Programación Orientada a Objetos

Proyecto Final:

Equipo No:__3__

Integrantes:

N.L.	Nombre
1	Antonio Martínez Rodrigo.
3	Briseño Vázquez Angel Geovany.
8	Nishimura Guerrero Christian Jesús.

Fecha de realización:

11/06/2023

Índice

IV. Pruebas y Validación

- Pruebas o capturas de pantalla
- Código de errores identificados y solucionados

V. Conclusiones

- Conclusiones individuales
- Conclusiones por equipo

Introducción:

En el contexto de la carrera de Ingeniería en Computación, es fundamental desarrollar habilidades y conocimientos en torno a la programación orientada a objetos y el desarrollo de aplicaciones. En este sentido, el proyecto que presentamos tiene como objetivo crear una tienda virtual de artículos de cómputo utilizando el lenguaje de programación C# y las bibliotecas necesarias para crear una aplicación de este tipo.

Los avances tecnológicos y la transformación digital que están teniendo lugar en la actualidad están cambiando la forma en que interactuamos con el mundo. El comercio electrónico es una de las áreas que ha experimentado un crecimiento exponencial en los últimos años y se ha convertido en una alternativa cada vez más popular para realizar compras de manera rápida y segura.

La creación de esta tienda virtual surge como una respuesta a la necesidad de adaptarse a los cambios en los hábitos de consumo de la población, cada vez más interesada en realizar compras en línea. Además, la pandemia mundial ha acelerado esta tendencia, lo que hace que este proyecto sea aún más relevante y actual.

En este sentido, la creación de una tienda virtual de artículos de cómputo es una propuesta que nos permite aplicar los conocimientos adquiridos en la carrera de Ingeniería en Computación para desarrollar una solución innovadora y relevante para la sociedad.

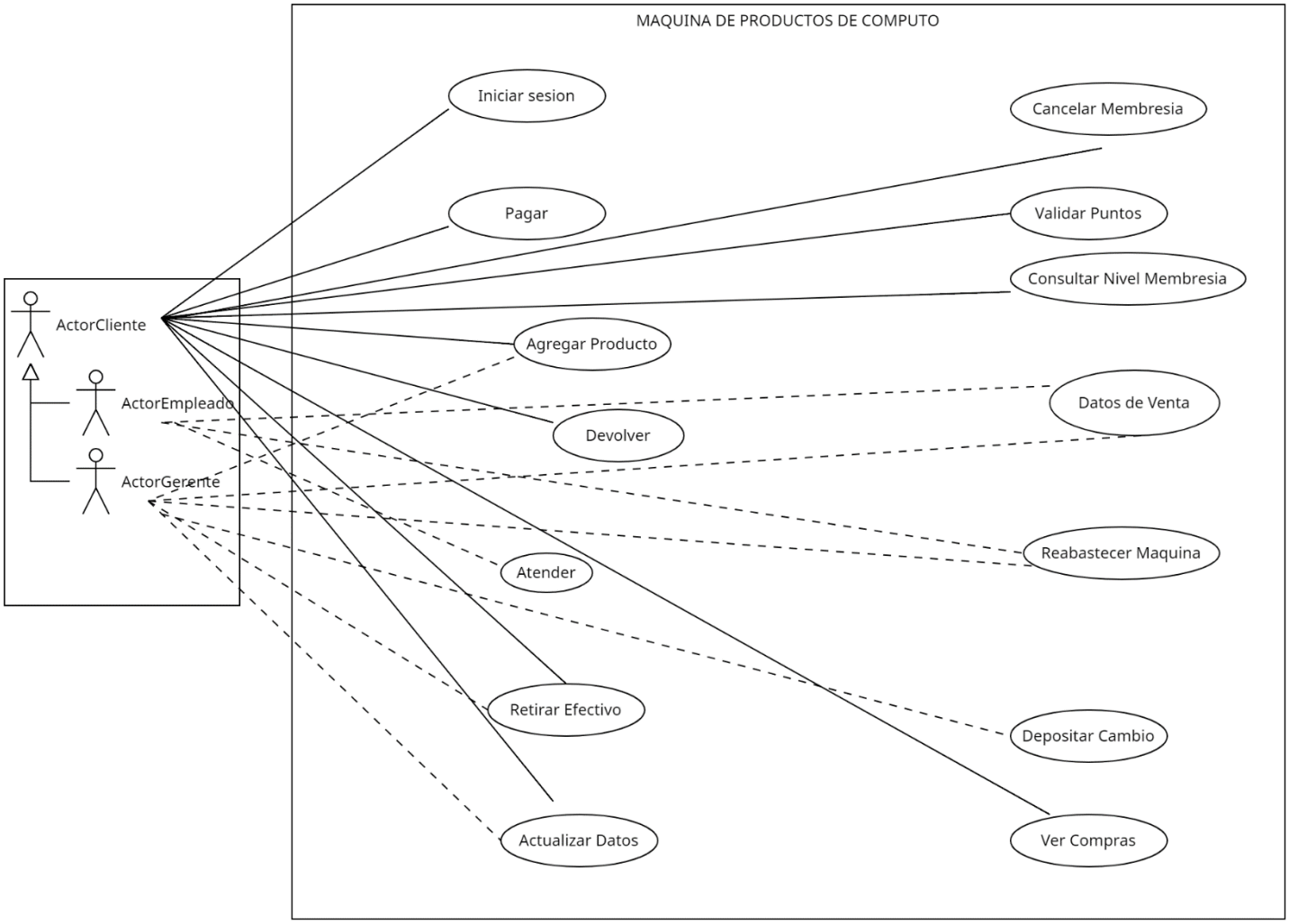
El lenguaje de programación C# y las bibliotecas asociadas son herramientas muy valiosas para el desarrollo de aplicaciones de alta calidad y con un alto grado de personalización. En este proyecto, utilizaremos estas herramientas para crear una tienda virtual de artículos de cómputo que ofrezca una experiencia de usuario satisfactoria y una interfaz gráfica atractiva.

Durante la realización de este proyecto, se utilizaron herramientas colaborativas para facilitar el trabajo en equipo y mejorar la eficiencia del proceso. Entre estas herramientas, se incluyen Visual Studio Code para el tratamiento del código, PowerShell y visual Studio 2022 para la ejecución de la tienda en línea y SQLite como gestor de base de datos para una mira futura para la implementación de una base de datos en el futuro. que nos permitieron trabajar de manera sincronizada y efectiva.

En conclusión, el desarrollo de esta tienda virtual de artículos de cómputo es una oportunidad para aplicar los conocimientos adquiridos en la carrera de Ingeniería en Computación y para contribuir a la creación de soluciones innovadoras y relevantes para la sociedad. Además, el uso de herramientas colaborativas nos permitirá trabajar de manera más eficiente y con mayor calidad en el resultado fina

Diagramas UML

Casos de Uso



Clases

USUARIO/CLIENTE
Nombre Apellidos Correo Direccion CP TEL NombreUsuario PASS Pago RFC
Metodos SET/GET Buscarproducto() Comprar() VerCarrito() ActualizarDatos() VerCompras() VerPuntos() IniciarSesion() CerrarSesion()

EMPLEADO
Nombre Apellidos Correo Direccion TEL NombreUsuario PASS RFC NumeroTrabajador
Metodos SET/GET Buscarproducto() AgregarProducto() RetiroEfectivo() DepositoCambio() IniciarSesion() CerrarSesion()

GERENTE
Nombre Apellidos Correo Direccion CP TEL PASS RFC
Metodos SET/GET VerEmpleados() AgregarEmpleados() SolicitarActualizacionStock() ReporteVentas() Solicitar retiro() DepositoEfectivo() IniciarSesion() CerrarSesion()

PRODUCTO
Serie CodigoProducto Tienda TipoProducto Precio Stock
Metodos SET/GET VerStock() VerInfoProducto()

CARRITO
Usuario Producto
Metodos SET/GET AgregarProducto() EliminarProducto() Pagar() ValidarPuntos()

PROGRAMA LEALTAD
NumMembresia Fecha de inicio Fecha de cancelacion Usuario Puntos Historial De compra
Metodos SET/GET ActualizarDatos() CancelarCuenta() IniciarSesion() CerrarSesion() VerHistorial() ValidarPuntos() VerPuntos()

PEDIDO
Nombre Apellidos Direccion CP TEL Pedidos Envio Fecha de salida Fecha entrega Monto Total Comprobante de Pago Tipo de Pago Numero de envio()
Metodos SET/GET buscarproducto() NumPedidos() NumEnvio() TipoPago() Comprobante dePago()

SUCURSAL
Nombre Sucursal Stock
Metodos SET/GET buscarproducto() InfoProducto() SolicitarActualizarStock() ReporteVentas() SolicitarRetiro() DepositoEfectivo() Envio()

TICKET
Nombre Apellidos Tienda NombreEmpresa Nombre Producto NumSerieProducto Precio por producto Monto total Descuento Pago Cambio NumSerieTicket
Metodos SET/GET PagoTotal() Nombre Producto Abreviado() Precio por producto() Descuento() NumSerieTicket()

VENTA
Productos Vendidos Cantidades de Productos Tipo Producto IDcompra Venta por tienda Venta Total Empleado Ejecuta Venta Cliente hace Compra
Metodos SET/GET VentaTotal() VentaTienda()

SISTEMA
Controla Operaciones empleado Controla Operaciones Clientes
Metodos SET/GET MetodosClientes() MetodosEmpleado()

METODO DE VENTA
Nombre Numero de tarjeta Saldo tarjeta Credito Tarjeta Debito Efectivo
Metodos SET/GET RealizarPago()

Diagrama de objetos

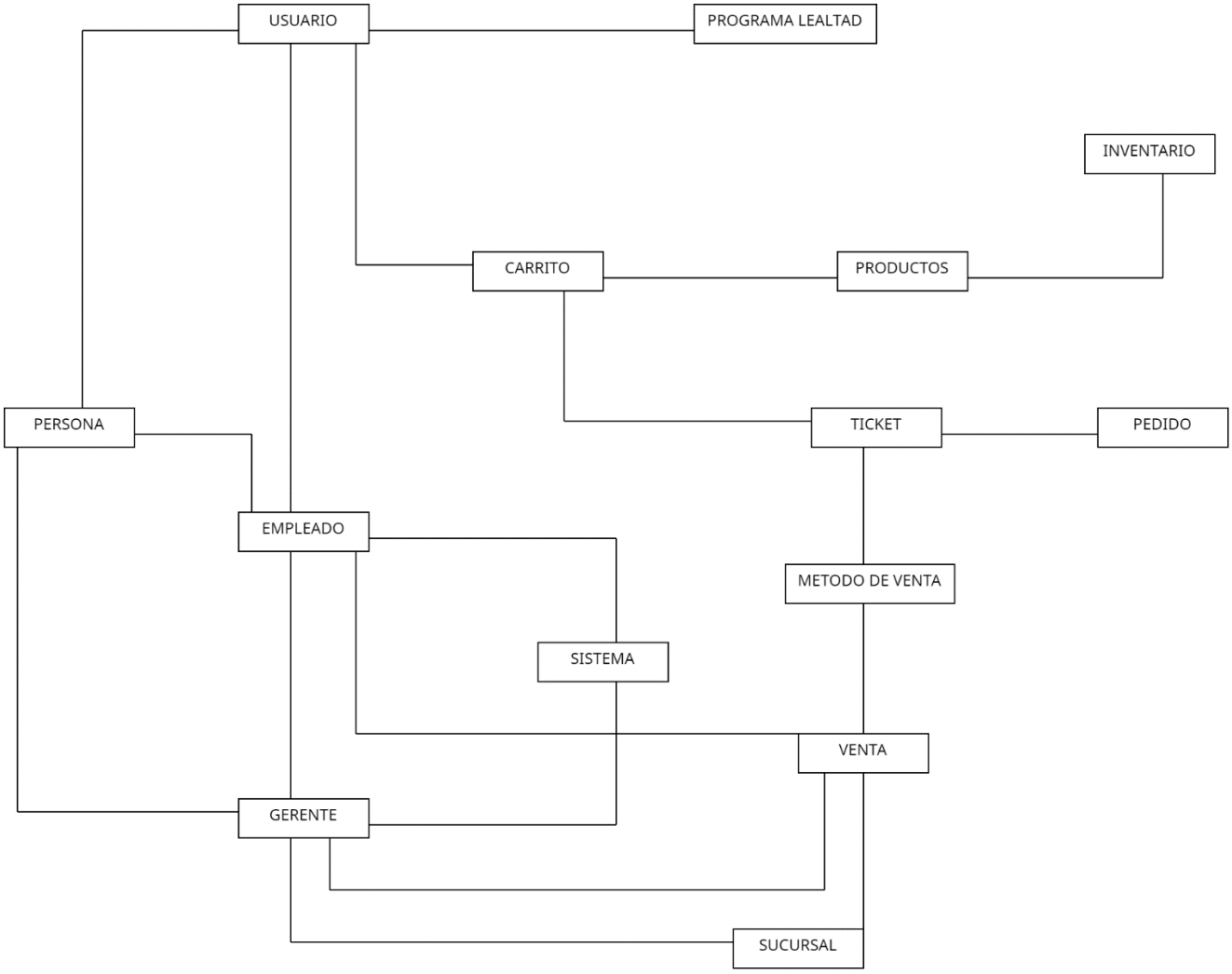
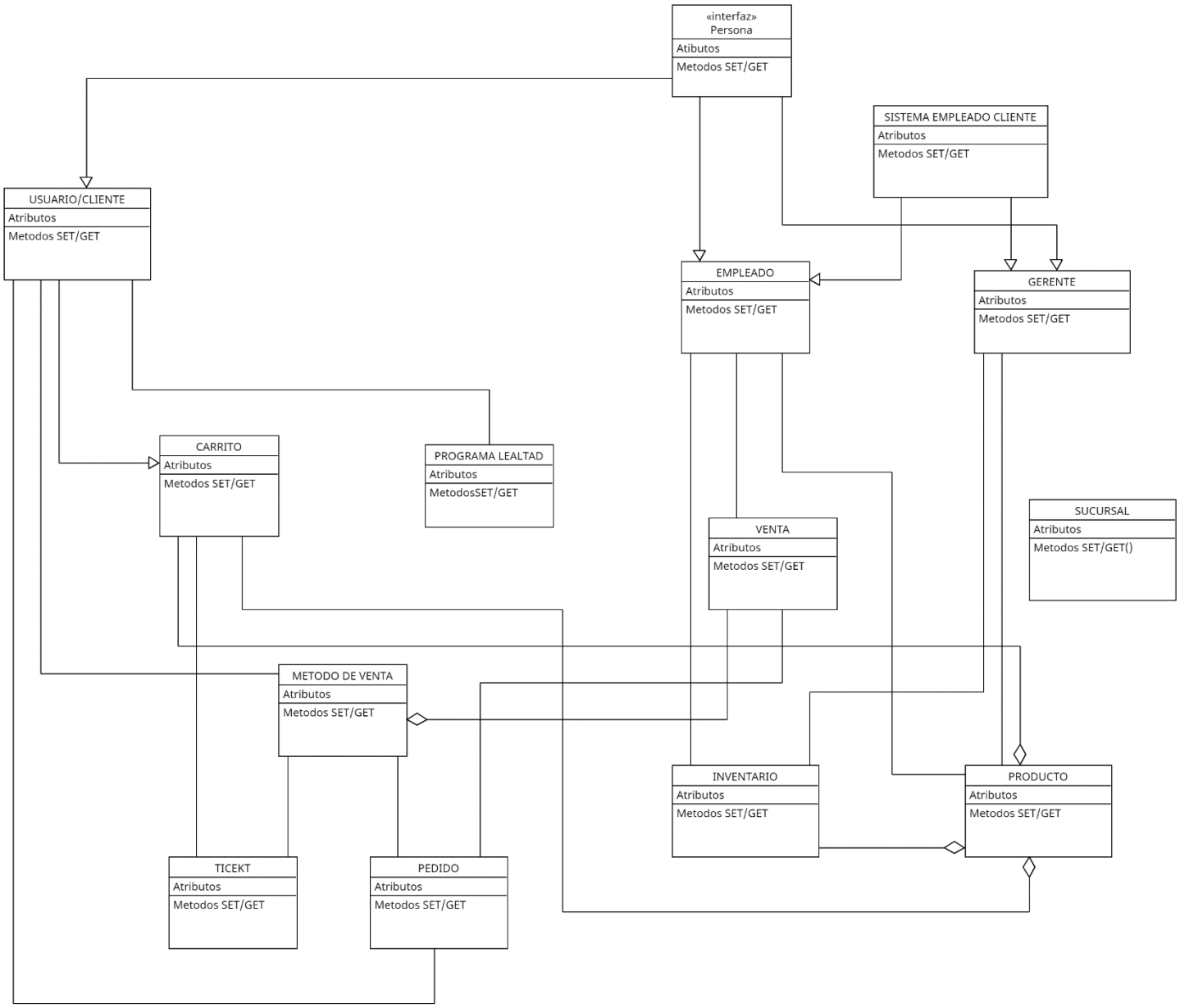


Diagrama de Clases



Código Fuente:

Programa Principal

```
using System;
using System.Collections.Generic;


class Program {
public static class Utils
{
    public static T Clamp<T>(T value, T min, T max) where T : IComparable<T>
    {
        if (value.CompareTo(min) < 0) return min;
        else if (value.CompareTo(max) > 0) return max;
        else return value;
```

```
    }
}

static void Main(string[] args) {
    // Información del equipo que desarrolló el proyecto
    Console.WriteLine("Proyecto Final");
    Console.WriteLine("Programación Orientada a Objetos");
    Console.WriteLine("Grupo: Equipo 03 - APPSOFT");
    Console.WriteLine("Integrantes:");
    Console.WriteLine("- Antonio Martínez Rodrigo. - Desarrolló la creacion de clases para
empleados y diagramas UML.");
    Console.WriteLine("- Nishimura Guerrero Christian Jesús. - Desarrolló la creacion de
clases para clientes y seleccion del flujo del programa.");
    Console.WriteLine("- Briseño Vázquez Angel Geovany - Desarrolló La creacion del resto de
las clases tanto para empleado y cliente, su implementaciín en el programa final y el intento
de conectar a base de datos Sqlite, mas control y manejo de exepciones.");
    Console.WriteLine("Presione cualquier tecla para continuar...");
    Console.ReadKey();
    Console.Clear();

    // Crear instancias de la sucursal y de los empleados y clientes
    Sucursal sucursal = new Sucursal("Av. Siempreviva 742", "(555) 123-4567");
    Empleado empleado1 = new Empleado("Juan Pérez", "juan.perez@example.com", "Calle Falsa
123", "(555) 987-6543", "Cajero", "password1", TipoTrabajador.Asistente, "001");
    Empleado empleado2 = new Empleado("María Gómez", "maria.gomez@example.com", "Calle Falsa
456", "(555) 555-5555", "Vendedor", "password2", TipoTrabajador.Gerente, "002");
    Cliente cliente1 = new Cliente("Ana", "González", "ana.gonzalez@example.com", "Calle
Falsa 789", "(555) 111-1111", "contrasena1");
    Cliente cliente2 = new Cliente("Pedro", "Rodríguez", "pedro.rodriguez@example.com",
"Calle Falsa 012", "(555) 222-2222", "contrasena2");
    Gerente gerente = new Gerente("Miguel Hernández", "miguel.hernandez@example.com", "Calle
Falsa 345", "(555) 333-3333", "Gerente General", "password3");

// Crear instancias de la sucursal y de los empleados y clientes


    // Agregar los empleados y clientes a la sucursal
    sucursal.AgregarEmpleado(empleado1);
    sucursal.AgregarEmpleado(empleado2);
    sucursal.AgregarCliente(cliente1);
    sucursal.AgregarCliente(cliente2);
    sucursal.AgregarGerente(gerente);

    Producto producto1 = new Producto("Monitor ASUS MXLZ27", 255, 1, "Monitor de 24” HDMI,
VGA");
Producto producto2 = new Producto("Monitor HP LS2001", 10, 1, "Monitor 21” VGA");
Producto producto3 = new Producto("Monitor Dell DS247", 255, 3, "Monitor 21” DP");
Producto producto4 = new Producto("Laptop Dell 1135G7", 255, 6, "1 TB DD, 8 GB RAM, Intel
Core i5");
Producto producto5 = new Producto("iPad MPQ03LZ/A", 255, 6, "64 GB, 10.9 Pulgadas");
```

```
Producto producto6 = new Producto("Combo Teclado Mouse Logitech K200", 255, 22, "Negro,
USB");
Producto producto7 = new Producto("Laptop HP B09BMGCVZH", 255, 6, "1 TB DD, 8 GB RAM, AMD
Ryzen 5");
Producto producto8 = new Producto("Laptop Alienware M15 R7", 255, 1, "512 SSD, 1 TB DD, 16 GB
RAM, Intel Core i7, Nvidia RTX 3060");
Producto producto9 = new Producto("Bocinas Logitech Z207", 255, 7, "Bocinas Bluetooth para
PC, Sonido Estéreo, 10W, Entrada Audio 3.5 mm");
Producto producto10 = new Producto("Monitor ASUS MXLZ27", 255, 12, "Monitor de 24” HDMI,
VGA");
Producto producto11 = new Producto("Monitor HP LS2001", 255, 4, "Monitor 21” VGA");
Producto producto12 = new Producto("Monitor Dell DS247", 255, 15, "Monitor 21” DP");
Producto producto13 = new Producto("Laptop Dell 1135G7", 255, 6, "1 TB DD, 8 GB RAM, Intel
Core i5");
Producto producto14 = new Producto("iPad MPQ03LZ/A", 255, 18, "64 GB, 10.9 Pulgadas");
Producto producto15 = new Producto("Combo Teclado Mouse Logitech K200", 255, 13, "Negro,
USB");
Producto producto16 = new Producto("Laptop HP B09BMGCVZH", 255, 12, "1 TB DD, 8 GB RAM, AMD
Ryzen 5");
Producto producto17 = new Producto("Laptop Alienware M15 R7", 255, 3, "512 SSD, 1 TB DD, 16
GB RAM, Intel Core i7, Nvidia RTX 3050");
Producto producto18 = new Producto("Bocinas Logitech Z207", 255, 2, "Bocinas Bluetooth para
PC, Sonido Estéreo, 10W, Entrada Audio 3.5 mm");
    // Agregar los productos a la sucursal
    sucursal.AgregarProductoNuevo(producto1);
    sucursal.AgregarProductoNuevo(producto2);
     sucursal.AgregarProductoNuevo(producto3);
    sucursal.AgregarProductoNuevo(producto4);
     sucursal.AgregarProductoNuevo(producto5);
    sucursal.AgregarProductoNuevo(producto6);
     sucursal.AgregarProductoNuevo(producto7);
    sucursal.AgregarProductoNuevo(producto8);
     sucursal.AgregarProductoNuevo(producto9);
    sucursal.AgregarProductoNuevo(producto10);
     sucursal.AgregarProductoNuevo(producto11);
    sucursal.AgregarProductoNuevo(producto12);
     sucursal.AgregarProductoNuevo(producto13);
    sucursal.AgregarProductoNuevo(producto14);
     sucursal.AgregarProductoNuevo(producto15);
    sucursal.AgregarProductoNuevo(producto16);
     sucursal.AgregarProductoNuevo(producto17);
    sucursal.AgregarProductoNuevo(producto18);

    // Mostrar el menú principal
    Console.ForegroundColor = ConsoleColor.Cyan;
    Console.WriteLine("=======================================================");
    Console.WriteLine("||              BIENVENIDO A                         ||");
    Console.WriteLine("||                   APPSOFT                         ||");
    Console.WriteLine("||                                                   ||");
    Console.ForegroundColor = ConsoleColor.DarkYellow;
```

```
    Console.WriteLine("||   La mejor tienda en línea de electrodomésticos   ||");
    Console.WriteLine("||   Estamos comprometidos en brindarte la mejor     ||");
    Console.WriteLine("||   experiencia de compra.                          ||");
    Console.ForegroundColor = ConsoleColor.Cyan;
    Console.WriteLine("============================================================");
    Console.WriteLine("||                                                        ||");
    Console.WriteLine("||   Si necesitas ayuda en cualquier momento,        ||");
    Console.WriteLine("||   no dudes en contactarnos.                       ||");
    Console.WriteLine("||                                                        ||");
    Console.WriteLine("||   ¡Que tengas un excelente día!                   ||");
    Console.WriteLine("||                                                        ||");
    Console.ForegroundColor = ConsoleColor.DarkYellow;
    Console.WriteLine("============================================================");

    // Menú para seleccionar la tienda
    Console.WriteLine("Seleccione la tienda de su agrado:");
    Console.WriteLine("1. Sucursal Coyoacán");
    Console.WriteLine("2. Sucursal Centro");
    Console.WriteLine("3. Sucursal de la FI");

    // Obtener la selección del usuario
    int seleccionTienda;

    while(true)
    {
        try
        {
            seleccionTienda = int.Parse(Console.ReadLine());
            break;
        }
        catch (FormatException)
       {
            Console.WriteLine("Selección invalida. Por favor ingrese un número válido.");

        }
    }

    // Seleccionar la tienda
    switch (seleccionTienda)
    {
            case 1:
                Console.WriteLine(" ");
                Console.WriteLine("Ha seleccionado la Sucursal Coyoacán.");
                Console.WriteLine("Información de la tienda:");
                Console.WriteLine("Nombre: Tienda Coyoacán");
                Console.WriteLine("Dirección: Calle 1, Coyoacán");
                Console.WriteLine("Teléfono: 55-1234-5678");
                break;
            case 2:
                Console.WriteLine(" ");
```

```
            Console.WriteLine("Ha seleccionado la Sucursal Centro.");
            Console.WriteLine("Información de la tienda:");
            Console.WriteLine("Nombre: Tienda Centro");
            Console.WriteLine("Dirección: Av. Juárez 123, Centro");
            Console.WriteLine("Teléfono: 55-2345-6789");
            break;
        case 3:
            Console.WriteLine(" ");
            Console.WriteLine("Ha seleccionado la Sucursal de la FI.");
            Console.WriteLine("Información de la tienda:");
            Console.WriteLine("Nombre: Tienda FI");
            Console.WriteLine("Dirección: Av. Universidad 3000, Coyoacán");
            Console.WriteLine("Teléfono: 55-3456-7890");
            break;
        default:
            Console.WriteLine(" ");
            Console.WriteLine("Opción inválida Será dirigido a la Sucursal de La FI.");
            break;
}

int seleccion=0;
do{
// Mostrar el menú principal de la tienda seleccionada
Console.WriteLine(" ");
Console.WriteLine("Seleccione una opción:");
Console.WriteLine("1. Iniciar sesión como empleado");
Console.WriteLine("2. Iniciar simulación como cliente");
Console.WriteLine("3. Salir de APPSOFT");

Console.Write("Ingrese el número de opción que desea seleccionar: ");

if (!int.TryParse(Console.ReadLine(), out seleccion)) {
    Console.WriteLine(" ");
    Console.WriteLine("Opción inválida. Por favor, seleccione una opción válida.");
    continue;
}
    // Verificar la opción elegida y actuar en consecuencia

// Ejecutar la acción seleccionada por el usuario
switch (seleccion)
{
    case 1:
        //Simulación empleado
    Console.ForegroundColor = ConsoleColor.Magenta;
    Console.WriteLine("Iniciar Sesión Empleado");
    Console.WriteLine("-----------------------");
    Console.Write("Ingrese su número de trabajador: ");
    string numTrabajador = Console.ReadLine();
    Console.Write("Ingrese su contraseña: ");
    string password = Console.ReadLine();
```

```
    Sucursal sucursalNueva = new Sucursal();
    // Buscar empleado por número de trabajador y contraseña
    Empleado empleado = sucursal.BuscarEmpleado(numTrabajador, password);

    if (empleado != null)
    {
        Console.WriteLine("Bienvenido, " + empleado.NombreCompleto);
        Console.WriteLine("-----------------------");

        // Menú de opciones para el empleado
        int opcionEmpleado = 0;
        do
        {
            Console.WriteLine("Menú de opciones:");
            Console.WriteLine("1. Actualizar Stock");
            Console.WriteLine("2. Retiro de Efectivo (ganancias)");
            Console.WriteLine("3. Depósito de cambio");
            Console.WriteLine("4. Ver inventario");
            Console.WriteLine("5. Ver lista de clientes de la sucursal");
            Console.WriteLine("6. Actualizar Información personal");
            Console.WriteLine("7. Cerrar sesión");
            Console.Write("Seleccione una opción: ");
            if (int.TryParse(Console.ReadLine(), out opcionEmpleado))
            {
                switch (opcionEmpleado)
                {
                    case 1:
                        // Actualizar Stock
                       if (empleado.Tipo == TipoTrabajador.Gerente)
                        {
                            Console.WriteLine("No tiene permisos para realizar esta
acción.");
                        }
                        else
                        {
                            // Se debe validar que el gerente haya enviado la solicitud
                            if (sucursal.SolicitudStockPendiente)
                            {
                                Console.WriteLine("Listado de productos a recibir:");
                                foreach (Producto producto in sucursal.ProductosSolicitados)
                                {
                                    Console.WriteLine(producto.ToString());
                                }

                                // Se debe permitir regresar productos que no se solicitaron
o de los cuales tienen suficientes
                                Console.WriteLine("¿Desea regresar algún producto?");
                                // ...
                            }
                            else
```

```
                        {
                            Console.WriteLine("No hay solicitudes de stock pendientes.");
                        }
                    }
                    break;
                case 2:
                    // Retiro de Efectivo (ganancias)
                    if (empleado.Tipo == TipoTrabajador.Gerente)
                    {
                        Console.WriteLine("No tiene permisos para realizar esta
acción.");
                    }
                    else
                    {
                        // Se debe llevar un registro sobre los depósitos y retiros con
el ID del Gerente que lo autorizó y el empleado que lo llevó a cabo.
                        Console.WriteLine("Ingrese la cantidad a retirar:");
                        double cantidad = double.Parse(Console.ReadLine());
                        Console.WriteLine("Ingrese el ID del Gerente que autoriza la
operación:");
                        string idGerente = Console.ReadLine();
                        // ...
                    }
                    break;
                case 3:
                    // Depósito de cambio
                    // ...
                    break;
                case 4:
                    // Ver inventario
                    Console.WriteLine("Inventario:");
                    foreach (Producto producto in sucursal.Inventario)
                    {
                        Console.WriteLine(producto.ToString());
                    }
                    break;
                case 5:
                    // Ver lista de clientes de la sucursal
                    Console.WriteLine("Lista de clientes:");
                    foreach (Cliente cliente in sucursal.Clientes)
                    {
                        Console.WriteLine(cliente.ToString());
                    }
                    break;
                case 6:
                    // Actualizar Información personal
                    // ...
                    break;
                case 7:
                    // Cerrar sesión
```

```
                    Console.WriteLine("Sesión cerrada.");
                    break;
                default:
                    Console.WriteLine("Opción inválida.");
                    break;
            }
        }
        else
        {
            Console.WriteLine("Ingrese un número válido.");
        }

    } while (opcionEmpleado != 7);
}
else
{
    Console.WriteLine("Número de trabajador o contraseña incorrectos.");
}
break;

        case 2:
//simulación cliente
Console.ForegroundColor = ConsoleColor.Green;
Console.WriteLine("Inicio de sesión Cliente");
Console.WriteLine("-----------------------");
Console.Write("Ingrese su correo electrónico: ");
string correo = Console.ReadLine();
Console.Write("Ingrese su contraseña: ");
string contrasena = Console.ReadLine();


// Buscar cliente por correo electrónico y contraseña
Cliente clienteExistente = sucursal.BuscarCliente(correo, contrasena);


if (clienteExistente != null)
{
    Console.WriteLine("Bienvenido, " + clienteExistente.NombreCompleto);
    Console.WriteLine("-----------------------");

    // Menú de opciones para el cliente
    int opcionCliente = 0;
    do
    {
        Console.WriteLine("Menú de opciones:");
        Console.WriteLine("1. Buscar producto");
        Console.WriteLine("2. Ver Carrito");
        Console.WriteLine("3. Editar carrito");
        Console.WriteLine("4. Comprar");
        Console.WriteLine("5. Actualizar Datos Personales");
```

```
            Console.WriteLine("6. Ver Mis Compras");
            Console.WriteLine("7. Cerrar sesión");
            Console.Write("Seleccione una opción: ");
            if (int.TryParse(Console.ReadLine(), out opcionCliente))
            {
                switch (opcionCliente)
                {
                    case 1:
                        // Buscar producto
                        Console.Write("Ingrese el nombre del producto: ");
                        string nombreProducto = Console.ReadLine();
                        Producto producto = sucursal.BuscarProducto(nombreProducto);
                        if (producto != null)
                        {
                            Console.WriteLine(producto.ToString());
                            Console.Write("¿Desea agregar el producto a su carrito? (S/N):
");
                            string respuesta = Console.ReadLine();
                            if (respuesta.ToUpper() == "S")
                            {
                                Console.Write("Ingrese la cantidad: ");
                                int cantidad = int.Parse(Console.ReadLine());
                                sucursal.AgregarProducto(producto, cantidad);
                                Console.WriteLine("Producto agregado al carrito.");
                            }
                        }
                        else
                        {
                            Console.WriteLine("Producto no encontrado.");
                        }
                        break;
                    case 2:
                        // Ver carrito
                        Console.WriteLine("Carrito de compras:");
                        foreach (ProductoCarrito productoCarrito in
clienteExistente.CarritoDeCompras)
                        {
                            Console.WriteLine(productoCarrito.Producto.ToString() + "
Cantidad: " + productoCarrito.Cantidad);
                        }
                        break;
                    case 3:
                        // Editar carrito
                        Console.WriteLine("Carrito de compras:");
                        foreach (ProductoCarrito productoCarrito in
clienteExistente.Carrito())
                        {
                        Console.WriteLine(productoCarrito.Producto.ToString() + " Cantidad: "
+ productoCarrito.Cantidad);
                        }
```

```
                        Console.Write("Ingrese el nombre del producto que desea eliminar o
modificar la cantidad: ");
                        string nombreProductoEditar = Console.ReadLine();
                        ProductoCarrito productoCarritoEditar =
clienteExistente.BuscarProducto(nombreProductoEditar);
                        if (productoCarritoEditar != null)
                        {
                            Console.Write("Ingrese la nueva cantidad o 0 para eliminar el
producto: ");
                            int nuevaCantidad = int.Parse(Console.ReadLine());
                            if (nuevaCantidad == 0)
                            {
                                clienteExistente.EliminarProductoDeCarrito(productoCarritoEdi
tar);
                                Console.WriteLine("Producto eliminado del carrito.");
                            }
                            else
                            {
                                productoCarritoEditar.Cantidad = nuevaCantidad;
                                Console.WriteLine("Cantidad modificada.");
                            }
                        }
                        else
                        {
                            Console.WriteLine("Producto no encontrado en el carrito.");
                        }
                        break;
                    case 4:
                        // Comprar
                        decimal totalCompraProducto = 0;
                        decimal totalCompra = 0;
                        bool hayProductosSinExistencia = false;

                        foreach (ProductoCarrito productoCarrito in
clienteExistente.Carrito())
                        {
                            if (productoCarrito.Producto.Existencia >=
productoCarrito.Cantidad)
                            {
                                productoCarrito.Producto.Existencia -=
productoCarrito.Cantidad;
                                totalCompraProducto = productoCarrito.Producto.Precio *
productoCarrito.Cantidad;
                            }
                            else
                            {
                                Console.WriteLine("No hay suficiente existencia de " +
productoCarrito.Producto.Nombre + ".");
                                hayProductosSinExistencia = true;
                            }
```

```
                        totalCompra += totalCompraProducto;
                    }

                    if (!hayProductosSinExistencia)
                    {
                        Console.Write("¿Desea factura? (S/N): ");
                        string respuestaFactura = Console.ReadLine();

                        if (respuestaFactura.ToUpper() == "S")
                        {
                            Console.Write("Ingrese su RFC: ");
                            string rfc = Console.ReadLine();
                            Console.Write("Ingrese su dirección fiscal: ");
                            string direccionFiscal = Console.ReadLine();
                            Console.Write("¿Es correcta la forma de pago? (Tarjeta de
Crédito) (S/N): ");
                            string respuestaFormaPago = Console.ReadLine();

                            CarritoDeCompras carrito = null;

                            if (respuestaFormaPago.ToUpper() == "N")
                            {
                                Console.Write("Ingrese la forma de pago: ");
                                string formaPago = Console.ReadLine();
                                carrito = new CarritoDeCompras();
                                carrito.RealizarCompra(clienteExistente, totalCompra,
rfc, direccionFiscal, formaPago);
                            }
                            else
                            {
                                carrito = new CarritoDeCompras();
                                carrito.RealizarCompra(clienteExistente, totalCompra,
rfc, direccionFiscal);
                            }
                        }
                        else
                        {
                            CarritoDeCompras carrito = new CarritoDeCompras();
                            carrito.RealizarCompra(clienteExistente, totalCompra);
                        }

                        Console.WriteLine("Compra realizada con éxito.");

                        // Vaciar carrito
                        clienteExistente.Carrito().VaciarCarrito();
                    }
                    break;
                case 5:
                    // Actualizar datos personales
```

```
                        Console.WriteLine("Actualización de Datos Personales");
                        Console.WriteLine("--------------------------------");
                        Console.Write("Ingrese su nombre completo: ");
                        string nombreCompleto = Console.ReadLine();
                        Console.Write("Ingrese su nuevo correo electrónico: ");
                        string nuevoCorreo = Console.ReadLine();
                        Console.Write("Ingrese su dirección: ");
                        string direccion = Console.ReadLine();
                        Console.Write("Ingrese su teléfono: ");
                        string telefono = Console.ReadLine();
                        clienteExistente.ActualizarDatosPersonales(nombreCompleto,
nuevoCorreo, direccion, telefono);
                        Console.WriteLine("Datos actualizados con éxito.");
                        break;
                    case 6:
                        // Ver mis compras
                        Console.WriteLine("Mis Compras:");
                        foreach (Compra compra in clienteExistente.Compras)
                        {
                            Console.WriteLine(compra.ToString());
                        }
                        break;
                    case 7:
                        // Cerrar sesión
                        Console.WriteLine("Sesión cerrada.");
                        break;
                    default:
                        Console.WriteLine("Opción inválida.");
                        break;
                }
            }
            else
            {
                Console.WriteLine("Opción inválida.");
            }
            Console.WriteLine();
        } while (opcionCliente != 7);
    }
    else
    {
        Console.WriteLine("Correo electrónico o contraseña incorrectos.");
    }
    break;
        case 3:
    Console.WriteLine("Saliendo del programa...");
    Environment.Exit(0);
    break;

    }
```

```
    Console.ReadKey();
} while (seleccion != 3);
}
}
```

Clase carrito de compras:

```
using System;
using System.Collections;
using System.Collections.Generic;

public static class ListExtensions
{
    public static void VaciarCarrito(this List<ProductoCarrito> carrito)
    {
        carrito.Clear();
    }
}
public class CarritoDeCompras : IEnumerable<ProductoCarrito>
{
    public List<Producto> ListaDeProductos { get; set; }
    public List<int> Cantidades { get; set; }

    public CarritoDeCompras()
    {
        this.ListaDeProductos = new List<Producto>();
        this.Cantidades = new List<int>();
    }

    public List<ProductoCarrito> Productos()
    {
        List<ProductoCarrito> productos = new List<ProductoCarrito>();
        for (int i = 0; i < this.ListaDeProductos.Count; i++)
        {
            Producto producto = this.ListaDeProductos[i];
            int cantidad = this.Cantidades[i];
            productos.Add(new ProductoCarrito(producto, cantidad));
        }
        return productos;

    }



    public ProductoCarrito BuscarProducto(string nombre)
    {
        int indice = this.ListaDeProductos.FindIndex(p => p.Nombre == nombre);
        if (indice != -1)
        {
            Producto producto = this.ListaDeProductos[indice];
```

```
            int cantidad = this.Cantidades[indice];
            return new ProductoCarrito(producto, cantidad);
        }
        else
        {
            return null;
        }
    }

    public void AgregarProducto(Producto producto, int cantidad)
    {
        this.ListaDeProductos.Add(producto);
        this.Cantidades.Add(cantidad);
    }

    public void EliminarProducto(Producto producto)
    {
        int indice = this.ListaDeProductos.IndexOf(producto);
        this.ListaDeProductos.RemoveAt(indice);
        this.Cantidades.RemoveAt(indice);
    }

    public void VaciarCarrito()
    {
        this.ListaDeProductos.Clear();
        this.Cantidades.Clear();
    }

    public decimal ObtenerTotal()
    {
        decimal total = 0;
        for (int i = 0; i < this.ListaDeProductos.Count; i++)
        {
            total += this.ListaDeProductos[i].Precio * this.Cantidades[i];
        }
        return total;
    }

    public void RealizarCompra(Cliente cliente, decimal totalCompra, string rfc, string
direccionFiscal, string formaPago)
    {
        // Lógica para realizar la compra con forma de pago
        // ...
    }

    public void RealizarCompra(Cliente cliente, decimal totalCompra, string rfc, string
direccionFiscal)
    {
        // Lógica para realizar la compra sin forma de pago
        // ...
```

```
    }

    public void RealizarCompra(Cliente cliente, decimal totalCompra)
    {
        // Lógica para realizar la compra sin información fiscal ni forma de pago
        // ...
    }

    public IEnumerator<ProductoCarrito> GetEnumerator()
    {
        foreach (Producto producto in this.ListaDeProductos)
        {
            int cantidad = this.Cantidades[this.ListaDeProductos.IndexOf(producto)];
            yield return new ProductoCarrito(producto, cantidad);
        }
    }

    IEnumerator IEnumerable.GetEnumerator()
    {
        return GetEnumerator();
    }
}

public class ProductoCarrito
{
    public Producto Producto { get; set; }
    public int Cantidad { get; set; }

    public ProductoCarrito(Producto producto, int cantidad)
    {
        this.Producto = producto;
        this.Cantidad = cantidad;
    }
}
```

Clase cliente:

```
using System;
using System.Collections.Generic;

// Clase Cliente
public class Cliente {
    public string Nombre { get; set; }
    public string Apellido { get; set; }
    public string Correo { get; set; }
    public string Direccion { get; set; }
    public string Telefono { get; set; }
    public ProgramaDeLealtad ProgramaDeLealtad { get; set; }
    public CarritoDeCompras CarritoDeCompras { get; set; }
    public decimal Saldo { get; set; }
```

```
    public string Contrasena { get; set; }
    public List<Compra> Compras { get; set; } // Agregar propiedad Compras

    private string _nombreCompleto;
    public string NombreCompleto {
        get { return _nombreCompleto; }
        private set { _nombreCompleto = value; }
    }

    public Cliente(string nombre, string apellido, string correo, string direccion, string
telefono, string contrasena) {
        this.Nombre = nombre;
        this.Apellido = apellido;
        this.Correo = correo;
        this.Direccion = direccion;
        this.Telefono = telefono;
        this.ProgramaDeLealtad = new ProgramaDeLealtad();
        this.CarritoDeCompras = new CarritoDeCompras();
        this.Saldo = 0;
        this.Contrasena = contrasena;
        this.NombreCompleto = nombre + " " + apellido;
        this.Compras = new List<Compra>(); // Inicializar la lista de compras
    }

    public void ActualizarDatosPersonales(string nombreCompleto, string correo, string
direccion, string telefono)
    {
        this.Correo = correo;
        this.Direccion = direccion;
        this.Telefono = telefono;
        this.NombreCompleto = nombreCompleto;
        string[] nombreApellido = nombreCompleto.Split(' ');
        if (nombreApellido.Length >= 1)
        {
            this.Nombre = nombreApellido[0];
        }
        if (nombreApellido.Length >= 2)
        {
            this.Apellido = nombreApellido[1];
        }
    }

    public void EliminarProductoDeCarrito(ProductoCarrito producto)
    {
        this.CarritoDeCompras.EliminarProducto(producto.Producto);
    }

    public void RealizarVenta(Ticket ticket)
    {
        // Agregar el ticket a la lista de tickets del cliente
```

```
        this.Compras.Add(new Compra(this, ticket));
        this.ProgramaDeLealtad.AñadirPuntos(ticket.Total);
        this.CarritoDeCompras.VaciarCarrito();
    }

    public void DescontarSaldo(decimal cantidad)
    {
        this.Saldo -= cantidad;
    }

    public List<ProductoCarrito> Carrito()
    {
        return this.CarritoDeCompras.Productos();
    }

    public ProductoCarrito BuscarProducto(string nombre)
    {
        return this.CarritoDeCompras.BuscarProducto(nombre);
    }
}
```

Clase compra:

```
using System;
using System.Collections.Generic;

// Clase Compra
public class Compra {
    public Cliente Cliente { get; set; }
    public Ticket Ticket { get; set; }

    public Compra(Cliente cliente, Ticket ticket)
    {
        this.Cliente = cliente;
        this.Ticket = ticket;
    }

    public override string ToString()
    {
        return "Fecha: " + this.Ticket.Fecha + ", Total: " + this.Ticket.Total;
    }
}
```

Clase Empleado:

```
using System;
using System.Collections.Generic;
```

```
public enum TipoTrabajador {
    Gerente,
    Asistente,
    Tecnico
}

public class Empleado {
    public string Nombre { get; set; }
    public string Correo { get; set; }
    public string Direccion { get; set; }
    public string Telefono { get; set; }
    public string Funcion { get; set; }
    public string Contrasena { get; set; }
    public bool Activo { get; set; }
    public string Rol { get; set; }
    public TipoTrabajador Tipo { get; set; }
    public string NumTrabajador { get; set; } // Agregar propiedad NumTrabajador

    public Empleado(string nombre, string correo, string direccion, string telefono, string
funcion, string contrasena, TipoTrabajador tipo, string numTrabajador) {
        this.Nombre = nombre;
        this.Correo = correo;
        this.Telefono = telefono;
        this.Funcion = funcion;
        this.Contrasena = contrasena;
        this.Activo = true;
        this.Direccion = direccion; // Puede ser null
        this.Tipo = tipo;
        this.NumTrabajador = numTrabajador; // Asignar valor a NumTrabajador
    }

    public Empleado(string nombre, string correo, string contrasena, TipoTrabajador tipo,
string numTrabajador) {
        this.Nombre = nombre;
        this.Correo = correo;
        this.Contrasena = contrasena;
        this.Activo = true;
        this.Tipo = tipo;
        this.NumTrabajador = numTrabajador; // Asignar valor a NumTrabajador
    }

    public Empleado(string nombre, string correo, string direccion, string telefono, string
funcion, TipoTrabajador tipo, string numTrabajador) {
        this.Nombre = nombre;
        this.Correo = correo;
        this.Direccion = direccion; // Puede ser null
        this.Telefono = telefono;
        this.Funcion = funcion;
        this.Activo = true;
        this.Tipo = tipo;
```

```
        this.NumTrabajador = numTrabajador; // Asignar valor a NumTrabajador
    }

    public string NombreCompleto {
        get {
            return string.Format("{0}", this.Nombre); // Return the full name as a string
        }
    }
}
```

Clase gerente:

```
using System.Collections.Generic;


public class Gerente {
    public Gerente(string nombre, string correo, string direccion, string telefono, string
funcion, string contrasena )  {
    }
}
```

Clase Pedido:

```
using System;
using System.Collections.Generic;

// Clase Pedido
class Pedido : CarritoDeCompras {
    public Cliente Cliente { get; set; }
    public DateTime Fecha { get; set; }

    public Pedido(Cliente cliente) {
        this.Cliente = cliente;
        this.Fecha = DateTime.Now;
    }
}
```

Clase Persona:

```
using System;
using System.Collections.Generic;

public class Sistema
{
    public List<Cliente> Clientes { get; set; }
    public List<Empleado> Empleados { get; set; }
    public List<Gerente> Gerentes { get; set; }
    public List<Producto> Productos { get; set; }
    public List<Sucursal> Sucursales { get; set; }
```

```
    public string Nombre { get; set; }
    public decimal CambioEnCaja { get; set; }

    public void ActualizarStock(Gerente gerente, Producto producto, int cantidad)
    {
        // validar que el gerente haya enviado la solicitud
        if (true /* validar que el gerente haya enviado la solicitud */)
        {
            // actualizar el stock del producto en la sucursal correspondiente
            producto.Stock += cantidad;
        }
    }

    public void RetirarEfectivo(Gerente gerente, Sucursal sucursal, decimal cantidad)
    {
        // retirar la cantidad de la caja y guardar el registro
        CambioEnCaja -= cantidad;
    }

    public void DepositarCambio(Gerente gerente, Sucursal sucursal, decimal cantidad)
    {
        // depositar la cantidad en la caja y guardar el registro
        CambioEnCaja += cantidad;
    }

    public void AgregarProducto(Gerente gerente, Sucursal sucursal, Producto producto, int
cantidad)
    {
        // agregar el producto a la lista de productos de la sucursal
        sucursal.Productos.Add(producto);
        // enviar la orden de abastecimiento de stock a la sucursal correspondiente
        ActualizarStock(gerente, producto, cantidad);
    }

    public void SolicitarActualizacionStock(Gerente gerente, Sucursal sucursal)
    {
        // enviar la solicitud de actualización de stock a la sucursal correspondiente
    }

    public void SolicitarRetiroEfectivo(Gerente gerente, Sucursal sucursal, decimal cantidad)
    {
        // enviar la solicitud de retiro de efectivo a la sucursal correspondiente
        RetirarEfectivo(gerente, sucursal, cantidad);
    }

    public void SolicitarDepositoCambio(Gerente gerente, Sucursal sucursal, decimal cantidad)
    {
        // enviar la solicitud de depósito de cambio a la sucursal correspondiente
        DepositarCambio(gerente, sucursal, cantidad);
    }
```

```
    public void RealizarVenta(Cliente cliente, Sucursal sucursal, Venta venta)
    {
        // validar que hay suficiente stock de cada producto
        bool haySuficienteStock = true;
        for (int i = 0; i < venta.Productos.Count; i++)
        {
            if (venta.Cantidades[i] > venta.Productos[i].Stock)
            {
                haySuficienteStock = false;
                break;
            }
        }
        if (haySuficienteStock)
        {
            // restar la cantidad vendida del stock de cada producto
            for(int i = 0; i < venta.Productos.Count; i++)
            {
                venta.Productos[i].Stock -= venta.Cantidades[i];
            }
            // agregar la venta a la lista de ventas de la sucursal y del cliente
            sucursal.Ventas.Add(venta);
            sucursal.Ventas.Add(venta);
        }
    }
}
```

Clase Producto:

```
using System;
using System.Collections.Generic;

public class Producto
{
    public string Nombre { get; set; }
    public decimal Precio { get; set; }
    public int Existencia { get; set; } // Cambiar el nombre de la propiedad a Existencia
    public string Descripcion { get; set; }
    public int Stock { get; set; }
    public Producto(string nombre, decimal precio, int existencia, string descripcion)
    {
        this.Nombre = nombre;
        this.Precio = precio;
        this.Existencia = existencia;
        this.Descripcion = descripcion;
    }

    public override string ToString()
    {
        return this.Nombre;
```

```
    }

    // Consultar productos desde una lista
    public static List<Producto> ConsultarProductos(List<Producto> listaProductos)
    {
        return listaProductos;
    }
}
```

Clase Sucursal:

```
using System;
using System.Collections.Generic;

public class Sucursal {
    public string Direccion { get; set; }
    public string Telefono { get; set; }
    public bool SolicitudStockPendiente { get; set; }
    public List<Empleado> empleados;
    public List<Cliente> clientes;
    public List<Producto> productos;
    public List<Venta> ventas;
    public List<Ticket> tickets;
    private List<Gerente> gerentes = new List<Gerente>();
    public List<Producto> ProductosSolicitados { get; set; }
    public List<Producto> Inventario
{
    get { return this.productos; }
}
    public Sucursal()
    {
        this.empleados = new List<Empleado>();
        this.clientes = new List<Cliente>();
        this.productos = new List<Producto>();
        this.ventas = new List<Venta>();
        this.tickets = new List<Ticket>();
        this.ProductosSolicitados = new List<Producto>();
    }

    public Sucursal(string direccion, string telefono)
    {
        this.Direccion = direccion;
        this.Telefono = telefono;
        this.empleados = new List<Empleado>();
        this.clientes = new List<Cliente>();
        this.productos = new List<Producto>();
        this.ventas = new List<Venta>();
        this.tickets = new List<Ticket>();
    }
```

```
    public List<Empleado> Empleados
    {
        get { return this.empleados; }
    }

    public List<Cliente> Clientes
    {
        get { return this.clientes; }
    }

    public List<Producto> Productos
    {
        get { return this.productos; }
    }

    public List<Venta> Ventas
    {
        get { return this.ventas; }
    }

    public List<Ticket> Tickets
    {
        get { return this.tickets; }
    }

    public void AgregarEmpleado(Empleado empleado)
    {
        this.empleados.Add(empleado);
    }

    public void AgregarGerente(Gerente gerente)
    {
        this.gerentes.Add(gerente);
    }

    public void AgregarCliente(Cliente cliente)
    {
        this.clientes.Add(cliente);
    }


public void AgregarProductoNuevo(Producto producto)
{
    if (producto != null) {
        Producto productoExistente = this.BuscarProducto(producto.Nombre);
        if (productoExistente != null) {
            productoExistente.Stock++;
        } else {
            producto.Stock = 1;
            this.productos.Add(producto);
```

```
        }
    }
}
   public void AgregarProducto(Producto producto, int cantidad)
{
    if (producto != null) {
        Producto productoExistente = this.BuscarProducto(producto.Nombre);
        if (productoExistente != null) {
            productoExistente.Stock += cantidad;
        } else {
            producto.Stock = cantidad;
            this.productos.Add(producto);
        }
    }
}

    public Producto BuscarProducto(string nombre)
{
    foreach (Producto producto in this.productos) {
        if (producto.Nombre == nombre) {
            return producto;
        }
    }
    return null;
}
    public Empleado AutenticarEmpleado(string usuario, string contrasena)
    {
        foreach (Empleado empleado in this.empleados) {
            if (empleado.Correo == usuario && empleado.Contrasena == contrasena) {
                return empleado;
            }
        }
        return null;
    }

    public Cliente AutenticarCliente(string correo, string contrasena)
    {
        foreach (Cliente cliente in this.clientes) {
            if (cliente.Correo == correo && cliente.Contrasena == contrasena) {
                return cliente;
            }
        }
        return null;
    }

    public void RealizarVenta(Venta venta, Empleado empleado)
    {
        this.ventas.Add(venta);
        venta.Ticket = new Ticket(venta, this);
        this.tickets.Add(venta.Ticket);
```

```
        for (int i = 0; i < venta.Productos.Count; i++) {
            venta.Productos[i].Stock -= venta.Cantidades[i];
        }
        if (venta.Cliente != null && venta.MetodoPago != null && venta.MetodoPago.Saldo > 0)
{
            venta.Cliente.Saldo -= venta.ObtenerTotal();
        }
        venta.Empleado = empleado;
    }

    // Método para obtener la lista de productos disponibles
    public List<Producto> ObtenerProductosDisponibles()
    {
        List<Producto> disponibles = new List<Producto>();
        foreach (Producto producto in this.productos) {
            if (producto.Stock > 0) {
                disponibles.Add(producto);
            }
        }
        return disponibles;
    }

    // Método para obtener la lista de empleados disponibles
    public List<Empleado> ObtenerEmpleadosDisponibles(string rol)
    {
        List<Empleado> disponibles = new List<Empleado>();
        foreach (Empleado empleado in this.empleados) {
            if (empleado.Activo) {
                disponibles.Add(empleado);
            }
        }
        return disponibles;
    }
    public List<Empleado> ObtenerEmpleadosDisponiblesPorRol(string rol)
{
    List<Empleado> disponibles = new List<Empleado>();
    foreach (Empleado empleado in this.empleados) {
        if (empleado.Activo && empleado.Rol.Equals(rol)) {
            disponibles.Add(empleado);
        }
    }
    return disponibles;
}
public List<Venta> RealizarVentas()
{
    return this.ventas;
}

public Empleado BuscarEmpleado(string numTrabajador, string password)
{
```

```
    foreach (Empleado empleado in this.empleados) {
        if (empleado.NumTrabajador == numTrabajador && empleado.Contrasena == password) {
            return empleado;
        }
    }
    return null;
}
public Cliente BuscarCliente(string correo, string contrasena)
{
    foreach (Cliente cliente in this.clientes) {
        if (cliente.Correo == correo && cliente.Contrasena == contrasena) {
            return cliente;
        }
    }
    return null;
}
}
```

Clase Ticket:

```
using System;
using System.Collections.Generic;


public class Ticket {
    public DateTime Fecha { get;set; }
    public Venta Venta { get; set; }
    public Sucursal Sucursal { get; set; }
    public decimal Total { get; set; }
    public string IdCompra { get; set; }

    public Ticket()
    {
        // initialize the properties
        Fecha = DateTime.Now;
        Venta = new Venta(); // or initialize with an existing instance of Venta
        Sucursal = new Sucursal(); // or initialize with an existing instance of Sucursal
        Total = 0;
        IdCompra = "";
    }

    public Ticket(Venta venta, Sucursal sucursal)
    {
        // initialize the properties
        Fecha = DateTime.Now;
        Venta = venta;
        Sucursal = sucursal;
        Total = venta.ObtenerTotal();
        IdCompra = Guid.NewGuid().ToString();
    }
```

```
}
```

Clase Venta:

```
using System;
using System.Collections.Generic;

// Definición de la clase MetodoPago
public class MetodoPago {
    public string Nombre { get; set; }
    public string NumeroTarjeta { get; set; }
    public decimal Saldo { get; set; }
    public bool TarjetaCredito { get; set; }
    public bool TarjetaDebito { get; set; }
    public bool Efectivo { get; set; }

    public MetodoPago() {
        this.Saldo = 0;
    }
}

 public class Venta {
    private static int contadorIdCompra = 0;
    public List<Producto> Productos { get; set; }
    public List<int> Cantidades { get; set; }
    public MetodoPago MetodoPago { get; set; }
    public Cliente Cliente { get; set; }
    public Ticket Ticket { get; set; }
    public Empleado Empleado { get; set; } // Agregar esta propiedad
    public int IdCompra { get; private set; }

    decimal saldo;

    public Venta() {
        this.Productos = new List<Producto>();
        this.Cantidades = new List<int>();
        this.MetodoPago = new MetodoPago();
        this.saldo = this.MetodoPago.Saldo;
    }
    public Venta(List<Producto> productos, List<int> cantidades, Cliente cliente) {
        this.Productos = productos;
        this.Cantidades = cantidades;
        this.Cliente = cliente;
        this.MetodoPago = new MetodoPago();
        this.saldo = this.MetodoPago.Saldo;
        this.IdCompra = ++contadorIdCompra;
    }

    public Venta(List<Producto> productos, List<int> cantidades, MetodoPago metodoPago,
Cliente cliente) {
```

```
        this.Productos = productos;
        this.Cantidades = cantidades;
        this.MetodoPago = metodoPago;
        this.Cliente = cliente;
        this.saldo = this.MetodoPago.Saldo;
        this.IdCompra = ++contadorIdCompra;
    }

    public decimal ObtenerTotal() {
        decimal total = 0;
        for (int i = 0; i < this.Productos.Count; i++) {
            total += this.Productos[i].Precio * this.Cantidades[i];
        }
        return total;
    }
}
```

Clase ProgramaDeLealtad:

```
using System;
using System.Collections.Generic;

// Clase ProgramaDeLealtad
// Clase ProgramaDeLealtad
public class ProgramaDeLealtad {
    public int Puntos { get; set; }
    public int Nivel { get; set; }

    public void AñadirPuntos(decimal puntos) {
        int puntosEnteros = (int) puntos;
        this.Puntos += puntosEnteros;
    }
}
```

Pruebas (Capturas de pantalla):

```
PS C:\Users\ciuda\Desktop\ProyectoFinal_EQ03\Código Fuente> .\Program.exe
Proyecto Final
Programación Orientada a Objetos
Grupo: Equipo 03 - APPSOFT
Integrantes:
- Antonio Martínez Rodrigo. - Desarrolló la creacion de clases para empleados y diagramas UML.
- Nishimura Guerrero Christian Jesús. - Desarrolló la creacion de clases para clientes y seleccion del flujo del program
a.
- Briseño Vázquez Angel Geovany - Desarrolló La creacion del resto de las clases tanto para empleado y cliente, su imple
mentaciín en el programa final y el intento de conectar a base de datos Sqlite, mas control y manejo de exepciones.
Presione cualquier tecla para continuar...
```

```
============================================================
||                  BIENVENIDO A                          ||
||                       APPSOFT                          ||
||                                                        ||
||   La mejor tienda en línea de electrodomésticos        ||
||   Estamos comprometidos en brindarte la mejor          ||
||   experiencia de compra.                               ||
============================================================
||                                                        ||
||   Si necesitas ayuda en cualquier momento,             ||
||   no dudes en contactarnos.                            ||
||                                                        ||
||   ¡Que tengas un excelente día!                        ||
||                                                        ||
============================================================
Seleccione la tienda de su agrado:
1. Sucursal Coyoacán
2. Sucursal Centro
3. Sucursal de la FI
```

```
Seleccione la tienda de su agrado:
1. Sucursal Coyoacán
2. Sucursal Centro
3. Sucursal de la FI
1

Ha seleccionado la Sucursal Coyoacán.
Información de la tienda:
Nombre: Tienda Coyoacán
Dirección: Calle 1, Coyoacán
Teléfono: 55-1234-5678

Seleccione una opción:
1. Iniciar sesión como empleado
2. Iniciar simulación como cliente
3. Salir de APPSOFT
Ingrese el número de opción que desea seleccionar: 1
Iniciar Sesión Empleado
-----------------------
Ingrese su número de trabajador: 001
Ingrese su contraseña: password1
Bienvenido, Juan Pérez
-----------------------
Menú de opciones:
1. Actualizar Stock
2. Retiro de Efectivo (ganancias)
3. Depósito de cambio
4. Ver inventario
5. Ver lista de clientes de la sucursal
```

```
Bienvenido, Juan Pérez
----------------------
Menú de opciones:
1. Actualizar Stock
2. Retiro de Efectivo (ganancias)
3. Depósito de cambio
4. Ver inventario
5. Ver lista de clientes de la sucursal
6. Actualizar Información personal
7. Cerrar sesión
Seleccione una opción: 4
Inventario:
Monitor ASUS MXLZ27
Monitor HP LS2001
Monitor Dell DS247
Laptop Dell 1135G7
iPad MPQ03LZ/A
Combo Teclado Mouse Logitech K200
Laptop HP B09BMGCVZH
Laptop Alienware M15 R7
Bocinas Logitech Z207
Menú de opciones:
1. Actualizar Stock
2. Retiro de Efectivo (ganancias)
3. Depósito de cambio
4. Ver inventario
5. Ver lista de clientes de la sucursal
6. Actualizar Información personal
7. Cerrar sesión
Seleccione una opción:
```

```
Inicio de sesión Cliente
-----------------------
Ingrese su correo electrónico: ana.gonzalez@example.com
Ingrese su contraseña: contrasena1
Bienvenido, Ana González
-----------------------
Menú de opciones:
1. Buscar producto
2. Ver Carrito
3. Editar carrito
4. Comprar
5. Actualizar Datos Personales
6. Ver Mis Compras
7. Cerrar sesión
Seleccione una opción: 4
¿Desea factura? (S/N): S
Ingrese su RFC: aw2dgrr
Ingrese su dirección fiscal: coyoacan
¿Es correcta la forma de pago? (Tarjeta de Crédito) (S/N): N
Ingrese la forma de pago: Efectivo
Compra realizada con éxito.

Menú de opciones:
1. Buscar producto
2. Ver Carrito
3. Editar carrito
4. Comprar
5. Actualizar Datos Personales
6. Ver Mis Compras
7. Cerrar sesión
Seleccione una opción: 2
Carrito de compras:
```

Código de Errores:

Fecha	Proyecto	Archivo	#Error	#Línea	Tipo de Excepción	Descripción	Estatus
23/06/2023	APPSOFT	Program.cs	1	-	Ejecución.	Clases a veces no llaman a los metodos a	
23/06/2023	APPSOFT	Program.cs	1	-	Ejecución.	cuando se intenta acceder a un objeto que no ha sido creado o que ha sido eliminado. Esto puede ocurrir si se intenta acceder a un objeto antes de que se haya inicializado o si se ha cerrado una instancia del objeto antes de la finalización del programa.	Pendiente.
23/06/2023	APPSOFT	Program.cs	1	-	Ejecución.	se utiliza un método que no ha sido definido en el programa. Esto puede ocurrir si se ha escrito mal el nombre del método o si el método no se ha definido antes de su uso.	Pendiente.
23/06/2023	APPSOFT	Program.cs	1	-	Ejecución.	el programa intenta utilizar una función o biblioteca que no es compatible con el sistema operativo o la versión de software utilizada. Esto puede requerir la actualización de software o la reescritura del código para utilizar una biblioteca compatible.	Pendiente.
23/06/2023	APPSOFT	Program.cs	1	-	Ejecución.	se utiliza un método que no ha sido definido en el programa. Esto puede ocurrir si se ha escrito mal el nombre del método o si el método no se ha definido antes de su uso.	Pendiente.
23/06/2023	APPSOFT	Program.cs	1	-	Ejecución.	Errores de tiempo de ejecución: Estos errores ocurren durante la ejecución del programa y pueden ser difíciles de prever o solucionar. Pueden incluir errores de lógica, problemas de entrada/salida, o problemas de conexión con bases de datos u otros sistemas externos.	Pendiente.
23/06/2023	APPSOFT	Program.cs	1	-	Ejecución.	Falta de desarrollo en metodos.	Pendiente.
23/06/2023	APPSOFT	Program.cs	1	-	Ejecución.	Mas control para exepciones cuando se da enter.	Pendiente.

CONCLUSIONES INDIVIDUALES:

Antonio Martínez Rodrigo:

La creación de esta aplicación fue un desafío interesante y gratificante para mí como programador experimentado. Aunque encontramos algunos problemas técnicos, como la integración de la base de datos y la optimización del código, pudimos resolverlos mediante un enfoque riguroso y colaborativo. Me siento satisfecho de haber contribuido a la creación de una solución de alta calidad y estoy emocionado de seguir trabajando en proyectos desafiantes en el futuro.

Briseño Vázquez Ángel Geovany:

Estoy muy satisfecho con el trabajo realizado por nuestro equipo en la creación de esta aplicación de tienda virtual de productos de cómputo. Aunque encontramos algunos desafíos técnicos y de gestión del tiempo, como la implementación de una base de datos y la coordinación del trabajo en equipo, logramos cumplir con nuestros objetivos y entregar un producto de alta calidad. Me siento orgulloso

de haber liderado un equipo talentoso y comprometido, y estoy emocionado de seguir trabajando en proyectos desafiantes y significativos en el futuro.

Nishimura Guerrero Christian Jesús:

Durante el proceso de creación de esta aplicación, aprendí mucho sobre programación orientada a objetos y el desarrollo de aplicaciones. Aunque tuvimos algunos desafíos técnicos, como la implementación de una base de datos y la gestión de errores, logramos trabajar en equipo y superar estos obstáculos juntos. Me siento orgulloso de haber contribuido a la creación de una solución innovadora y relevante para la sociedad, y estoy emocionado de seguir aprendiendo y mejorando mis habilidades como desarrollador.

CONCLUSIONES POR EQUIPO:

Durante el proceso de creación de esta aplicación de tienda virtual de productos de cómputo, como equipo, hemos aprendido y experimentado una gran cantidad de desafíos técnicos y de gestión del proyecto. A través de nuestros esfuerzos colaborativos, hemos logrado superar estos desafíos y entregar un producto final que estamos orgullosos de presentar.

Uno de los desafíos que enfrentamos fue la implementación y unión de las clases. Aunque inicialmente encontramos dificultades en la creación, logramos crear una tabla de inventario y conectarla a la clase de productos. Este problema nos llevó a aprender más sobre el uso de clases y cómo integrarlas en una aplicación.

Otro desafío que enfrentamos fue la optimización del código. A medida que avanzamos en el proceso de desarrollo, nos dimos cuenta de que algunas partes del código eran redundantes o poco eficientes. Trabajamos juntos para identificar y corregir estos problemas, lo que resultó en una aplicación más rápida y eficiente.

Además, enfrentamos desafíos de gestión del tiempo y de coordinación del equipo. En general, estamos muy orgullosos de lo que hemos logrado como equipo en la creación de esta aplicación de tienda virtual de productos de cómputo. Hemos demostrado nuestra capacidad para trabajar juntos y superar desafíos técnicos y de gestión del proyecto. Además, hemos adquirido una valiosa experiencia en programación orientada a objetos y desarrollo de aplicaciones que nos servirá en futuros proyectos y trabajos. Estamos emocionados de seguir aprendiendo y mejorando nuestras habilidades como desarrolladores y esperamos tener la oportunidad de trabajar juntos en proyectos igualmente desafiantes y gratificantes en el futuro.