y si vemos que se hace click en uno de ellos
            lanzamos a la funcionalidad de btnComprar
            for (let i = 0; i < btn.length; i++) {
                btn[i].addEventListener("click", btnComprar);
            }

```

```

        //almacenamos todos los botones btnActualizarCantidad en la
variable btnCantidad
        let btnCantidad =
document.querySelectorAll(".btnActualizarCantidad")
        //ahora podemos recorrerlos y ver en cual se hace click,
luego ejecutamos actualizarCantidad.
        for (let i = 0; i < btnCantidad.length; i++) {
            btnCantidad[i].addEventListener("click",
actualizarCantidad);
        }
    }
}
}

function mostrarCompras() {
    //Obtenemos el tipo de usuario y lo almacenamos
    let tipoUsuario = document.querySelector("#slcTipoUsuario").value
    if (tipoUsuario === "user") {
        document.querySelector("#tblMisCompras").innerHTML = "";
        //recorremos el array: sistema.miscompras, colando a [i] como
objeto, para obtener las diferentes propiedades allí detalladas, que se
mostrarán en tabla
        for (let i = 0; i < sistema.misCompras.length; i++) {
            const unaCompra = sistema.misCompras[i];
            document.querySelector("#tblMisCompras").innerHTML += `
<tr>
    <td>${unaCompra.id}</td>
    <td>${unaCompra.idProductoCompra}</td>
    <td>${unaCompra.idCliente}</td>
    <td>${unaCompra.cantidad}</td>
    <td>${unaCompra.monto}</td>
    <td>${unaCompra.estadoCompra}</td>
    <td><input type="button" value="Cancelar"
class="btnMisCompras${i} btn" data-id="${unaCompra.id}"></td>
</tr> `;

            //Aquí deshabilitamos la posibilidad de cancelar mediante
boton a todas aquellas compras que ya esten aprobadas o sean canceladas
            if (unaCompra.estadoCompra === "Aprobado" ||
unaCompra.estadoCompra === "Cancelada") {
                document.querySelector(`.btnMisCompras${i}`).setAttribute
("disabled", "disabled");
            }
        }

        //Aqui recorremos todos los botones en busca de aquel en el que
se realiza el "click"

```

```

        let btn = document.querySelectorAll(".btn")

        for (let i = 0; i < btn.length; i++) {
            btn[i].addEventListener("click", modificarCompra);
        }
    }
}

function modificarCompra() {
    /*En esta caso guardamos el id de compra del id sobre el que se hace
    click en "dataCompra", con la finalidad de poder buscarlo en "misCompras"
    (a través de sistema.obtenerObjeto, con sus parámetros correspondientes).
    Una vez que se obtiene este dato, se cambia su estado a "cancelada".
    Con esta funcionalidad se logra que el usuario comprador cancele una
    compra que se encuentra Pendiente.
    */
    let dataCompra = Number(this.getAttribute("data-id"));
    let objCompra = sistema.obtenerObjeto(sistema.misCompras, "id",
dataCompra);

    objCompra.estadoCompra = "Cancelada";

    mostrarCompras()
}

function mostrarOfertas() {
    //Esta funcionalidad nos permite ver la tabla de Mis compras, primero
    limpiamos la tabla, asi no se crea una tabla larguísima.
    document.querySelector("#tblProductos").innerHTML = "";
    //se obtiene el resultado del tipo de usuario, si es user entonces
    puede continuar.
    let tipoUsuario = document.querySelector("#slcTipoUsuario").value
    if (tipoUsuario === "user") {
        //limpiamos la tabla ofertas
        document.querySelector("#tblOfertas").innerHTML = "";
        //almacenamos el array de productos en unProducto
        const unProducto = sistema.productos

        /*arrEncontrados contiene un array de objetos que contiene el
        valor de la propiedad buscada*/
        let arrEncontrados = sistema.obtenerObjetoEnArray(unProducto,
"condicion", "Oferta")

        //Los vamos recorriendo uno a uno y vamos imprimiendo sus
        propiedades
    }
}

```

```

        for (let i = 0; i < arrEncontrados.length; i++) {
            document.querySelector("#tblOfertas").innerHTML += `
                <tr>
                    <td>${arrEncontrados[i].id}</td>
                    <td>${arrEncontrados[i].nombre}</td>
                    <td></td>
                    <td>${arrEncontrados[i].descripcion}</td>
                    <td>${arrEncontrados[i].precio}</td>
                    <td>${arrEncontrados[i].condicion}</td>
                    <td><input type="number"
id="CantidadUnidades${arrEncontrados[i].id}" value="1"></td>
                    <td><input type="button" value="Comprar"
class="btnComprarOferta${i}" btn" data-id="${arrEncontrados[i].id}"></td>
                </tr> `;

            //Si el estado del producto recorrido NO es activo, entonces
            vamos a setearle el atributo disabled.
            if (unProducto[i].estado === "Activo") {
            } else {
                document.querySelector(`.btnComprarOferta${i}`)
                .setAttribute("disabled", "disabled");
            }
        }

        //en estas lineas obtenemos todas las clases .btn y las
        almacenamos en btn
        let btn = document.querySelectorAll(".btn")
        //luego las recorremos para ver cual se hizo click, asi
        ejecutamos btnComprar
        for (let i = 0; i < btn.length; i++) {
            btn[i].addEventListener("click", btnComprar);
        }

        //aca almacenamos todas las clases .btnActualizarCantidad en la
        variable btnCantidad
        let btnCantidad =
        document.querySelectorAll(".btnActualizarCantidad")
        //Ahora las recorremos y en la que se haga click, le ejecutamos
        la funcion actualizarCantidad
        for (let i = 0; i < btnCantidad.length; i++) {
            btnCantidad[i].addEventListener("click", actualizarCantidad);
        }
    }
}

function actualizarCantidad() {
    //Esta funcion se inicia una vez que se haga click en el boton
    .btnActualizarCantidad

```

```

    //Obtenemos el idProducto del data-id, para buscar el objeto que fue
    cliqueado
    let idProducto = this.getAttribute("data-id");
    //buscamos ese objeto y lo almacenamos en objProducto
    let objProducto = sistema.obtenerObjeto(sistema.productos, "id",
idProducto);
    //vamos a castear a number el valor del html que se haya escrito en
    cantidad.
    let cantidadNueva = Number(document.querySelector("#CantidadUnidades"
+ idProducto).value);//2
    objProducto.cantidad += cantidadNueva;
    //luego ahi, actualizamos la cantidad por la que se escribio en el
    html

    //mostramos las tablas.
    mostrarOfertas()
    mostrarTabla()
}

function btnComprar() {
    //Cuando se hace click en Comprar se guarda el id del producto al
    cual se le hizo el click
    let idProducto = this.getAttribute("data-id");//idProducto
    //se busca ese producto por su id.
    let objProducto = sistema.obtenerObjeto(sistema.productos, "id",
idProducto);
    //se guardan todas las variables cada valor de las propiedades que
    nos interesan
    let id = sistema.misCompras[sistema.misCompras.length - 1].id
    let idProductoCompra = objProducto.id
    let idCliente = usuarioLogeado.id;
    let precio = objProducto.precio;
    //por defecto el estadoCompra es Pendiente.
    let estadoCompra = "Pendiente";
    //la cantidad se extrae del HTML del valor que haya colocado el
    usuario.
    let cantidad = Number(document.querySelector("#CantidadUnidades" +
idProducto).value);

    //Si se cumple los requisitos necesarios entonces se procede a crear
    una nueva instancia del objeto ListadoCompras
    if (!isNaN(id) && isNaN(idProductoCompra) && !isNaN(idCliente) &&
cantidad > 0 && precio > 0 && estadoCompra !== "") {
        //Creando la instancia del objeto ListadoCompras, le pasamos
        todas las variables que creamos a paritr de la linea 736

```

```

        let nuevaCompra = new ListadoCompras(aumentarIdCompra(id),
idProductoCompra, idCliente, cantidad, precio, estadoCompra);
        //esta parte, se encarga de agregar efectivamente el ibjeto nuevo
con sus propiedades a la lista de sistemas.misCompras, se encarga de
hacerlo la funcion agregarcompra en sistema
        sistema.agregarCompra(nuevaCompra);
        alert("Compra exitosa!")
    } else {
        alert("Error: El precio y el stock deben ser valores numéricos y
mayores a 0");
    }
}

function agregarCompra() {

    let id = sistema.misCompras[sistema.misCompras.length - 1].id

    //se extrae del producto que se hizo click.
    let idProductoCompra = sistema.productos;

    let idCliente = usuarioLogeado.id;

    //se extrae del producto que se hizo click.
    let cantidades = sisema.productos;

    //se extrae del producto que se hizo click
    let monto = sistema.productos;

    let estadoCompra = "Pendiente";

    if (!isNaN(id) && !isNaN(idProductoCompra) && !isNaN(idCliente) &&
cantidades > 0 && monto > 0 && estadoCompra !== "") {

        let producto = new ListadoCompras(aumentarIdCompra(id),
idProductoCompra, idCliente, cantidades, monto, estadoCompra);
        sistema.agregarProducto(producto);
        alert("Registro exitoso!")

    } else {
        alert("Error: El precio y el stock deben ser valores numéricos y
mayores a 0");
    }
}

```

```

function salir() {
    /*Primero oculta todas las secciones */
    ocultarSecciones();
    /*Luego muestra lo que deberia verse */
    document.querySelector("#seccionLogin").style.display = "block";
    document.querySelector("#seccionRegistro").style.display = "block";
    document.querySelector("#navPrincipal").style.display = "none";
    document.querySelector("#aux").style.display = "none";
    /*Luego oculta la seccion de registro */
    ocultarSeccionRegistro()
    /*Luego resetea los valores al salir*/
    usuarioLogeado = null;
    usuarioLogeadoAdmin = null;
}

```

4.3 SISTEMA.JS:

```

class Sistema {
    constructor() {
        this.usuarios = [
            new Usuario(1, "Pablo", "123456aS", 3000, "1111-1111-1111-1111", 123),
            new Usuario(2, "Natalia", "123456As", 3000, "1111-1111-1111-1111", 123),
            new Usuario(3, "Mauro", "222222Fd", 3000, "1111-1111-1111-1111", 123),
            new Usuario(4, "Santiago", "111111Gg", 3000, "1111-1111-1111-1111", 123),
            new Usuario(5, "Shirley", "333333sH", 3000, "1111-1111-1111-1111", 123)
        ];
        this.usuariosAdmin = [
            new Admin(1, "admin", "admin"),
            new Admin(2, "a", "a"),
            new Admin(3, "admin1", "admin1"),
            new Admin(4, "admin2", "admin2"),
            new Admin(5, "admin3", "admin3")
        ]
        this.productos = [
            new Producto("PROD_ID_1", "Botella", "botella.jpg", "Botella de plastico duro", 100, "Precio Regular", 30, "Activo"),

```

```

        new Producto("PROD_ID_2", "Inflador", "inflador.jpg", "Un
inflador de petola con múltiples punteros", 450, "Oferta", 35, "Activo"),
        new Producto("PROD_ID_3", "Mesa de Juego", "mesa_juego.jpg",
"Mesa de Tejo para tu diversion", 200, "Precio Regular", 25, "Inactivo"),
        new Producto("PROD_ID_4", "Bolsa Grande", "bolso.jpeg",
"Bolso de tela impermeable", 100, "Precio Regular", 30, "Activo"),
        new Producto("PROD_ID_5", "Kit de Entrenamiento",
"kit_de_entrenamiento.jpg", "Todo lo necesario para tu entrenamiento",
300, "Oferta", 35, "Activo"),
        new Producto("PROD_ID_6", "Pack de Ciclismo",
"Kit_deportivo.jpg", "Articulos para ciclismo", 250, "Precio Regular",
25, "Inactivo"),
        new Producto("PROD_ID_7", "Mesa de Juego Simple",
"mesa_juego_2.jpg", "Mesa de juego portatil de tejo", 234, "Precio
Regular", 30, "Activo"),
        new Producto("PROD_ID_8", "Kit de Pelotas", "pelotas.jpg",
"Pelotas para practicar distintos deportes", 368, "Oferta", 35,
"Activo"),
        new Producto("PROD_ID_9", "Pelota de Ruggby",
"pelota_ruggbie.jpg", "Pelota de material liviano", 500, "Precio
Regular", 25, "Inactivo"),
        new Producto("PROD_ID_10", "Pelota de Ruggby chica",
"pelota_ruggbie_2.jpg", "Pelota de Ruggby para los chicos", 400, "Precio
Regular", 25, "Inactivo")
    ]

    this.misCompras = [
        //| posicion del obj |id | idProductoCompra |
idCliente | cantidad | monto | estadoCompra |
        new ListadoCompras(1, this.productos[0].id,
this.usuarios[0].id, 2, this.productos[0].precio, "Aprobado"),
        new ListadoCompras(2, this.productos[1].id,
this.usuarios[1].id, 2, this.productos[1].precio, "Aprobado"),
        new ListadoCompras(3, this.productos[2].id,
this.usuarios[2].id, 2, this.productos[2].precio, "Aprobado"),
        new ListadoCompras(4, this.productos[2].id,
this.usuarios[2].id, 2, this.productos[2].precio, "Aprobado"),
        new ListadoCompras(5, this.productos[1].id,
this.usuarios[1].id, 2, this.productos[1].precio, "Aprobado"),
        new ListadoCompras(6, this.productos[3].id,
this.usuarios[4].id, 2, this.productos[3].precio, "Pendiente"),
        new ListadoCompras(7, this.productos[4].id,
this.usuarios[3].id, 2, this.productos[4].precio, "Cancelada")
    ]
}

```



```

    buscarNombre(arrElementos, propiedad, busqueda) { // usuarios,
"nombre", "Pablo"
    let existe = false;

    for (let i = 0; i < arrElementos.length; i++) {
        const unElemento = arrElementos[i];
        const valorPropiedad = unElemento[propiedad]
        if (valorPropiedad.toLowerCase() === busqueda.toLowerCase())
{ //usuarios[i].nombre === usuarios[i]["nombre"] => unaPelícula.nombre ===
unaPelícula["nombre"] === "spiderman"
            existe = true;
            break;
        }
    }
    return existe;
}

    buscarElemento(arrElementos, propiedad, busqueda) { // productos,
"condicion", "Oferta"
    let existe = false;

    for (let i = 0; i < arrElementos.length; i++) {
        const unElemento = arrElementos[i];
        if (unElemento[propiedad] === busqueda) { //usuarios[i].nombre
=== usuarios[i]["nombre"] => unaPelícula.nombre === unaPelícula["nombre"]
=== "spiderman"
            existe = true;
            break;
        }
    }
    return existe;
}

    obtenerObjeto(arrElementos, propiedad, busqueda) {
    let objeto = null;

    for (let i = 0; i < arrElementos.length; i++) {
        const unElemento = arrElementos[i];
        if (unElemento[propiedad] === busqueda)
{ //películas[i].nombre === películas[i]["nombre"] => unaPelícula.nombre
=== unaPelícula["nombre"] === "spiderman"
            objeto = unElemento;
            break;
        }
    }
    return objeto;
}

```

```

    obtenerObjetoEnArray(arrElementos, propiedad, busqueda) {
        let objeto = [];

        for (let i = 0; i < arrElementos.length; i++) {
            const unElemento = arrElementos[i];
            if (unElemento[propiedad] === busqueda)
                { //peliculas[i].nombre === peliculas[i]["nombre"] => unaPelicula.nombre
                === unaPelicula["nombre"] === "spiderman"
                }
            objeto.push(unElemento);
        } else {
            // (`no se encontro nada ${[i]}`)
        }
    }
    return objeto;
}

//METODO VINCULADO AL REGISTRO
validarCamposVaciosRegistro(nombre, clave) {
    let camposValidos = false;
    if (nombre !== "" && clave !== "") {
        camposValidos = true;
    }
    return camposValidos;
}

//METODO VINCULADO AL REGISTRO
verificarFormatoContrasena(clave) {
    let valido = false;
    let contMayusculas = 0;
    let contMinusculas = 0;
    let contNumeros = 0;

    for (let i = 0; i < clave.length; i++) {

        if (clave.charAt(i) === clave.charAt(i).toUpperCase() &&
clave.charAt(1) !== " " && isNaN(clave.charAt(i))) {
            contMayusculas++;
        }

        if (clave.charAt(i) === clave.charAt(i).toLowerCase() &&
clave.charAt(1) !== " " && isNaN(clave.charAt(i))) {
            contMinusculas++;
        }

        if (!isNaN(clave.charAt(i))) {

```

```

        contNumeros++;
    }
}

    if (clave.length > 5 && contMayusculas >= 1 && contMinusculas >=
1 && contNumeros >= 1) {
        valido = true;
    }

    return valido;
}

verificarCVC(numero) {
    let validar = false;
    let contNumeros = 0;
    let contCaracterInvalido = 0;

    if (numero.length === 3) {
        for (let i = 0; i < numero.length; i++) {

            if (!isNaN(numero.charAt(i))) {
                contNumeros++;
            } else {
                contCaracterInvalido++;
            }
        }

        if (contNumeros === 3 && contCaracterInvalido === 0) {
            validar = true;
        }
    }
    return validar
}

verificarTarjetaCredito(numero) {
    let validar = false;
    let contNumerico = 0;
    let contGuiones = 0;
    let contCaracterInvalido = 0;
    if (numero.length < 19) {
        validar = false
    } else {
        for (let i = 0; i < numero.length; i++) {
            if (!isNaN(numero.charAt(i))) {
                contNumerico++;
            } else if (numero.charAt(i) === "-") {
                contGuiones++;
            }
        }
    }
}

```

```

        } else {
            contCaracterInvalido++;
        }
    }
    if (contNumerico === 16 && contCaracterInvalido === 0 &&
contGuiones === 3) {
        validar = true;
    }
}
return validar;
}

verificarTarjetaCreditoLuhn(numeroTarjeta) {
    let nroTarjeta = "";
    for (let i = 0; i < numeroTarjeta.length; i++) {
        if (numeroTarjeta.charAt(i) !== "-") {
            nroTarjeta += numeroTarjeta.charAt(i);
        }
    }

    let dev = false;
    let digitoVerificar = nroTarjeta.charAt(nroTarjeta.length - 1);
    let acumulador = 0;
    let cont = 0;
    for (let i = nroTarjeta.length - 2; i >= 0; i--) {
        let num = Number(nroTarjeta.charAt(i));
        if (cont % 2 === 0) {
            //se duplica
            let duplicado = num * 2;
            if (duplicado >= 10) {
                acumulador += (duplicado - 9);
            } else {
                acumulador += duplicado;
            }
        } else {
            acumulador += num;
        }
        cont++;
    }

    let multiplicado = acumulador * 9;
    let multiplicadoStr = String(multiplicado);
    let digitoVerificador =
multiplicadoStr.charAt(multiplicadoStr.length - 1);

    if (digitoVerificar === digitoVerificador) {
        dev = true;
    }
}

```

```

    }
    return dev;
}

//METODO VINCULADO AL REGISTRO
agregarUsuario(usuario) {
    this.usuarios.push(usuario);
}

agregarProducto(producto) {
    this.productos.push(producto)
}

agregarCompra(compra) {
    this.misCompras.push(compra)
}

//METODO VINCULADO AL LOGIN
/**Verificar Login, inicia el resultado en false, luego crea variable
unUsuario el cual Accede al metodo obtenerObjeto
* allí le pasa 3 parametros, 1) la lista de usuarios 2) el nombre de
propiedad 3) el valor del nombre.
*/
verificarLogin(nombre, clave, tipoUsuario) {
    let resultado = false;
    let unUsuario = this.obtenerObjeto(this.usuarios, "nombre",
nombre);
    let unUsuarioAdmin = this.obtenerObjeto(this.usuariosAdmin,
"nombre", nombre);

    if (tipoUsuario === "admin") {
        if (unUsuarioAdmin !== null) {
            if (clave === unUsuarioAdmin.contrasena) {
                return resultado = true;
            }
        }
    }

    } else {
        if (tipoUsuario === "user")
            if (unUsuario !== null) {
                if (clave === unUsuario.contrasena) {
                    resultado = true;
                }
            }
        return resultado;
    }
}

```

```

    obtenerCantidadProductos(idProducto) {
        let cantidad = 0;
        for (const compra of this.misCompras) {
            if(compra.idProductoCompra === idProducto &&
compra.estadoCompra === "Aprobado"){
                cantidad += compra.cantidad;
            }
        }
        return cantidad
    }

    obtenerGananciasTotales(){
        let total = 0;
        for (const compra of this.misCompras) {
            let producto = this.obtenerObjeto(this.productos, "id",
compra.idProductoCompra);
            if(compra.estadoCompra === "Aprobado"){
                total += producto.precio * compra.cantidad;
            }
        }
        return `$` + total
    }
}

```

4.4 CLASES.JS:

```

class Usuario {
    constructor(unId, unNombreUsuario, unaClave, unSaldo, nroTarjeta,
nroCVC) {
        this.id = unId;
        this.nombre = unNombreUsuario;
        this.contrasena = unaClave;
        this.saldo = unSaldo;
        this.tarjeta = nroTarjeta;
        this.cvc = nroCVC;
    }
}

class Admin {
    constructor(unId, unNombreUsuarioAdmin, unaClave) {
        this.id = unId;
        this.nombre = unNombreUsuarioAdmin;
    }
}

```

```

        this.contrasena = unaClave;
    }
}

class Producto {
    constructor(unId, unNombreProducto ,unaFoto,unaDescripcion, unPrecio,
unaCondicion, unStock, unEstado) {
        this.id = unId;
        this.nombre = unNombreProducto;
        this.imagen = unaFoto;
        this.descripcion = unaDescripcion;
        this.precio = unPrecio;
        this.condicion = unaCondicion;
        this.stock = unStock;
        this.estado = unEstado;

    }
}

class ListadoCompras {
    constructor(unId, unIdProducto, unIdCliente ,cantidades, montoTotal,
estadoCompra) {
        this.id = unId;
        this.idProductoCompra = unIdProducto;
        this.idCliente = unIdCliente;
        this.cantidad = cantidades;
        this.monto = montoTotal;
        this.estadoCompra = estadoCompra;

    }
}

```