

Universidad ORT Uruguay
Facultad de Ingeniería
Escuela de Tecnología

OBLIGATORIO PROGRAMACIÓN 1

DOCUMENTO DE ANÁLISIS



Martín Etchebarne – 325175



Martin Leib – 323250

M1D

Docente: Andres de Sosa

Analista Programador / 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 | 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 – Inicio de sesión | 8 |
| 2.2.1 | Acceso | 8 |
| 2.2.2 | Descripción | 8 |
| 2.2.3 | Interfaz de usuario | 8 |
| 2.2.4 | Validaciones | 8 |
| 2.3 | F03 – Compra de productos | 9 |
| 2.3.1 | Acceso | 9 |
| 2.3.2 | Descripción | 9 |
| 2.3.3 | Interfaz de usuario | 9 |
| 2.3.4 | Validaciones | 10 |
| 2.4 | F04 – Listado de Compras | 11 |
| 2.4.1 | Acceso | 11 |
| 2.4.2 | Descripción | 11 |
| 2.4.3 | Interfaz de usuario | 11 |
| 2.4.4 | Validaciones | 12 |
| 2.5 | F05 – Productos en Oferta | 13 |
| 2.5.1 | Acceso | 13 |

| | | |
|--------|---|----|
| 2.5.2 | Descripción | 13 |
| 2.5.3 | Interfaz de usuario | 13 |
| 2.5.4 | Validaciones | 13 |
| 2.6 | F06 – Ingresar al sistema - <i>como administrador</i> | 14 |
| 2.6.1 | Acceso | 14 |
| 2.6.2 | Descripción | 14 |
| 2.6.3 | Interfaz de usuario | 14 |
| 2.6.4 | Validaciones | 14 |
| 2.7 | F07 – Listado y aprobación de ventas | 15 |
| 2.7.1 | Acceso | 15 |
| 2.7.2 | Descripción | 15 |
| 2.7.3 | Interfaz de usuario | 15 |
| 2.7.4 | Validaciones | 16 |
| 2.8 | F08 – Crear productos | 17 |
| 2.8.1 | Acceso | 17 |
| 2.8.2 | Descripción | 17 |
| 2.8.3 | Interfaz de usuario | 17 |
| 2.8.4 | Validaciones | 18 |
| 2.9 | F09 – Administrar productos | 19 |
| 2.9.1 | Acceso | 19 |
| 2.9.2 | Descripción | 19 |
| 2.9.3 | Interfaz de usuario | 19 |
| 2.9.4 | Validaciones | 19 |
| 2.10 | F10 – Informe ganancias | 20 |
| 2.10.1 | Acceso | 20 |
| 2.10.2 | Descripción | 20 |
| 2.10.3 | Interfaz de usuario | 20 |
| 2.10.4 | Validaciones | 20 |

| | |
|---|----|
| 3. Datos precargados | 22 |
| 3.1 Usuarios | 22 |
| 3.1.1 Tipo administrador | 22 |
| 3.1.2 Tipo comprador | 24 |
| 3.2 Productos en venta | 26 |
| 3.3 Ventas | 30 |
| 4. Código fuente | 32 |
| 4.1 Código HTML | 32 |
| 4.2 Código JS - archivo “app.js” | 37 |
| 4.3 Código JS - archivo “classes.js” | 57 |
| 5. Casos de prueba para funcionalidades | 63 |
| 5.1 Casos asociados al usuario tipo comprador | 63 |
| 5.2 Casos asociados al usuario tipo administrador | 65 |

1. Descripción general del problema a resolver

Se debe crear un sistema de ventas tipo e-commerce que permite a una casa de venta de artículos deportivos vender sus productos de forma online

1.1. Tipos de usuario del sistema

Cliente

Administrador

1.2. Listado de funcionalidades

F01 – Registro – Usuario: Comprador

F02 – Inicio de sesión – Usuario: Comprador

F03 – Compra de productos – Usuario: Comprador

F04 – Listado de compras – Usuario: Comprador

F05 – Ver productos en oferta – Usuario: Comprador

F06 – Inicio de sesión – Usuario: Administrador

F07 – Listado y Aprobación de Ventas – Usuario: Administrador

F08 – Crear productos – Usuario: Administrador

F09 – Administrar productos – Usuario: Administrador

F10 – Informe ganancia – Usuario: 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

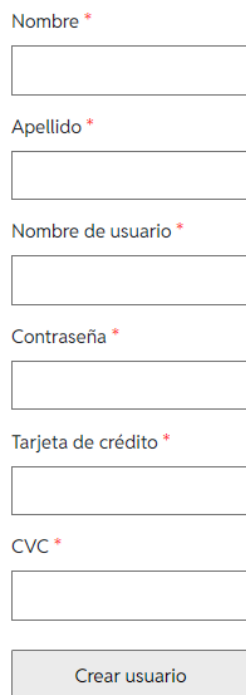
2.1.1 Acceso

Comprador

2.1.2 Descripción

Permite que se creen usuarios de tipo comprador. El usuario deberá de ingresar un nombre, apellido, nombre de usuario, contraseña, número de tarjeta de crédito y CVC (código de verificación).

2.1.3 Interfaz de usuario



Formulario de registro de usuario con los siguientes campos:

- Nombre *
- Apellido *
- Nombre de usuario *
- Contraseña *
- Tarjeta de crédito *
- CVC *
- Botón: Crear usuario

2.1.4 Validaciones

- Se valida que el nombre de usuario sea case insensitive (es decir, no importan las mayúsculas ni las minúsculas) y que sea único en el sistema. No pueden haber dos usuarios registrados bajo el mismo nombre. En caso de que el nombre dado por el usuario coincida con otro previamente registrado se mostrará un error: 'Ese nombre ya está en uso.'
- Que la contraseña contenga por lo menos 5 caracteres de los cuales uno tiene que ser una letra mayúscula, uno una letra minúscula y uno un número. Los datos serán procesados y validados uno por uno de manera tal que si alguna de las condiciones no se cumplen se muestre un error específico: 'Tu contraseña debe contener al menos una mayúscula', 'Tu contraseña debe contener al menos una minúscula', 'Tu contraseña debe contener al menos un número' o 'Tu contraseña debe contener al menos 5 caracteres'.
- El número de tarjeta de crédito tenga el formato correcto (WWWW-XXXX-YYYY-ZZZZ, 4 grupos de 4 dígitos separados por guiones) y que el número se valide con el algoritmo de Luhn. En caso de que la estructura sea incorrecta o que el número no se valide se mostrará un error: 'El número de tarjeta es incorrecto'
- El CVC sea un número de 3 dígitos. En caso de que el CVC tenga menos o más de 3 dígitos se mostrará un error: 'El CVC tiene que tener 3 dígitos'.
- Si alguno de los campos es vacío, se muestra un mensaje de error.

2.2 F02 – Inicio de Sesión

2.2.1 Acceso

Comprador

2.2.2 Descripción

Los usuarios de tipo comprador deberán iniciar sesión en el sistema para poder utilizarlo. El usuario tendrá que ingresar su Nombre de Usuario y Contraseña.

2.2.3 Interfaz de usuario

Nombre de usuario

Contraseña

Iniciar sesión

No tenés un usuario? [Registrate](#)

2.2.4 Validaciones

- El nombre debe de ser case insensitive (no importan las mayúsculas ni las minúsculas) y la contraseña debe ser case sensitive (importan las mayúsculas y las minúsculas).

- El usuario debe existir y la contraseña debe ser válida. En caso de que las credenciales ingresadas no se encuentren en el sistema se mostrará un error: 'Los datos no son correctos'.

2.3 F03 – Compra de Productos

2.3.1 Acceso

Comprador

2.3.2 Descripción


Luego de haber iniciado sesión, los usuarios podrán ver el listado de productos para comprar, pudiendo ver los siguientes detalles:

1. nombre,
2. descripción,
3. precio por unidad,
4. si se encuentra en oferta o no,
5. imagen ilustrativa,
6. cantidad de stock disponible.

El usuario podrá observar si el producto está disponible. En caso de que el producto esté disponible, podrá comprarlo. Si el producto está pausado, el comprador no podrá verlo.

Luego de la elección del producto, podrá seleccionar cuantas unidades desea, al presionar el botón ‘comprar’ se efectúa la orden de compra. La compra pasa a un estado ‘pendiente’ en hasta que un usuario administrador la confirme.

2.3.3 Interfaz de usuario

| Nombre | Precio | Descripción | Imagen | Stock | Unidades | Comprar |
|------------------|--------|--------------------------|---|-------|---|--|
| Air Zoom Pegasus | 100 | Descripción del producto |  | 7 | <input type="text" value="Cantidad de unidades"/> | <input type="button" value="Comprar"/> |

2.3.4 Validaciones

- Se valida si el producto está en oferta o no. En caso de que esté en oferta, se mostrará el precio descontado al cliente.
- El producto debe estar en estado 'publicado', contando con el stock suficiente como para realizar la compra. En caso de que no haya stock suficiente se mostrará el mensaje de error: 'no hay suficiente stock como para procesar tu pedido'. En caso de que el producto no esté en estado publicado, es decir, esté en estado pausado, el usuario no podrá ver el producto y no podrá comprarlo.

2.4 F04 – Listado de Compras

2.4.1 Acceso

Comprador

2.4.2 Descripción

El usuario podrá visualizar una lista con todas los pedidos de compra que haya realizado. Podrá ver la siguiente información:

1. nombre del producto,
2. cantidad de unidades compradas
3. monto total de la compra
4. estado de la compra [aprobado, pendiente, cancelado]
5. botón que permita cancelar compras [únicamente aquellas en estado 'pendiente']
6. monto total de todas las compras aprobadas
7. saldo disponible
 - a. filtro [que permita mostrar o todas las compras, o aquellas aprobadas, canceladas o pendientes]

2.4.3 Interfaz de usuario

| Nombre producto | Unidades | Precio | Estado compra | Acción |
|-------------------------|----------|--------|---------------|---|
| Air Zoom Pegasus | 2 | 200 | Pendiente | <input type="button" value="Cancelar"/> |
| Air Zoom Pegasus Shield | 2 | 200 | Pendiente | <input type="button" value="Cancelar"/> |
| Quest 5 | 2 | 300 | Pendiente | <input type="button" value="Cancelar"/> |
| Air Max | 4 | 800 | Pendiente | <input type="button" value="Cancelar"/> |

2.4.4 Validaciones

Se deberá de validar que solo se pueda cancelar una compra si esta se encuentra en estado 'pendiente'. El botón de cancelación solo es visible en la UI (interfaz de usuario) si el estado de la orden de compra es 'pendiente'. Al momento de presionar el botón de cancelar deberá aparecer un mensaje tipo pop-up (ventana emergente) pidiéndole al usuario la confirmación de la cancelación de la misma.

2.5 F05 – Productos en Oferta


2.5.1 Acceso

Comprador

2.5.2 Descripción

El comprador contará con una vista en la que pueda observar los productos en oferta disponibles, también tendrá la posibilidad de comprarlos.

2.5.3 Interfaz de usuario

| Nombre | Precio | Descripción | Imagen | Stock | Unidades | Comprar |
|------------------|------------------|--------------------------|---|-------|---|--|
| Air Zoom Pegasus | 100 En Oferta | Descripción del producto |  | 7 | <input type="text" value="Cantidad de unidades"/> | <input type="button" value="Comprar"/> |

2.5.4 Validaciones

Se deberá validar qué productos están en oferta. Aquellos que estén en oferta van a ser mostrados en la vista llamada ‘productos en oferta’. Si no hay ningún producto en oferta se mostrará ‘No hay productos en oferta en este momento’.

2.6 F06 – Ingresar al Sistema

2.6.1 Acceso

Administrador

2.6.2 Descripción

Los usuarios con perfil administrador podrán administrar los productos y compras de los usuarios perfil comprador.

2.6.3 Interfaz de usuario

Nombre de usuario

admin

Contraseña

.....

Iniciar sesión

No tenés un usuario? [Registrate](#)

2.6.4 Validaciones

Se valida que el nombre de usuario y contraseña sean correctos. En caso de que las credenciales ingresadas no sean encontradas en el sistema se mostrará un error: ‘Las credenciales ingresadas no son correctas’.

2.7 F07 – Listado y aprobación de ventas

2.7.1 Acceso

Administrador

2.7.2 Descripción

Los administradores tendrán una vista donde se observarán las compras del usuario, mostrando tres tipos diferentes de listas:

1. Lista de compras pendientes
2. Lista de compras canceladas
3. Lista de compras aprobadas

2.7.3 Interfaz de usuario

| Nombre comprador | Balance comprador | Unidades | Nombre producto | Stock disponible | Estado compra | Acción |
|------------------|-------------------|----------|--------------------|------------------|---------------|--|
| admin5 | 3000 USD | 5 | Air Zoom Pegasus | 7 | Aprobada | <div>Aprobar</div> <div>Cancelar</div> |
| admin3 | 3000 USD | 8 | Air Zoom Structure | 8 | Cancelada | <div>Aprobar</div> <div>Cancelar</div> |
| admin2 | 3000 USD | 2 | Air Zoom Pegasus | 7 | Pendiente | <div>Aprobar</div> <div>Cancelar</div> |

2.7.4 Validaciones

La orden de compra solo podrá ser confirmada si el comprador tiene saldo suficiente para realizar la compra, si hay stock suficiente y si el estado del producto es 'activo' al momento de aprobar la compra - es posible que el usuario haya efectuado su pedido cuando el producto estaba activo y a la hora de ser aprobada la compra por un administrador el estado del producto haya cambiado a 'pausado'. En caso de que no pueda ser confirmada la compra porque una de estas condiciones no se cumple la compra queda en estado 'cancelada' sin afectar el balance del usuario y el stock del producto.

Al ser aprobada la compra el stock del producto se actualiza al igual que el balance del comprador. Si al aprobar la compra el stock del producto queda en 0 el estado del producto pasa a ser 'pausado'.

Ante cualquier cambio la página se refresca automáticamente.

2.8 F08 – Crear Productos

2.8.1 Acceso

Administrador

2.8.2 Descripción

Los administradores tienen una ventana emergente en la cual pueden crear nuevos productos proveyendo la siguiente información necesaria:

1. nombre del producto
2. precio
3. descripción
4. URL de la imagen del producto
5. cantidad de stock disponible

2.8.3 Interfaz de usuario

| |
|--|
| Nombre del producto |
| Precio del producto |
| Descripción del producto |
| Stock del producto |
| <div>Seleccionar imagen Seleccionar archivo Sin archivos seleccionados</div> |
| Crear producto |

2.8.4 Validaciones

Todos los campos deberán tener su dato correspondiente. En caso de que un dato sea invalido, como en el caso de que se ingrese un número en un campo donde solo puede haber texto o viceversa se procesa dato por dato y se muestra en qué campo está el error. Por ejemplo: 'la cantidad de stock tiene que ser un número'.

Se valida que el precio y el stock sean mayores a 0 y se muestra un error en caso de que esta condición no se cumpla. El identificador del producto será asignado automáticamente al momento de crear el producto.

Verificar que al crear el producto no sea puesto en oferta automáticamente - el producto podrá ser puesto en oferta luego utilizando la función de Administrar Productos.

2.9 F09 – Administrar Productos

2.9.1 Acceso


Administrador

2.9.2 Descripción

Los usuarios administradores tienen una ventana emergente en la cual pueden modificar un producto pudiendo realizar las siguientes acciones:

1. modificar stock
2. pausar/activar producto
3. habilitar/deshabilitar oferta

2.9.3 Interfaz de usuario

| | | | | | | |
|-------------------------|-----|--------------------------|---|--------|---|--|
| Air Zoom Pegasus Shield | 100 | Descripción del producto |  | Activo | 4 | <div>Cambiar estado</div> <div>Editar producto</div> <div>Oferta</div> |
|-------------------------|-----|--------------------------|---|--------|---|--|

2.9.4 Validaciones

Al momento de modificar el stock, verificar que la cantidad ingresada no sea menor que cero unidades. En caso de que lo sea, cambiar el estado de la publicación (pausado -> activo).

2.10 F10 – Informe de Ganancias

2.10.1 Acceso

Administrador

2.10.2 Descripción

Los usuarios administradores pueden ver la cantidad de unidades vendidas de cada producto y la ganancia total de todas las ventas realizadas.

2.10.3 Interfaz de usuario

| Nombre comprador | Nombre producto | Unidades compradas | Total |
|------------------|------------------|--------------------|---------|
| admin5 | Air Zoom Pegasus | 5 | 500 USD |
| admin2 | Air Zoom Pegasus | 3 | 300 USD |

Ganancias totales: 800 USD

Unidades Compradas: 8

2.10.4 Validaciones

No hay validación alguna.

DATOS PRECARGADOS

Usuarios tipo: administrador

| | |
|------------------------------|-----------------------|
| Nombre | Usuario Administrador |
| Apellido | |
| Nombre de usuario | admin |
| contraseña | admin |
| Número de Tarjeta de Crédito | |
| Código de Seguridad | |
| Tipo de usuario | Administrador |

| | |
|------------------------------|-----------------------|
| Nombre | Usuario Administrador |
| Apellido | |
| Nombre de usuario | admin2 |
| contraseña | admin2 |
| Número de Tarjeta de Crédito | |
| Código de Seguridad | |
| Tipo de usuario | Administrador |

| | |
|------------------------------|-----------------------|
| Nombre | Usuario Administrador |
| Apellido | |
| Nombre de usuario | admin3 |
| contraseña | admin3 |
| Número de Tarjeta de Crédito | |
| Código de Seguridad | |
| Tipo de usuario | Administrador |

| | |
|-------------------------------------|-----------------------|
| Nombre | Usuario Administrador |
| Apellido | |
| Nombre de usuario | admin4 |
| contraseña | admin4 |
| Número de Tarjeta de Crédito | |
| Código de Seguridad | |
| Tipo de usuario | Administrador |

| | |
|-------------------------------------|-----------------------|
| Nombre | Usuario Administrador |
| Apellido | |
| Nombre de usuario | admin5 |
| contraseña | admin5 |
| Número de Tarjeta de Crédito | |
| Código de Seguridad | |
| Tipo de usuario | Administrador |

DATOS PRECARGADOS

Usuarios tipo: Comprador

| | |
|------------------------------|---------------------|
| Nombre | Martin |
| Apellido | Leib |
| Nombre de usuario | mleib |
| contraseña | 123 |
| Número de Tarjeta de Crédito | 3566-0020-2036-0505 |
| Código de Seguridad | 821 |
| Tipo de usuario | Comprador |

| | |
|------------------------------|---------------------|
| Nombre | Martín |
| Apellido | Etchebarne |
| Nombre de usuario | tinchoet |
| contraseña | 123 |
| Número de Tarjeta de Crédito | 4012-8888-8888-1881 |
| Código de Seguridad | 652 |
| Tipo de usuario | Comprador |

| | |
|------------------------------|---------------------|
| Nombre | Juan |
| Apellido | Perez |
| Nombre de usuario | jperez |
| contraseña | 123 |
| Número de Tarjeta de Crédito | 6011-0009-9013-9424 |
| Código de Seguridad | 362 |
| Tipo de usuario | Comprador |

| | |
|-------------------------------------|---------------------|
| Nombre | Carlos |
| Apellido | Rodriguez |
| Nombre de usuario | crodriguez |
| contraseña | 123 |
| Número de Tarjeta de Crédito | 3852-0000-0232-3711 |
| Código de Seguridad | 990 |
| Tipo de usuario | Comprador |

| | |
|-------------------------------------|---------------------|
| Nombre | Marcos |
| Apellido | Gonzalez |
| Nombre de usuario | mgonzalez |
| contraseña | 123 |
| Número de Tarjeta de Crédito | 3787-3449-3671-0002 |
| Código de Seguridad | 119 |
| Tipo de usuario | Comprador |

DATOS PRECARGADOS

Productos en venta

| | |
|---------------------|---|
| Nombre del Producto | Air Zoom Pegasus |
| Precio | 100 |
| Descripción | Descripción del producto |
| Imagen | URL: nike-air-zoom-pegasus.webp |
| Stock | 7 |
| Visible | Si |
| En Oferta | No |

| | |
|---------------------|--|
| Nombre del Producto | Air Zoom Pegasus Shield |
| Precio | 100 |
| Descripción | Descripción del producto |
| Imagen | URL: nike-air-zoom-pegasus-shield.webp |
| Stock | 4 |
| Visible | Si |
| En Oferta | No |

| | |
|---------------------|---|
| Nombre del Producto | Air Zoom Structure |
| Precio | 100 |
| Descripción | Descripción del producto |
| Imagen | URL: nike-air-zoom-structure.webp |
| Stock | 8 |
| Visible | No |
| En Oferta | No |

| | |
|---------------------|---|
| Nombre del Producto | Legend Essential 3 |
| Precio | 200 |
| Descripción | Descripción del producto |
| Imagen | URL: nike-legend-essential-3.webp |
| Stock | 11 |
| Visible | No |
| En Oferta | No |

| | |
|---------------------|--|
| Nombre del Producto | Quest 5 |
| Precio | 150 |
| Descripción | Descripción del producto |
| Imagen | URL: nike-quest-5.webp |
| Stock | 21 |
| Visible | Si |
| En Oferta | Si |

| | |
|---------------------|--------------------------------|
| Nombre del Producto | Air Max |
| Precio | 200 |
| Descripción | Descripción del producto |
| Imagen | URL: nike1.jpg |
| Stock | 15 |
| Visible | Si |
| En Oferta | Si |

| | |
|----------------------------|--------------------------------|
| Nombre del Producto | Air Max 2 |
| Precio | 150 |
| Descripción | Descripción del producto |
| Imagen | URL: nike2.jpg |
| Stock | 5 |
| Visible | No |
| En Oferta | Si |

| | |
|----------------------------|--------------------------------|
| Nombre del Producto | LeBron 15 Low |
| Precio | 250 |
| Descripción | Descripción del producto |
| Imagen | URL: nike3.jpg |
| Stock | 8 |
| Visible | No |
| En Oferta | Si |

| | |
|----------------------------|---------------------------------------|
| Nombre del Producto | Medias |
| Precio | 50 |
| Descripción | Descripción del producto |
| Imagen | URL: medias_nike2.jpg |
| Stock | 15 |
| Visible | Si |
| En Oferta | Si |

| | |
|----------------------------|-----------------------------|
| Nombre del Producto | Medias Negras |
| Precio | 50 |
| Descripción | Descripción del producto |
| Imagen | URL: <code>image.jpg</code> |
| Stock | 10 |
| Visible | Si |
| En Oferta | Si |

DATOS PRECARGADOS

Ventas

| | |
|-----------------------|------------------|
| Usuario del comprador | tinchoet |
| Producto comprado | Air Zoom Pegasus |
| Precio por unidad | 100 |
| Cantidad comprada | 5 |
| Estado | Aprobada |

| | |
|-----------------------|--------------------|
| Usuario del comprador | jperez |
| Producto comprado | Air Zoom Structure |
| Precio por unidad | 100 |
| Cantidad comprada | 8 |
| Estado | Cancelada |

| | |
|-----------------------|------------------|
| Usuario del comprador | crodriguez |
| Producto comprado | Air Zoom Pegasus |
| Precio por unidad | 100 |
| Cantidad comprada | 2 |
| Estado | Pendiente |

| | |
|-----------------------|------------------|
| Usuario del comprador | mgonzales |
| Producto comprado | Air Zoom Pegasus |
| Precio por unidad | 100 |
| Cantidad comprada | 5 |
| Estado | Cancelada |

| | |
|-----------------------|------------------|
| Usuario del comprador | tinchoet |
| Producto comprado | Air Zoom Pegasus |
| Precio por unidad | 100 |
| Cantidad comprada | 3 |
| Estado | Aprobada |

| | |
|-----------------------|-------------------------|
| Usuario del comprador | mleib |
| Producto comprado | Air Zoom Pegasus Shield |
| Precio por unidad | 100 |
| Cantidad comprada | 5 |
| Estado | Pendiente |

Código HTML

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

<head>
  <!-- Meta tags -->
  <meta charset="UTF-8" />
  <meta http-equiv="X-UA-Compatible" content="IE=edge" />
  <meta name="viewport" content="width=device-width, initial-scale=1.0" />
  <meta name="robots" content="index, follow" />
  <meta name="description" content="Tienda online oficial de Nike en Uruguay" />

  <!-- CSS -->
  <link rel="stylesheet"
href="https://cdnjs.cloudflare.com/ajax/libs/bootstrap/5.3.3/css/bootstrap.min.css"

integrity="sha512-jnSuA4Ss2PkkikSOLtYs8BlYIeeIK1h99ty4YfvRPAlzr377vr3CXDb7sb7eEEBYj
DtcYj+AjbH3FLv5uSJuxg==" crossorigin="anonymous" referrerpolicy="no-referrer" />

  <!-- Google Fonts -->
  <link rel="preconnect" href="https://fonts.googleapis.com" />
  <link rel="preconnect" href="https://fonts.gstatic.com" crossorigin />
  <link
href="https://fonts.googleapis.com/css2?family=Reddit+Sans:ital,wght@0,200..900;1,2
00..900&display=swap"
rel="stylesheet" />

  <link rel="stylesheet" href="./css/styles.css">

  <link rel="shortcut icon" href="img/favicon.png" type="image/x-icon">
  <script src="./js/classes.js"></script>
  <script src="./js/app.js"></script>

  <title>Nike Uruguay</title>
</head>

<body class="bg-white text-dark">
  <header>
    <!-- Header: visible al admin -->
    <div id="header-hidden-actions">
      <input type='button' id="header-hidden-logout-button" value="Cerrar Sesión"
class="btn btn-danger lg-3">
      <input type='button' id="header-hidden-create-products-button" value="Crear
Producto"
class="btn btn-dark lg-3">
      <input type='button' id="header-hidden-show-products-button" value="Productos"
class="btn btn-dark lg-3">
      <input type='button' id="header-hidden-show-sales-button" value="Ver ventas"
class="btn btn-dark lg-3">
      <input type='button' id="header-hidden-show-earnings-button" value="Ver
ganancias" class="btn btn-dark lg-3">
    </div>
```



```
<!-- Header: visible al usuario -->
<div id="header-hidden-actions-user">
  <input type='button' id="user-hidden-logout-button" value="Cerrar Sesión"
class="btn btn-danger lg-3">
  <input type='button' id="user-hidden-show-products-button" value="Productos"
class="btn btn-dark lg-3">
  <input type='button' id="user-hidden-show-purchases-button" value="Historial
de compras"
  class="btn btn-dark lg-3">
</div>

<!-- Filtrar ventas: visible al admin -->
<div id="header-sales-actions">
  <select name="sales-actions" id="sales-actions" class="me-2">
    <option value="" selected disabled>Filtrar por estado</option>
    <option value="Ver-Todo">Ver Todo</option>
    <option value="Aprobadas">Aprobadas</option>
    <option value="Canceladas">Canceladas</option>
    <option value="Pendientes">Pendientes</option>
  </select>
</div>

<!-- Flecha para ir hacia atrás y logo de la tienda visibles durante el registro
-->
<div class="arrow-container" id="arrow-container">
  <svg xmlns="http://www.w3.org/2000/svg" height="20" width="17.5" class="mt-5
mx-4" viewBox="0 0 448 512">
    <path
      d="M9.4 233.4c-12.5 12.5-12.5 32.8 0 45.31160 160c12.5 12.5 32.8 12.5 45.3
0s12.5-32.8 0-45.3L109.2 288 416 288c17.7 0 32-14.3 32-32s-14.3-32-32-32L
0L214.6 118.6c12.5-12.5 12.5-32.8 0-45.3s-32.8-12.5-45.3 0l-160 160z" />
  </svg>
</div>
<div class="logo-container" id="logo-container">
  <a href="#"></a>
</div>
</header>

<main>
  <!-- Panel de login -->
  <div class="login-container" id="login-container">
    <form action="#" method="get">
      <div class="login-flex-container">
        <div class="form-group">
          <label for="login-username">Nombre de usuario</label>
          <input type="text" id="login-username" value="admin">
        </div>
        <div class="form-group">
          <label for="login-password">Contraseña</label>
          <input type="password" id="login-password" value="admin">
        </div>
        <input type="button" id="login-button" value="Iniciar sesión">
      </div>
      <span class="me-2">No tenés un usuario?</span><a href="#" id="hideLogin"
class="ms-2">Registrate</a>
```

```

        <p id="login-messages"></p>
    </form>
</div>

<!-- Panel de registro -->
<div class="register-container" id="register-container" style="display: none;">
    <form action="#" method="get" autocomplete="on">
        <div class="register-flex-container">
            <div class="form-group">
                <label class="mt-2 mb-2" for="register-name">Nombre
<span>*</span></label>
                <input type="text" id="register-name" required>
            </div>
            <div class="form-group">
                <label class="mt-2 mb-2" for="register-surname">Apellido
<span>*</span></label>
                <input type="text" id="register-surname" required>
            </div>
            <div class="form-group">
                <label class="mt-2 mb-2" for="register-username">Nombre de usuario
<span>*</span></label>
                <input type="text" id="register-username" required>
            </div>
            <div class="form-group">
                <label class="mt-2 mb-2" for="register-password">Contraseña
<span>*</span></label>
                <input type="password" id="register-password" required>
            </div>
            <div class="form-group">
                <label class="mt-2 mb-2" for="register-card">Tarjeta de crédito
<span>*</span></label>
                <input type="text" id="register-card" required>
            </div>
            <div class="form-group">
                <label class="mt-2 mb-2" for="register-cvc">CVC <span>*</span></label>
                <input type="number" id="register-cvc" maxlength="3" required>
            </div>
            <div class="form-group">
                <input type="button" id="register-button" value="Crear usuario"
required>
            </div>
        </div>
    </form>
    <div class="register-text">
        <p id="register-error-fullName"></p>
        <p id="register-error-username"></p>
        <p id="register-error-password"></p>
        <p id="register-error-card"></p>
        <p id="register-error-cvc"></p>
        <p id="register-messages"></p>
        <a href="#" id="register-success-login"></a>
    </div>
</div>

<!-- Tabla productos: visible al usuario -->
<div class="table-container">
    <div class="hidden-list" id="products-list-user"></div>
</div>

```

```

    <!-- Tabla historial de compras: visible al usuario -->
    <div class="table-container">
        <div class="hidden-list" id="user-purchases-list" style="display:
none;"></div>
    </div>

    <!-- Tabla productos: visible al admin -->
    <div class="table-container">
        <div class="hidden-list" id="products-list">
        </div>
    </div>

    <!-- Tabla editar productos: visible al admin -->
    <div class="edit-product-options" id="edit-product-options" style="display:
none;">
        <input type="text" id="editing-product-name" placeholder="Nombre del
producto"><br>
        <input type="text" id="editing-product-price" placeholder="Precio del
producto"><br>
        <textarea id="editing-product-description" placeholder="Descripción del
producto"></textarea><br>
        <label for="editing-product-image">Imagen del producto</label>
        <input type="file" id="editing-product-image"><br>
        <input type="number" id="editing-product-stock" placeholder="Stock del
producto"><br>
        <label for="editing-product-onSale">En oferta</label>
        <input type="checkbox" id="editing-product-onSale">
        <input type="button" id="finished-editing" value="Guardar cambios">
    </div>

    <!-- Crear productos: visible al admin -->
    <div class="create-products-container" id="create-products-container"
style="display: none;">
        <input type="text" placeholder="Nombre del producto"
id="create-products-product-name">
        <input type="number" placeholder="Precio del producto"
id="create-products-product-price">
        <input type="text" placeholder="Descripción del producto"
id="create-products-product-description">
        <input type="number" placeholder="Stock del producto"
id="create-products-product-stock">
        <label for="create-products-product-image">Seleccionar imagen</label><input
type="file"
        id="create-products-product-image">
        <input type="button" value="Crear producto" id="create-products-button">
        <p id="create-products-message"></p>
    </div>

    <!-- Tabla ventas: visible al admin -->
    <div class="table-container">
        <div class="hidden-list" id="sales-list"></div>
    </div>

    <!-- Tabla ganancias: visible al admin -->
    <div class="table-container" id="earnings-list-and-text">
        <div class="hidden-list" id="earnings-list"></div>
        <p id="total-earnings"></p>
    </div>
</main>

```

```
<!-- Footer -->
<footer class="bg-dark text-white">
  <div class="footer-flex-container">
    
    <p class="mt-3 mb-3 mx-3">
      &copy; 2024 Nike Uruguay - Todos los derechos reservados
    </p>
  </div>
</footer>
</body>

</html>
```

Código JS - app.js

```
// Globals
let productTableVisible = false; // bool que guarda si se está mostrando la tabla
de productos al admin como un true o false
let productTableUserVisible = false; // bool que guarda si se está mostrando la
tabla de productos al usuario como un true o false
let showingAdminOrUsers = false; // bool que guarda si se está mostrando la vista
de administrador o usuario como un true o false
let createProductsVisible = false; // bool que guarda si se está mostrando la
función de crear productos al admin como un true o false

let mainApp = new App();

// Función de inicio
window.addEventListener("load", () => {
  mainApp.preloadUsers();
  mainApp.preloadProducts();
  mainApp.preloadSales();

  // Función de registro y display none al login al hacer click en el botón de
registro
  document.querySelector("#register-button").addEventListener("click",
registerFunction);
  document.querySelector("#hideLogin").addEventListener("click", showRegister);

  // Función de login y display none al registro al hacer click en el botón de login
  document.querySelector("#login-button").addEventListener("click", loginFunction);
  document.querySelector("#arrow-container").addEventListener("click", showLogin);

  // Al finalizar el registro muestra un hipervínculo que te permite redirigirte
para iniciar sesión con la cuenta recién registrada
  document.querySelector("#register-success-login").addEventListener("click",
showLogin);

  // Header: vista administrador
  hideHeaderHiddenActions();

  document.querySelector("#header-hidden-show-products-button").addEventListener("cli
ck", showAndHideProducts);

  document.querySelector("#header-hidden-create-products-button").addEventListener("c
lick", showAndHideCreateProducts);
  document.querySelector("#header-hidden-logout-button").addEventListener("click",
logout); // logout

  document.querySelector("#header-hidden-show-sales-button").addEventListener("click"
, showSales);

  document.querySelector("#header-hidden-show-sales-button").addEventListener("click"
, toggleSalesListDisplay);

  document.querySelector("#header-hidden-show-earnings-button").addEventListener("cli
ck", showEarnings);

  document.querySelector("#header-hidden-show-earnings-button").addEventListener("cli
ck", toggleEarningsDisplay);
```

```

    document.querySelector("#sales-actions").addEventListener("change",
showFilterSales);

    // Header: vista usuario
    hideUserHiddenActions();

document.querySelector("#user-hidden-show-products-button").addEventListener("click
", showAndHideProductsUser);
    document.querySelector("#user-hidden-logout-button").addEventListener("click",
logout); // logout

document.querySelector("#user-hidden-show-purchases-button").addEventListener("clie
k", showUserPurchases);

document.querySelector("#user-hidden-show-purchases-button").addEventListener("clie
k", toggleUserPurchases);


    // Crear producto
    document.querySelector("#create-products-button").addEventListener("click",
createProduct);
});

// Función de login
function showLogin() {
    document.querySelector("#login-container").style.display = "flex"; // Muestra
login
    document.querySelector("#register-container").style.display = "none"; // Display
none al registro
    document.querySelector("#arrow-container").style.display = "none"; // Display none
a la flecha para ir hacia atrás
    document.querySelector("#logo-container").style.display = "none"; // Display none
al logo de la tienda
}

// Función de registro
function showRegister() {
    document.querySelector("#login-container").style.display = "none"; // Display none
al login
    document.querySelector("#register-container").style.display = "flex"; // Muestra
registro
    document.querySelector("#arrow-container").style.display = "block"; // Muestra
flecha para ir hacia atrás
    document.querySelector("#logo-container").style.display = "block"; // Muestra logo
de la tienda
}

// Función de logout
function logout() {
    document.querySelector("#products-list").style.display = "none";
    document.querySelector("#products-list-user").style.display = "none";
    document.querySelector("#login-container").style.display = "flex";
    document.querySelector("#register-container").style.display = "none";
    document.querySelector("#header-sales-actions").style.display = "none";
    document.querySelector("#edit-product-options").style.display = "none";
    document.querySelector("#user-purchases-list").style.display = "none";

```

```

document.querySelector("#earnings-list-and-text").style.display = "none";

// Al hacer logout muestra la pestaña de Login y esconde el header (opciones de
administrador)
showLogin();
hideHeaderHiddenActions();
hideUserHiddenActions();
}

// Validaciones

// Función registro
function registerFunction() {
  let firstName = document.querySelector("#register-name").value;
  let lastName = document.querySelector("#register-surname").value;
  let username = document.querySelector("#register-username").value;
  let password = document.querySelector("#register-password").value;
  let creditCard = document.querySelector("#register-card").value;
  let cvc = Number(document.querySelector("#register-cvc").value);

  if (
    !checkVoidInputs(firstName, lastName, username, password, creditCard, cvc)
    // SOLAMENTE si no hay espacios en blanco -> inicio la validación del registro
    // en caso de que todo se valide -> creo el usuario
  ) {
    validateName(firstName, lastName);
    validateUsername(username);
    validatePassword(password);
    validateCard(creditCard);
    validateCVC(cvc);

    if (
      validateName(firstName, lastName) &&
      !validateUsername(username) &&
      validatePassword(password) &&
      validateCard(creditCard) &&
      validateCVC(cvc)
    ) {
      mainApp.createNewUser(
        firstName,
        lastName,
        username,
        password,
        creditCard,
        cvc
      );
      document.querySelector("#register-messages").innerHTML =
        "Usuario registrado con éxito";
      document.querySelector("#register-success-login").innerHTML =
        "Iniciar sesión";
      console.log("Usuario creado:" + mainApp.userList[length - 1]);
    }
  } else {
    document.querySelector("#register-messages").innerHTML =
      "No pueden haber espacios vacíos";
  }
}

```

```

// Función login
function loginFunction() {
  let username = document.querySelector("#login-username").value;
  let password = document.querySelector("#login-password").value;
  let foundUser = false;

  if (!checkVoidInputs(username, password)) {
    for (let i = 0; i < mainApp.userList.length; i++) {
      if (
        username === mainApp.userList[i].username &&
        password === mainApp.userList[i].password
        // busco si dentro de un mismo Objeto user hay un nombre y contraseña
        // coincidente con la que acabo de ingresar
      ) {
        mainApp.loggedUser = mainApp.userList[i];
        foundUser = true;
      }
    }

    if (foundUser) {
      document.querySelector("#login-container").style.display = "none";
      document.querySelector("#login-messages").textContent = "";
      if (mainApp.loggedUser.power) {
        showAdminFunctions();
      } else {
        showUserFunctions();
      }
    } else {
      document.querySelector("#login-messages").innerHTML =
        "Credenciales incorrectas, intenta otra vez.";
    }
  } else {
    document.querySelector("#login-messages").innerHTML =
      "No pueden haber espacios vacíos";
  }
}

// Validación de nombre
function validateName(firstName, lastName) {
  let nameValidated = false;
  document.querySelector("#register-error-fullName").innerHTML = "";

  if (
    firstName.charAt(0) === firstName.charAt(0).toUpperCase() &&
    lastName.charAt(0) === lastName.charAt(0).toUpperCase()
    // valido que el primer caracter de tanto el nombre como el apellido estén en
    mayúsculas
  ) {
    nameValidated = true;
  } else if (firstName === "" || lastName === "") {
    document.querySelector("#register-error-fullName").innerHTML = "";
  } else {
    document.querySelector("#register-error-fullName").innerHTML =
      "El Nombre y Apellido deben comenzar en mayúsculas";
  }

  return nameValidated;
}

```



```

function validateUsername(username) {
  let usernameExists = false;
  document.querySelector("#register-error-username").innerHTML = "";

  for (let i = 0; i < mainApp.userList.length; i++) {
    if (username === mainApp.userList[i].username) {
      usernameExists = true;
      // busco si el nombre que ingrese es coincidente con el nombre asignado a otro
      // objeto
      // si es coincidente, muestro mensaje de error y le doy a usernameExists valor
      true
      document.querySelector("#register-error-username").innerHTML =
        "El nombre de usuario ya existe";
    }
  }

  return usernameExists;
}

// Validación de contraseña
function validatePassword(password) {
  let passwordValidated = false;
  let passwordUppercaseCount = 0;
  let passwordLowercaseCount = 0;
  let passwordNumberCount = 0;
  document.querySelector("#register-error-password").innerHTML = "";

  for (let i = 0; i < password.length; i++) {
    if (password.charCodeAt(i) >= 65 && password.charCodeAt(i) <= 90) {
      // chequeo si el caracter de la posición en la que estoy dentro del bucle
      // corresponde a una letra mayúscula según la tabla ascii
      passwordUppercaseCount++;
    }
    if (password.charCodeAt(i) >= 97 && password.charCodeAt(i) <= 122) {
      // chequeo si el caracter de la posición en la que estoy dentro del bucle
      // corresponde a una letra minúscula según la tabla ascii
      passwordLowercaseCount++;
    }
    if (!isNaN(password.charAt(i))) {
      // chequeo si el caracter de la posición en la que estoy dentro del bucle
      // corresponde a un número
      passwordNumberCount++;
    }
  }

  if (
    password.length < 5 ||
    passwordLowercaseCount === 0 ||
    passwordUppercaseCount === 0 ||
    passwordNumberCount === 0
    // muestro mensajes de error uno por uno
  ) {
    if (password.length < 5) {
      document.querySelector("#register-error-password").innerHTML +=
        "La contraseña debe contener al menos 5 caracteres. <br>";
    }
    if (passwordUppercaseCount === 0) {
      document.querySelector("#register-error-password").innerHTML +=
        "La contraseña debe contener al menos una mayúscula. <br>";
    }
  }
}

```

```

    }
    if (passwordLowercaseCount === 0) {
        document.querySelector("#register-error-password").innerHTML +=
            "La contraseña debe contener al menos una minúscula. <br>";
    }
    if (passwordNumberCount === 0) {
        document.querySelector("#register-error-password").innerHTML +=
            "La contraseña debe contener al menos un número.";
    }
} else {
    passwordValidated = true;
}

return passwordValidated;
}

// Validación de tarjeta de crédito
function validateCard(creditCard) {
    let creditCardNoDashes = "";

    for (let i = 0; i < creditCard.length; i++) {
        if (creditCard.charAt(i) !== "-") {
            creditCardNoDashes += creditCard.charAt(i);
        }
    }
    creditCardNoDashes.toString();

    let addedUpNumber = 0;
    let cardValidated = false;

    document.querySelector("#register-error-card").innerHTML = "";

    // recorrer string de derecha a izquierda
    for (let pos = creditCardNoDashes.length - 1; pos >= 0; pos--) {
        let digit = Number(creditCardNoDashes.charAt(pos));

        // verificar si la posición actual (contando desde la derecha) es par
        if ((creditCardNoDashes.length - pos) % 2 === 0) {
            digit *= 2;

            // si la posición actual es mayor a 10, le resto 9, que es lo mismo que sumar
            // cada uno de sus dígitos individualmente
            // por ejemplo: 12. 1+2 = 3. 12-9 = 3. restarle 9 es una forma más sencilla de
            // implementar la funcionalidad
            if (digit >= 10) {
                digit -= 9;
            }
        }

        // le sumo el dígito actual al total, independientemente de si fue multiplicado
        // por 2 o no
        addedUpNumber += digit;
    }

    if (addedUpNumber % 10 === 0) {
        // si el resultado total es divisible entre 10, la tarjeta es válida
        cardValidated = true;
    } else {
        cardValidated = false;
    }
}

```

```

        document.querySelector("#register-error-card").innerHTML =
            "La tarjeta ingresada es incorrecta. <br>";
    }

    return cardValidated;
}

// Validación de CVC
function validateCVC(cvc) {
    let cvcValidated = false;
    let stringCVC = cvc.toString();
    document.querySelector("#register-error-cvc").innerHTML = "";

    for (let i = 0; i < stringCVC.length; i++) {
        if (stringCVC.length === 3) {
            // chequeo que el CVC tenga tres digitos
            cvcValidated = true;
        } else {
            document.querySelector("#register-error-cvc").innerHTML =
                "El CVC solamente puede tener 3 digitos";
        }
    }

    return cvcValidated;
}

// Checkear si hay inputs vacíos
function checkVoidInputs(input1, input2, input3, input4, input5, input6) {
    let voidInputs = false;

    if (
        input1 === "" ||
        input2 === "" ||
        input3 === "" ||
        input4 === "" ||
        input5 === "" ||
        input6 === ""
    ) {
        voidInputs = true;
    }

    return voidInputs;
}

// Función para extraer la url de la imagen sin el C:\fakepath\
function getFileName(filename) {
    let backslashPosition = -1;
    for (let i = filename.length - 1; i >= 0 && backslashPosition == -1; i--) {
        if (filename.charAt(i) == "\\") {
            backslashPosition = i;
        }
    }
    return filename.substring(backslashPosition + 1);
}

// Funciones del panel de control del admin
// Tabla de productos admin
function productsTableAdmin() {
    let HTMLtable = "<table border='1' align='center'>";

```

```

HTMLtable += `<tr>
    <th>Nombre</th>
    <th>Precio</th>
    <th>Descripcion</th>
    <th>Imagen</th>
    <th>Estado</th>
    <th>Stock</th>
    <th>Acción</th>
</tr>`;

for (let i = 0; i < mainApp.productList.length; i++) {
    let loadingItem = mainApp.productList[i];
    let loadingItemStatus = "Activo";
    let loadingItemOnSale = 'En Oferta'

    if (!loadingItem.status) {
        loadingItemStatus = "Pausado";
    }
    if (!loadingItem.onSale) {
        loadingItemOnSale = '';
    }

    HTMLtable += `<tr>
        <td>${loadingItem.name}</td>
        <td>${loadingItem.price}</td>
        <td>${loadingItem.description}</td>
        <td></td>
        <td>${loadingItemStatus} <br> ${loadingItemOnSale}</td>
        <td>${loadingItem.stock}</td>
        <td><input type='button' value='Cambiar estado' id='p${i}'>
            <br><br>
            <input type='button' value='Editar producto' id='edit${i}'>
            <br><br>
            <input type='button' value='Oferta' id='offer${i}'>
        </td>
    </tr>`;
}
HTMLtable += "</table>";

document.querySelector("#products-list").innerHTML = HTMLtable;
document.querySelector("#products-list").style.display = "block";

for (let i = 0; i < mainApp.productList.length; i++) {
    let currentButton = document.querySelector("#p" + i);
    currentButton.addEventListener("click", changeStatus);
}

for (let i = 0; i < mainApp.productList.length; i++) {
    let currentButton = document.querySelector("#edit" + i);
    currentButton.addEventListener("click", editProduct);
}

for (let i = 0; i < mainApp.productList.length; i++) {
    let currentButton = document.querySelector("#offer" + i);
    currentButton.addEventListener("click", offerProduct);
}
}

```

```

// Marcar un producto como "en oferta"
function offerProduct() {
  let clickedButton = this;
  let buttonID = clickedButton.id;
  let productPosition = Number(buttonID.substring(5));
  let currentProduct = mainApp.productList[productPosition];
  if (currentProduct.onSale) {
    currentProduct.onSale = false;
  } else {
    currentProduct.onSale = true;
  }
  productsTableAdmin();
}

// Cambiar estado de una compra de activo a pausado y viceversa
function changeStatus() {
  let clickedButton = this;
  let buttonID = clickedButton.id;
  let productPosition = Number(buttonID.substring(1));
  let currentProduct = mainApp.productList[productPosition];
  if (currentProduct.status) {
    currentProduct.status = false;
  } else {
    currentProduct.status = true;
  }
  productsTableAdmin();
}

// Editar producto
function editProduct() {
  document.querySelector("#products-list").style.display = "none";
  document.querySelector("#edit-product-options").style.display = "block";

  let clickedButton = this;
  let buttonID = clickedButton.id;
  let productPosition = Number(buttonID.substring(4));
  let currentProduct = mainApp.productList[productPosition];

  document.querySelector("#editing-product-name").placeholder = currentProduct.name;
  document.querySelector("#editing-product-price").placeholder = `Precio del
producto: ${currentProduct.price}`;
  document.querySelector("#editing-product-description").placeholder =
currentProduct.description;
  document.querySelector("#editing-product-stock").placeholder = `Stock del
producto: ${currentProduct.stock}`;
  document.querySelector("#editing-product-onSale").checked = currentProduct.onSale;
// Corregido el selector

  document.querySelector("#editing-product-name").addEventListener("input", () => {
    if (!checkVoidInputs(document.querySelector("#editing-product-name").value)) {
      currentProduct.name = document.querySelector("#editing-product-name").value;
    }
  });

  document.querySelector("#editing-product-price").addEventListener("input", () => {
    if (!checkVoidInputs(document.querySelector("#editing-product-price").value)) {
      currentProduct.price = document.querySelector("#editing-product-price").value;
    }
  });
}

```

```

    document.querySelector("#editing-product-description").addEventListener("input",
function () {
    if
(!checkVoidInputs(document.querySelector("#editing-product-description").value)) {
        currentProduct.description =
document.querySelector("#editing-product-description").value;
    }
});

    document.querySelector("#editing-product-image").addEventListener("input",
function () {
    if (!checkVoidInputs(document.querySelector("#editing-product-image").value)) {
        currentProduct.image =
getFileName(document.querySelector("#editing-product-image").value);
    }
});

    document.querySelector("#editing-product-stock").addEventListener("input",
function () {
    if (!checkVoidInputs(document.querySelector("#editing-product-stock").value)) {
        currentProduct.stock = document.querySelector("#editing-product-stock").value;
    }
});

    document.querySelector("#editing-product-onSale").addEventListener("change",
function () {
        currentProduct.onSale =
document.querySelector("#editing-product-onSale").checked;
    });

    document.querySelector("#finished-editing").addEventListener("click", () => {
        document.querySelector("#edit-product-options").style.display = "none";
        document.querySelector("#products-list").innerHTML = "";
        productsTableAdmin();
    });
}

// Función para crear producto
function createProduct() {
    let productName = document.querySelector(
        "#create-products-product-name"
    ).value;
    let productPrice = document.querySelector(
        "#create-products-product-price"
    ).value;
    let productDescription = document.querySelector(
        "#create-products-product-description"
    ).value;
    let productStock = document.querySelector(
        "#create-products-product-stock"
    ).value;
    let productImage = document.querySelector(
        "#create-products-product-image"
    ).value;
    let newFilePath = "../img/" + getFileName(productImage);

    if (
        !checkVoidInputs(

```

```

        productName,
        productPrice,
        productDescription,
        productStock,
        productImage
    )
} {
    mainApp.productPush(
        productName,
        productPrice,
        productDescription,
        newFilePath,
        productStock
    );
    document.querySelector("#create-products-message").innerHTML =
        "Producto creado con éxito";
} else {
    document.querySelector("#create-products-message").innerHTML =
        "No pueden haber espacios vacíos";
}
}

// Mostrar tablas con ventas al admin
function showSales() {
    document.querySelector("#header-sales-actions").style.display = "block";
    let salesListContainer = document.querySelector("#sales-list");

    // Crear tabla
    let HTMLtable = "<table border='1' align='center'>";
    HTMLtable += `<tr>
        <th>Nombre comprador</th>
        <th>Balance comprador</th>
        <th>Unidades</th>
        <th>Nombre producto</th>
        <th>Stock disponible</th>
        <th>Estado compra</th>
        <th>Acción</th>
    </tr>`;

    for (let i = 0; i < mainApp.salesList.length; i++) {
        let loadingItem = mainApp.salesList[i];
        HTMLtable += `<tr>
            <td>${loadingItem.buyer.username}</td>
            <td>${loadingItem.buyer.balance} USD</td>
            <td>${loadingItem.amountPurchased}</td>
            <td>${loadingItem.product.name}</td>
            <td>${loadingItem.product.stock}</td>
            <td>${loadingItem.purchaseStatus}</td>
            <td><input type='button' value='Aprobar' id='confirmP${i}'>
                <br><br>
                <input type='button' value='Cancelar' id='cancelP${i}'>
                <br>
            </td>
        </tr>`;
    }
    HTMLtable += "</table>";
    salesListContainer.innerHTML = HTMLtable;

    // Llamar a la función para confirmar o cancelar compras pendientes

```

```

for (let i = 0; i < mainApp.salesList.length; i++) {
  let confirmButton = document.querySelector("#confirmP" + i);
  let cancelButton = document.querySelector("#cancelP" + i);

  if (mainApp.salesList[i].purchaseStatus === "Pendiente") {
    confirmButton.disabled = false;
    cancelButton.disabled = false;

    confirmButton.addEventListener("click", confirmSale);
    cancelButton.addEventListener("click", cancelSale);

  } else if (mainApp.salesList[i].purchaseStatus === "Aprobada") {
    confirmButton.disabled = true;
    cancelButton.disabled = true;
  } else if (mainApp.salesList[i].purchaseStatus === "Cancelada") {
    confirmButton.disabled = true;
    cancelButton.disabled = true;
  }
}

// Confirmar compras pendientes
function confirmSale() {
  let clickedButton = this;
  let buttonID = clickedButton.id;
  let salePosition = Number(buttonID.substring(8));
  let currentSale = mainApp.salesList[salePosition];

  if (currentSale.product.stock - currentSale.amountPurchased >= 0) {
    currentSale.buyer.balance -=
      currentSale.amountPurchased * currentSale.product.price;
    currentSale.product.stock -= currentSale.amountPurchased;
    currentSale.purchaseStatus = "Aprobada";
    clickedButton.disabled = true;
  } else {
    alert("No hay stock suficiente para procesar esta orden de compra");
  }

  showSales();
}

// Cancelar compras pendientes
function cancelSale() {
  let clickedButton = this;
  let buttonID = clickedButton.id;
  let salePosition = Number(buttonID.substring(7));
  let currentSale = mainApp.salesList[salePosition];

  currentSale.buyer.balance +=
    currentSale.amountPurchased * currentSale.product.price;
  currentSale.product.stock += currentSale.amountPurchased;
  currentSale.purchaseStatus = "Cancelada";
  clickedButton.disabled = true;

  showSales();
}

// Genera la tabla con las ventas al hacer click en el botón de ver ganancias
function generateEarningsTable(salesList) {

```



```

let earningsListContainer = document.querySelector("#earnings-list");
let HTMLtable = "<table border='1' align='center'>";
HTMLtable += `<tr>
    <th>Nombre comprador</th>
    <th>Nombre producto</th>
    <th>Unidades compradas</th>
    <th>Total</th>
</tr>`;

for (let i = 0; i < salesList.length; i++) {
    let saleItem = salesList[i];
    if (saleItem.purchaseStatus === "Aprobada") {
        HTMLtable += `<tr>
            <td>${saleItem.buyer.username}</td>
            <td>${saleItem.product.name}</td>
            <td>${saleItem.amountPurchased}</td>
            <td>${saleItem.amountPurchased * saleItem.product.price}
USD</td>
        </tr>`;
    }
}
HTMLtable += "</table>";
earningsListContainer.innerHTML = HTMLtable;
}

// Hace el calculo de las ganancias totales al hacer click en el botón de ver ganancias
function calculateTotalEarningsAndPurchases(salesList) {
    let totalEarnings = 0;
    let totalPurchase = 0;
    for (let i = 0; i < salesList.length; i++) {
        let saleItem = salesList[i];
        if (saleItem.purchaseStatus === "Aprobada") {
            totalEarnings += saleItem.amountPurchased * saleItem.product.price;
            totalPurchase += saleItem.amountPurchased;
        }
    }
    document.querySelector("#total-earnings").innerHTML = `Ganancias totales:
${totalEarnings} USD <br> Unidades Compradas: ${totalPurchase}`;
}

// Función que llama las dos funciones anteriores y muestra ambas
function showEarnings() {
    let salesList = mainApp.salesList;
    generateEarningsTable(salesList);
    calculateTotalEarningsAndPurchases(salesList);
}

// Filtrar ventas según estado: vista administrador
function showFilterSales() {
    let salesFilter = document.querySelector("#sales-actions").value;
    let HTMLtable = "";

    switch (salesFilter) {
        case "Aprobadas":
            let approved = mainApp.salesList.filter(
                (Sale) => Sale.purchaseStatus === "Aprobada"
            );
            HTMLtable = "<table border='1' align='center'>";
    }
}

```

```

HTMLtable += `<tr>
    <th>Nombre comprador</th>
    <th>Balance comprador</th>
    <th>Unidades</th>
    <th>Nombre producto</th>
    <th>Stock disponible</th>
    <th>Estado compra</th>
</tr>`;
for (let i = 0; i < approved.length; i++) {
    let loadingItem = approved[i];
    HTMLtable += `<tr>
        <td>${loadingItem.buyer.username}</td>
        <td>${loadingItem.buyer.balance} USD</td>
        <td>${loadingItem.amountPurchased}</td>
        <td>${loadingItem.product.name}</td>
        <td>${loadingItem.product.stock}</td>
        <td>${loadingItem.purchaseStatus}</td>
    </tr>`;
}
HTMLtable += "</table>";
document.querySelector("#sales-list").innerHTML = HTMLtable;
break;

case "Pendientes":
    let pending = mainApp.salesList.filter(
        (Sale) => Sale.purchaseStatus === "Pendiente"
    );
    HTMLtable = "<table border='1' align='center'>";
    HTMLtable += `<tr>
        <th>Nombre comprador</th>
        <th>Balance comprador</th>
        <th>Unidades</th>
        <th>Nombre producto</th>
        <th>Stock disponible</th>
        <th>Estado compra</th>
    </tr>`;
    for (let i = 0; i < pending.length; i++) {
        let loadingItem = pending[i];
        HTMLtable += `<tr>
            <td>${loadingItem.buyer.username}</td>
            <td>${loadingItem.buyer.balance} USD</td>
            <td>${loadingItem.amountPurchased}</td>
            <td>${loadingItem.product.name}</td>
            <td>${loadingItem.product.stock}</td>
            <td>${loadingItem.purchaseStatus}</td>
        </tr>`;
    }
    HTMLtable += "</table>";
    document.querySelector("#sales-list").innerHTML = HTMLtable;
    break;

case "Canceladas":
    let cancelled = mainApp.salesList.filter(
        (Sale) => Sale.purchaseStatus === "Cancelada"
    );
    HTMLtable = "<table border='1' align='center'>";
    HTMLtable += `<tr>
        <th>Nombre comprador</th>
        <th>Balance comprador</th>

```

```

        <th>Unidades</th>
        <th>Nombre producto</th>
        <th>Stock disponible</th>
        <th>Estado compra</th>
    </tr>`;
    for (let i = 0; i < cancelled.length; i++) {
        let loadingItem = cancelled[i];
        HTMLtable += `<tr>
            <td>${loadingItem.buyer.username}</td>
            <td>${loadingItem.buyer.balance} USD</td>
            <td>${loadingItem.amountPurchased}</td>
            <td>${loadingItem.product.name}</td>
            <td>${loadingItem.product.stock}</td>
            <td>${loadingItem.purchaseStatus}</td>
        </tr>`;
    }
    HTMLtable += "</table>";
    document.querySelector("#sales-list").innerHTML = HTMLtable;
    break;

    default:
        showSales();
}
}

// Funciones del panel del usuario
// Crear tabla con los productos visibles para el usuario
function productsTableUser() {
    let HTMLtable = "<table border='1' align='center'>";

    HTMLtable += `<tr>
        <th>Nombre</th>
        <th>Precio</th>
        <th>Descripcion</th>
        <th>Imagen</th>
        <th>Stock</th>
        <th>Unidades</th>
        <th>Comprar</th>
    </tr>`;

    for (let i = 0; i < mainApp.productList.length; i++) {
        let loadingItem = mainApp.productList[i];
        let loadingSale = loadingItem.onSale ? 'En Oferta' : '';
        if (mainApp.productList[i].status) {
            HTMLtable += `<tr>
                <td>${loadingItem.name}</td>
                <td>${loadingItem.price} <br> <span style="color:
red;">${loadingSale}</span></td>
                <td>${loadingItem.description}</td>
                <td></td>
                <td>${loadingItem.stock}</td>
                <td><input type='number' id='products-list-stock${i}'
placeholder='Cantidad de unidades'></td>
                <td><input type='button' value='Comprar' id='purchaseP${i}'></td>
            </tr>`;
        }
    }
    HTMLtable += "</table>";
}

```

```

document.querySelector("#products-list-user").innerHTML = HTMLtable;
document.querySelector("#products-list-user").style.display = "block";

for (let i = 0; i < mainApp.productList.length; i++) {
    let purchaseButton = document.querySelector("#purchaseP" + i);
    if (purchaseButton) {
        purchaseButton.addEventListener("click", buyProduct);
    }
}

// Función crear orden de compra
function buyProduct() {
    let clickedButton = this;

    let buttonID = clickedButton.id;

    let productPosition = Number(buttonID.substring(9));
    let currentProduct = mainApp.productList[productPosition];
    let amountPurchased = Number(
        document.querySelector("#products-list-stock" + productPosition).value
    );

    if (currentProduct.stock >= 1) {
        mainApp.createSale(mainApp.loggedUser, currentProduct, amountPurchased);
    } else {
        alert("No hay suficientes unidades como para realizar este pedido");
    }
    productsTableUser();
}

// Mostrar historial de compras: vista usuario
function showUserPurchases() {
    let showPurchasesContainer = document.querySelector("#user-purchases-list");
    let totalSpent = 0;
    let totalUnits = 0;
    let initialBalance = mainApp.loggedUser.balance; // Asumiendo que este es 3000 y
    no cambia

    // Calcular total gastado por usuario + unidades compradas para mostrarlas
    posteriormente
    for (let i = 0; i < mainApp.salesList.length; i++) {
        if (mainApp.salesList[i].buyer.username === mainApp.loggedUser.username &&
            mainApp.salesList[i].purchaseStatus === "Aprobada") {
            totalUnits += mainApp.salesList[i].amountPurchased;
            totalSpent += mainApp.salesList[i].amountPurchased *
            mainApp.salesList[i].product.price;
        }
    }

    // Crear tabla
    let HTMLtable = "<table border='1' align='center'>";
    HTMLtable += "<tr>
        <th>Nombre producto</th>
        <th>Unidades</th>
        <th>Precio</th>
        <th>Estado compra</th>
        <th>Acción</th>

```

```

        </tr>`;

for (let i = 0; i < mainApp.salesList.length; i++) {
    let loadingItem = mainApp.salesList[i];

    if (loadingItem.buyer.username == mainApp.loggedUser.username) {
        let totalPrice = loadingItem.product.price * loadingItem.amountPurchased;

        HTMLtable += `<tr>
            <td>${loadingItem.product.name}</td>
            <td>${loadingItem.amountPurchased}</td>
            <td>${totalPrice}</td>
            <td>${loadingItem.purchaseStatus}</td>
            <td><input type='button' value='Cancelar'
id='cancelP${i}'><br></td>
        </tr>`;
    }
}

// Agregar total gastado y unidades compradas al final de la tabla
HTMLtable += `<tr><td colspan="5" align="right">Total gastado: ${totalSpent} -
Unidades compradas: ${totalUnits}</td></tr>`;

// El saldo restante ahora se muestra como el saldo inicial ya que no se restan
las compras precargadas
let remainingBalance = initialBalance; // Mostrar el saldo inicial como saldo
restante
HTMLtable += `<tr><td colspan="5" align="right">Saldo restante:
${remainingBalance}</td></tr>`;

HTMLtable += "</table>";

showPurchasesContainer.innerHTML = HTMLtable;

// Agregar eventos a los botones de cancelar
for (let i = 0; i < mainApp.salesList.length; i++) {
    if (mainApp.salesList[i].buyer.username === mainApp.loggedUser.username) {
        let cancelButton = document.querySelector("#cancelP" + i);

        if (
            mainApp.salesList[i].purchaseStatus === "Cancelada" ||
            mainApp.salesList[i].purchaseStatus === "Aprobada"
        ) {
            cancelButton.disabled = true;
        }

        if (mainApp.salesList[i].purchaseStatus === "Pendiente") {
            cancelButton.addEventListener("click", cancelPendingSale);
        }
    }
}

// Cancelar compra pendiente: vista usuario (al hacer click en panel de historial
compras)
function cancelPendingSale() {
    let clickedButton = this;
    let buttonID = clickedButton.id;
    let salePosition = Number(buttonID.substring(7));

```

```

let currentSale = mainApp.salesList[salePosition];

if (currentSale.purchaseStatus === "Pendiente") {
    currentSale.purchaseStatus = "Cancelada";
    showUserPurchases();
    clickedButton.disabled = true;
} else {
    clickedButton.disabled = false;
}
}

// Funciones de visibilidad
// Se ejecuta al inicio y al logout. Esconde el header de vista administrador
function hideHeaderHiddenActions() {
    document.querySelector("#header-hidden-actions").style.display = "none";
    document.querySelector("#header-sales-actions").style.display = "none";
}

// Se ejecuta al inicio y al logout. Esconde el header de vista usuario
function hideUserHiddenActions() {
    document.querySelector("#header-hidden-actions-user").style.display = "none";
}

// Al hacer login como admin se muestra el header y opciones de administrador
function showAdminFunctions() {
    document.querySelector("#header-hidden-actions").style.display = "block";
}

// Al hacer login como usuario se muestra el header y opciones de usuario
function showUserFunctions() {
    document.querySelector("#header-hidden-actions-user").style.display = "block";
}

// Prende y apaga la vista de la lista de compras al usuario
function toggleUserPurchases() {
    let userPurchasesContainer = document.getElementById("user-purchases-list");

    if (userPurchasesContainer.style.display === "none") {
        userPurchasesContainer.style.display = "block";
    } else {
        userPurchasesContainer.style.display = "none";
    }
}

// Oculta todos los paneles que el usuario pudiese haber tenido abiertos
// previamente para que no se abran los dos al mismo tiempo
document.querySelector("#create-products-container").style.display = "none";
document.querySelector("#products-list").style.display = "none";
document.querySelector("#products-list-user").style.display = "none";
document.querySelector("#sales-list").style.display = "none";
document.querySelector("#edit-product-options").style.display = "none";
document.querySelector("#header-sales-actions").style.display = "none";
}

// Prende y apaga la vista de la lista de productos al admin
function showAndHideProducts() {
    productTableVisible = !productTableVisible;
    if (productTableVisible) {

```

```

    productsTableAdmin();
} else {
    document.querySelector("#products-list").style.display = "none";
}

// Oculta todos los paneles que el usuario pudiese haber tenido abiertos
previamente para que no se abran los dos al mismo tiempo
document.querySelector("#create-products-container").style.display = "none";
document.querySelector("#sales-list").style.display = "none";
document.querySelector("#earnings-list-and-text").style.display = "none";
document.querySelector("#header-sales-actions").style.display = "none";
document.querySelector("#user-purchases-list").style.display = "none";
}

// Prende y apaga la vista de la lista de productos al usuario
function showAndHideProductsUser() {
    productTableUserVisible = !productTableUserVisible;
    if (productTableUserVisible) {
        productsTableUser();
    } else {
        document.querySelector("#products-list-user").style.display = "none";
    }

    document.querySelector("#user-purchases-list").style.display = "none";
}

// Prende y apaga la vista de la lista de ventas al admin
function toggleSalesListDisplay() {
    let salesContainer = document.getElementById("sales-list");
    let salesActions = document.getElementById("header-sales-actions");

    if (salesContainer.style.display === "block") {
        salesActions.style.display = "none";
        salesContainer.style.display = "none";
    } else {
        salesActions.style.display = "block";
        salesContainer.style.display = "block";
    }
}

// Oculta todos los paneles que el usuario pudiese haber tenido abiertos
previamente para que no se abran los dos al mismo tiempo
document.querySelector("#create-products-container").style.display = "none";
document.querySelector("#products-list").style.display = "none";
document.querySelector("#earnings-list-and-text").style.display = "none";
document.querySelector("#edit-product-options").style.display = "none";
document.querySelector("#user-purchases-list").style.display = "none";
}

// Prende y apaga la vista de creación de productos al admin
function showAndHideCreateProducts() {
    createProductsVisible = !createProductsVisible;
    if (createProductsVisible) {
        document.querySelector("#create-products-container").style.display =
            "block";
    } else {
        document.querySelector("#create-products-container").style.display = "none";
    }
}

```

```

// Oculta todos los paneles que el usuario pudiese haber tenido abiertos
previamente para que no se abran los dos al mismo tiempo
document.querySelector("#products-list").style.display = "none";
document.querySelector("#sales-list").style.display = "none";
document.querySelector("#earnings-list-and-text").style.display = "none";
document.querySelector("#header-sales-actions").style.display = "none";
document.querySelector("#user-purchases-list").style.display = "none";
}

// Prende y apaga la vista de la lista con las ganancias al admin
function toggleEarningsDisplay() {
  let earningsContainer = document.getElementById("earnings-list-and-text");

  if (earningsContainer.style.display === "block") {
    earningsContainer.style.display = "none";
  } else {
    earningsContainer.style.display = "block";
  }

  // Oculta todos los paneles que el usuario pudiese haber tenido abiertos
  previamente para que no se abran los dos al mismo tiempo
  document.querySelector("#create-products-container").style.display = "none";
  document.querySelector("#products-list").style.display = "none";
  document.querySelector("#sales-list").style.display = "none";
  document.querySelector("#edit-product-options").style.display = "none";
  document.querySelector("#header-sales-actions").style.display = "none";
  document.querySelector("#user-purchases-list").style.display = "none";
}

```


Código JS - classes.js

```
class App {
  constructor() {
    this.userList = new Array();
    this.productList = new Array();
    this.salesList = new Array();
    this.loggedUser = null;
  }

  // Precarga usuarios admin y compradores
  preloadUsers() {
    let preloadedUser = new User(
      "Usuario administrador",
      "",
      "admin",
      "admin",
      "",
      "",
      true
    );
    this.userList.push(preloadedUser);
    preloadedUser = new User(
      "Usuario Administrador",
      "",
      "admin2",
      "admin2",
      "",
      "",
      true
    );this.userList.push(preloadedUser);
    preloadedUser = new User(
      "Usuario Administrador",
      "",
      "admin3",
      "admin3",
      "",
      "",
      true
    );this.userList.push(preloadedUser);
    preloadedUser = new User(
      "Usuario Administrador",
      "",
      "admin4",
      "admin4",
      "",
      "",
      true
    );this.userList.push(preloadedUser);
    preloadedUser = new User(
      "Usuario Administrador",
      "",
      "admin5",
      "admin5",
      "",
      "",
      true
    );
    this.userList.push(preloadedUser);
  }
}
```

```

preloadedUser = new User(
  "Martin",
  "Leib",
  "mleib",
  "123",
  "3566-0020-2036-0505",
  "821",
  false
);
this.userList.push(preloadedUser);
preloadedUser = new User(
  "Martín",
  "Etchebarne",
  "tinchoet",
  "123",
  "4012-8888-8888-1881",
  "652",
  false
);
this.userList.push(preloadedUser);
preloadedUser = new User(
  "Juan",
  "Perez",
  "jperez",
  "123",
  "6011-0009-9013-9424",
  "362",
  false
);
this.userList.push(preloadedUser);
preloadedUser = new User(
  "Carlos",
  "Rodriguez",
  "crodriguez",
  "123",
  "3852-0000-0232-3711",
  "990",
  false
);
this.userList.push(preloadedUser);
preloadedUser = new User(
  "Marcos",
  "Gonzales",
  "mgonzales",
  "123",
  "3787-3449-3671-0002",
  "119",
  false
);
this.userList.push(preloadedUser);

console.log(
  `Cantidad de usuarios en la base de datos: ${this.userList.length}.`
);
console.log(this.userList);
}

// Precarga productos
preloadProducts() {

```

```

    let preloadedProduct = new Product(
        "Air Zoom Pegasus",
        100,
        "Champions cómodos y livianos para correr.",
        "nike-air-zoom-pegasus.webp",
        7,
        true,
        false
    );
    this.productList.push(preloadedProduct);
    preloadedProduct = new Product(
        "Air Zoom Pegasus Shield",
        100, // precio
        "Champions ideales para correr en días de lluvia.",
        "nike-air-zoom-pegasus-shield.webp",
        4, // stock
        true, // apagado o prendido
        false // false = no esta en descuento true = esta en descuento
    );
    this.productList.push(preloadedProduct);
    preloadedProduct = new Product(
        "Air Zoom Structure",
        100,
        "Champions para correr con soporte en el arco del pie.",
        "nike-air-zoom-structure.webp",
        8,
        false,
        false
    );
    this.productList.push(preloadedProduct);
    preloadedProduct = new Product(
        "Legend Essential 3",
        200,
        "Champions para entrenamiento en el gimnasio.",
        "nike-legend-essential-3.webp",
        11,
        false,
        false
    );
    this.productList.push(preloadedProduct);
    preloadedProduct = new Product(
        "Quest 5",
        150,
        "Champions para correr con soporte en el talón.",
        "nike-quest-5.webp",
        21,
        true,
        true
    );
    this.productList.push(preloadedProduct);
    preloadedProduct = new Product(
        "Air Max",
        200,
        "Champions urbanos para el día a día.",
        "nike1.jpg",
        15,
        true,
        true
    );

```

```

this.productList.push(preloadedProduct);
preloadedProduct = new Product(
    "Air Max 2",
    150,
    "Champions urbanos para el día a día.",
    "nike2.jpg",
    5,
    false,
    true
);
this.productList.push(preloadedProduct);
preloadedProduct = new Product(
    "LeBron 15 Low",
    250,
    "Champions de basquet para jugar en interiores.",
    "nike3.jpg",
    8,
    false,
    true
);
this.productList.push(preloadedProduct);
preloadedProduct = new Product(
    "Medias",
    50,
    "Medias deportivas para correr.",
    "medias_Nike2.jpg",
    5,
    true,
    true
);
this.productList.push(preloadedProduct);
preloadedProduct = new Product(
    "Medias Negras",
    50,
    "Medias deportivas para correr.",
    "image.jpg",
    10,
    true,
    true
);
this.productList.push(preloadedProduct);
}

// Precarga ventas
preloadSales() {
    let preloadedSale = new Sale(this.userList[6], this.productList[0], 5,
"Aprobada");
    this.salesList.push(preloadedSale);
    preloadedSale = new Sale(this.userList[7], this.productList[2], 8, "Cancelada");
    this.salesList.push(preloadedSale);
    preloadedSale = new Sale(this.userList[8], this.productList[0], 2, "Pendiente");
    this.salesList.push(preloadedSale);
    preloadedSale = new Sale(this.userList[9], this.productList[0], 5, "Cancelada");
    this.salesList.push(preloadedSale);
    preloadedSale = new Sale(this.userList[6], this.productList[0], 3, "Aprobada");
    this.salesList.push(preloadedSale);
    preloadedSale = new Sale(this.userList[5], this.productList[1], 5, "Pendiente");
    this.salesList.push(preloadedSale);
}

```

```

// Función que pushea un producto al array al crearlo
productPush(
    pProductName,
    pProductPrice,
    pProductDescription,
    pFilePath,
    pProductStock
) {
    let newProduct = new Product(
        pProductName,
        pProductPrice,
        pProductDescription,
        pFilePath,
        pProductStock,
        true,
        false
    );
    this.productList.push(newProduct);
}

// Función que pushea un usuario al array al crearlo ("crea" un usuario)
createNewUser(firstName, lastName, username, password, creditCard, cvc) {
    let newUser = new User(
        firstName,
        lastName,
        username,
        password,
        creditCard,
        cvc,
        false
    );
    this.userList.push(newUser);
}

// Función que pushea una venta al array al crearlo ("crea" una venta)
createSale(pBuyer, pProduct, pAmountPurchased) {
    let newSale = new Sale(pBuyer, pProduct, pAmountPurchased, "Pendiente");
    this.salesList.push(newSale);
}
}

class User {
    constructor(
        pFirstName,
        pLastName,
        pUsername,
        pPassword,
        pCreditCard,
        pCvc,
        pPower
    ) {
        this.firstName = pFirstName;
        this.lastName = pLastName;
        this.username = pUsername;
        this.password = pPassword;
        this.creditCard = pCreditCard;
        this.cvc = pCvc;
        // Por defecto el balance es $3000
    }
}

```

```

        this.balance = 3000;
        // Tipo de usuario. True es admin, false es user
        this.power = pPower;
    }
}
let productCounter = 1;
class Product {
    constructor(pName, pPrice, pDescription, pImage, pStock, pStatus, pOnSale) {
        this.name = pName;
        this.price = pPrice;
        this.description = pDescription;
        this.image = pImage;
        this.stock = pStock;
        this.status = pStatus;
        this.onSale = pOnSale;
        this.id = "PROD_ID_" + productCounter;
        productCounter++;
    }
}

class Sale {
    constructor(pBuyer, pProductName, pAmountPurchased, pPurchaseStatus) {
        this.buyer = pBuyer;
        this.product = pProductName;
        this.amountPurchased = pAmountPurchased;
        this.purchaseStatus = pPurchaseStatus;
    }
}

```

CASOS DE PRUEBA PARA FUNCIONALIDADES ASOCIADAS AL USUARIO REGISTRADO:

F-01-T01: REGISTRO DE USUARIO.**F-01-T02: REGISTRO DE USUARIO FALLIDO.****F-02-T01: INICIO DE SESIÓN: USUARIO.****F-02-T02: INICIO DE SESIÓN: USUARIO.**

| ID | Escenario de test | Pasos | Resultado esperado | Resultado obtenido | Estado (F=Falla, P=pasa) |
|---------|---------------------------|---|--|--|--------------------------|
| F01-T01 | Registro de usuario. | 1. Completa los campos de Nombre y Apellido 2. Ingresa un nombre de usuario 3. Ingresa una contraseña que cumpla con las condiciones 4. Completa el campo con una tarjeta de credito valida 5. Ingresa un codigo de seguridad | El usuario fue creado con éxito. | El usuario fue creado con exito | P |
| F01-T02 | Registro de usuario. | 1. Ingresa en los campos los datos correspondientes sin cumplir con las condiciones | Mensaje de error de los respectivos campos | Mensaje de error de los respectivos campos | F |
| F02-T01 | Inicio de sesión: Usuario | 1. Ingresa el nombre de usuario creado. 2. Ingresa una contraseña valida | Oculto el formulario en pantalla. | Oculto el formulario en pantalla | P |
| F02-T02 | Inicio de sesión: Usuario | 1. Completa los campos con los datos equivocados | Mensaje de error. "Las credenciales son incorrectas." | Mensaje de error. "Las credenciales son incorrectas." | F |

F-03-T01: COMPRAR PRODUCTO.

F-03-T02: ERROR COMPRAR PRODUCTO.

F-04-T01: CANCELAR COMPRAS.

F-04-T02: ERROR COMPRA.

| ID | Escenario de test | Pasos | Resultado esperado | Resultado obtenido | Estado (F=Falla, P=pasa) |
|---------|---------------------------|---|---|---|--------------------------|
| F03-T01 | Comprar Producto | 1. Seleccionar un producto. 2. Elegir la cantidad. 3. Oprimir boton de compra. | Producto agregado a la lista de compras. | Producto agregado a la lista de compras. | P |
| F03-T02 | Comprar Producto | 1. Seleccionar un producto 2. Elegir cantidad mayor al stock disponible. 3. Oprimir boton de compra | El producto no se carga en la lista de compras. | El producto no se carga en la lista de compras. | F |
| F04-T01 | Cancelar compras: Usuario | 1. Seleccionar producto 2. Cancelar compra mediante boton | Deshabilitacion del boton y el estado de la compra cambia a cancelado | Deshabilitacion del boton y el estado de la compra cambia a cancelado | P |
| F04-T02 | Cancelar compras | 1. Selecciona el producto con estado "Aprobado" o "Cancelado" | Boton de cancelar deshabilitado porque la compra ya fue aprobada o fue cancelada por el administrador | Boton de cancelar deshabilitado porque la compra ya fue aprobada o fue cancelada por el administrador | F |

CASOS DE PRUEBA PARA FUNCIONALIDADES ASOCIADAS AL ADMINISTRADOR:**F-05-T01: INICIAR SESION****F-05-T02: ERROR INICIAR SESION****F-06-T01: LISTADO Y APROBACION DE COMPRAS****F-06-T02: ERROR LISTADO Y APROBACION DE COMPRAS**

| ID | Escenario de test | Pasos | Resultado esperado | Resultado obtenido | Estado (F=Falla, P=pasa) |
|---------|----------------------------------|--|--|--|--------------------------|
| F05-T01 | Iniciar Sesión: Administrador | 1. Ingresa nombre de usuario 2. Ingresa contraseña | Ocultar el formulario en pantalla. | Ocultar el formulario en pantalla. | P |
| F05-T02 | Iniciar Sesión: Administrador | 1. Ingresar credenciales incorrectas | Mensaje de error en pantalla. | Mensaje de error en pantalla. | F |
| F06-T01 | Listado y Aprobación de compras. | 1. Selecciona un producto en estado pendiente 2. Verifica los datos del usuario(saldo disponible, stock del producto disponible). 3. Aprueba o cancela la compra mediante un botón | Botones deshabilitados, cambia el estado de la compra dependiendo del botón pulsado. | Botones deshabilitados, cambia el estado de la compra dependiendo del botón pulsado. | P |
| F06-T02 | Listado y Aprobación de compras. | 1. Verifica si hay stock suficiente para realizar la compra | El estado de la compra pasa a ser cancelado. | El estado de la compra pasa a ser cancelado. | F |
| F08-T02 | Administrar Productos | 1. Verifica si hay stock suficiente para realizar la compra | El estado de la compra pasa a ser cancelado. | El estado de la compra pasa a ser cancelado. | F |
| F09-T01 | Informe de Ganancias | 1. Selecciona el botón. "Ver Ganancias". | Despliega en pantalla una tabla con todos los productos vendidos, mostrando cuantos se vendieron y cual es la suma total de dinero generada. | Despliega en pantalla una tabla con todos los productos vendidos, mostrando cuantos se vendieron y cual es la suma total de dinero generada. | P |