

Universidad ORT Uruguay Facultad de
Ingeniería Escuela de Tecnología

OBLIGATORIO PROGRAMACIÓN 2 DOCUMENTACION

Federico Gallo- 210578



Marcio Huertas – 279268



N2B – Remoto

Docente: Santiago Baillo

Analista en Tecnologías de la información 10/10/2024

Contenido

Diagrama de clases completo del Dominio del problema.....	3
Tabla de datos precargados:	3
Usuarios Clientes:	3
Usuarios Administradores:	4
Precarga de Artículos:	4
Subastas.....	5
Ventas:	9
Código fuente cosola:	12
Program.cs.....	12
Dominio	23
Usuario.cs	23
Administrador.cs	25
Cliente.cs.....	27
Publicacion.cs	28
Venta.cs.....	31
Subasta.cs	33
Articulo.cs	36
Estado.cs	38
Oferta.cs.....	39
Sistema.cs	39

Diagrama de clases completo del Dominio del problema.

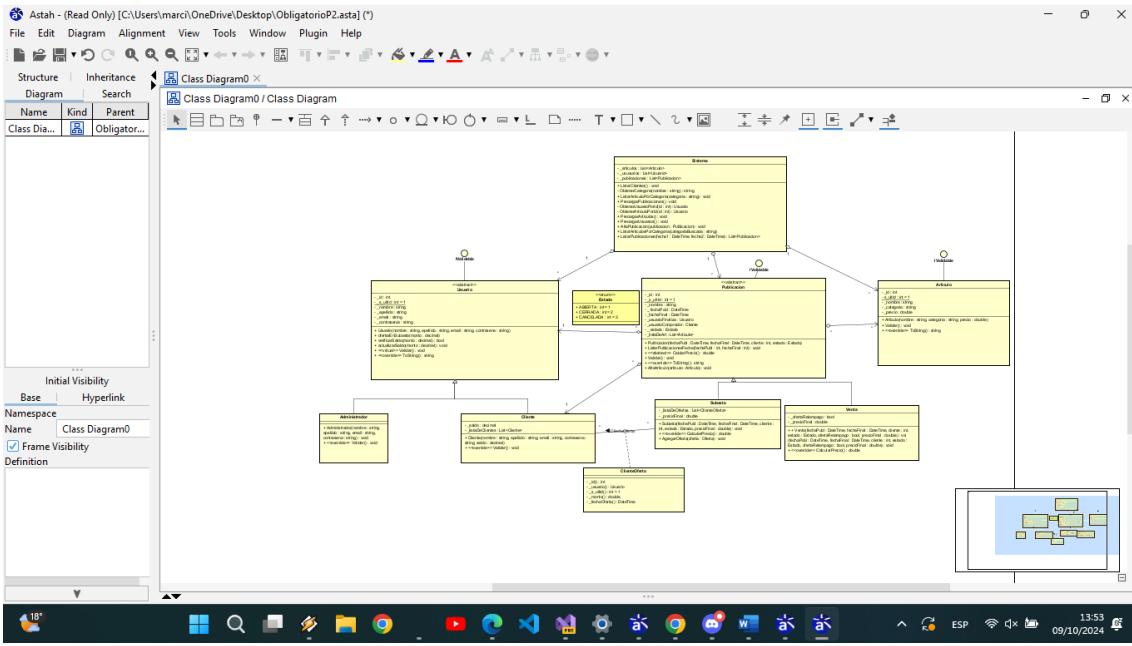


Tabla de datos precargados:

Usuarios Clientes:

Marcio Pérez, marcio@example.com, password123, 500

Federico Cuello, fede@example.com, fede123, 300

Carlos Gómez, carlos@example.com, passCarlos, 700

Sofia Rodríguez, sofia@example.com, passSofia, 200

Diego Martínez, diego@example.com, passDiego, 600

Laura Fernández, laura@example.com, passLaura, 400
Pablo Sánchez, pablo@example.com, passPablo, 350
Lucía Hernández, lucia@example.com, passLucia, 800
Javier Ramírez, javier@example.com, passJavier, 150
Valentina Ruiz, valentina@example.com, passValentina, 550
Lucía Hernández, lucia@example.com, passLucia, 800
Lucía Hernández, lucia@example.com, passLucia, 800

Usuarios Administradores:

Marcio Huertas, marciohuertasrial1995@outlook.com, marcio123
Fede Gallo, fede@outlook.com, fede123

Precarga de Artículos:

Pelota futbol, Deporte, 100
Raqueta de tenis, Deporte, 150
Bicicleta, Deporte, 300
Pesas 10kg, Deporte, 80
Balón de básquet, Deporte, 120

Teléfono móvil, Tecnología, 500
Laptop, Tecnología, 1000
Tablet, Tecnología, 300
Auriculares inalámbricos, Tecnología, 200
Monitor 24 pulgadas, Tecnología, 250

Sofá, Hogar, 700
Mesa de comedor, Hogar, 400
Lámpara de pie, Hogar, 100
Refrigerador, Hogar, 1200
Microondas, Hogar, 150

Muñeca, Juguetería, 30
Coche de juguete, Juguetería, 40
Rompecabezas 1000 piezas, Juguetería, 25
Videojuego, Juguetería, 60
Juguete de construcción, Juguetería, 90

Reloj inteligente, Tecnología, 300
Cámara fotográfica, Tecnología, 700

Teclado mecánico, Tecnología, 120

Mouse inalámbrico, Tecnología, 80

Impresora, Tecnología, 200

Cama, Hogar, 500

Silla de oficina, Hogar, 150

Aspiradora, Hogar, 180

Cafetera, Hogar, 90

Tostadora, Hogar, 50

Patinete, Deporte, 100

Casco de bicicleta, Deporte, 50

Zapatillas deportivas, Deporte, 120

Cinta para correr, Deporte, 800

Guantes de boxeo, Deporte, 60

Drone, Tecnología, 500

Consola de videojuegos, Tecnología, 600

Smart TV, Tecnología, 900

Cargador portátil, Tecnología, 40

Router WiFi, Tecnología, 120

Silla de comedor, Hogar, 100

Ventilador, Hogar, 60

Estufa, Hogar, 200

Planchita de ropa, Hogar, 70

Cortinas, Hogar, 40

Juego de mesa, Juguetería, 50

Pelota de playa, Juguetería, 15

Peluche, Juguetería, 25

Lego, Juguetería, 100

Trompo, Juguetería, 5

Subastas

1. Subasta 1

- Título: Tecnología
- Fecha Inicio: 10/10/2024

- Fecha Fin: 12/12/2024
- Cliente: Cliente 1
- Artículos: Artículo 8, Artículo 9
- Estado: Abierta
- Precio Base: 1500
- Administrador: Administrador
- Oferta: 1500 (Cliente 1)

2. Subasta 2

- Título: Balón de Basket
- Fecha Inicio: 26/10/2024
- Fecha Fin: 29/12/2024
- Cliente: Cliente 2
- Artículos: Artículo 6, Artículo 5
- Estado: Abierta
- Precio Base: 800
- Administrador: Administrador
- Oferta: 900 (Cliente 2)

3. Subasta 3

- Título: Cámara Fotográfica Profesional
- Fecha Inicio: 10/06/2024
- Fecha Fin: 12/09/2024
- Cliente: Cliente 1
- Artículos: Artículo 11, Artículo 12
- Estado: Abierta
- Precio Base: 2500
- Administrador: Administrador

4. Subasta 4

- Título: Smartphone de Última Generación

- Fecha Inicio: 12/17/2024
- Fecha Fin: 01/06/2025
- Cliente: Cliente 1
- Artículos: Artículo 5, Artículo 7
- Estado: Abierta
- Precio Base: 1200
- Administrador: Administrador

5. Subasta 5

- Título: Consola de Videojuegos
- Fecha Inicio: 10/24/2024
- Fecha Fin: 11/20/2024
- Cliente: Cliente 2
- Artículos: Artículo 26, Artículo 31
- Estado: Abierta
- Precio Base: 2000
- Administrador: Administrador

6. Subasta 6

- Título: Monitor 4K
- Fecha Inicio: 09/06/2023
- Fecha Fin: 10/10/2023
- Cliente: Cliente 1
- Artículos: Artículo 44, Artículo 49
- Estado: Abierta
- Precio Base: 1500
- Administrador: Administrador

7. Subasta 7

- Título: Tablet Samsung
- Fecha Inicio: 12/12/2022

- Fecha Fin: 01/06/2023
- Cliente: Cliente 2
- Artículos: Artículo 19, Artículo 10
- Estado: Abierta
- Precio Base: 800
- Administrador: Administrador

8. Subasta 8

- Título: Reloj Inteligente
- Fecha Inicio: 16/15/2023 (Error en la fecha)
- Fecha Fin: 20/20/2024 (Error en la fecha)
- Cliente: Cliente 1
- Artículos: Artículo 33, Artículo 40
- Estado: Abierta
- Precio Base: 700
- Administrador: Administrador

9. Subasta 9

- Título: Drone Profesional
- Fecha Inicio: 12/08/2024
- Fecha Fin: 10/10/2024
- Cliente: Cliente 2
- Artículos: Artículo 23, Artículo 37
- Estado: Abierta
- Precio Base: 3000
- Administrador: Administrador

10. Subasta 10

- Título: Kit de Herramientas
- Fecha Inicio: 10/10/2023
- Fecha Fin: 12/18/2023

- **Cliente:** Cliente 1
- **Artículos:** Artículo 15, Artículo 22
- **Estado:** Abierta
- **Precio Base:** 400
- **Administrador:** Administrador.

Ventas:

1. Venta 1

- **Título:** Kit Deporte
- **Fecha Inicio:** 10/10/2023
- **Fecha Fin:** 12/10/2024
- **Cliente:** Cliente 1
- **Artículos:** Artículo 11, Artículo 12
- **Estado:** Abierta
- **Precio:** 200

2. Venta 2

- **Título:** Artículos de Pesca
- **Fecha Inicio:** 10/05/2024
- **Fecha Fin:** 10/20/2024
- **Cliente:** Cliente 2
- **Artículos:** Artículo 5, Artículo 7
- **Estado:** Abierta
- **Precio:** 1200

3. Venta 3

- **Título:** Juego de Sillas
- **Fecha Inicio:** 11/05/2024
- **Fecha Fin:** 11/15/2024

- **Cliente:** Cliente 2
- **Artículos:** Artículo 11, Artículo 12
- **Estado:** Abierta
- **Precio:** 500

4. **Venta 4**

- **Título:** Televisor 4K + Xbox-one
- **Fecha Inicio:** 12/01/2024
- **Fecha Fin:** 12/15/2024
- **Cliente:** Cliente 2
- **Artículos:** Artículo 5, Artículo 7
- **Estado:** Abierta
- **Precio:** 1000

5. **Venta 5**

- **Título:** Juegos de Mesa
- **Fecha Inicio:** 10/01/2024
- **Fecha Fin:** 10/15/2024
- **Cliente:** Cliente 1
- **Artículos:** Artículo 11, Artículo 12
- **Estado:** Abierta
- **Precio:** 200

6. **Venta 6**

- **Título:** Libros
- **Fecha Inicio:** 10/05/2024
- **Fecha Fin:** 10/20/2024
- **Cliente:** Cliente 2
- **Artículos:** Artículo 5, Artículo 7
- **Estado:** Abierta
- **Precio:** 1200

7. **Venta 7**

- **Título:** Kit Boxeo
- **Fecha Inicio:** 09/20/2024
- **Fecha Fin:** 10/05/2024
- **Cliente:** Cliente 2
- **Artículos:** Artículo 8, Artículo 9
- **Estado:** Abierta
- **Precio:** 150

8. **Venta 8**

- **Título:** Útiles Escolares
- **Fecha Inicio:** 08/15/2024
- **Fecha Fin:** 09/01/2024
- **Cliente:** Cliente 1
- **Artículos:** Artículo 6, Artículo 5
- **Estado:** Abierta
- **Precio:** 900

9. **Venta 9**

- **Título:** Kit Limpieza
- **Fecha Inicio:** 07/10/2024
- **Fecha Fin:** 07/25/2024
- **Cliente:** Cliente 1
- **Artículos:** Artículo 11, Artículo 12
- **Estado:** Abierta
- **Precio:** 800

10. **Venta 10**

- **Título:** Kit de Vuelo
- **Fecha Inicio:** 09/25/2024
- **Fecha Fin:** 10/10/2024

- Cliente: Cliente 2
- Artículos:

Código fuente cosola:

Program.cs

using Dominio;

using System.Text.RegularExpressions;

namespace Consola

{

 internal class Program

 {

 private static Sistema miSistema;

 static void Main(string[] args)

 {

 miSistema = new Sistema();

 string opcion = "";

 while (opcion != "0")

 {

 MostrarMenu();

```
opcion = PedirPalabras("Ingrese una opcion -> ");
```

```
switch (opcion)
```

```
{
```

```
case "1":
```

```
ListarClientes();
```

```
break;
```

```
case "2":
```

```
ListarArticulosPorCategoria();
```

```
break;
```

```
case "3":
```

```
AltaArticulo();
```

```
break;
```

```
case "4":
```

```
ListarPublicacionesEntreFechas();
```

```
break;
```

```
case "0":
```

```
Console.WriteLine("Saliendo ...");
```

```
break;
```

```
default:
```

```
Console.WriteLine("Opcion incorrecta");
```

```
break;
```

```
}
```

```
}
```

```
}
```

```
#region METODOS AUXILIARES
```

```
static void MostrarMenu()
{
    Console.Clear();
    MostrarMensajeColor(ConsoleColor.Cyan, "*****");
    MostrarMensajeColor(ConsoleColor.Cyan, "  MENU  ");
    MostrarMensajeColor(ConsoleColor.Cyan, "*****");
    Console.WriteLine();
    Console.WriteLine("1 - Listar Clientes");
    Console.WriteLine("2 - Listado de Articulos por Categoria");
    Console.WriteLine("3 - Alta Articulo");
    Console.WriteLine("4 - Listado de Publicaciones dada 2 fechas"); //MENU
CONSOLA
```

```
    Console.WriteLine("0 - Salir");
}
```

```
static string PedirPalabras(string mensaje)
{
    Console.Write(mensaje);
    string datos = Console.ReadLine();
    return datos;
}
```

```
static int PedirNumeros(string mensaje)
{
    bool exito = false;
    int valorConvertido = 0;
```

```
_____ while (!exito)
_____ {
_____ Console.Write(mensaje);
_____ exito = int.TryParse(Console.ReadLine(), out valorConvertido);
```

```
_____ if (!exito)
_____ {
_____ MostrarError("ERROR: Debe ingresar solo numeros");
_____ }
_____ }
```

```
_____ return valorConvertido;
_____ }
```

```
_____ static DateTime PedirFecha(string mensaje)
_____ {
_____ bool exito = false;
_____ DateTime fecha = new DateTime();
```

```
_____ while (!exito)
_____ {
_____ Console.Write($"{mensaje} [dd/MM/yyyy]:");
_____ exito = DateTime.TryParse(Console.ReadLine(), out fecha);
```

```
_____ if (!exito)
_____ {
_____ MostrarError("ERROR: La fecha no respeta el formato dd/MM/yyyy");
_____ }
```

_____}

_____return fecha;

_____}

_____static double PedirNumerosDouble(string mensaje)

_____{"

_____bool exito = false;

_____double valorConvertido = 0;

_____while (!exito)

_____{"

_____Console.Write(mensaje);

_____exito = double.TryParse(Console.ReadLine(), out valorConvertido);

_____if (!exito)

_____{"

_____MostrarError("ERROR: Debe ingresar solo numeros");

_____}"

_____}"

_____return valorConvertido;

_____}"

_____static void MostrarMensajeColor(ConsoleColor color1, string mensaje)

_____{"

_____Console.ForegroundColor = color1;

_____Console.WriteLine(mensaje);


```
Console.ForegroundColor = ConsoleColor.Gray;  
}
```

```
static void MostrarError(string mensaje)  
{  
Console.ForegroundColor = ConsoleColor.Red;  
Console.WriteLine(mensaje);  
Console.ForegroundColor = ConsoleColor.Gray;  
}
```

```
static void MostrarExito(string mensaje)  
{  
Console.ForegroundColor = ConsoleColor.Green;  
Console.WriteLine(mensaje);  
Console.ForegroundColor = ConsoleColor.Gray;  
}
```

```
static void PressToContinue()  
{  
Console.WriteLine();  
Console.WriteLine("Presione cualquier tecla para volver al menu...");  
Console.ReadKey();  
}
```

```
#endregion
```

```
#region METODOS DE MENU
```

```
static void Saludo()
{
    Console.Clear();

    MostrarMensajeColor(ConsoleColor.Yellow, "FUNCIONALIDAD SALUDO");

    Console.WriteLine();

    Console.Write("Ingrese nombre: ");

    string nombre = Console.ReadLine();

    Console.WriteLine($"Bienvenido {nombre}");

    PressToContinue();
}

static void ListarClientes()
{
    Console.Clear();

    MostrarMensajeColor(ConsoleColor.Yellow, "Lista de Clientes");

    Console.WriteLine();

    try
    {
        // Obtener la lista de clientes desde el sistema

        List<Cliente> todosLosClientes = miSistema.ListarClientes();

        // Verificar si la lista está vacía

        if (todosLosClientes == null || todosLosClientes.Count == 0)
        {
            MostrarError("No existen Clientes en el sistema.");
        }
    }
}
```

```
_____ else
_____ {
_____ // Mostrar los clientes en formato limpio
_____ foreach (Cliente cliente in todosLosClientes)
_____ {
_____ Console.WriteLine(cliente.ToString()); // Asume que tienes un método
ToString en Cliente
_____ }
_____ }
_____ }
_____ catch (Exception ex)
_____ {
_____ MostrarError($"Error al listar los clientes: {ex.Message}");
_____ }

_____ PressToContinue(); // Esperar para continuar
_____ }
```

```
_____ static void ListarArticulosPorCategoria()
_____ {
_____ Console.Clear();
_____ MostrarMensajeColor(ConsoleColor.Yellow, "Lista de Artículos por Categoría");
_____ Console.WriteLine();

_____ try
_____ {
```

```
_____ // Pedir la categoría al usuario
_____ string categoriaBuscada = PedirPalabras("Ingrese la categoría: ");

_____ // Obtener artículos filtrados por categoría
_____ List<Articulo> articulosFiltrados =
_____ miSistema.ListarArticulosPorCategoria(categoriaBuscada);

_____ // Verificar si no hay artículos en la categoría buscada
_____ if (articulosFiltrados == null || articulosFiltrados.Count == 0)
_____ {
_____     MostrarError($"No hay artículos en la categoría '{categoriaBuscada}'.");
_____ }
_____ else
_____ {
_____     // Mostrar los artículos filtrados
_____     foreach (Articulo articulo in articulosFiltrados)
_____     {
_____         Console.WriteLine(articulo.ToString()); // Asume que tienes un método
_____         ToString en Articulo
_____     }
_____ }
_____ }
_____ catch (Exception ex)
_____ {
_____     MostrarError($"Error al listar los artículos por categoría: {ex.Message}");
_____ }

_____ PressToContinue(); // Esperar para continuar
_____ }
```

```
static void AltaArticulo()
{
    Console.Clear();

    MostrarMensajeColor(ConsoleColor.Yellow, "Alta de Articulos");

    Console.WriteLine();

    string nombre = PedirPalabras("Ingrese nombre : ");
    string categoria = PedirPalabras("Ingrese categoria: ");
    double precio = PedirNumerosDouble("Ingrese precio: ");

    try
    {
        if (string.IsNullOrEmpty(nombre)) // Valida si el nombre está vacío o es
solo espacios
        {
            throw new Exception("El nombre no puede estar vacío.");
        }

        if (string.IsNullOrEmpty(categoria)) // Valida si la categoría está vacía o
es solo espacios
        {
            throw new Exception("La categoría no puede estar vacía.");
        }

        if (precio <= 0) // Valida que el precio sea mayor que cero
        {
            throw new Exception("El precio debe ser mayor a cero.");
        }
    }
}
```

```
_____ miSistema.AltaArticulo(new Articulo(nombre, categoria, precio));  
_____ MostrarExito("Artículo registrado exitosamente.");  
_____  
_____ catch (Exception ex)  
_____  
_____ {  
_____ MostrarError($"Error: {ex.Message}");  
_____  
_____ }  
  
_____ PressToContinue();  
_____  
  
_____ static void ListarPublicacionesEntreFechas()  
_____  
_____ {  
_____ Console.Clear();  
_____ MostrarMensajeColor(ConsoleColor.Yellow, "Listar Publicaciones entre Fechas");  
_____ Console.WriteLine();  
  
_____ DateTime fecha1 = PedirFecha("Ingrese la primera fecha: ");  
_____ DateTime fecha2 = PedirFecha("Ingrese la segunda fecha: ");  
  
_____ try  
_____ {  
_____ List<Publicacion> publicacionesFiltradas = miSistema.ListarPublicaciones(  
_____ fecha1, fecha2);  
_____ if (publicacionesFiltradas == null || publicacionesFiltradas.Count == 0)  
_____ {  
_____ throw new Exception("No hay publicaciones entre esas fechas.");
```

```

    }

    foreach (Publicacion publicacion in publicacionesFiltradas)
    {
        Console.WriteLine($"{publicacion.Id} - {publicacion.Nombre} -
        {publicacion.Estado} - {publicacion.FechaPublic}");
    }
}

catch (Exception ex)
{
    MostrarError(ex.Message);
}

PressToContinue();
}

#endregion
}
}

```

Dominio

Usuario.cs

```

using Dominio.Interfaces;

using System;

using System.Collections.Generic;

```

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Dominio

{

public abstract class Usuario : IValidable

{

private int id;

private static int s_ultId = 1;

private string nombre;

private string apellido;

private string email;

private string contrasena;

protected Usuario(string nombre, string apellido, string email, string contrasena)

{

id = s_ultId;

s_ultId++;

nombre = nombre;

apellido = apellido;

email = email;

contrasena = contrasena;

}

public string Nombre

{


```
_____get { return nombre;}  
_____  
_____}
```

```
_____public int Id  
_____ { get { return id; } }
```

```
_____public virtual void Validar()  
_____  
_____if (string.IsNullOrEmpty( nombre)) throw new ArgumentNullException("El  
nombre no puede ser vacio");  
_____if (string.IsNullOrEmpty( apellido)) throw new ArgumentNullException("El  
nombre no puede ser vacio");  
_____if (string.IsNullOrEmpty( contrasena)) throw new ArgumentNullException("La  
contraseña no puede ser vacio"); //VALIDACIONES BASICAS.  
_____if (string.IsNullOrEmpty( email)) throw new ArgumentNullException("El emial  
no puede ser vacio");  
_____}
```

```
_____  
_____}
```

[Administrador.cs](#)

using Dominio.Interfaces;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Dominio

{

public class Administrador : Usuario, IValidable

{

public Administrador(string nombre, string apellido, string email, string
contrasena)

: base(nombre, apellido, email, contrasena)

{

}

public override string ToString()

{

return \$"{Nombre}"; // CORREGIR

}

public override void Validar()

{

base.Validar();

}

}

}

Cliente.cs

using Dominio.Interfaces;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Dominio

{

public class Cliente : Usuario, IValidable

{

private decimal saldo;

// Constructor que llama al constructor de la clase base (Usuario)

public Cliente(string nombre, string apellido, string email, string contrasena,
decimal saldo):base(nombre, apellido, email, contrasena)

{

saldo = saldo;

}

```

    public decimal Saldo
    {
        get { return saldo; }

    }

    public override string ToString()
    {
        return $"{Nombre}, Saldo: { saldo}";
    }

    public override void Validar()
    {
        base.Validar();
    }
}

```

Publicacion.cs

```

using Dominio.Interfaces;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

```

namespace Dominio

{

public class Publicacion : IValidable

{

private int id;

private static int s_ultId = 1;

private string nombre;

private DateTime fechaPublic;

private DateTime fechaFinaliz;

private Usuario usuarioFinaliza;

private Cliente usuarioComprador;

private List<Articulo> articulos = new List<Articulo>();

private Estado estado;

public Publicacion(string name, DateTime fechaPublic, DateTime fechaFinaliz,
Cliente cliente, List<Articulo> articulos, Estado estado)

{

nombre = name;

id = s_ultId;

s_ultId++;

fechaPublic = fechaPublic;

fechaFinaliz = fechaFinaliz;

usuarioFinaliza = cliente;

articulos = articulos;

estado = estado;

}

```
____ public int Id
____ { get { return id; } }
____ public string Estado
____ {
____ get { return estado.ToString(); }
____ }
```

```
____ public string Nombre
____ {
____ get { return nombre; }
____ }
```

```
____ public List<Articulo> Articulos
____ { get { return articulos; } }
```

```
____ public DateTime FechaPublic
____ {
____ get { return fechaPublic; }
____ }
```

```
____ public DateTime FechaFinaliz
____ {
____ get { return fechaFinaliz; }
____ }
```

```
____ public virtual void Validar()
____ {
____ if (string.IsNullOrEmpty( nombre))
____ throw new ArgumentNullException("El nombre no puede ser vacio");
```

```

        if ( fechaFinaliz <= fechaPublic)
            throw new Exception ("La fecha de finalización debe ser posterior a la fecha
de publicación."); //VALIDACIONES BASICAS.

    }

    public override string ToString()
    {
        return $"Publicación: Id: { id}Nombre: { nombre}, Estado: { estado}, Artículos:
{Articulos.Count}";
    }
}
}
}

```

Venta.cs

```

using Dominio.Interfaces;
using System;
using System.Collections.Generic;

namespace Dominio
{
    public class Venta : Publicacion, IValidable
    {
        private bool ofertaRelampago;

        private decimal precioFinal;
    }
}

```

```

____private Usuario comprador;

____// Atributos específicos de la clase Venta
____public bool OfertaRelampago
____{
____    get { return ofertaRelampago; }
____}

____public decimal PrecioFinal
____{
____    get { return precioFinal; }
____}

____public Venta(string nombre, DateTime fechaPublicacion, DateTime
fechaFinalizacion, Cliente cliente, List<Articulo> articulos, Estado estado, bool
ofertaRelampago, decimal precioFinal)
____: base(nombre, fechaPublicacion, fechaFinalizacion, cliente, articulos, estado) //
Llama al constructor de Publicacion
____{
____    ofertaRelampago = ofertaRelampago;
____    precioFinal = precioFinal;
____}

____public override void Validar()
____{
____    base.Validar();
____}

```



```

____ // Método ToString para facilitar la visualización
____ public override string ToString()
____ {
____     return $"Venta: {Nombre}, Precio Final: { _precioFinal}, Oferta Relámpago:
____     { ofertaRelampago}";
____ }
____ }
____ }
____ }

```

Subasta.cs

```

____ using System;
____ using System.Collections.Generic;

____ namespace Dominio
____ {
____     public class Subasta : Publicacion
____     {
____         // Atributos específicos de la clase Subasta
____         private List<Oferta> ofertas; // Lista de ofertas realizadas por clientes
____         private decimal _precioFinal;
____         private Cliente _cliente;
____         private Administrador _usuario;

____         //Propiedades solo de lectura para acceder a los atributos
____         public List<Oferta> Ofertas
____         {

```

_____get { return ofertas; }

_____}

_____public decimal PrecioFinal

_____ {

_____get { return precioFinal; }

_____}

_____public Cliente Cliente

_____ {

_____get { return cliente; }

_____}

_____public Administrador Administrador

_____ {

_____get { return usuario; }

_____}

_____// Constructor de la clase Subasta

_____public Subasta(string nombre, DateTime fechaPublicacion, DateTime
fechaFinalizacion, Cliente cliente, List<Articulo> articulos, Estado estado, decimal
precioFinal, Administrador admin, List<Oferta> ofertas = null)

_____ : base(nombre, fechaPublicacion, fechaFinalizacion, cliente, articulos, estado)

_____ {

_____precioFinal = precioFinal;

_____usuario = admin;

_____ofertas = ofertas ?? new List<Oferta>();

_____}

```
____ public void AgregarOferta(Oferta nuevaOferta)
____ {
____     if (nuevaOferta == null) throw new Exception("La oferta no puede ser nula");

____
____     // Verificar si hay ofertas previas
____     if ( ofertas.Count > 0) // Si hay al menos una oferta
____     {
____         // Tomar la última oferta (la más alta debería ser la última agregada)
____         Oferta ofertaActual = ofertas[ ofertas.Count - 1]; // Última oferta en la lista

____
____         // Comparar el monto de la nueva oferta con la última oferta
____         if (nuevaOferta.Monto <= ofertaActual.Monto)
____         {
____             throw new Exception("La nueva oferta debe ser mayor que la oferta
actual.");
____         }
____     }

____
____     // Si la nueva oferta es válida, agregarla a la lista
____     ofertas.Add(nuevaOferta);
____ }

____
____ //oferta mayor y cliente que no haya otra oferta

____
____ // Método ToString para facilitar la visualización
____ public override string ToString()
____ {
```

```

        return $"Subasta: {Nombre}, Precio Final: { _precioFinal}, Ofertas:
        {Ofertas.Count}";
    }

    public override void Validar()
    {
        base.Validar();
    }
}
}

```

Articulo.cs

```

using Dominio.Interfaces;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Dominio
{
    public class Articulo : IValidable
    {
        private int _id;
    }
}

```

```
____ private static int s_ultId = 1;

____ private string nombre;

____ private string categoria;

____ private double precio;


____ public Artículo( string nombre, string categoria, double precio)
____ {
____     id = s_ultId;
____     s_ultId++;
____     nombre = nombre;
____     categoria = categoria;
____     precio = precio;
____ }

____ public string Categoria
____ {
____     get { return categoria; }

____ }


____ public int Id
____ {
____     get { return id; }

____ }

____ public void Validar()
____ {
____     if (string.IsNullOrEmpty( nombre)) throw new ArgumentNullException("El
nombre no puede estar vacio");
```

```
_____ if (string.IsNullOrEmpty( categoria)) throw new ArgumentNullException("la  
categoria puede estar vacio"); //VALIDACIONES BASICAS.
```

```
_____ if ( precio < 0) throw new ArgumentOutOfRangeException("El precio no puede  
ser negativo");
```

```
_____ }
```

```
_____ public override string ToString()
```

```
_____ {
```

```
_____ return $"Nombre: { nombre}, Categoría: { categoria}, Precio: { precio}";
```

```
_____ }
```

```
_____ }
```

```
_____ }
```

Estado.cs

```
_____ using System;
```

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

```
_____ using System.Linq;
```

```
_____ using System.Text;
```

```
_____ using System.Threading.Tasks;
```

```
_____ namespace Dominio
```

```
_____ {
```

```
_____ public enum Estado
```

```
_____ {
```

```
_____ ABIERTA = 1,
```

```
_____ CERRADA = 2,
```

CANCELADA = 3

}

}

Oferta.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

Sistema.cs

namespace Dominio

{

public class Sistema

{

//LISTAS GENERALES DE SISTEMA

private List<Articulo> articulos = new List<Articulo>();

private List<Oferta> ofertas = new List<Oferta>();

private List<Usuario> usuarios = new List<Usuario>();

private List<Publicacion> publicaciones = new List<Publicacion>();

```
____public Sistema()  
____{  
____    PrecargarUsuarios();  
____    PrecargarArticulos();  
____    PrecargarPublicaciones();  
  
____}
```

```
____//METODO PARA PRECARGAR PUBLICACIONES  
____private void PrecargarPublicaciones()  
____{  
____    // Obtener clientes y administrador por sus ID  
____    Cliente cliente1 = (Cliente)ObtenerUsuarioPorId(2); //METODO PARA BUSCAR  
____    UN USUAURIO DADO UN ID POR PARAMETRO  
____    Cliente cliente2 = (Cliente)ObtenerUsuarioPorId(3);  
____    Usuario usuario1 = ObtenerUsuarioPorId(1);  
____    Administrador administrador = usuario1 as Administrador; // Conversion de  
____    usuario1 a adminsitrador mediante palabra res as  
  
____    // Artículos para las publicaciones  
____    List<Articulo> articulos1 = new List<Articulo>
```



```
_{  
    ObtenerArticuloPorId(8),  
    ObtenerArticuloPorId(9),
```

```
};
```

```
List<Articulo> articulos2 = new List<Articulo>
```

```
{  
    ObtenerArticuloPorId(6),  
    ObtenerArticuloPorId(5),
```

```
};
```

```
List<Articulo> articulos3 = new List<Articulo>
```

```
{  
    ObtenerArticuloPorId(11),  
    ObtenerArticuloPorId(12)
```

```
};
```

```
List<Articulo> articulos4 = new List<Articulo>
```

```
{  
    ObtenerArticuloPorId(5),  
    ObtenerArticuloPorId(7)
```

```
};
```

```
List<Articulo> articulos5 = new List<Articulo>
```

```
{
```

```
    ObtenerArticuloPorId(26),  
    ObtenerArticuloPorId(31)  
};
```

```
    List<Articulo> articulos6 = new List<Articulo>  
{  
    ObtenerArticuloPorId(44),  
    ObtenerArticuloPorId(49)  
};
```

```
    List<Articulo> articulos7 = new List<Articulo>  
{  
    ObtenerArticuloPorId(19),  
    ObtenerArticuloPorId(10)  
};
```

```
    List<Articulo> articulos8 = new List<Articulo>  
{ ObtenerArticuloPorId(33),  
    ObtenerArticuloPorId(40)};
```

```
    List<Articulo> articulos9 = new List<Articulo>  
{ ObtenerArticuloPorId(23),  
    ObtenerArticuloPorId(37)};
```

```
    List<Articulo> articulos10 = new List<Articulo>  
{ ObtenerArticuloPorId(15),
```

ObtenerArticuloPorId(22));

// Subasta 1

if (administrador != null)

{

Subasta subasta1 = new Subasta(

"Tecnologia",

new DateTime(10 / 10 / 2024),

new DateTime(10, 12 / 2024),

cliente1,

articulos1,

Estado.ABIERTA,

1500,

administrador

);

Oferta oferta1 = new Oferta(cliente1, 1500, new DateTime(2024, 11, 15));

subasta1.AgregarOferta(oferta1); //agregamos oferta a subasta

AltaPublicacion(subasta1);

}

// Subasta 2

if (administrador != null)

```
_____{  
____Subasta subasta2 = new Subasta(  
____    "Balon Basket",  
____    new DateTime(15 / 26 / 2024),  
____    new DateTime(22 / 29 / 2024),  
____    cliente2,  
____    articulos2,  
____    Estado.ABIERTA,  
____    800,  
____    administrador  
____);  
____Oferta oferta2 = new Oferta(cliente2, 900, new DateTime(2024, 12, 01));  
____subasta2.AgregarOferta(oferta2);  
____AltaPublicacion(subasta2);  
____}
```

```
____    if (administrador != null)  
____    {  
____        Subasta subasta3 = new Subasta(  
____            "Cámara Fotográfica Profesional",  
____            new DateTime(06, 10, 2024),  
____            new DateTime(09, 12, 2024),  
____            cliente1,  
____            articulos3,  
____            Estado.ABIERTA,  
____            2500,  
____            administrador
```

_____);

_____}

_____// Subasta 4

_____if (administrador != null)

_____{"

_____Subasta subasta4 = new Subasta(

_____ "Smartphone de Última Generación",

_____ new DateTime(17, 12, 2024),

_____ new DateTime(06, 01, 2025),

_____ cliente1,

_____ articulos4,

_____ Estado.ABIERTA,

_____ 1200,

_____ administrador

_____);

_____}

_____// Subasta 5

_____if (administrador != null)

_____{"

_____Subasta subasta5 = new Subasta(

_____ "Consola de Videojuegos",

_____ new DateTime(24, 10, 2024),

_____ new DateTime(20, 11, 2024),

_____ cliente2,

```
_____articulos5,  
____Estado.ABIERTA,  
____2000,  
____administrador  
____);  
  
_____}
```

```
_____// Subasta 6  
____if (administrador != null)  
____{  
____Subasta subasta6 = new Subasta(  
____"Monitor 4K",  
____new DateTime(06, 09, 2023),  
____new DateTime(10, 10, 2023),  
____cliente1,  
____articulos6,  
____Estado.ABIERTA,  
____1500,  
____administrador  
____);  
  
_____}
```

```
_____// Subasta 7  
____if (administrador != null)  
____{  
____Subasta subasta7 = new Subasta(  
____
```

```
        "Tablet Samsung",  
        new DateTime(12, 12, 2022),  
        new DateTime(06, 01, 2023),  
        cliente2,  
        articulos7,  
        Estado.ABIERTA,  
        800,  
        administrador  
    );
```

```
    }
```

```
    // Subasta 8  
    if (administrador != null)  
    {  
        Subasta subasta8 = new Subasta(  
            "Reloj Inteligente",  
            new DateTime(15, 16, 2023),  
            new DateTime(20, 20, 2024),  
            cliente1,  
            articulos8,  
            Estado.ABIERTA,  
            700,  
            administrador  
        );
```

```
    }
```

```
____ // Subasta 9
____ if (administrador != null)
____ {
____     Subasta subasta9 = new Subasta(
____         "Drone Profesional",
____         new DateTime(08, 12, 2024),
____         new DateTime(10, 10, 2024),
____         cliente2,
____         articulos9,
____         Estado.ABIERTA,
____         3000,
____         administrador
____     );
```

```
____ }
```

```
____ // Subasta 10
____ if (administrador != null)
____ {
____     Subasta subasta10 = new Subasta(
____         "Kit de Herramientas",
____         new DateTime(10, 10, 2023),
____         new DateTime(18, 12, 2023),
____         cliente1,
____         articulos10,
____         Estado.ABIERTA,
____         400,
____         administrador
```


_____);

_____}

_____// Venta 1

_____Venta venta1 = new Venta(

_____ "Kit deporte",

_____ new DateTime(2023, 10, 10),

_____ new DateTime(2024, 12, 10),

_____ cliente1,

_____ articulos3,

_____ Estado.ABIERTA,

_____ true,

_____ 200

_____);

_____AltaPublicacion(venta1);

_____// Venta 2

_____Venta venta2 = new Venta(

_____ "Articulos de pesca",

_____ new DateTime(2024, 10, 05),

_____ new DateTime(2024, 10, 20),

_____ cliente2,

_____ articulos4,

_____ Estado.ABIERTA,

_____ false,

_____ 1200

_____);

AltaPublicacion(venta2);

// Venta 3

Venta venta3 = new Venta(

"Juego de sillas",

new DateTime(2024, 11, 05),

new DateTime(2024, 11, 15),

cliente2,

articulos3,

Estado.ABIERTA,

true,

500

);

AltaPublicacion(venta3);

// Venta 4

Venta venta4 = new Venta(

"Televisor 4K + Xbox-one",

new DateTime(2024, 12, 01),

new DateTime(2024, 12, 15),

cliente2,

articulos4,

Estado.ABIERTA,

true,

1000

);

AltaPublicacion(venta4);

```
// Venta 5  
Venta venta5 = new Venta(  
    "Juegos de mesa",  
    new DateTime(2024, 10, 01),  
    new DateTime(2024, 10, 15),  
    cliente1,  
    articulos3,  
    Estado.ABIERTA,  
    true,  
    200  
);  
AltaPublicacion(venta5);
```

```
// Venta 6  
Venta venta6 = new Venta(  
    "Libros",  
    new DateTime(2024, 10, 05),  
    new DateTime(2024, 10, 20),  
    cliente2,  
    articulos4,  
    Estado.ABIERTA,  
    false,  
    1200  
);  
AltaPublicacion(venta6);
```

```
// Venta 7  
Venta venta7 = new Venta(
```

```
_____"Kit Boxeo",  
____new DateTime(2024, 09, 20),  
____new DateTime(2024, 10, 05),  
____cliente2,  
____articulos1,  
____Estado.ABIERTA,  
____false,  
____150  
____);  
____AltaPublicacion(venta7);
```

```
____// Venta 8  
____Venta venta8 = new Venta(  
____"Utiles escolares",  
____new DateTime(2024, 08, 15),  
____new DateTime(2024, 09, 01),  
____cliente1,  
____articulos2,  
____Estado.ABIERTA,  
____true,  
____900  
____);  
____AltaPublicacion(venta8);
```

```
____// Venta 9  
____Venta venta9 = new Venta(  
____"Kit limpieza",  
____new DateTime(2024, 07, 10),
```

```
        new DateTime(2024, 07, 25),  
        cliente1,  
        articulos3,  
        Estado.ABIERTA,  
        false,  
        800  
    );  
    AltaPublicacion(venta9);
```

```
    // Venta 10  
    Venta venta10 = new Venta(  
        "Kit de vuelo",  
        new DateTime(2024, 09, 25),  
        new DateTime(2024, 10, 10),  
        cliente2,  
        articulos5,  
        Estado.ABIERTA,  
        true,  
        250  
    );  
    AltaPublicacion(venta10);
```

```
    }
```

```
    public Usuario ObtenerUsuarioPorId(int id)
```

```
    {
```

```
____ Usuario buscada = null;

____ int i = 0;

____ while (i < usuarios.Count && buscada == null)
____ {
____     if ( usuarios[i].Id == id) buscada = usuarios[i];
____     i++;
____ }
```

```
____ return buscada;
____ }
```

```
____ public Articulo ObtenerArticuloPorId(int id)
____ {
____     Articulo buscada = null;

____     int i = 0;

____     while (i < articulos.Count && buscada == null)
____     {
____         if ( articulos[i].Id == id) buscada = articulos[i];
____         i++;
____     }
```

```
____ return buscada;
____ }
```

```
____ //precarga de articulos

____ private void PrecargarArticulos()
____ {
____     AltaArticulo(new Articulo("Pelota futbol", "deporte", 100));
```

AltaArticulo(new Articulo("Pelota futbol", "Deporte", 100));

AltaArticulo(new Articulo("Raqueta de tenis", "Deporte", 150));

AltaArticulo(new Articulo("Bicicleta", "Deporte", 300));

AltaArticulo(new Articulo("Pesas 10kg", "Deporte", 80));

AltaArticulo(new Articulo("Balón de básquet", "Deporte", 120));

AltaArticulo(new Articulo("Teléfono móvil", "Tecnología", 500));

AltaArticulo(new Articulo("Laptop", "Tecnología", 1000));

AltaArticulo(new Articulo("Tablet", "Tecnología", 300));

//precargas de articulos

AltaArticulo(new Articulo("Auriculares inalámbricos", "Tecnología", 200));

AltaArticulo(new Articulo("Monitor 24 pulgadas", "Tecnología", 250));

AltaArticulo(new Articulo("Sofá", "Hogar", 700));

AltaArticulo(new Articulo("Mesa de comedor", "Hogar", 400));

AltaArticulo(new Articulo("Lámpara de pie", "Hogar", 100));

AltaArticulo(new Articulo("Refrigerador", "Hogar", 1200));

AltaArticulo(new Articulo("Microondas", "Hogar", 150));

AltaArticulo(new Articulo("Muñeca", "Juguetería", 30));

AltaArticulo(new Articulo("Coche de juguete", "Juguetería", 40));

AltaArticulo(new Articulo("Rompecabezas 1000 piezas", "Juguetería", 25));

AltaArticulo(new Articulo("Videojuego", "Juguetería", 60));

AltaArticulo(new Articulo("Juguete de construcción", "Juguetería", 90));

AltaArticulo(new Articulo("Reloj inteligente", "Tecnología", 300));

AltaArticulo(new Articulo("Cámara fotográfica", "Tecnología", 700));

AltaArticulo(new Articulo("Teclado mecánico", "Tecnología", 120));

AltaArticulo(new Articulo("Mouse inalámbrico", "Tecnología", 80));

AltaArticulo(new Articulo("Impresora", "Tecnología", 200));

AltaArticulo(new Articulo("Cama", "Hogar", 500));

AltaArticulo(new Articulo("Silla de oficina", "Hogar", 150));

AltaArticulo(new Articulo("Aspiradora", "Hogar", 180));

AltaArticulo(new Articulo("Cafetera", "Hogar", 90));

AltaArticulo(new Articulo("Tostadora", "Hogar", 50));

AltaArticulo(new Articulo("Patinete", "Deporte", 100));

AltaArticulo(new Articulo("Casco de bicicleta", "Deporte", 50));

AltaArticulo(new Articulo("Zapatillas deportivas", "Deporte", 120));

AltaArticulo(new Articulo("Cinta para correr", "Deporte", 800));

AltaArticulo(new Articulo("Guantes de boxeo", "Deporte", 60));

AltaArticulo(new Articulo("Drone", "Tecnología", 500));

AltaArticulo(new Articulo("Consola de videojuegos", "Tecnología", 600));

AltaArticulo(new Articulo("Smart TV", "Tecnología", 900));

AltaArticulo(new Articulo("Cargador portátil", "Tecnología", 40));

AltaArticulo(new Articulo("Router WiFi", "Tecnología", 120));

AltaArticulo(new Articulo("Silla de comedor", "Hogar", 100));

AltaArticulo(new Articulo("Ventilador", "Hogar", 60));

AltaArticulo(new Articulo("Estufa", "Hogar", 200));

AltaArticulo(new Articulo("Planchita de ropa", "Hogar", 70));

AltaArticulo(new Articulo("Cortinas", "Hogar", 40));

AltaArticulo(new Articulo("Juego de mesa", "Juguetería", 50));


```
AltaArticulo(new Articulo("Pelota de playa", "Juguetería", 15));  
AltaArticulo(new Articulo("Peluche", "Juguetería", 25));  
AltaArticulo(new Articulo("Lego", "Juguetería", 100));  
AltaArticulo(new Articulo("Trompo", "Juguetería", 5));  
}
```

```
//precarga de usuarios(clientes y administradores)  
private void PrecargarUsuarios()  
{  
AltaUsuario(new Cliente("Marcio", "Pérez", "marcio@example.com",  
"password123", 500));  
AltaUsuario(new Cliente("Federico", "Cuello", "fede@example.com", "fede123",  
300));  
AltaUsuario(new Cliente("Carlos", "Gómez", "carlos@example.com",  
"passCarlos", 700));  
AltaUsuario(new Cliente("Sofia", "Rodríguez", "sofia@example.com",  
"passSofia", 200));  
AltaUsuario(new Cliente("Diego", "Martínez", "diego@example.com",  
"passDiego", 600)); //Precarga de usuarios  
AltaUsuario(new Cliente("Laura", "Fernández", "laura@example.com",  
"passLaura", 400));  
AltaUsuario(new Cliente("Pablo", "Sánchez", "pablo@example.com",  
"passPablo", 350));  
AltaUsuario(new Cliente("Lucía", "Hernández", "lucia@example.com",  
"passLucia", 800));  
AltaUsuario(new Cliente("Javier", "Ramírez", "javier@example.com",  
"passJavier", 150));  
AltaUsuario(new Cliente("Valentina", "Ruiz", "valentina@example.com",  
"passValentina", 550));
```

```
____ AltaUsuario(new Cliente("Lucía", "Hernández", "lucia@example.com",  
____ "passLucia", 800));  
  
____ AltaUsuario(new Cliente("Lucía", "Hernández", "lucia@example.com",  
____ "passLucia", 800));  
  
____ AltaUsuario(new Administrador("Marcio", "Huertas",  
____ "marciohuertasrial1995@outlook.com", "marcio"));  
  
____ AltaUsuario(new Administrador("Fede", "Gallo", "fede@outlook.com", "fede"));
```

```
____ }
```

```
____ public void AltaPublicacion(Publicacion publicacion)  
____ {  
____ if (publicacion == null) throw new Exception("El articulo no puede ser nulo");  
____ publicacion.Validar();  
  
____ publicaciones.Add(publicacion);  
____ }
```

```
____ //Metodo que devuelve una lista de Clientes  
____ public List<Cliente> ListarClientes()  
____ {  
____ // Crear una lista de clientes filtrando la liista de usuarios  
____ List<Cliente> clientesFiltrados = new List<Cliente>();  
  
____ foreach (Usuario usuario in usuarios)  
____ {
```

```
_____ if (usuario is Cliente cliente)
_____ {
_____ clientesFiltrados.Add(cliente);
_____ }
_____ }
```

```
_____ return clientesFiltrados;
_____ }
```

```
_____ public void AltaArticulo(Articulo articulo)
_____ {
_____ if (articulo == null) throw new Exception("El articulo no puede ser nulo");
_____ articulo.Validar();
```

```
_____ articulos.Add(articulo);
_____ }
```

```
_____ public void AltaUsuario(Usuario usuario)
_____ {
_____ if (usuario == null) throw new ArgumentNullException("El usuario no puede ser
nulo");
_____ usuario.Validar();
_____ usuarios.Add(usuario);
_____ }
```

```
_____ public List<Articulo> ListarArticulosPorCategoria(string categoriaBuscada)
_____ {
_____ // convertimos la categoría buscada a minúsculas
_____ string categoriaBuscadaLower = categoriaBuscada.ToLower();

_____ List<Articulo> articulosFiltrados = new List<Articulo>();
_____ foreach (Articulo art in articulos)
_____ {
_____ if (art.Categoria.ToLower() == categoriaBuscadaLower) //si las categoria
_____ coincide con la categoria ingresada como parametro, agregamos a la lista el articulo
_____ {
_____ articulosFiltrados.Add(art);
_____ }
_____ }

_____ return articulosFiltrados;
_____ }

_____ public List <Publicacion> ListarPublicaciones(DateTime fecha1, DateTime fecha2)
_____ {
_____ List <Publicacion> publicaciones = new List<Publicacion>();

_____ foreach(Publicacion p in publicaciones)
_____ {

_____ if (p.FechaPublic >= fecha1 && p.FechaFinaliz <= fecha2)
_____ publicaciones.Add(p);
_____ }
```

```
_____}
```

```
_____return publicaciones;
```

```
_____}
```

```
_____}
```

```
_____}
```

Evidencia de las consultas realizadas a la inteligencia artificial
para la precarga.


Precargas de Ventas, Subastas, Administradores, Clientes

ChatGPT

Memoria completa

Compartir

M



Aquí te dejo el código completado hasta llegar a 10 ventas:

csharp


Copiar código

```
// Venta 1
Venta venta1 = new Venta(
    "Kit deporte",
    new DateTime(2024, 10, 01),
    new DateTime(2024, 10, 15),
    cliente1,
    articulos3,
    Estado.ABIERTA,
    true,
    200
);
AltaPublicacion(venta1);

// Venta 2
Venta venta2 = new Venta(
```

Envía un mensaje a ChatGPT

ChatGPT puede cometer errores. Comprueba la información importante.




15:14
10/10/2024

ChatGPT

Memoria completa

Compartir

M



Aquí tienes la precarga de las subastas y ventas:


csharp

Copiar código

```
// Subasta 1
if (administrador != null)
{
    Subasta subasta1 = new Subasta(
        "Tecnologia",
        new DateTime(10 / 10 / 2024),
        new DateTime(10, 12 / 2024),
        cliente1,
        articulos1,
        Estado.ABIERTA,
        1500,
        administrador
    );
    Oferta oferta1 = new Oferta(cliente1, 500, new DateTime(2024, 11, 15));
    subasta1.AgregarOferta(oferta1); // agregamos oferta a subasta
    AltaPublicacion(subasta1);
}
```

Envía un mensaje a ChatGPT

ChatGPT puede cometer errores. Comprueba la información importante.



16:30
10/10/2024

