

Universidad ORT Uruguay
Facultad de Ingeniería
Escuela de Tecnología

OBLIGATORIO PROGRAMACION 1



Sebastián Hohl – 327007



Matías Oreiro - 239479

Vespertino 1 A – V1A

Docente: Guillermo Vieira

Analista Programador / Analista en Tecnologías de la Información

22/05/2024

Contenido

1. Documento de análisis.....	5
1. Descripción general del problema a resolver	5
1. Tipos de usuario del sistema.....	5
2. Listado de funcionalidades	5
2. Detalle de Funcionalidades.....	6
2.1 F01 – Registro en la tienda.....	6
2.1.1 Acceso	6
2.1.2 Descripción.....	6
2.1.3 Interfaz de usuario	6
2.1.4 Validaciones	6
2.2 F02 – Iniciar sesión en el sistema.....	7
2.2.1 Acceso	7
2.2.2 Descripción.....	7
2.2.3 Interfaz de usuario	7
2.2.4 Validaciones	7
2.3 F03 – Compra de productos	8
2.3.1 Acceso	8
2.3.2 Descripción.....	8
2.3.3 Interfaz de usuario	8
2.3.4 Validaciones	9
2.4 F04 – Ver listado de compra	9
2.4.1 Acceso	9
2.4.2 Descripción.....	9
2.4.3 Interfaz de usuario	10
2.4.4 Validaciones	10

2.5	F05 – Ver 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	11
2.6	F06 – Listado y aprobación de compras de los usuarios	11
2.6.1	Acceso	11
2.6.2	Descripción	12
2.6.3	Interfaz de usuario	12
2.6.4	Validaciones	12
2.7	F07 – Crear productos	13
2.7.1	Acceso	13
2.7.2	Descripción	13
2.7.3	Interfaz de usuario	13
2.7.4	Validaciones	13
2.8	F08 – Administrar productos	14
2.8.1	Acceso	14
2.8.2	Descripción	14
2.8.3	Interfaz de usuario	14
2.8.4	Validaciones	14
2.9	F09 – Ver el informe de ganancias	15
2.9.1	Acceso	15
2.9.2	Descripción	15
2.9.3	Interfaz de usuario	15
2.9.4	Validaciones	15
2.-	El código completo (HTML y JS)	16

2.1.- Index.html	16
2.2.- Funciones.js.....	20
2.3.- Clases.js.....	33
3.- Listado de información precargada	42

1. Documento de análisis

1. Descripción general del problema a resolver

Un comercio que se dedica a la venta de artículos deportivos nos solicitó implementar un sistema para poder vender sus productos de forma online a los usuarios (e-commerce).

1. Tipos de usuario del sistema

- Comprador
- Administrador

2. Listado de funcionalidades

- F01 – Registro en la tienda – Usuarios: Comprador
 - Datos del cliente
 - ID único de usuario
- F02 – Iniciar sesión en el sistema – Usuarios: Comprador/Administrador
- F03 – Compra de productos – Usuarios: Comprador
 - Productos disponibles para comprar
 - Efectuar orden de compra
- F04 – Ver listado de compras – Usuarios: Comprador
- F05 – Ver los productos en oferta – Usuarios: Comprador
- F06 – Listado y aprobación de compras de los usuarios – Usuarios: Administrador
 - Listas de estados de compra
 - Botón para aprobar/rechazar compras
- F07 – Crear productos – Usuarios: Administrador
- F08 – Administrar productos – Usuarios: Administrador
 - Modificar el stock
 - Modificar el estado
 - Asignar ofertas a los productos
- F09 – Ver el informe de ganancias – Usuarios: 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 en la tienda

2.1.1 Acceso

Solamente puede acceder a ella el perfil de usuario Comprador.

2.1.2 Descripción

El comprador deberá completar el registro en el sistema ingresando su Nombre, Apellido, Nombre de usuario, Contraseña, Número de tarjeta de crédito y CVC.

2.1.3 Interfaz de usuario

Registrarse

Nombre:

Apellido:

UserName:

Contraseña:

Numero de tarjeta:

CVC:

2.1.4 Validaciones

- Todos los campos son obligatorios.
- Verificar que el nombre de usuario no exista en el sistema.
- Verificar que el formato de la tarjeta de crédito es el correcto.
(XXXX-XXXX-XXXX-XXXX)
- Validar el número de tarjeta de crédito contra el algoritmo de Luhn.
- Verificar que el CVC tenga 3 dígitos.

2.2 F02 – Iniciar sesión en el sistema

2.2.1 Acceso

Tanto el perfil de usuario Administrador como Comprador pueden acceder a esta función.

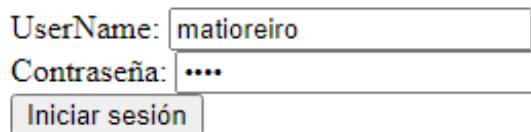
2.2.2 Descripción

Esta función da acceso a la página.

Los usuarios registrados deben ingresar su nombre de Usuario y su Contraseña ingresadas en el registro.

2.2.3 Interfaz de usuario

Iniciar sesión



The screenshot shows a login form with the following elements:

- A label "UserName:" followed by a text input field containing the value "matioreiro".
- A label "Contraseña:" followed by a password input field containing four dots "....".
- A button labeled "Iniciar sesión" located below the password field.

2.2.4 Validaciones

- Se requieren ambos datos
- Nombre de usuario registrado en el sistema
- Contraseña case sensitive, debe ser ingresada tal cual lo fue al momento del registro

2.3 F03 – Compra de productos

2.3.1 Acceso

Solamente puede acceder a ella el perfil de usuario Comprador.

2.3.2 Descripción

Una vez el inicio de sesión fue exitoso, los usuarios pueden ver en la pantalla principal los productos disponibles para comprar. Debe incluir nombre del producto, descripción, precio unitario, si está de oferta o no y la imagen ilustrativa.

2.3.3 Interfaz de usuario

Comprador

Productos: Fútbol - \$14.99 ▼

☐ Solo ofertas

Fútbol



Una pelota de futbol Nike

Precio unitario: \$14.99

Cantidad:

2.3.4 Validaciones

- El stock de los artículos mostrados debe ser mayor a 0
- Los artículos mostrados no pueden tener estado pausado, sea el stock que sea
- Para realizar la compra, la cantidad no puede ser menor a 1
- Para que se efectúe la compra, ésta quedará pendiente hasta ser aprobada o denegada por un administrador

2.4 F04 – Ver listado de compra

2.4.1 Acceso

Solamente puede acceder a ella el perfil de usuario Comprador.

2.4.2 Descripción

El comprador debe tener la posibilidad de ver una lista con sus compras.

Debe incluir el Nombre del producto, la cantidad que compró, monto total de la compra, estado de la compra, un botón que permita cancelar las compras aun pendientes, y al final mostrar un párrafo que muestre el monto total de las compras aprobadas y el saldo disponible del usuario.

2.4.3 Interfaz de usuario

Mis compras

☒ Todas ☐ Aprobadas ☐ Pendientes ☐ Canceladas

Producto	Precio total	Cantidad	Estado	Acciones
Fútbol	\$74.95	5	Aprobada	
Golf	\$74.97	3	Pendiente	Cancelar
Bowling	\$269.97	3	Pendiente	Cancelar
Futsal	\$95.92	8	Aprobada	
Padel	\$19990.00	1000	Cancelada	

Monto total de compras aprobadas: \$170.87.

Su saldo actual es de \$2829.13

2.4.4 Validaciones

- Validar estado actual de la compra
- Si la compra está aprobada o rechazada, ocultar el botón de cancelar
- Si la compra fue aprobada, sumar al acumulado del monto total de compras aprobadas
- Descontar del saldo disponible del usuario el monto total de la compra aprobada
- Descontar el monto total de las compras aprobadas del saldo disponible del usuario
- Descontar la cantidad de productos en el stock cuando la compra fue aprobada
- Validar que el filtro muestre únicamente las compras con el estado seleccionado

2.5 F05 – Ver productos en oferta

2.5.1 Acceso

Solamente puede acceder a ella el perfil de usuario Comprador.

2.5.2 Descripción

El comprador deberá contar con una vista en la que se muestren únicamente los productos marcados como oferta, incluyendo la opción de comprar.

2.5.3 Interfaz de usuario

Comprador

Productos: Voley - \$10.99 - En oferta ▼

☒ **Solo ofertas**

2.5.4 Validaciones

- Verificar que funcione el mismo sistema de compras de la función F03
- El stock de los artículos mostrados debe ser mayor a 0
- Los artículos mostrados no pueden tener estado pausado, sea el stock que sea
- Para realizar la compra, la cantidad no puede ser menor a 1
- Para que se efectúe la compra, ésta quedará pendiente hasta ser aprobada o denegada por un administrador

2.6 F06 – Listado y aprobación de compras de los usuarios

2.6.1 Acceso

Solamente puede acceder a ella el perfil de usuario Administrador.

2.6.2 Descripción

El administrador deberá tener una vista donde se muestren las compras de los usuarios. Esta vista debe contener 3 listas, una con las compras aprobadas, otra con las pendientes y la última con las compras rechazadas.

También debe incluir un botón que permita al administrador aprobar una compra.

2.6.3 Interfaz de usuario

Admin

Lista de compras de los usuarios

☒ Todas ☐ Aprobadas ☐ Pendientes ☐ Canceladas

Comprador	Producto	Precio total	Cantidad	Estado	Acciones
nahuelinho	Fútbol	\$74.95	5	Aprobada	
madridista	Basket	\$25.98	2	Pendiente	Aprobar
elmanyá	Bowling	\$539.94	6	Pendiente	Aprobar
elmessias	Ping Pong	\$55.92	8	Pendiente	Aprobar
comandante	Padel	\$199.90	10	Pendiente	Aprobar
nahuelinho	Golf	\$74.97	3	Pendiente	Aprobar
nahuelinho	Bowling	\$269.97	3	Pendiente	Aprobar
nahuelinho	Futsal	\$95.92	8	Aprobada	
nahuelinho	Padel	\$19990.00	1000	Cancelada	

2.6.4 Validaciones

- El select tipo de lista debe mostrar la lista indicada
- El botón “aprobar” aprobaría si y solo si se dan las validaciones correctas
- Para aprobar una compra el stock debe ser mayor a 0
- El stock del producto debe ser suficiente para la cantidad a comprar
- El producto de la compra a aprobar no debe estar en estado “pausado”
- Si la compra puede efectuarse, cambiar el estado de la misma a aprobado
- Si una compra fue aprobada, descontar la cantidad comprada del stock

2.7 F07 – Crear productos

2.7.1 Acceso

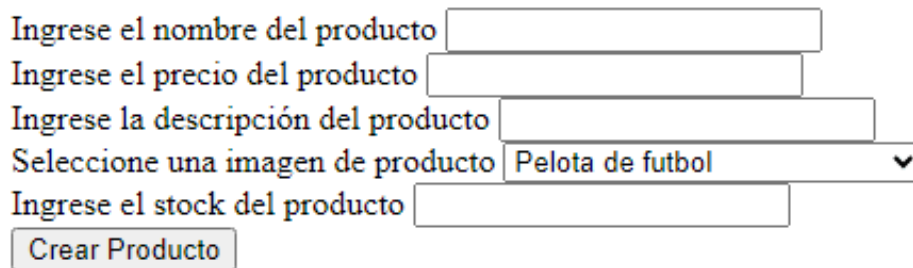
Solamente puede acceder a ella el perfil de usuario Administrador.

2.7.2 Descripción

El administrador podrá crear un producto nuevo, de manera que aparezcan a los compradores para comprarlos. Se solicitará nombre del producto, precio unitario, una breve descripción, url de la imagen y la cantidad de stock disponible.

2.7.3 Interfaz de usuario

Crear productos



Formulario de creación de productos:

- Ingresa el nombre del producto:
- Ingresa el precio del producto:
- Ingresa la descripción del producto:
- Selecciona una imagen de producto: ▼
- Ingresa el stock del producto:
- Botón:

2.7.4 Validaciones

- Todos los campos son obligatorios.
- Verificar que la cantidad de stock y el precio sean numéricos y mayores a 0
- Si alguna de las validaciones no se cumple, se informará mediante HTML
- Al crear el producto debe estar en estado activo
- Asignarle un ID único al crear el producto
- Al ser creado no estará de oferta

2.8 F08 – Administrar productos

2.8.1 Acceso

Solamente puede acceder a ella el perfil de usuario Administrador.

2.8.2 Descripción

El administrador contará con una vista que permita administrar los productos creados. Las acciones que pueden realizar serían modificar el stock, modificar el estado de un producto y asignar a un producto si el mismo está de oferta o desmarcarlo si ya lo estaba previamente.

2.8.3 Interfaz de usuario

Administrar productos

Producto	Precio	Stock	Modificar stock		Estado	Modificar estado	Oferta	Modificar oferta
Fútbol	14.99	45	<input type="text"/>	Modificar	Activo	<input type="button" value="Pausar"/>	No	<input type="button" value="Agregar oferta"/>
Basket	12.99	15	<input type="text"/>	Modificar	Activo	<input type="button" value="Pausar"/>	No	<input type="button" value="Agregar oferta"/>
Voley	10.99	11	<input type="text"/>	Modificar	Activo	<input type="button" value="Pausar"/>	Si	<input type="button" value="Quitar oferta"/>
Golf	24.99	60	<input type="text"/>	Modificar	Activo	<input type="button" value="Pausar"/>	No	<input type="button" value="Agregar oferta"/>
Bowling	89.99	44	<input type="text"/>	Modificar	Activo	<input type="button" value="Pausar"/>	No	<input type="button" value="Agregar oferta"/>
Tenis	9.99	70	<input type="text"/>	Modificar	Activo	<input type="button" value="Pausar"/>	No	<input type="button" value="Agregar oferta"/>
Ping Pong	6.99	13	<input type="text"/>	Modificar	Activo	<input type="button" value="Pausar"/>	No	<input type="button" value="Agregar oferta"/>
Futsal	11.99	52	<input type="text"/>	Modificar	Activo	<input type="button" value="Pausar"/>	Si	<input type="button" value="Quitar oferta"/>
Padel	19.99	15	<input type="text"/>	Modificar	Activo	<input type="button" value="Pausar"/>	No	<input type="button" value="Agregar oferta"/>
Bolas de pool	149.99	8	<input type="text"/>	Modificar	Activo	<input type="button" value="Pausar"/>	No	<input type="button" value="Agregar oferta"/>

2.8.4 Validaciones

- Si se ingresa un stock valor 0, el estado del producto se cambiará automáticamente a pausado
- Al modificar el estado o la oferta, tanto el párrafo estado como el select modificar estado/oferta deben coincidir

2.9 F09 – Ver el informe de ganancias

2.9.1 Acceso

Solamente puede acceder a ella el perfil de usuario Administrador.

2.9.2 Descripción

En esta vista, el administrador, deberá ver en un párrafo, la cantidad total de vendida (unidades) de cada producto y la ganancia total por todas las compras realizadas por los usuarios.

2.9.3 Interfaz de usuario

Ver informe de ganancias

Se ha vendido 5 de Fútbol

Se ha vendido 8 de Futsal

Ganancias totales: \$170.87

2.9.4 Validaciones

- Se muestren la cantidad de unidades vendidas distinguidos por el producto
- Mostrar el acumulado de ganancias de ventas aprobadas en otro párrafo

2.- El código completo (HTML y JS)

2.1.- Index.html

```
<!DOCTYPE html>
<html lang="es">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-
scale=1.0">
  <title>Obligatorio Programación 1</title>
  <link rel="stylesheet" href="css/estilos.css">
  <script src="js/clases.js"></script>
  <script src="js/funciones.js"></script>
</head>
<body>
  <h1>Obligatorio</h1>
  <section id="sectionInicial">
    <h2>Iniciar sesión</h2>
    <form id="formIniciarSesion">
      <label for="userName" >UserName: </label>
      <input type="text" id="userName" value="mationeiro"> <br>
      <label for="contraseña" >Contraseña: </label>
      <input type="password" id="password" value="1891"> <br>
      <input type="button" value="Iniciar sesión"
id="btnIniciarSesion">
    </form>
    <br><br>
    <h2>Registrarse</h2>
    <form id="formRegistrar">
      <label for="registroDeNombre">Nombre: </label>
      <input type="text" id="registroDeNombre"> <br>
      <label for="registroDeApellido">Apellido: </label>
      <input type="text" id="registroDeApellido"> <br>
      <label for="registroDeUserName">UserName: </label>
      <input type="text" id="registroDeUserName"> <br>
      <label for="registroDeContraseña">Contraseña: </label>
      <input type="password" id="registroDeContraseña"> <br>
      <label for="registroDeNumeroDeTarjeta">Numero de tarjeta:
</label>
      <input type="text" id="registroDeNumeroDeTarjeta"
placeholder="XXXX-XXXX-XXXX-XXXX"> <br>
      <label for="registroDeCVC">CVC: </label>
      <input type="text" id="registroDeCVC" placeholder="XXX"> <br>
      <input type="button" value="Registrarse" id="btnRegistro">
```



```

        </form>
        <p id="pContraseniaValida"></p>
    </section>
    <section id="sectionComprador">
        <h2>Comprador</h2>
        <label for="selectProductosDisponibles"><strong>Productos:
    </strong></label>
        <select id="selectProductosDisponibles">

        </select><br>
        <input type="checkbox" id="checkProdEnOferta">
        <label for="checkProdEnOferta">Solo ofertas</label><br><br>
        <div id="muestraDelProductoSeleccionado">

        </div>
        <label for="cantProducto">Cantidad: </label>
        <input type="number" id="cantProducto">
        <input type="button" value="Comprar" id="btnComprarProducto">
    <br><br>

    <h3>Mis compras</h3>
    <input type="radio" name="radioFiltrarMisCompras"
id="radioFiltrarMisComprasTodas" checked>
    <label for="radioFiltrarMisComprasTodas">Todas</label>
    <input type="radio" name="radioFiltrarMisCompras"
id="radioFiltrarMisComprasAprobadas">
    <label for="radioFiltrarMisComprasAprobadas">Aprobadas</label>
    <input type="radio" name="radioFiltrarMisCompras"
id="radioFiltrarMisComprasPendientes">
    <label for="radioFiltrarMisComprasPendientes">Pendientes</label>
    <input type="radio" name="radioFiltrarMisCompras"
id="radioFiltrarMisComprasCanceladas">
    <label for="radioFiltrarMisComprasCanceladas">Canceladas</label>
    <table>
        <thead>
            <tr>
                <th>Producto</th>
                <th>Precio total</th>
                <th>Cantidad</th>
                <th>Estado</th>
                <th>Acciones</th>
            </tr>
        </thead>
        <tbody id="tablaDeMisCompras">

        </tbody>
    </table><br>

```

```

<p id="parrafoDeComprasAprobadas"></p>
<p id="parrafoSaldoDelComprador"></p>
<input type="button" value="Cerrar sesión" id="btnCerrarSesion">

</section>
<section id="sectionAdministrador">
  <h2>Admin</h2>
  <h3>Lista de compras de los usuarios</h3>
  <input type="radio" name="radioFiltrarComprasDeUsuario"
id="radioFiltrarComprasDeUsuarioTodas" checked>
  <label for="radioFiltrarComprasDeUsuarioTodas">Todas</label>
  <input type="radio" name="radioFiltrarComprasDeUsuario"
id="radioFiltrarComprasDeUsuarioAprobadas">
  <label
for="radioFiltrarComprasDeUsuarioAprobadas">Aprobadas</label>
  <input type="radio" name="radioFiltrarComprasDeUsuario"
id="radioFiltrarComprasDeUsuarioPendientes">
  <label
for="radioFiltrarComprasDeUsuarioPendientes">Pendientes</label>
  <input type="radio" name="radioFiltrarComprasDeUsuario"
id="radioFiltrarComprasDeUsuarioCanceladas">
  <label
for="radioFiltrarComprasDeUsuarioCanceladas">Canceladas</label>
  <table>
    <thead>
      <tr>
        <th>Comprador</th>
        <th>Producto</th>
        <th>Precio total</th>
        <th>Cantidad</th>
        <th>Estado</th>
        <th>Acciones</th>
      </tr>
    </thead>
    <tbody id="tablaDeComprasDeUsuarios">

    </tbody>
  </table><br><br>
  <h3>Crear productos</h3>
  <form id="formCrearProducto">
    <label for="nombreCrearProducto">Ingrese el nombre del
producto</label>
    <input type="text" id="nombreCrearProducto"><br>
    <label for="precioCrearProducto">Ingrese el precio del
producto</label>
    <input type="text" id="precioCrearProducto"><br>

```

```

        <label for="descripcionCrearProducto">Ingrese la descripción
del producto</label>
        <input type="text" id="descripcionCrearProducto"><br>
        <label for="imagenCrearProducto">Seleccione una imagen de
producto</label>
        <select id="imagenCrearProducto">
            <option value="imagenes/futbol.jpg">Pelota de
futbol</option>
            <option value="imagenes/basket.jpg">Pelota de
basket</option>
            <option value="imagenes/voley.jpg">Pelota de
voley</option>
            <option value="imagenes/golf.avif">Pelota de
golf</option>
            <option value="imagenes/bowling.webp">Pelota de
bowling</option>
            <option value="imagenes/tenis.webp">Pelota de
tenis</option>
            <option value="imagenes/pingpong.webp">Caja de pelotas de
ping pong</option>
            <option value="imagenes/futsal.webp">Pelota de
futsal</option>
            <option value="imagenes/padel.webp">Pelota de
padel</option>
            <option value="imagenes/pool.webp">Bolas de pool</option>
            <option value="imagenes/rugby.jpg">Pelota de
rugby</option>
            <option value="imagenes/futbolamericano.webp">Pelota de
futbol americano</option>
            <option value="imagenes/waterpolo.webp">Pelota de
waterpolo</option>
            <option value="imagenes/pilates.webp">Pelota de
pilates</option>
            <option value="imagenes/beisbol.webp">Pelota de
beisbol</option>
        </select><br>
        <label for="stockCrearProducto">Ingrese el stock del
producto</label>
        <input type="number" id="stockCrearProducto"><br>
        <input type="button" value="Crear Producto"
id="btnCrearProducto">
    </form>
    <p id="pErroresCrearProducto"></p> <br><br>
    <h3>Administrar productos</h3>
    <table>
        <thead>
            <tr>

```

```

        <th>Producto</th>
        <th>Precio</th>
        <th>Stock</th>
        <th>Modificar stock</th>
        <th>Estado</th>
        <th>Modificar estado</th>
        <th>Oferta</th>
        <th>Modificar oferta</th>
    </tr>
</thead>
<tbody id="tablaDeTodosLosProductos">

    </tbody>
</table> <br> <br>
<h3>Ver informe de ganancias</h3>
<p id="pInformeDeGanancias"></p> <br><br>
<input type="button" value="Cerrar sesión"
id="btnCerrarSesionAdministrador">
</section>
</body>
</html>

```

2.2.- Funciones.js

```

window.addEventListener("load", inicio);

let sistema = new Sistema();
sistema.precargaDeDatos();

function inicio(){
    document.querySelector("#btnIniciarSesion").addEventListener("click",
realizarInicioSesion);
    document.querySelector("#btnCerrarSesion").addEventListener("click",
realizarCierreDeSesion);
    document.querySelector("#btnCerrarSesionAdministrador").addEventListener("click",
realizarCierreDeSesion);
    document.querySelector("#btnRegistro").addEventListener("click",
realizarRegistro);
    document.querySelector("#selectProductosDisponibles").addEventListener("change",
actualizarMuestraDeProducto);
    document.querySelector("#btnComprarProducto").addEventListener("click",
comprar);
    document.querySelector("#checkProdEnOferta").addEventListener("click",
actualizarSelectDeProductos);
}

```

```

        document.querySelector("#checkProdEnOferta").addEventListener("click",
        actualizarMuestraDeProducto);
        document.querySelector("#radioFiltrarMisComprasTodas").addEventListener("click",
        actualizarTablaMisCompras);
        document.querySelector("#radioFiltrarMisComprasAprobadas").addEventListener("click",
        actualizarTablaMisCompras);
        document.querySelector("#radioFiltrarMisComprasPendientes").addEventListener("click",
        actualizarTablaMisCompras);
        document.querySelector("#radioFiltrarMisComprasCanceladas").addEventListener("click",
        actualizarTablaMisCompras);
        document.querySelector("#radioFiltrarComprasDeUsuarioTodas").addEventListener("click",
        actualizarTablaComprasDeUsuario);
        document.querySelector("#radioFiltrarComprasDeUsuarioAprobadas").addEventListener("click",
        actualizarTablaComprasDeUsuario);
        document.querySelector("#radioFiltrarComprasDeUsuarioPendientes").addEventListener("click",
        actualizarTablaComprasDeUsuario);
        document.querySelector("#radioFiltrarComprasDeUsuarioCanceladas").addEventListener("click",
        actualizarTablaComprasDeUsuario);
        document.querySelector("#btnCrearProducto").addEventListener("click",
        validarCreacionDeNuevoProducto);
    }

    //Inicio/cierre de sesion, registro y validaciones
    function realizarInicioSesion(){
        let userName = document.querySelector("#userName").value;
        let password = document.querySelector("#password").value;

        if(sistema.validarInicioSesionUsuario(userName, password)){
            actualizarComprador();
            mostrarYOcultarPanel("#sectionComprador");
        }else if(sistema.validarInicioSesionAdministrador(userName,
        password)){
            actualizarAdministrador();
            mostrarYOcultarPanel("#sectionAdministrador");
        }else{
            alert("No se encontraron los datos")
        }

        document.querySelector("#formIniciarSesion").reset();
    }

    function realizarCierreDeSesion(){
        sistema.cierreDeSesion();
        reestablecerInputs();
        mostrarYOcultarPanel("#sectionInicial");
    }

```

```

function realizarRegistro(){
    let nombre = document.querySelector("#registroDeNombre").value;
    let apellido = document.querySelector("#registroDeApellido").value;
    let userName = document.querySelector("#registroDeUserName").value;
    let password = document.querySelector("#registroDeContraseña").value;
    let numeroDeTarjeta =
document.querySelector("#registroDeNumeroDeTarjeta").value;
    let cvc = parseInt(document.querySelector("#registroDeCVC").value);
    let tarjetaSinGuiones = numeroDeTarjeta.replaceAll("-", "");

    if(nombre == "" || apellido == "" || userName == "" || password == ""
|| numeroDeTarjeta == "" || cvc == ""){
        alert("Todos los campos son obligatorios")
    }else if(sistema.existeComprador(userName) ||
sistema.existeAdministrador(userName)){
        alert("Error, vuelva a verificar los datos")
    }else if(!passwordValida(password)){
        let errores = "";
        errores += "La contraseña tiene que tener al menos 5 caracteres.
<br>"
        errores += "La contraseña debe tener mínimo una mayúscula. <br>"
        errores += "La contraseña debe tener mínimo una minúscula. <br>"
        errores += "La contraseña debe tener mínimo un número. <br>"
        document.querySelector("#pContraseniasValida").innerHTML =
errores;
    }else if(!numeroDeTarjetaValida(numeroDeTarjeta)){
        alert("La tarjeta no es válida");
    }else if(!algoritmoLuhn(tarjetaSinGuiones)){
        alert("La tarjeta no se pudo verificar.");
    }else if(cvc < 100 || cvc > 999){
        alert("Ingresa un CVC válido");
    }else{
        sistema.agregarComprador(nombre, apellido, userName, password,
numeroDeTarjeta, cvc);
        document.querySelector("#formRegistrar").reset();
        alert("Registro con éxito")
    }
}

function passwordValida(password){
    let resp = false;
    let cantMayus = 0;
    let cantMinus = 0;
    let cantNumeros = 0;

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

```

```

        if(password.charCodeAt(i) >= 65 && password.charCodeAt(i) <= 90){
            cantMayus++;
        }else if(password.charCodeAt(i) >= 97 && password.charCodeAt(i)
<= 122){
            cantMinus++;
        }else if(password.charCodeAt(i) >= 48 && password.charCodeAt(i)
<= 57){
            cantNumeros++;
        }
    }

    if(!(password.length < 5 || cantMayus < 1 || cantMinus < 1 ||
cantNumeros < 1)){
        resp = true;
    }

    return resp;
}

function numeroDeTarjetaValida(numeroDeTarjeta){
    let valida = false;
    numeroDeTarjeta = numeroDeTarjeta.replaceAll(" ", ``);
    let cantNumeros = 0;
    let cantGuiones = 0;

    for (let i = 0; i < numeroDeTarjeta.length; i++) {
        if(numeroDeTarjeta.charCodeAt(i) >= 48 &&
numeroDeTarjeta.charCodeAt(i) <= 57){
            cantNumeros++
        }else if(numeroDeTarjeta.charAt(i) == "-"){
            cantGuiones++;
        }
    }

    if(!(numeroDeTarjeta.length != 19 || cantNumeros != 16 || cantGuiones
!= 3)){
        valida = true;
    }

    return valida;
}

function validarCreacionDeNuevoProducto(){
    let nombre = document.querySelector("#nombreCrearProducto").value;
    let valuePrecio =
document.querySelector("#precioCrearProducto").value;

```

```

        valuePrecio = valuePrecio.replaceAll(",", ".");
        let precio = parseFloat(valuePrecio);
        let descripcion =
document.querySelector("#descripcionCrearProducto").value;
        let img = document.querySelector("#imagenCrearProducto").value;
        let stock =
parseInt(document.querySelector("#stockCrearProducto").value);
        let texto = "";

        if(isNaN(precio) || isNaN(stock) || nombre.length == 0 ||
descripcion.length == 0 || img.length == 0 ){
            texto += "Todos los campos son obligatorios";
        }else if(precio <= 0 || stock <= 0){
            texto += "El precio y el stock deben ser mayores a 0";
        }else{
            sistema.agregarProducto(nombre, precio, descripcion, img, stock);
            document.querySelector("#formCrearProducto").reset();
            alert("Producto creado con éxito")
            texto = "";
        }
    }

    document.querySelector("#pErroresCrearProducto").innerHTML = texto;
    mostrarTablaDeTodosLosProductos();
}

//Actualizar usuario
function actualizarComprador(){
    actualizarSelectDeProductos();
    actualizarMuestraDeProducto();
    actualizarTablaMisCompras();
    mostrarSaldo();
    mostrarMontoDeComprasAprobadas();
}

function actualizarSelectDeProductos(){
    let productos = sistema.obtenerListadoDeProductos();
    let texto = "";
    if(document.querySelector("#checkProdEnOferta").checked ){
        for (let i = 0; i < productos.length; i++) {
            let objProducto = productos[i];
            if(objProducto.oferta){
                texto += `<option
value="${objProducto.id}">${objProducto.nombre} - $$${objProducto.precio}
- En oferta</option>`
            }
        }
    }
}

```



```

    }else{
        for (let i = 0; i < productos.length; i++) {
            let objProducto = productos[i];
            if(objProducto.oferta == true){
                texto += `<option
value="${objProducto.id}">${objProducto.nombre} - ${objProducto.precio}
- En oferta</option>`
            }else{
                texto += `<option
value="${objProducto.id}">${objProducto.nombre} -
${objProducto.precio}</option>`
            }
        }
    }
    actualizarMuestraDeProducto();
    document.querySelector("#selectProductosDisponibles").innerHTML =
texto;
}

function actualizarMuestraDeProducto(){
    let producto =
document.querySelector("#selectProductosDisponibles").value;
    let lista = sistema.obtenerListadoDeProductos();
    let texto = "";
    for (let i = 0; i < lista.length; i++) {
        let objProducto = lista[i];
        if(producto == objProducto.id){
            texto = `<h3>${objProducto.nombre}</h3>
                
                <p>${objProducto.descripcion}</p>
                <p>Precio unitario: ${objProducto.precio}</p>`
        }
    }
    document.querySelector("#muestraDelProductoSeleccionado").innerHTML =
texto;
}

function actualizarTablaMisCompras(){
    let lista;
    if(document.querySelector("#radioFiltrarMisComprasTodas").checked){
        lista = sistema.obtenerMisCompras("Todas");
    }else
    if(document.querySelector("#radioFiltrarMisComprasAprobadas").checked){
        lista = sistema.obtenerMisCompras("Aprobada");
    }else
    if(document.querySelector("#radioFiltrarMisComprasPendientes").checked){

```

```

        lista = sistema.obtenerMisCompras("Pendiente");
    }else
if(document.querySelector("#radioFiltrarMisComprasCanceladas").checked){
        lista = sistema.obtenerMisCompras("Cancelada");
    }

    mostrarTabla(lista);
}

function mostrarSaldo(){
    document.querySelector("#parrafoSaldoDelComprador").innerHTML = `Su
saldo actual es de ${sistema.obtenerSaldoComprador().toFixed(2)}`;
}

function mostrarMontoDeComprasAprobadas(){
    document.querySelector("#parrafoDeComprasAprobadas").innerHTML =
`Monto total de compras aprobadas:
${sistema.calcularMontoDeComprasAprobadas().toFixed(2)}.`
}

// Actualizar administrador

function actualizarAdministrador(){
    actualizarTablaComprasDeUsuario();
    mostrarTablaDeTodosLosProductos();
    mostrarInformeDeGanancias();
}

function actualizarTablaComprasDeUsuario(){
    let lista;
    if(document.querySelector("#radioFiltrarComprasDeUsuarioTodas").checked){
        lista = sistema.obtenerListadoDeCompras();
    }else
if(document.querySelector("#radioFiltrarComprasDeUsuarioAprobadas").checked){
        lista = sistema.obtenerComprasDeUsuario("Aprobada");
    }else
if(document.querySelector("#radioFiltrarComprasDeUsuarioPendientes").checked){
        lista = sistema.obtenerComprasDeUsuario("Pendiente");
    }else
if(document.querySelector("#radioFiltrarComprasDeUsuarioCanceladas").checked){
        lista = sistema.obtenerComprasDeUsuario("Cancelada");
    }
}

```

```

    mostrarTablaDeAdministrador(lista);
}

function mostrarTablaDeTodosLosProductos(){
    let texto = "";

    for (let i = 0; i < sistema.listaDeProductos.length; i++) {
        let objProducto = sistema.listaDeProductos[i];
        texto += `<tr>
            <td>${objProducto.nombre}</td>
            <td>${objProducto.precio.toFixed(2)}</td>
            <td>${objProducto.stock}</td>
            <td><input type="number" id="${objProducto.id}-
stockProducto" class="stock"><input type="button" value="Modificar"
id="${objProducto.id}" class="modificarStock"></td>
            `

            if(objProducto.estado){
                texto += `<td>Activo</td>`
                texto += `<td><input type="button" value="Pausar"
id="${objProducto.id}-estadoProducto" class="pausar"></td>`

            }else{
                texto += `<td>Pausado</td>`
                texto += `<td><input type="button" value="Activar"
id="${objProducto.id}-estadoProducto" class="activar"></td>`

            }

            if(objProducto.oferta){
                texto += `<td>Si</td>`
                texto += `<td><input type="button" value="Quitar oferta"
id="${objProducto.id}-ofertaProducto" class="quitarOferta"></td>`

            }else{
                texto += `<td>No</td>`
                texto += `<td><input type="button" value="Agregar oferta"
id="${objProducto.id}-ofertaProducto" class="agregarOferta"></td>`

            }

        }

        document.querySelector("#tablaDeTodosLosProductos").innerHTML =
texto;

        let listaDeBotonesParaModificarStock =
document.querySelectorAll(".modificarStock");

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

        let listaDeBotonesParaPausarEstado =
document.querySelectorAll(".pausar");

```

```

        for (let i = 0; i < listaDeBotonesParaPausarEstado.length; i++) {
            let botonActual = listaDeBotonesParaPausarEstado[i];
            botonActual.addEventListener("click", pausarEstadoDinamico)
        }
        let listaDeBotonesParaActivarEstado =
document.querySelectorAll(".activar");
        for (let i = 0; i < listaDeBotonesParaActivarEstado.length; i++) {
            let botonActual = listaDeBotonesParaActivarEstado[i];
            botonActual.addEventListener("click", activarEstadoDinamico)
        }
        let listaDeBotonesParaAgregarOferta =
document.querySelectorAll(".agregarOferta");
        for (let i = 0; i < listaDeBotonesParaAgregarOferta.length; i++) {
            let botonActual = listaDeBotonesParaAgregarOferta[i];
            botonActual.addEventListener("click", agregarOfertaDinamico)
        }
        let listaDeBotonesParaQuitarOferta =
document.querySelectorAll(".quitarOferta");
        for (let i = 0; i < listaDeBotonesParaQuitarOferta.length; i++) {
            let botonActual = listaDeBotonesParaQuitarOferta[i];
            botonActual.addEventListener("click", quitarOfertaDinamico)
        }
    }

function mostrarInformeDeGanancias(){
    let texto = "";
    let listaDeProductos = sistema.obtenerListadoDeTodosLosProductos()

    for (let i = 0; i < listaDeProductos.length; i++) {
        let objProducto = listaDeProductos[i];
        if(objProducto.cantidadDeVendidos > 0){
            texto += `Se ha vendido ${objProducto.cantidadDeVendidos} de
${objProducto.nombre} <br>`;
        }

    }

    texto += `<strong>Ganancias totales:</strong>
${sistema.obtenerMontoTotalDeComprasAprobadas().toFixed(2)}`;
    document.querySelector("#pInformeDeGanancias").innerHTML = texto;
}

// funcionalidades
function comprar(){

```

```

    let idProducto =
document.querySelector("#selectProductosDisponibles").value;
    let cantidad =
parseInt(document.querySelector("#cantProducto").value);

    if(isNaN(cantidad) || cantidad <= 0){
        alert("Debe ingresar una cantidad")
    }else{
        sistema.comprarProducto(idProducto, cantidad);
        actualizarComprador();
        alert("Compra realizada exitosamente, aguarde a que un
administrador la apruebe.");
    }
}

function cancelarCompraDinamico(){
    let id = parseInt(this.id);
    sistema.cancelarCompra(id);
    actualizarTablaMisCompras();
}

function aprobarCompraDinamico(){
    let id = parseInt(this.id);
    sistema.procesarCompra(id);
    actualizarTablaComprasDeUsuario();
    mostrarTablaDeTodosLosProductos();
    mostrarInformeDeGanancias();
}

function modificarStockDinamico(){
    let id = this.id
    let cantidad = parseInt(document.querySelector(`#${this.id}-
stockProducto`).value);
    if(isNaN(cantidad) || cantidad < 0){
        alert("Debe ingresar un valor válido de stock")
    }else{
        sistema.actualizarStock(id, cantidad);
    }
    mostrarTablaDeTodosLosProductos();
}

function pausarEstadoDinamico(){
    let id = this.id
    let idProducto = id.substring(0, id.indexOf("-"))
    sistema.modificarEstado(idProducto);
    mostrarTablaDeTodosLosProductos();
}

```

```

function activarEstadoDinamico(){
    let id = this.id
    let idProducto = id.substring(0, id.indexOf("-"))
    sistema.modificarEstado(idProducto);
    mostrarTablaDeTodosLosProductos();
}

function agregarOfertaDinamico(){
    let id = this.id
    let idProducto = id.substring(0, id.indexOf("-"))
    sistema.modificarOferta(idProducto);
    mostrarTablaDeTodosLosProductos();
}

function quitarOfertaDinamico(){
    let id = this.id
    let idProducto = id.substring(0, id.indexOf("-"))
    sistema.modificarOferta(idProducto);
    mostrarTablaDeTodosLosProductos();
}

// Mostrar datos
function mostrarTabla(lista){
    let texto = "";

    for (let I = 0; I < lista.length; i++) {
        let objCompra = lista[i];
        texto += `<tr>
                    <td>${objCompra.producto.nombre}</td>
                    <td>${objCompra.montoTotal().toFixed(2)}</td>
                    <td>${objCompra.cantidad}</td>
                    <td>${objCompra.estado}</td>`
        if(objCompra.estado === "Pendiente"){
            texto += `<td><input type="button" value="Cancelar"
id="${objCompra.id}-estadoCompra" class="cancelar"></td>`
        }

    }
    document.querySelector("#tablaDeMisCompras").innerHTML = texto;

    let listaDeBotonesParaCancelar =
document.querySelectorAll(".cancelar");
    for (let I = 0; I < listaDeBotonesParaCancelar.length; i++) {
        let botonActual = listaDeBotonesParaCancelar[i];
        botonActual.addEventListener("click", cancelarCompraDinamico);
    }
}

```

```

}

function mostrarTablaDeAdministrador(lista){
    let texto = "";

    for (let I = 0; I < lista.length; i++) {
        let objCompra = lista[i];
        texto += `<tr>
                    <td>${objCompra.comprador.userName}</td>
                    <td>${objCompra.producto.nombre}</td>
                    <td>${objCompra.montoTotal().toFixed(2)}</td>
                    <td>${objCompra.cantidad}</td>
                    <td>${objCompra.estado}</td>`
        if(objCompra.estado === "Pendiente"){
            texto += `<td><input type="button" value="Aprobar"
id="${objCompra.id}-estadoCompra" class="aprobar"></td>`
        }

    }
    document.querySelector("#tablaDeComprasDeUsuarios").innerHTML =
    texto;

    let listaDeBotonesParaAprobar =
    document.querySelectorAll(".aprobar");
    for (let I = 0; I < listaDeBotonesParaAprobar.length; i++) {
        let botonActual = listaDeBotonesParaAprobar[i];
        botonActual.addEventListener("click", aprobarCompraDinamico);
    }
}

function reestablecerInputs(){
    document.querySelector("#checkProdEnOferta").checked = false;
    document.querySelector("#cantProducto").value = "";
    document.querySelector("#radioFiltrarMisComprasTodas").checked =
    true;
    document.querySelector("#radioFiltrarComprasDeUsuarioTodas").checked
    = true;
}

function mostrarYOCultarPanel(idPanel) {
    document.querySelector("#sectionInicial").style.display = "none";
    document.querySelector("#sectionComprador").style.display = "none";
    document.querySelector("#sectionAdministrador").style.display =
    "none";
    document.querySelector(idPanel).style.display = "block";
}

```

```
// Algoritmo de Luhn

function algoritmoLuhn(pNumero) {
  /*Se óley iterando numero a numero, desde el final del string hasta
  el primer imágenes, se estarán
  sumando y sustituyendo por duplicado cuando sea par, ya que sería el
  segundo nro. */
  let suma = 0;
  let digitoVerificadorX = Number(pNumero.charAt(pNumero.length - 1));
  let contador = 0; //para saber cuando estamos en los segundos, lo
  pares.
  let haynro = true;
  let I = pNumero.length - 2; //el penúltimo.

  //Mientras los óley r sea mayor o igual a 0 se óley tomando cada
  imágenes
  while (i >= 0 && haynro) {
    //Obtener el numero
    let caracter = pNumero.charAt(i);
    //Valida que el número sea válido
    if (!isNaN(caracter)) {
      let num = Number(caracter);
      //Duplicando cada segundo dígito
      if (contador % 2 == 0) {
        num = duplicarPar(num); //porque si es mayor a 9 se deben sumar.
      }
      suma += num;
    } else {
      haynro = false;
    }
    i--;
    contador++;
  }
  let digitoVerificadorValido = checkDigito(suma, digitoVerificadorX);
  let modulodelasumaValiado = checkModulo(suma, digitoVerificadorX);
  return digitoVerificadorValido && modulodelasumaValiado;
}

function duplicarPar(pNum) {
  pNum = pNum * 2;
  if (pNum > 9) {
    /*Si el resultado del multiplicación es mayor a 9 entonces lo
    descomponemos y sumamos.
    Como el numero óle x>=10 && x<=19
    Entonces es 1+(num % 10) 1 más el resto de dividir entre 10.*/
  }
}
```



```

    pNum = 1 + (pNum % 10);
  }
  return pNum;
}

function checkDigito(pSuma, pDigito) {
  /* 1. Calcular la suma de los dígitos (67).
  2. Multiplicar por 9 (603).
  3. Tomar el último dígito (3).
  4. El resultado es el dígito de chequeo.*/
  let total = 9 * pSuma;
  let ultimoNro = total % 10
  return ultimoNro === pDigito;
}

function checkModulo(pSuma, pDigito) {
  /*
  Si el total del módulo 10 es igual a 0 (si el total termina en cero),
  entonces el número es válido
  de acuerdo con la fórmula Luhn, de lo contrario no es válido.
  */
  let total = pSuma + pDigito;
  let validacionFinal = false;
  if (total % 10 === 0 && total !== 0) {
    validacionFinal = true;
  }
  return validacionFinal;
}

```

2.3.- Clases.js

```

class Sistema{
  constructor(){
    this.listaDeCompradores = [];
    this.listaDeAdministradores = [];
    this.listaDeProductos = [];
    this.listaDeCompras = [];
    this.usuarioLogueado = null;
  }

  // Precarga de datos
  precargaDeDatos(){
    // Precarga de compradores
    this.listaDeCompradores.push(new Comprador("Nahuel", "Acosta",
    "nahuelinho", "Goat33", "4506-1234-1891-2024", "132"))
  }
}

```

```

        this.listaDeCompradores.push(new Comprador("Kylian", "Mbappe",
"madridista", "Madrid7", "1111-1111-1111-1111", "123"))
        this.listaDeCompradores.push(new Comprador("Leo", "Fernandez",
"elmanyá", "Peñarol1891", "1111-1111-1111-1891", "111"))
        this.listaDeCompradores.push(new Comprador("Lionel", "Messi",
"elmessias", "Leo10", "1888-1111-2011-1111", "777"))
        this.listaDeCompradores.push(new Comprador("Cristiano",
"Ronaldo", "comandante", "CRis7", "1111-2018-1111-2017", "123"))

// Precarga de Administradores
this.listaDeAdministradores.push(new Administrador("Sebastian",
"Hohl", "sebahohl", "Sh1891"))
this.listaDeAdministradores.push(new Administrador("Matias",
"Oreiro", "matioreiro", "Sh1891"))
this.listaDeAdministradores.push(new Administrador("Guillermo",
"Vieira", "giveira", "Idolo2024"))
this.listaDeAdministradores.push(new Administrador("Giancarlo",
"Federico", "óley rlo", "Tipazo11"))
this.listaDeAdministradores.push(new Administrador("Diego",
"Aguirre", "lafiera", "Lider1891"))

// Precarga de productos
this.listaDeProductos.push(new Producto("Fútbol",
"imágenes/futbol.jpg", "Una pelota de futbol Nike", 14.99, 50))
this.listaDeProductos.push(new Producto("Basket",
"imágenes/basket.jpg", "Una pelota de basketball genérica", 12.99, 15))
this.listaDeProductos.push(new Producto("Voley",
"imágenes/óley.jpg", "Una pelota de voleyball Mikasa", 10.99, 11))
this.listaDeProductos.push(new Producto("Golf",
"imágenes/golf.avif", "Una pelota de golf Inesis", 24.99, 60))
this.listaDeProductos.push(new Producto("Bowling",
"imágenes/bowling.webp", "Una pelota de bolos azul", 89.99, 44))
this.listaDeProductos.push(new Producto("Tenis",
"imágenes/tenis.webp", "Una pelota de tenis Wilson", 9.99, 70))
this.listaDeProductos.push(new Producto("Ping Pong",
"imágenes/pingpong.webp", "Una caja de 6 pelotas de ping pong", 6.99,
13))
this.listaDeProductos.push(new Producto("Futsal",
"imágenes/futsal.webp", "Una pelota de futsal Mikasa", 11.99, 60))
this.listaDeProductos.push(new Producto("Padel",
"imágenes/padel.webp", "Una pelota de padel profesional", 19.99, 5))
this.listaDeProductos.push(new Producto("Bolas de pool",
"imágenes/pool.webp", "Un juego de 16 bolas de pool", 149.99, 8))
this.listaDeProductos[6].estado = false;

```

```

        // Precarga de compras
        this.listaDeCompras.push(new Compra(this.listaDeCompradores[0],
this.listaDeProductos[0], 5))
        this.listaDeCompras.push(new Compra(this.listaDeCompradores[1],
this.listaDeProductos[1], 2))
        this.listaDeCompras.push(new Compra(this.listaDeCompradores[2],
this.listaDeProductos[4], 6))
        this.listaDeCompras.push(new Compra(this.listaDeCompradores[3],
this.listaDeProductos[6], 8))
        this.listaDeCompras.push(new Compra(this.listaDeCompradores[4],
this.listaDeProductos[8], 10))
    }

    //Validaciones
    validarInicioSesionUsuario( óley r1, password){
        let resp = false;
        óley r1 = óley r1.toUpperCase();
        for (let I = 0; I < this.listaDeCompradores.length; i++) {
            let objActual = this.listaDeCompradores[i];
            let userActual = objActual.userName;
            let passActual = objActual.password;
            if( óley r1 == userActual.toUpperCase() && password ==
passActual){
                resp = true
                this.usuarioLogueado = objActual
            }
        }
        return resp;
    }

    validarInicioSesionAdministrador( óley r1, password){
        let resp = false;
        óley r1 = óley r1.toUpperCase();
        for (let I = 0; I < this.listaDeAdministradores.length; i++) {
            let objActual = this.listaDeAdministradores[i];
            let userActual = objActual.userName;
            let passActual = objActual.password;
            if( óley r1 == userActual.toUpperCase() && password ==
passActual){
                resp = true;
                this.usuarioLogueado = objActual;
            }
        }
        return resp;
    }
}

```

```

    cierreDeSesion(){
        this.usuarioLogueado = null;
    }

    existeComprador(userName){
        let existe = false;
        óley r1 = óley r1.toUpperCase();
        for (let I = 0; I < this.listaDeCompradores.length; i++) {
            let objCompradorActual = this.listaDeCompradores[i];
            if(objCompradorActual.userName.toUpperCase() == óley r1){
                existe = true;
            }
        }
        return existe;
    }

    existeAdministrador(userName){
        let existe = false;
        óley r1 = óley r1.toUpperCase();
        for (let I = 0; I < this.listaDeAdministradores.length; i++) {
            let objCompradorActual = this.listaDeAdministradores[i];
            if(objCompradorActual.userName.toUpperCase() == userName){
                existe = true;
            }
        }
        return existe;
    }

    //Acciones del Usuario
    agregarComprador(nombre, apellido, userName, password,
numeroTarjetaDeCredito, cvcTarjetaDeCredito){
        let objComprador = new Comprador(nombre, apellido, userName,
password, numeroTarjetaDeCredito, cvcTarjetaDeCredito)
        this.listaDeCompradores.push(objComprador)
    }

    comprarProducto(idProducto, cantidad){
        let producto = this.obtenerProductoPorID(idProducto);
        let compra = new Compra(this.usuarioLogueado, producto,
cantidad);
        this.listaDeCompras.push(compra);
    }

    cancelarCompra(idProducto){
        let sigo = true;
        for (let I = 0; I < this.listaDeCompras.length; i++) {

```

```

        let objProducto = this.listaDeCompras[i];
        if(objProducto.id === idProducto){
            objProducto.estado = "Cancelada";
            sigo = false;
        }
    }
}

calcularMontoDeComprasAprobadas(){
    let lista = sistema.obtenerMisCompras("Aprobada");
    let montoFinal = 0;

    for (let I = 0; I < lista.length; i++) {
        let objCompra = lista[i];
        montoFinal += parseFloat(objCompra.montoTotal());
    }

    return montoFinal;
}

//Acciones del Administrador
procesarCompra(idCompra){
    let compra = this.obtenerCompraPorID(idCompra);
    let ok = false;
    if( !compra.producto.estado || compra.producto.stock <
compra.cantidad || compra.montoTotal() > compra.comprador.saldo ){
        compra.estado = "Cancelada";
    }else{
        compra.producto.stock -= compra.cantidad;
        compra.producto.cantidadDeVendidos += compra.cantidad;
        compra.comprador.saldo -= compra.montoTotal();
        compra.estado = "Aprobada";
        if(compra.producto.stock == 0){
            compra.producto.estado = false;
        }
        ok = true
    }
    return ok;
}

agregarProducto(nombre, precio, óley rlo n, imagen, stock){
    let objProducto = new Producto(nombre, imagen, óley rlo n,
precio, stock);
    this.listaDeProductos.push(objProducto);
}

```

```

actualizarStock(idProducto, nuevoStock){
    let óley r = this.obtenerProductoPorID(idProducto);
    óley r.stock = nuevoStock;
    if( óley r.stock == 0){
        producto.estado = false;
    }

    return nuevoStock
}

modificarEstado(idProducto){
    let producto = this.obtenerProductoPorID(idProducto);
    if( óley r.stock == 0){

    }else{
        if( óley r.estado == false){
            producto.estado = true;
        }else{
            producto.estado = false;
        }
    }
}

modificarOferta(idProducto){
    let producto = this.obtenerProductoPorID(idProducto);
    if (producto.oferta == false) {
        producto.oferta = true;
    }else{
        producto.oferta = false;
    }
}

//Obtener Datos
obtenerListadoDeTodosLosProductos(){
    return this.listaDeProductos;
}

obtenerListadoDeProductos(){
    let lista = [];
    for (let I = 0; I < this.listaDeProductos.length; i++) {
        let objProducto = this.listaDeProductos[i];
        if(objProducto.estado){
            lista.push(objProducto);
        }
    }
    return lista;
}

```

```

    obtenerProductoPorID(idProducto){
        for (let I = 0; I < this.listaDeProductos.length; i++) {
            let objProducto = this.listaDeProductos[i];
            if (objProducto.id == idProducto) {
                return objProducto;
            }
        }
    }

    obtenerListadoDeCompras(){
        return this.listaDeCompras;
    }

    obtenerCompraPorID(idCompra){
        for (let I = 0; I < this.listaDeCompras.length; i++) {
            let objCompra = this.listaDeCompras[i];
            if(objCompra.id == idCompra){
                return objCompra;
            }
        }
    }

    obtenerMisCompras(estado){
        let lista = [];
        for (let I = 0; I < this.listaDeCompras.length; i++) {
            let objCompra = this.listaDeCompras[i];
            if(objCompra.comprador.userName ===
this.usuarioLogueado.userName && estado == "Aprobada" &&
objCompra.estaAprobada()){
                lista.push(objCompra);
            }else if(objCompra.comprador.userName ===
this.usuarioLogueado.userName && estado == "Pendiente" &&
objCompra.estaPendiente()){
                lista.push(objCompra);
            }else if(objCompra.comprador.userName ===
this.usuarioLogueado.userName && estado == "Cancelada" &&
objCompra.estaCancelada()){
                lista.push(objCompra);
            }else if(objCompra.comprador.userName ===
this.usuarioLogueado.userName && estado == "Todas"){
                lista.push(objCompra);
            }
        }
        return lista
    }

```

```

    }

    obtenerComprasDeUsuario(estado){
        let lista = [];
        for (let I = 0; I < this.listaDeCompras.length; i++) {
            let objCompra = this.listaDeCompras[i];
            if(estado == "Aprobada" && objCompra.estaAprobada()){
                lista.push(objCompra);
            }else if(estado == "Pendiente" && objCompra.estaPendiente()){
                lista.push(objCompra);
            }else if(estado == "Cancelada" && objCompra.estaCancelada()){
                lista.push(objCompra);
            }
        }
        return lista;
    }

    obtenerSaldoComprador(){
        for (let I = 0; I < this.listaDeCompradores.length; i++) {
            let objCompradorActual = this.listaDeCompradores[i];
            if(objCompradorActual.userName ==
this.usuarioLogueado.userName){
                return objCompradorActual.obtenerSaldo();
            }
        }
    }

    obtenerMontoTotalDeComprasAprobadas(){
        let lista = this.obtenerComprasDeUsuario("Aprobada");
        let montoFinal = 0;

        for (let I = 0; I < lista.length; i++) {
            let objCompraAprobada = lista[i];
            montoFinal += objCompraAprobada.montoTotal();
        }

        return montoFinal;
    }
}

sigIdCompra = 1;
class Compra{
    constructor(comprador, producto, cantidad){
        this.id = sigIdCompra++;
        this.comprador = comprador;
    }
}

```



```

        this.producto = producto;
        this.cantidad = cantidad;
        this.estado = "Pendiente";
    }

    montoTotal(){
        return this.producto.precio * this.cantidad;
    }

    estaAprobada(){
        return this.estado == "Aprobada";
    }

    estaPendiente(){
        return this.estado == "Pendiente";
    }

    estaCancelada(){
        return this.estado == "Cancelada";
    }
}

let sigIDComprador = 1;
class Comprador{
    constructor(nombre, apellido, userName, password,
numeroTarjetaDeCredito, cvcTarjetaDeCredito){
        this.id = sigIDComprador++;
        this.nombre = nombre;
        this.apellido = apellido;
        this.userName = userName;
        this.password = password;
        this.numeroTarjetaDeCredito = numeroTarjetaDeCredito;
        this.cvcTarjetaDeCredito = cvcTarjetaDeCredito;
        this.saldo = 3000;
    }

    obtenerSaldo(){
        return this.saldo;
    }
}

class Administrador{
    constructor(nombre, apellido, userName, password){
        this.nombre = nombre;
        this.apellido = apellido;
        this.userName = óley rl;
    }
}

```

```

        this.password = password;
    }
}

sigIDProducto = 1;
class Producto{
    constructor(nombre, imagen, óley rlo n, precio, stock){
        this.id = "PROD_ID_" + sigIDProducto++;
        this.nombre = nombre;
        this.imagen = imagen;
        this.descripcion = óley rlo n;
        this.precio = precio;
        this.stock = stock;
        this.cantidadDeVendidos = 0;
        this.oferta = false;
        this.estado = true;
    }
}

```

3.- Listado de información precargada

```

// Precarga de compradores
    this.listaDeCompradores.push(new Comprador("Nahuel", "Acosta",
"nahuelinho", "Goat33", "4506-1234-1891-2024", "132"))
    this.listaDeCompradores.push(new Comprador("Kylilan", "Mbappe",
"madridista", "Madrid7", "1111-1111-1111-1111", "123"))
    this.listaDeCompradores.push(new Comprador("Leo", "Fernandez",
"elmany", "Peñarol1891", "1111-1111-1111-1891", "111"))
    this.listaDeCompradores.push(new Comprador("Lionel", "Messi",
"elmessias", "Leo10", "1888-1111-2011-1111", "777"))
    this.listaDeCompradores.push(new Comprador("Cristiano",
"Ronaldo", "comandante", "CRis7", "1111-2018-1111-2017", "123"))

// Precarga de Administradores
    this.listaDeAdministradores.push(new Administrador("Sebastian",
"Hohl", "sebahohl", "Sh1891"))
    this.listaDeAdministradores.push(new Administrador("Matias",
"Oreiro", "matioreiro", "Mo1891"))
    this.listaDeAdministradores.push(new Administrador("Guillermo",
"Vieira", "giveira", "Idolo2024"))

```

```

        this.listaDeAdministradores.push(new Administrador("Giancarlo",
"Federico", "óley rlo", "Tipazo11"))
        this.listaDeAdministradores.push(new Administrador("Diego",
"Aguirre", "lafiera", "Lider1891"))

// Precarga de productos
        this.listaDeProductos.push(new Producto("Fútbol",
"imágenes/futbol.jpg", "Una pelota de futbol Nike", 14.99, 50))
        this.listaDeProductos.push(new Producto("Basket",
"imágenes/basket.jpg", "Una pelota de basketball genérica", 12.99, 15))
        this.listaDeProductos.push(new Producto("Voley",
"imágenes/óley.jpg", "Una pelota de voleyball Mikasa", 10.99, 11))
        this.listaDeProductos.push(new Producto("Golf",
"imágenes/golf.avif", "Una pelota de golf Inesis", 24.99, 60))
        this.listaDeProductos.push(new Producto("Bowling",
"imágenes/bowling.webp", "Una pelota de bolos azul", 89.99, 44))
        this.listaDeProductos.push(new Producto("Tenis",
"imágenes/tenis.webp", "Una pelota de tenis Wilson", 9.99, 70))
        this.listaDeProductos.push(new Producto("Ping Pong",
"imágenes/pingpong.webp", "Una caja de 6 pelotas de ping pong", 6.99,
13))

        this.listaDeProductos.push(new Producto("Futsal",
"imágenes/futsal.webp", "Una pelota de futsal Mikasa", 11.99, 60))
        this.listaDeProductos.push(new Producto("Padel",
"imágenes/padel.webp", "Una pelota de padel profesional", 19.99, 5))
        this.listaDeProductos.push(new Producto("Bolas de pool",
"imágenes/pool.webp", "Un juego de 16 bolas de pool", 149.99, 8))
        this.listaDeProductos[6].estado = false;

// Precarga de compras
        this.listaDeCompras.push(new Compra(this.listaDeCompradores[0],
this.listaDeProductos[0], 5))
        this.listaDeCompras.push(new Compra(this.listaDeCompradores[1],
this.listaDeProductos[1], 2))
        this.listaDeCompras.push(new Compra(this.listaDeCompradores[2],
this.listaDeProductos[4], 6))
        this.listaDeCompras.push(new Compra(this.listaDeCompradores[3],
this.listaDeProductos[6], 8))
        this.listaDeCompras.push(new Compra(this.listaDeCompradores[4],
this.listaDeProductos[8], 10))

```

4. Testing

4.1. Registro de usuario

Datos ingresados	Resultado esperado	Resultado obtenido
Uno o varios datos están vacíos	No se permite continuar el registro debido a que todos los campos son obligatorios. Se lanza un Alert avisando de la obligatoriedad de todos los campos.	OK
UserName ya registrado. Caso de username: gvieira	No se pudo seguir con el registro porque ya existe un usuario con ese nombre. Se imprime un Alert: Error, verificar datos	OK
UserName disponible. Contraseña no cumple los requisitos. Casos de contraseñas: - 123	No se continua con el registro y en un párrafo debajo del registro se imprime los parámetros de la contraseña.	OK

<p>- ort</p> <p>- ORT</p> <p>- Ort</p> <p>- ORT123</p> <p>- ort123</p>		
<p>Username disponible y contraseña válida.</p> <p>Formato de número de tarjeta no válido.</p> <p>Caso de n° inválido:</p> <p>- 123456</p> <p>- 1234123412341234</p> <p>- 1234 1234 1234 1234</p>	<p>Se valida que la tarjeta sea ingresada con el formato válido (XXXX-XXXX-XXXX-XXXX) con los 16 números de la tarjeta y los guiones que separan en grupos de 4 el número. En caso de no cumplir con el formato no se sigue con el registro y se ejecuta un alert diciendo que el formato de la tarjeta no es válido.</p>	OK
	<p>Se ejecuta el algoritmo de Luhn con el número de tarjeta ingresado, y si no lo valida no se sigue el registro y se lanza un Alert diciendo</p>	OK

<p>Username disponible, contraseña y formato del n° de tarjeta válido.</p> <p>Número de tarjeta no pasa el algoritmo de Luhn</p> <p>Caso de n° de tarjeta:</p> <p>1234-1234-1234-1234</p>	<p>que no se pudo verificar la tarjeta.</p>	
<p>Username disponible, contraseña válida, tarjeta válida y verificada por el algoritmo de Luhn.</p> <p>El CVC es menor que 100 o mayor que 999</p> <p>Casos de CVC:</p> <p>-50</p> <p>-1891</p>	<p>El registro no pudo seguir adelante debido a que el CVC debe tener 3 cifras de forma obligatoria, cualquier número con menos o más cifras que 3 es incorrecto. Se muestra un Alert indicando que se ingrese un CVC válido.</p>	<p>OK</p>
<p>Todos los datos ingresados son válidos</p> <p>Nombre: José</p> <p>Apellido: Artigas</p>	<p>Se valida todo el registro y se lanza la función para registrar, lo que ingresa al usuario en la lista de compradores, se limpia el formulario de registro y se</p>	<p>OK</p>

Username: elprocer Contraseña: Uru2024 N° de tarjeta: 7343-8884-0385-8854 CVC: 753	lanza un alert especificando el Registro con éxito	
---	--	--

4.2. Sección Administrador

Datos ingresados	Resultado esperado	Resultado obtenido
Aprobar compra del comprador mientras el producto tiene estado false (Pausado) Caso de producto pausado: ID compra: 4 (elmessias, Ping Pong)	Se cancela la compra debido al estado false (pausado), y no se hace ningún descuento.	OK
Aprobar la compra del comprador mientras el monto total de la compra	Se cancela la compra debido a la falta de saldo. No se realiza ningún descuento.	OK

<p>es mayor al saldo disponible.</p> <p>Caso del saldo menor al monto total de la compra:</p> <p>ID compra: 3</p> <p>(elmanyá, Bowling)</p>		
<p>Aprobar la compra del comprador mientras la cantidad de producto comprado es mayor al stock disponible de ese producto.</p> <p>Caso del stock menor a la cantidad de la compra:</p> <p>ID compra: 5</p> <p>(elcomandante, Padel)</p>	<p>El stock del producto no es suficiente para la cantidad que quiere comprar el comprador, se cancela la compra y no se realizan descuentos.</p>	OK
<p>Aprobar compra del comprador</p>	<p>La compra pasa a estado aprobada, se descuenta del saldo del comprador y el stock del producto.</p>	OK

Al crear un producto uno o varios datos están vacíos	No se crea el producto y se lanza en un párrafo que todos los campos son obligatorios	OK
<p>Al crear un producto, se ingresan todos los datos, pero el precio y/o el stock son 0.</p> <p>Caso de producto creado:</p> <p>1)</p> <p>Nombre: Beisbol</p> <p>Precio: 0</p> <p>Descripción: Pelota de beisbol</p> <p>Imagen: Pelota de beisbol</p> <p>Stock: 0</p> <p>2)</p> <p>Nombre: Beisbol</p> <p>Precio: 100</p> <p>Descripción: Pelota de beisbol</p> <p>Imagen: Pelota de beisbol</p> <p>Stock: 0</p>	No se crea el producto y en un párrafo se muestra que el precio y el stock deben de ser mayores a 0.	OK

<p>3)</p> <p>Nombre: Beisbol</p> <p>Precio: 0</p> <p>Descripción: Pelota de beisbol</p> <p>Imagen: Pelota de beisbol</p> <p>Stock: 100</p>		
<p>Al crear un producto con todos los campos válidos y con el precio y stock mayores a 0.</p> <p>Caso de crear producto:</p> <p>Nombre: Beisbol</p> <p>Precio: 50</p> <p>Descripción: Pelota de beisbol</p> <p>Imagen: Pelota de beisbol</p> <p>Stock: 100</p>	<p>Se manda un alert mostrando que el producto fue creado con éxito y se ingresa en la lista de productos el producto creado.</p>	<p>OK</p>

<p>Se modifica el stock y no es un stock válido.</p> <p>Casos de stock inválido:</p> <ul style="list-style-type: none"> - (vacío) - -100 	<p>Se lanza un alert diciendo que se ingrese un stock válido y no se modifica el stock</p>	<p>OK</p>
<p>Se modifica el stock y es un stock válido igual a 0</p>	<p>Se modifica con éxito el stock, se muestra en la tabla su nuevo stock y su estado cambia a false (pausado).</p>	<p>OK</p>
<p>Se modifica el stock y es un stock válido mayor a 0</p> <p>Casos de stock válido:</p> <ul style="list-style-type: none"> - 1 - 100 - 256 	<p>Se modifica con éxito el stock y se muestra en la tabla su nuevo valor de stock.</p>	<p>OK</p>

Se pulsa el botón de modificar estado.	Se modifica el estado del producto de true a false o viceversa. En caso de que el stock sea 0 no se puede activar	OK
Se pulsa el botón de modificar oferta.	Se modifica la oferta del producto de true a false y viceversa, indistintamente del estado o del stock del producto.	OK
Cierre de sesión correcto	Al apretar el botón de cierre de sesión se borra el usuario logueado y se oculta el section administrador y se muestra la sección para iniciar sesión y registrarse.	OK

4.3. Sección comprador

Datos ingresados	Resultado esperado	Resultado obtenido
Checkbox de solo ofertas	Muestra en el select de productos únicamente los productos en oferta	OK
	Añade a la lista de compras el producto y la cantidad del mismo producto que se	

<p>Cantidad de productos que el usuario quiere comprar.</p> <p>Casos de cantidades válidas:</p> <p>-10</p> <p>-100</p> <p>Caso de cantidades inválidas:</p> <p>- 0</p> <p>- -100</p>	<p>compraron y se envía un alert diciendo que el administrador debe aprobar su compra. En caso de que la cantidad sea menor o igual a 0, se considera inválida y se envía un alert diciendo que se debe ingresar una cantidad</p>	<p>OK</p>
<p>Cancelar compra en caso de que esté pendiente</p>	<p>Al apretar el botón de cancelar compra, que debe mostrarse sólo en las compras pendientes, la compra pasa a estado “Cancelada” y no puede ser aprobada. No se realizan descuento alguno.</p>	<p>OK</p>
<p>Cierre de sesión correcto</p>	<p>Al apretar el botón de cierre de sesión se borra el usuario logueado y se oculta el section comprador y se muestra la sección para iniciar sesión y registrarse.</p>	<p>OK</p>