

**Universidad ORT Uruguay**  
**Facultad de Ingeniería**  
**Escuela de Tecnología**

**OBLIGATORIO**  
**PROGRAMACIÓN 2**



**Pablo Larnaudie – 340181**



**Natalia Rebella – 327283**

**Grupo: N2A**

**Docente: Alejandro Martinis**

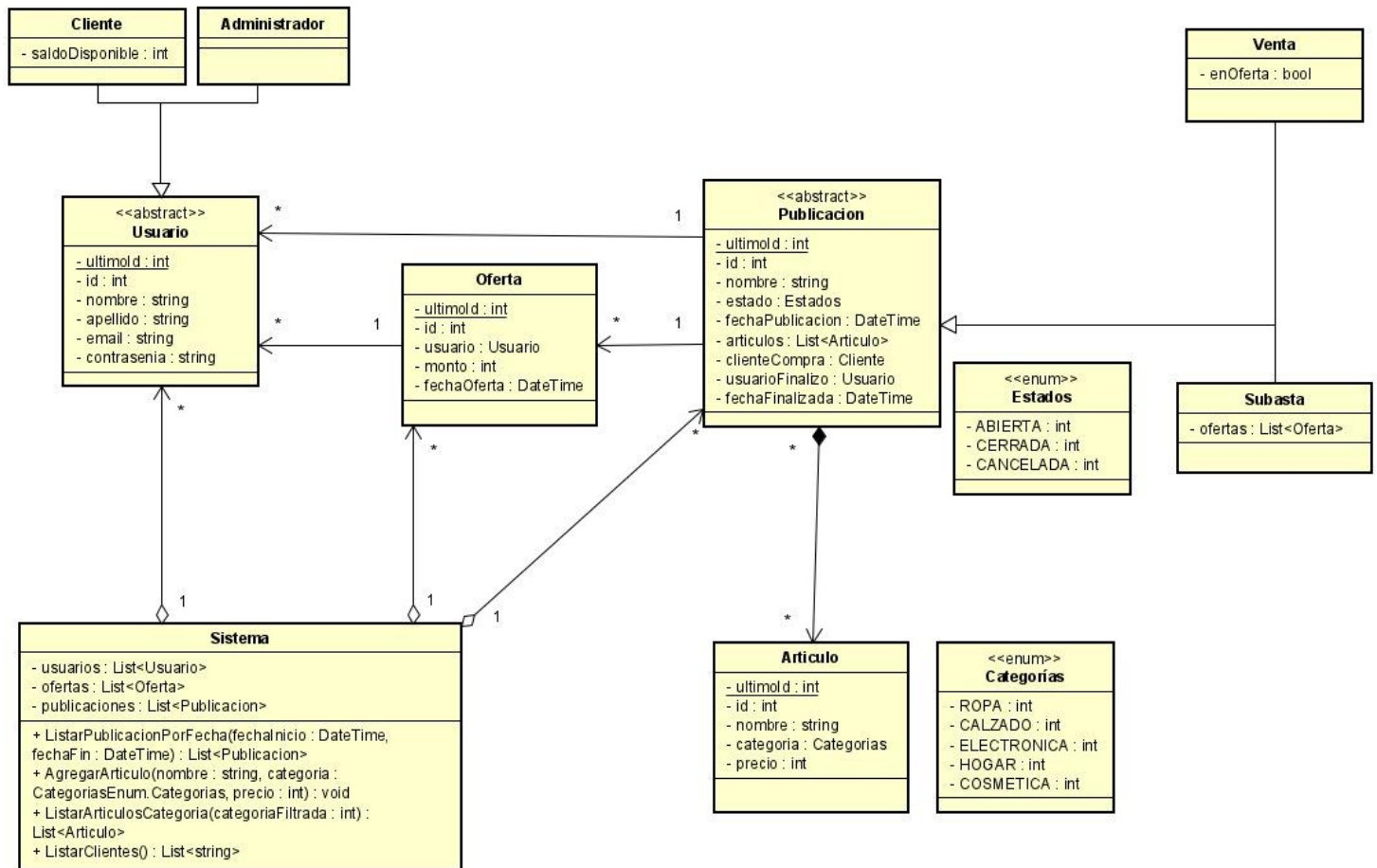
**ATI**

**Fecha de entrega: 10-10-2024**

# Índice

1.	Diagrama de clases UML .....	3
2.	Tabla de precargas .....	3
2.1.	Precarga Cliente .....	3
2.2.	Precarga Administrador .....	4
2.3.	Precarga Artículo .....	5
2.4.	Precarga Publicación.....	8
3.	Código comentado .....	10
3.1.	Consola .....	10
3.1.1.	Program.cs .....	10
3.2.	Dominio .....	14
3.2.1.	Sistema.cs .....	14
3.2.2.	Usuario.cs .....	35
3.2.3.	Cliente.cs .....	36
3.2.4.	Administrador.cs .....	37
3.2.5.	Articulo.cs .....	37
3.2.6.	CategoriasEnum.cs .....	38
3.2.7.	Publicacion.cs .....	38
3.2.8.	Venta.cs .....	39
3.2.9.	Subasta.cs .....	40
3.2.10.	EstadosEnum.cs .....	40
3.2.11.	Oferta.cs .....	40
4.	Evidencias de ejecución de consultas a la IA (precargas) .....	41

## 1. Diagrama de clases UML



## 2. Tabla de precargas

### 2.1. Precarga Cliente

PRECARGA CLIENTE				
NOMBRE	APELLIDO	MAIL	CONTRASEÑA	MONTO

JUAN	PEREZ	juan.perez@example.com	contrasenia1	500
MARIA	GOMEZ	maria.gomez@example.com	Contrasenia2	1000
CARLOS	LOPEZ	carlos.lopez@example.com	Contrasenia3	750
ANA	MARTINEZ	ana.martinez@example.com	Contrasenia4	300
LUIS	FERNANDEZ	luis.fernandez@example.com0	Contrasenia5	1500
LAURA	HERNANDEZ	laura.hernandez@example.com00000	Contrasenia6	200
JORGE	SANCHEZ	jorge.sanchez@example.com	Contrasenia7	1200
ELENA	TORRES	elena.torres@example.com	Contrasenia8	800
PEDRO	RAMIREZ	pedro.ramirez@example.com	Contrasenia9	400
SOFIA	VASQUEZ	sofia.vasquez@example.com	Contrasenia10	950

## 2.2.Precarga Administrador

<b><u>PRECARGA ADMINSTRADOR</u></b>			
NOMBRE	APELLIDO	MAIL	CONTRASEÑA
IGNACIO	QUIROGA	ignacio.admin@example.com	adminpass1

LUCIA	NERUDA	lucia.admin@example.com	Adminpass2
-------	--------	-------------------------	------------

### 2.3.Precarga Artículo

<u>PRECARGA ARTÍCULO</u>		
NOMBRE	CATEGORÍA	MONTO
CAMISETA DE ALGODÓN	ROPA	20
JEANS AJUSTADOS	ROPA	40
CHAQUETA DE CUERO	ROPA	100
VESTIDO DE VERANO	ROPA	50
SUDADERA CON CAPUCHA	ROPA	35
CAMISA DE MANGA LARGA	ROPA	30
SHORTS DE PLAYA	ROPA	25
FALDA LAPIZ	ROPA	45
PANTALONES CORTOS	ROPA	30
ABRIGO DE INVIERNO	ROPA	120
ZAPATILLAS DEPORTIVAS	CALZADO	60
BOTINES DE CUERO	CALZADO	80
SANDALIAS	CALZADO	25
BAILARINAS	CALZADO	35

BOTAS DE LLUVIA	CALZADO	50
ZAPATOS DE VESTIR	CALZADO	90
ZAPATILLAS DE RUNNING	CALZADO	70
ZAPATOS CASUALES	CALZADO	55
ZAPATILLAS DE SKATE	CALZADO	65
CHANCLAS	CALZADO	15
SMARTPHONE	ELECTRONICA	300
TABLETA	ELECTRONICA	250
LAPTOP	ELECTRONICA	800
AURICULARES INALAMBRICOS	ELECTRONICA	100
TELEVISOR 4K	ELECTRONICA	600
RELOJ INTELIGENTE	ELECTRONICA	200
ALTAVOZ BLUETOOTH	ELECTRONICA	80
CAMARA DIGITAL	ELECTRONICA	400
DISCO DURO EXTERNO	ELECTRONICA	120
CONSOLA DE VIDEOJUEGOS	ELECTRONICA	500
LAMPARA DE MESA	HOGAR	40
COJIN DECORATIVO	HOGAR	15
ESPEJO DE PARED	HOGAR	50
CUADRO ARTISTICO	HOGAR	75

CORTINAS	HOGAR	30
MESA DE CENTRO	HOGAR	150
SILLON	HOGAR	200
RELOJ DE PARED	HOGAR	25
JARRON DECORATIVO	HOGAR	20
ALFOMBRA	HOGAR	100
CREMA HIDRATANTE	COSMETICA	30
CHAMPU	COSMETICA	15
JABON EXOFILANTE	COSMETICA	10
PERFUME	COSMETICA	50
DESODORANTE	COSMETICA	8
MAQUILLAJE	COSMETICA	25
CREMA PARA MANOS	COSMETICA	12
ACONDICIONADOR	COSMETICA	15
GEL DE DUCHA	COSMETICA	12
ESPEJO DE MAQUILLAJE	COSMETICA	20

## 2.4.Precarga Publicación

<b><u>PRECARGA PUBLICACION</u></b>				
<b><u>VENTAS</u></b>				
<b>TIENE OFERTA</b>	<b>NOMBRE</b>	<b>ESTADO</b>	<b>FECHA PUBLICACION</b>	<b>ARTICULOS</b>
TRUE	VERANO EN LA PLAYA	ABIERTA	2024/07/15	VERANOPLAYAARTICULOS
FALSE	DEPORTIVOS Y URBANO	ABIERTA	2024/07/20	DEPORTIVOURBANOARTICULOS
FALSE	AVENTURA EN MONTAÑA	ABIERTA	2024/07/25	AVENTURAMONTAÑAARTICULOS
TRUE	ESTILO CASUAL	ABIERTA	2024/08/02	ESTILOCASUALARTICULOS
FALSE	EJERCICIO AL AIRE LIBRE	ABIERTA	2024/08/05	EJERCICIOAIRELIBREARTICULOS
TRUE	TECNOLOGIA MODERNA	ABIERTA	2024/08/10	TECNOLOGIAMODERNAARTICULOS
TRUE	ELEGANCIA EN EL HOGAR	ABIERTA	2024/08/15	ELEGANCIAHOGARARTICULOS
TRUE	BELLEZA NATURAL	ABIERTA	2024/08/20	BELLEZANATURALARTICULOS
TRUE	RELAX EN CASA	ABIERTA	2024/08/25	RELAXCASAARTICULOS
FALSE	FIESTA DE VERANO	ABIERTA	2024/09/01	FIESTAVERANOARTICULOS



<b><u>SUBASTAS</u></b>				
<b>OFERTA</b>	<b>NOMBRE</b>	<b>ESTADO</b>	<b>FECHA PUBLICACION</b>	<b>ARTICULOS</b>
NULL	DIA DE OFICINA	ABIERTA	2024/09/05	OFICINAARTICULOS
NULL	VIAJE DE NEGOCIOS	ABIERTA	3034/09/10	VIAJENEGOCIOSARTICULOS
NULL	GAMERZONE	ABIERTA	2024/09/15	GAMERZONAARTICULOS
NULL	INVIERNO CALIENTE	ABIERTA	2024/11/22	INVIERNOCALIENTEARTICULOS
NULL	DIA DE CAMPO	ABIERTA	2024/07/16	DIACAMPOARTICULOS
NULL	RUTINA DE CUIDADO FACIAL	ABIERTA	2024/03/12	RUTINAFACIALARTICULOS
NULL	AVENTURA EN EL DESIERTO	ABIERTA	2024/02/09	AVENTURADESIERTOARTICULOS
NULL	ESCAPADA ROMANTICA	ABIERTA	2024/05/30	ESCAPADAROMANTICAARTICULOS
OFERTA FIESTAS CASAS ARTICULOS	FIESTA DE CASA	ABIERTA	2024/06/13	FIESTACASAARTICULOS
OFERTA CLASICOS DEL OTOÑO ARTICULOS	CLASICOS DEL OTOÑO	ABIERTA	2024/12/17	CLASICOSOTONOARTICULOS

### 3. Código comentado

#### 3.1. Consola

##### 3.1.1. Program.cs

```
using System.Net;
namespace Consola
{
    using Dominio;
    using System;
    using static Dominio.CategoriasEnum;
    using static System.Runtime.InteropServices.JavaScript.JSType;

    internal class Program
    {
        static void Main(string[] args)
        {
            //Método que se visualizará al momento de iniciar la Consola
            Opciones();
        }

        static void Menu()
        {
            string[] titulos = { "Listar Clientes", "Listar Articulos por
Categoria", "Alta de articulo", "Listar publicaciones por fecha" };

            int opcion = 1;
            foreach (string titulo in titulos)
            {
                Console.WriteLine($"{opcion} - {titulo}");
                opcion++;
            }
        }

        static void Opciones()
        {
            int valor = -1;
            while (valor != 0)
            {
                Console.Clear();
                Menu();
                Console.Write("Ingrese opcion:");
                valor = LeerEntero();
                switch (valor)
                {
                    case 1:
                        Punto1();
                        break;
                    case 2:
                        Punto2();
                        break;
                    case 3:
                        Punto3();
                        break;
                    case 4:
```

```

        Punto4();
        break;
    default:
        break;
    }
}

// Llamar al método que lista clientes e indicar que hacer para
seguir
static void Punto1()
{
    ListarClientes();
    Console.WriteLine("Enter para seguir");
    Console.ReadLine();
}
static void Punto2()
{
    Console.WriteLine("Para acceder a los artículos por categoría,
digita el número correspondiente:");
    Console.WriteLine("1 - ROPA");
    Console.WriteLine("2 - CALZADO");
    Console.WriteLine("3 - ELECTRONICA");
    Console.WriteLine("4 - HOGAR");
    Console.WriteLine("5 - COSMETICA");

    int parametro;
    if (int.TryParse(Console.ReadLine(), out parametro))
    {
        // Llama a ListarArticulosCategoría con el número de
categoría
        ListarArticulosCategoría(parametro);
    }
    else
    {
        Console.WriteLine("Entrada no válida.");
    }

    Console.WriteLine("Enter para seguir");
    Console.ReadLine();
}
static void Punto3()
{
    AltaArticulos();
    Console.WriteLine("Punto 3");
    Console.WriteLine("Enter para seguir");
    Console.ReadLine();
}

static void Punto4()
{
    Console.WriteLine("Punto 4");

    // Leer fecha de inicio
    Console.WriteLine("Ingresar fecha de inicio (AAAA/MM/DD):");
    DateTime fechaInicio;
    while (!DateTime.TryParse(Console.ReadLine(), out fechaInicio))
    {
        Console.WriteLine("Formato inválido. Intente de nuevo
(AAAA/MM/DD):");
    }
}

```

```

    }

    // Leer fecha de fin
    Console.WriteLine("Ingresar fecha de fin (AAAA/MM/DD):");
    DateTime fechaFin;
    while (!DateTime.TryParse(Console.ReadLine(), out fechaFin))
    {
        Console.WriteLine("Formato inválido. Intente de nuevo (AAAA/MM/DD):");
    }

    // Llamar al método con las fechas ingresadas
    ListarPublicacionPorFecha(fechaInicio, fechaFin);

    Console.WriteLine("Enter para seguir");
    Console.ReadLine();
}

static int LeerEntero()
{
    string texto = Console.ReadLine();
    int resultado = 0;
    while (!int.TryParse(texto, out resultado))
    {
        Console.WriteLine("Error. Vuelva a ingresar el valor:");
        texto = Console.ReadLine();
    }
    return resultado;
}

static void ListarClientes()
{
    // Llamada al método ListarClientes que se encuentra en la
    // instancia de Sistema
    List<string> detalleClientes =
    Sistema.Instancia.ListarClientes();

    // Verificar si la lista está vacía
    if (detalleClientes.Count == 0)
    {
        Console.WriteLine("No hay clientes para listar.");
    }
    else
    {
        //Mostrar la lista de clientes
        Console.WriteLine("Lista de Clientes:");
        foreach (string unCliente in detalleClientes)
        {
            Console.WriteLine(unCliente);
        }
    }
}

static void ListarArticulosCategoria(int categoriaFiltrada)
{
    // Llamada al método ListarArticulosCategoria que se encuentra
    // en la instancia de Sistema
    List<Articulo> listaPorCategoria =
    Sistema.Instancia.ListarArticulosCategoria(categoriaFiltrada);

    // Verificar si la lista está vacía
    if (listaPorCategoria.Count == 0)

```

```

        {
            Console.WriteLine("No hay articulos para listar.");
        }
        else
        {
            //Filtrar por categoría y mostrar
            Console.WriteLine($"Hay articulos para listar");
            foreach (Articulo i in listaPorCategoria)
            {
                Console.WriteLine($"{i.Id}, {i.Nombre}, ${i.Precio},
{i.CategoriaArt}");
            }
        }
    }

    static void AgregarArticulo(string nombre,
CategoriasEnum.Categorias categoria, int precio)
    {
        // Crea un nuevo artículo con los datos recibidos
        Articulo nuevoArticulo = new Articulo
        {
            Nombre = nombre,
            CategoriaArt = categoria,
            Precio = precio
        };

        // Llama al método en el sistema que agrega el artículo a la
lista
        Sistema.Instancia.AgregarArticulo(nuevoArticulo);

        Console.WriteLine("Artículo agregado correctamente.");
    }
}

static void AltaArticulos()
{
    //Nombre artículo
    Console.WriteLine("Ingrese el nombre del artículo:");
    string nombre = Console.ReadLine();

    // Ingreso y validación de la categoría
    Console.WriteLine("Seleccione el número de la categoría (1 -
ROPA, 2 - CALZADO, 3 - ELECTRONICA, 4 - HOGAR, 5 - COSMETICA):");
    int categoriaSeleccionada;
    while (!int.TryParse(Console.ReadLine(), out
categoriaSeleccionada) || categoriaSeleccionada < 1 || categoriaSeleccionada
> 5)
    {
        Console.WriteLine("Categoría no válida. Por favor, ingrese
un número entre 1 y 5.");
    }

    // Convertir el número ingresado en la categoría correspondiente
    CategoriasEnum.Categorias categoria =
(CategoriasEnum.Categorias)categoriaSeleccionada;

    // Ingreso del precio del artículo y validar que sea numérico y
mayor que 0.
    Console.WriteLine("Ingrese el precio del artículo:");

```

```

        int precio;
        while (!int.TryParse(Console.ReadLine(), out precio) || precio <
0)
        {
            Console.WriteLine("Precio no válido. Por favor, ingrese un
número entero positivo.");
        }
        // Llama al método para agregar el artículo
        Sistema.Instancia.AgregarArticulo(nombre, categoria, precio);

        Console.WriteLine("Artículo agregado exitosamente.");
    }

    static void ListarPublicacionPorFecha(DateTime fechaInicio, DateTime
fechaFin)
    {
        List<Publicacion> listaPublicacionesFecha =
Sistema.Instancia.ListarPublicacionPorFecha(fechaInicio, fechaFin);
        // Verificar si la lista está vacía o si la fecha inicial es
mayor a la fecha final
        if (listaPublicacionesFecha.Count == 0 || fechaInicio <
fechaFin)
        {
            Console.WriteLine("No hay publicaciones entre esas fechas
para listar o la fecha inicial es mayor a la fecha final para el rango
seleccionado.");
        }
        else
        {
            //Filtrar y mostrar
            Console.WriteLine($"Los articulos encontrados en ese rango
de fechas son: ");
            foreach (Publicacion unaPublicacion in
listaPublicacionesFecha)
            {
                Console.WriteLine($"ID: {unaPublicacion.Id}, NOMBRE:
{unaPublicacion.Nombre}, ESTADO: {unaPublicacion.Estado}, FECHA:
{unaPublicacion.FechaPublicacion}");
            }
        }
    }
}

```

## 3.2. Dominio

### 3.2.1. Sistema.cs

```

using System;

using System.Collections.Generic;

```

```

using System.Diagnostics.Metrics;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using static Dominio.CategoriasEnum;
using static Dominio.EstadosEnum;
using static System.Runtime.InteropServices.JavaScript.JSType;

namespace Dominio
{
    //DEFINIMOS LA CLASE CON SUS ATRIBUTOS
    public class Sistema
    {
        List<Usuario> usuarios = new List<Usuario>();
        List<Oferta> ofertas = new List<Oferta>();
        List<Publicacion> publicaciones = new List<Publicacion>();
        List<Articulo> articulos = new List<Articulo>();
        List<Cliente> clientes = new List<Cliente>();
        List<Administrador> administradores = new List<Administrador>();
        List<Venta> ventas = new List<Venta>();
        List<Subasta> subastas = new List<Subasta>();

        private static Sistema instancia;

        #region Singleton

```

```

public static Sistema Instancia
{
    get
    {
        if (instancia == null)
        {
            instancia = new Sistema();
        }
        return instancia;
    }
}

// Constructor del sistema donde se realiza la precarga de datos
private Sistema()
{
    PrecargaArticulos();
    PrecargaClientes();
    PrecargaAdministradores();
    PrecargaPublicaciones();
    PrecargaOfertas();
}

#endregion

public void PrecargaArticulos()
{
    AgregarArticulo(new Articulo("Camiseta de algodón",
    CategoriasEnum.Categorias.ROPA, 20));
}

```



```

        AgregarArticulo(new Articulo("Jeans ajustados",
CategoriasEnum.Categorias.ROPA, 40));

        AgregarArticulo(new Articulo("Chaqueta de cuero",
CategoriasEnum.Categorias.ROPA, 100));

        AgregarArticulo(new Articulo("Vestido de verano",
CategoriasEnum.Categorias.ROPA, 50));

        AgregarArticulo(new Articulo("Sudadera con capucha",
CategoriasEnum.Categorias.ROPA, 35));

        AgregarArticulo(new Articulo("Camisa de manga larga",
CategoriasEnum.Categorias.ROPA, 30));

        AgregarArticulo(new Articulo("Shorts de playa",
CategoriasEnum.Categorias.ROPA, 25));

        AgregarArticulo(new Articulo("Falda lápiz", CategoriasEnum.Categorias.ROPA,
45));

        AgregarArticulo(new Articulo("Pantalones cortos",
CategoriasEnum.Categorias.ROPA, 30));

        AgregarArticulo(new Articulo("Abrigo de invierno",
CategoriasEnum.Categorias.ROPA, 120));

        AgregarArticulo(new Articulo("Zapatillas deportivas",
CategoriasEnum.Categorias.CALZADO, 60));

        AgregarArticulo(new Articulo("Botines de cuero",
CategoriasEnum.Categorias.CALZADO, 80));

        AgregarArticulo(new Articulo("Sandalias",
CategoriasEnum.Categorias.CALZADO, 25));

        AgregarArticulo(new Articulo("Bailarinas",
CategoriasEnum.Categorias.CALZADO, 30));

        AgregarArticulo(new Articulo("Botas de lluvia",
CategoriasEnum.Categorias.CALZADO, 50));

        AgregarArticulo(new Articulo("Zapatos de vestir",
CategoriasEnum.Categorias.CALZADO, 90));

        AgregarArticulo(new Articulo("Zapatillas de running",
CategoriasEnum.Categorias.CALZADO, 70));

```

AgregarArticulo(new Articulo("Zapatos casuales",  
CategoriasEnum.Categorias.CALZADO, 55));

AgregarArticulo(new Articulo("Zapatillas de skate",  
CategoriasEnum.Categorias.CALZADO, 65));

AgregarArticulo(new Articulo("Chanclas",  
CategoriasEnum.Categorias.CALZADO, 15));

AgregarArticulo(new Articulo("Smartphone",  
CategoriasEnum.Categorias.ELECTRONICA, 300));

AgregarArticulo(new Articulo("Tableta",  
CategoriasEnum.Categorias.ELECTRONICA, 250));

AgregarArticulo(new Articulo("Laptop",  
CategoriasEnum.Categorias.ELECTRONICA, 800));

AgregarArticulo(new Articulo("Auriculares inalámbricos",  
CategoriasEnum.Categorias.ELECTRONICA, 100));

AgregarArticulo(new Articulo("Televisor 4K",  
CategoriasEnum.Categorias.ELECTRONICA, 600));

AgregarArticulo(new Articulo("Reloj inteligente",  
CategoriasEnum.Categorias.ELECTRONICA, 200));

AgregarArticulo(new Articulo("Altavoz Bluetooth",  
CategoriasEnum.Categorias.ELECTRONICA, 80));

AgregarArticulo(new Articulo("Cámara digital",  
CategoriasEnum.Categorias.ELECTRONICA, 400));

AgregarArticulo(new Articulo("Disco duro externo",  
CategoriasEnum.Categorias.ELECTRONICA, 120));

AgregarArticulo(new Articulo("Consola de videojuegos",  
CategoriasEnum.Categorias.ELECTRONICA, 500));

AgregarArticulo(new Articulo("Lámpara de mesa",  
CategoriasEnum.Categorias.HOGAR, 40));

AgregarArticulo(new Articulo("Cojín decorativo",  
CategoriasEnum.Categorias.HOGAR, 15));

```

        AgregarArticulo(new Articulo("Espejo de pared",
CategoriasEnum.Categorias.HOGAR, 50));

        AgregarArticulo(new Articulo("Cuadro artístico",
CategoriasEnum.Categorias.HOGAR, 75));

        AgregarArticulo(new Articulo("Cortinas", CategoriasEnum.Categorias.HOGAR,
30));

        AgregarArticulo(new Articulo("Mesa de centro",
CategoriasEnum.Categorias.HOGAR, 150));

        AgregarArticulo(new Articulo("Sillón", CategoriasEnum.Categorias.HOGAR,
200));

        AgregarArticulo(new Articulo("Reloj de pared",
CategoriasEnum.Categorias.HOGAR, 25));

        AgregarArticulo(new Articulo("Jarrón decorativo",
CategoriasEnum.Categorias.HOGAR, 20));

        AgregarArticulo(new Articulo("Alfombra",
CategoriasEnum.Categorias.HOGAR, 100));

        AgregarArticulo(new Articulo("Crema hidratante",
CategoriasEnum.Categorias.COSMETICA, 30));

        AgregarArticulo(new Articulo("Champú",
CategoriasEnum.Categorias.COSMETICA, 15));

        AgregarArticulo(new Articulo("Jabón exfoliante",
CategoriasEnum.Categorias.COSMETICA, 10));

        AgregarArticulo(new Articulo("Perfume",
CategoriasEnum.Categorias.COSMETICA, 50));

        AgregarArticulo(new Articulo("Desodorante",
CategoriasEnum.Categorias.COSMETICA, 8));

        AgregarArticulo(new Articulo("Maquillaje",
CategoriasEnum.Categorias.COSMETICA, 25));

        AgregarArticulo(new Articulo("Crema para manos",
CategoriasEnum.Categorias.COSMETICA, 12));

        AgregarArticulo(new Articulo("Acondicionador",
CategoriasEnum.Categorias.COSMETICA, 15));

```

```
        AgregarArticulo(new Articulo("Gel de ducha",
CategoriasEnum.Categorias.COSMETICA, 12));
```

```
        AgregarArticulo(new Articulo("Espejo de maquillaje",
CategoriasEnum.Categorias.COSMETICA, 20));
```

```
    }
```

```
    public void PrecargaClientes()
```

```
    {
```

```
        AgregarCliente(new Cliente(500, "Juan", "Pérez", "juan.perez@example.com",
"contrasenia1"));
```

```
        AgregarCliente(new Cliente(1000, "María", "Gómez",
"maria.gomez@example.com", "contrasenia2"));
```

```
        AgregarCliente(new Cliente(750, "Carlos", "Lopez",
"carlos.lopez@example.com", "contrasenia3"));
```

```
        AgregarCliente(new Cliente(300, "Ana", "Martínez",
"ana.martinez@example.com", "contrasenia4"));
```

```
        AgregarCliente(new Cliente(1500, "Luis", "Fernández",
"luis.fernandez@example.com", "contrasenia5"));
```

```
        AgregarCliente(new Cliente(200, "Laura", "Hernández",
"laura.hernandez@example.com", "contrasenia6"));
```

```
        AgregarCliente(new Cliente(1200, "Jorge", "Sánchez",
"jorge.sanchez@example.com", "contrasenia7"));
```

```
        AgregarCliente(new Cliente(800, "Elena", "Torres",
"elena.torres@example.com", "contrasenia8"));
```

```
        AgregarCliente(new Cliente(400, "Pedro", "Ramírez",
"pedro.ramirez@example.com", "contrasenia9"));
```

```
        AgregarCliente(new Cliente(950, "Sofía", "Vásquez",
"sofia.vasquez@example.com", "contrasenia10"));
```

```
    }
```

```
    public void PrecargaAdministradores()
```

```
    {
```

```

        AgregarAdministrador(new Administrador("Ignacio", "Quiroga",
"ignacio.admin@example.com", "adminpass1"));

        AgregarAdministrador(new Administrador("Lucía", "Neruda",
"lucia.admin@example.com", "adminpass2"));
    }

```

```

public void PrecargaOfertas()
{
    AgregarOferta(new Oferta( clientes[2],123,new DateTime(2024, 05,12)) );
    AgregarOferta(new Oferta( clientes[8], 563, new DateTime(2024, 05, 12)) );
}

```

```

public void AgregarOferta(Oferta unaOferta)
{
    try
    {
        VerificarOferta(unaOferta);
        ofertas.Add(unaOferta);
    }
    catch (Exception ex)
    {
        throw ex;
    }
}

```

```

public void VerificarOferta(Oferta unaOferta)
{

```

```

        if (ofertas.Contains(unaOferta))
            throw new Exception("Oferta ya existe.");
    }

    public void PrecargaPublicaciones()
    {
        // Publicación 1: "Verano en la Playa"

        List<Articulo> veranoPlayaArticulos = new List<Articulo>
        {
            new Articulo { Nombre = "Shorts de playa", Precio = 25 },
            new Articulo { Nombre = "Sandalias", Precio = 25 },
            new Articulo { Nombre = "Camiseta de algodón", Precio = 20 }
        };

        AgregarPublicacion(new Venta(true, "Verano en la Playa",
            EstadosEnum.Estados.ABIERTA, new DateTime(2024, 7, 15), veranoPlayaArticulos,
            null, null, null));

        // Publicación 2: "Deportivo y Urbano"

        List<Articulo> deportivoUrbanoArticulos = new List<Articulo>
        {
            new Articulo { Nombre = "Zapatillas deportivas", Precio = 60 },
            new Articulo { Nombre = "Sudadera con capucha", Precio = 35 },
            new Articulo { Nombre = "Pantalones cortos", Precio = 30 }
        };

        AgregarPublicacion(new Venta(false, "Deportivo y Urbano",
            EstadosEnum.Estados.ABIERTA, new DateTime(2024, 7, 20),
            deportivoUrbanoArticulos, null, null, null));
    }

```

```

// Publicación 3: "Aventura en la Montaña"

List<Articulo> aventuraMontanaArticulos = new List<Articulo>
{
    new Articulo { Nombre = "Chaqueta de cuero", Precio = 100 },
    new Articulo { Nombre = "Botas de lluvia", Precio = 50 },
    new Articulo { Nombre = "Zapatillas de running", Precio = 70 }
};

AgregarPublicacion(new Venta(false, "Aventura en la Montaña",
    EstadosEnum.Estados.ABIERTA, new DateTime(2024, 7, 25),
    aventuraMontanaArticulos, null, null, null));

// Publicación 4: "Estilo Casual"

List<Articulo> estiloCasualArticulos = new List<Articulo>
{
    new Articulo { Nombre = "Jeans ajustados", Precio = 40 },
    new Articulo { Nombre = "Camisa de manga larga", Precio = 30 },
    new Articulo { Nombre = "Zapatos casuales", Precio = 55 }
};

AgregarPublicacion(new Venta(true, "Estilo Casual",
    EstadosEnum.Estados.ABIERTA, new DateTime(2024, 8, 2),
    estiloCasualArticulos, null, null, null));

// Publicación 5: "Ejercicio al Aire Libre"

List<Articulo> ejercicioAireLibreArticulos = new List<Articulo>
{
    new Articulo { Nombre = "Zapatillas de running", Precio = 70 },
    new Articulo { Nombre = "Camiseta de algodón", Precio = 20 },
    new Articulo { Nombre = "Shorts de playa", Precio = 25 }
}

```

```

    };

    AgregarPublicacion(new Venta(false, "Ejercicio al Aire Libre",
    EstadosEnum.Estados.ABIERTA, new DateTime(2024, 8, 5),
    ejercicioAireLibreArticulos, null, null, null));

```

```

// Publicación 6: "Tecnología Moderna"

```

```

List<Articulo> tecnologiaModernaArticulos = new List<Articulo>
{
    new Articulo { Nombre = "Smartphone", Precio = 300 },
    new Articulo { Nombre = "Reloj inteligente", Precio = 200 },
    new Articulo { Nombre = "Altavoz Bluetooth", Precio = 80 }
};

```

```

    AgregarPublicacion(new Venta(true, "Tecnología Moderna",
    EstadosEnum.Estados.ABIERTA, new DateTime(2024, 8, 10),
    tecnologiaModernaArticulos, null, null, null));

```

```

// Publicación 7: "Elegancia en el Hogar"

```

```

List<Articulo> eleganciaHogarArticulos = new List<Articulo>
{
    new Articulo { Nombre = "Lámpara de mesa", Precio = 40 },
    new Articulo { Nombre = "Espejo de pared", Precio = 50 },
    new Articulo { Nombre = "Cuadro artístico", Precio = 75 }
};

```

```

    AgregarPublicacion(new Venta(true, "Elegancia en el Hogar",
    EstadosEnum.Estados.ABIERTA, new DateTime(2024, 8, 15),
    eleganciaHogarArticulos, null, null, null));

```

```

// Publicación 8: "Belleza Natural"

```

```

List<Articulo> bellezaNaturalArticulos = new List<Articulo>

```



```

{
    new Artículo { Nombre = "Crema hidratante", Precio = 30 },
    new Artículo { Nombre = "Jabón exfoliante", Precio = 10 },
    new Artículo { Nombre = "Perfume", Precio = 50 }
};

AgregarPublicacion(new Venta(true, "Belleza Natural",
    EstadosEnum.Estados.ABIERTA, new DateTime(2024, 8, 20), bellezaNaturalArticulos,
    null, null, null));

// Publicación 9: "Relax en Casa"

List<Articulo> relaxCasaArticulos = new List<Articulo>
{
    new Artículo { Nombre = "Sillón", Precio = 200 },
    new Artículo { Nombre = "Cojín decorativo", Precio = 15 },
    new Artículo { Nombre = "Jarrón decorativo", Precio = 20 }
};

AgregarPublicacion(new Venta(true, "Relax en Casa",
    EstadosEnum.Estados.ABIERTA, new DateTime(2024, 8, 25), relaxCasaArticulos, null,
    null, null));

// Publicación 10: "Fiesta de Verano"

List<Articulo> fiestaVeranoArticulos = new List<Articulo>
{
    new Artículo { Nombre = "Vestido de verano", Precio = 50 },
    new Artículo { Nombre = "Sandalias", Precio = 25 },
    new Artículo { Nombre = "Falda lápiz", Precio = 45 }
};

```

```
        AgregarPublicacion(new Venta(false, "Fiesta de Verano",
EstadosEnum.Estados.ABIERTA, new DateTime(2024, 9, 1), fiestaVeranoArticulos,
null, null, null));
```

```
// Publicación 11: "Día de Oficina"
```

```
List<Articulo> oficinaArticulos = new List<Articulo>
```

```
{
```

```
    new Articulo { Nombre = "Zapatos de vestir", Precio = 90 },
```

```
    new Articulo { Nombre = "Pantalones cortos", Precio = 30 },
```

```
    new Articulo { Nombre = "Camisa de manga larga", Precio = 30 }
```

```
};
```

```
        AgregarPublicacion(new Subasta(null, "Día de Oficina",
EstadosEnum.Estados.ABIERTA, new DateTime(2024, 9, 5), oficinaArticulos, null,
null, null));
```

```
// Publicación 12: "Viaje de Negocios"
```

```
List<Articulo> viajeNegociosArticulos = new List<Articulo>
```

```
{
```

```
    new Articulo { Nombre = "Laptop", Precio = 800 },
```

```
    new Articulo { Nombre = "Auriculares inalámbricos", Precio = 100 },
```

```
    new Articulo { Nombre = "Reloj inteligente", Precio = 200 }
```

```
};
```

```
        AgregarPublicacion(new Subasta(null, "Viaje de Negocios",
EstadosEnum.Estados.ABIERTA, new DateTime(2024, 9, 10), viajeNegociosArticulos,
null, null, null));
```

```
// Publicación 13: "Gamer Zone"
```

```
List<Articulo> gamerZoneArticulos = new List<Articulo>
```

```
{
```

```

new Artículo { Nombre = "Consola de videojuegos", Precio = 500 },
new Artículo { Nombre = "Auriculares inalámbricos", Precio = 100 },
new Artículo { Nombre = "Disco duro externo", Precio = 120 }
};

AgregarPublicacion(new Subasta(null, "Gamer Zone",
EstadosEnum.Estados.ABIERTA, new DateTime(2024, 9, 15), gamerZoneArticulos,
null, null, null));

// Publicación 14: "Invierno Caliente"

List<Artículo> inviernoCalienteArticulos = new List<Artículo>
{
new Artículo { Nombre = "Abrigo de invierno", Precio = 120 },
new Artículo { Nombre = "Botines de cuero", Precio = 80 },
new Artículo { Nombre = "Guantes de lana", Precio = 15 }
};

AgregarPublicacion(new Subasta(null, "Invierno Caliente",
EstadosEnum.Estados.ABIERTA, new DateTime(2024, 11, 22),
inviernoCalienteArticulos, null, null, null));

```

```

// Publicación 15: "Día de Campo"

List<Artículo> diaCampoArticulos = new List<Artículo>
{
new Artículo { Nombre = "Chanclas", Precio = 15 },
new Artículo { Nombre = "Zapatillas deportivas", Precio = 60 },
new Artículo { Nombre = "Pantalones cortos", Precio = 30 }
};

```

```
AgregarPublicacion(new Subasta(null, "Día de Campo",  
EstadosEnum.Estados.ABIERTA, new DateTime(2024, 7, 16), diaCampoArticulos, null,  
null, null));
```

```
// Publicación 16: "Rutina de Cuidado Facial"
```

```
List<Articulo> rutinaFacialArticulos = new List<Articulo>  
{  
    new Articulo { Nombre = "Crema hidratante", Precio = 30 },  
    new Articulo { Nombre = "Champú", Precio = 15 },  
    new Articulo { Nombre = "Jabón exfoliante", Precio = 10 }  
};
```

```
AgregarPublicacion(new Subasta(null, "Rutina de Cuidado Facial",  
EstadosEnum.Estados.ABIERTA, new DateTime(2024, 3, 12), rutinaFacialArticulos,  
null, null, null));
```

```
// Publicación 17: "Aventura en el Desierto"
```

```
List<Articulo> aventuraDesiertoArticulos = new List<Articulo>  
{  
    new Articulo { Nombre = "Botines de cuero", Precio = 80 },  
    new Articulo { Nombre = "Sombrero de playa", Precio = 25 },  
    new Articulo { Nombre = "Camiseta de algodón", Precio = 20 }  
};
```

```
AgregarPublicacion(new Subasta(null, "Aventura en el Desierto",  
EstadosEnum.Estados.ABIERTA, new DateTime(2024, 2, 9),  
aventuraDesiertoArticulos, null, null, null));
```

```
// Publicación 18: "Escapada Romántica"
```

```
List<Articulo> escapadaRomanticaArticulos = new List<Articulo>  
{
```

```

new Artículo { Nombre = "Vestido de verano", Precio = 50 },
new Artículo { Nombre = "Bailarinas", Precio = 30 },
new Artículo { Nombre = "Perfume", Precio = 50 }
};

AgregarPublicacion(new Subasta(null, "Escapada Romántica",
EstadosEnum.Estados.ABIERTA, new DateTime(2024, 5, 30),
escapadaRomanticaArticulos, null, null, null));

```

// Publicación 19: "Fiesta en Casa"

```

List<Articulo> fiestaCasaArticulos = new List<Articulo>
{
new Artículo { Nombre = "Altavoz Bluetooth", Precio = 80 },
new Artículo { Nombre = "Lámpara de mesa", Precio = 40 },
new Artículo { Nombre = "Cojín decorativo", Precio = 15 },
};

List<Oferta> ofertaFiestaCasaArticulos = new List<Oferta>
{
new Oferta( clientes[2],123,new DateTime(2024, 05,12))
};

AgregarPublicacion(new Subasta(ofertaFiestaCasaArticulos, "Fiesta en Casa",
EstadosEnum.Estados.ABIERTA, new DateTime(2024, 6, 13), fiestaCasaArticulos, null,
null, null));

```

// Publicación 20: "Clásicos del Otoño"

```

List<Articulo> clasicosOtonoArticulos = new List<Articulo>
{
new Artículo { Nombre = "Abrigo de invierno", Precio = 120 },
new Artículo { Nombre = "Botines de cuero", Precio = 80 },

```

```

        new Artículo { Nombre = "Bufanda de lana", Precio = 30 }
    };

    List<Oferta> ofertaclasicosOtonoArticulos = new List<Oferta>
    {
        new Oferta( clientes[8],526,new DateTime(2024, 04,12))
    };

    AgregarPublicacion(new Subasta(ofertaclasicosOtonoArticulos, "Clásicos del
    Otoño", EstadosEnum.Estados.ABIERTA, new DateTime(2024, 12, 17),
    clasicosOtonoArticulos, null, null, null));
    }

    public void AgregarAdministrador(Administrador administradorNuevo)
    {
        try
        {
            VerificarAdminNoExiste(administradorNuevo);

            administradores.Add(administradorNuevo);
        }
        catch (Exception ex)
        {
            throw ex;
        }
    }

    public void VerificarAdminNoExiste(Administrador administradorNuevo)
    {
        if (administradores.Contains(administradorNuevo))
        {
            throw new Exception("Administrador ya existe.");
        }
    }

```

```
}
```

```
public void AgregarCliente(Cliente clienteNuevo)
```

```
{
```

```
    try
```

```
    {
```

```
        VerificarClienteNoExiste(clienteNuevo);
```

```
        clientes.Add(clienteNuevo);
```

```
    }
```

```
    catch (Exception ex)
```

```
    {
```

```
        throw ex;
```

```
    }
```

```
}
```

```
public void VerificarClienteNoExiste(Cliente clienteNuevo)
```

```
{
```

```
    if (clientes.Contains(clienteNuevo))
```

```
        throw new Exception("Cliente ya existe.");
```

```
}
```

```
public void AgregarArticulo(Articulo articuloNuevo)
```

```
{
```

```
    try
```

```
    {
```

```
        VerificarArticuloNoExiste(articuloNuevo);
```

```
        articulos.Add(articuloNuevo);
```

```

    }

    catch (Exception ex)

    {

        throw ex;

    }

}

public void VerificarArticuloNoExiste(Articulo articuloNuevo)

{

    if (articulos.Contains(articuloNuevo))

        throw new Exception("Articulo ya existe.");

}


// Método para listar nombres de clientes

public List<string> ListarClientes()

{

    List<string> detallesClientes = new List<string>();

    foreach (Cliente unCliente in clientes)

    {

        detallesClientes.Add($"ID:{unCliente.Id} Nombre: {unCliente.Nombre},
Apellido: {unCliente.Apellido}, Email: {unCliente.Mail} Saldo Disponible:
{unCliente.SaldoDisponible}");

    }

    return detallesClientes;

}

public List<Articulo> ListarArticulosCategoria(int categoriaFiltrada)

{

    List<Articulo> listaPorCategoria = new List<Articulo>();

```



```

// Convertir el entero a CategoriasEnum.Categorias

if (Enum.IsDefined(typeof(CategoriasEnum.Categorias), categoriaFiltrada))
{
    CategoriasEnum.Categorias categoriaConvertida =
(CategoriasEnum.Categorias)categoriaFiltrada;

    foreach (Articulo unArticulo in articulos)
    {
        if (unArticulo.CategoriaArt == categoriaConvertida)
        {
            listaPorCategoria.Add(unArticulo);
        }
    }
}
else
{
    Console.WriteLine("Categoría no válida.");
}

return listaPorCategoria;
}

public void AgregarArticulo(string nombre, CategoriasEnum.Categorias categoria,
int precio)
{
    Articulo nuevoArticulo = new Articulo (nombre, categoria, precio);

    // Verificamos que el artículo no exista en la lista antes de agregarlo

```

```

        foreach (Articulo articulo in articulos)
        {
            if (articulo.Nombre == nombre && articulo.CategoriaArt == categoria &&
articulo.Precio == precio)
            {
                throw new Exception("Este artículo ya existe.");
            }
        }

        // Agregar el artículo a la lista
        articulos.Add(nuevoArticulo);
    }

    public void AgregarPublicacion(Publicacion publicacionNueva)
    {
        try
        {
            VerificarPublicacionNoExiste(publicacionNueva);
            publicaciones.Add(publicacionNueva);
        }
        catch (Exception ex)
        {
            throw ex;
        }
    }

    public void VerificarPublicacionNoExiste(Publicacion publicacionNueva)
    {

```

```

        if (publicaciones.Contains(publicacionNueva)){
            throw new Exception("Publicacion ya existe.");
        }
    }

    public List<Publicacion> ListarPublicacionPorFecha(DateTime fechaInicio,
    DateTime fechaFin)
    {
        List<Publicacion> listaFiltradaPublicacionesFecha = new List<Publicacion>();
        foreach (Publicacion unaPublicacion in publicaciones)
        {
            if(unaPublicacion.FechaPublicacion >= fechaInicio &&
            unaPublicacion.FechaPublicacion <= fechaFin)
            {
                listaFiltradaPublicacionesFecha.Add(unaPublicacion);
            }
        }
        return listaFiltradaPublicacionesFecha;
    }
}
}

```

### 3.2.2. Usuario.cs

```

namespace Dominio
{
    //DEFINIMOS LA CLASE CON SUS ATRIBUTOS

```

```

public class Usuario
{
    int id;
    static int ultimoId;
    string nombre;
    string apellido;
    string mail;
    string contrasenia;

    public Usuario() { }

    //DEFINIMOS SU CONSTRUCTOR
    public Usuario(string nombre, string apellido, string mail, string
contrasenia)
    {
        this.id = ++ultimoId;
        this.nombre = nombre;
        this.apellido = apellido;
        this.mail = mail;
        this.contrasenia = contrasenia;
    }

    //DEFINIMOS SUS PROPIEDADES
    public int Id { get => id; set => id = value; }
    public string Nombre { get => nombre; set => nombre = value; }
    public string Apellido { get => apellido; set => apellido = value; }
    public string Mail { get => mail; set => mail = value; }
    public string Contraseña { get => contrasenia; set => contrasenia =
value; }
}

```

### 3.2.3. Cliente.cs

```

namespace Dominio
{
    //DEFINIMOS LA CLASE CON SUS ATRIBUTOS
    public class Cliente : Usuario
    {
        private int saldoDisponible;

        //DEFINIMOS SUS PROPIEDADES

        public int SaldoDisponible { get => saldoDisponible; set =>
saldoDisponible = value; }

        public Cliente() { }

        //DEFINIMOS SU CONSTRUCTOR

        public Cliente(int saldoDisponible, string nombre, string apellido,
string mail, string contrasenia) : base(nombre, apellido, mail, contrasenia)
        {
            this.saldoDisponible = saldoDisponible;
        }
    }
}

```

### 3.2.4. Administrador.cs

```
namespace Dominio
{
    public class Administrador : Usuario
    {
        //DEFINIR CONSTRUCTOR
        public Administrador (string nombre, string apellido, string mail,
string contrasenia) : base (nombre, apellido, mail, contrasenia)
        {
        }
    }
}
```

### 3.2.5. Artículo.cs

```
namespace Dominio
{
    //DEFINIMOS LA CLASE CON SUS ATRIBUTOS
    public class Artículo
    {
        int id;
        static int ultimoId;
        string nombre;
        CategoriasEnum.Categorias categoriaArt;
        int precio;

        //DEFINIMOS SUS PROPIEDADES
        public int Id { get => id; set => id = value; }
        public string Nombre { get => nombre; set => nombre = value; }
        public CategoriasEnum.Categorias CategoriaArt { get => categoriaArt;
set => categoriaArt = value; }
        public int Precio { get => precio; set => precio = value; }

        public Artículo() { }

        //DEFINIMOS SU CONSTRUCTOR
        public Artículo(string nombre, CategoriasEnum.Categorias
categoriaArt, int precio)
        {
            this.id = ++ultimoId;
            this.nombre = nombre;
            this.categoriaArt = categoriaArt;
            this.precio = precio;
        }
    }
}
```

### 3.2.6. CategoriasEnum.cs

```
namespace Dominio

{

    // Clase que contiene una enumeración para las categorías de artículos

    public class CategoriasEnum

    {

        // Enumeración que define las diferentes categorías posibles para los artículos,
        // asignados con sus respectivos valores numericos

        public enum Categorias { ROPA = 1, CALZADO = 2, ELECTRONICA = 3,
        HOGAR = 4, COSMETICA = 5 };

    }

}
```

### 3.2.7. Publicacion.cs

```
namespace Dominio
{
    //DEFINIMOS LA CLASE CON SUS ATRIBUTOS
    public class Publicacion
    {
        int id;
        static int ultimoId;
        string nombre;
        EstadosEnum.Estados estado;
        DateTime fechaPublicacion;
        List<Articulo> articulos;
        Cliente? clienteCompra; // Permitir que sea null
        Usuario? usuarioFinalizo; // Permitir que sea null
        DateTime? fechaFinalizada; // Cambiar a DateTime? para permitir null

        // PROPIEDADES
        public int Id { get => id; set => id = value; }
        public string Nombre { get => nombre; set => nombre = value; }
        public EstadosEnum.Estados Estado { get => estado; set => estado =
value; }
        public DateTime FechaPublicacion { get => fechaPublicacion; set =>
fechaPublicacion = value; }
    }
}
```

```

        public List<Articulo> Articulos { get => articulos; set => articulos
= value; }
        public Cliente? ClienteCompra { get => clienteCompra; set =>
clienteCompra = value; } // Permitir null
        public Usuario? UsuarioFinalizo { get => usuarioFinalizo; set =>
usuarioFinalizo = value; } // Permitir null
        public DateTime? FechaFinalizada { get => fechaFinalizada; set =>
fechaFinalizada = value; } // Cambiar a DateTime?

        // CONSTRUCTOR VACÍO
        public Publicacion() { }

        // CONSTRUCTOR CON PARÁMETROS
        public Publicacion(string nombre, EstadosEnum.Estados estado,
DateTime fechaPublicacion, List<Articulo> articulos, Cliente? clienteCompra,
Usuario? usuarioFinalizo, DateTime? fechaFinalizada = null)
        {
            this.id = ++ultimoId;
            this.nombre = nombre;
            this.estado = estado;
            this.fechaPublicacion = fechaPublicacion;
            this.articulos = articulos;
            this.clienteCompra = clienteCompra; // Se permite asignar null
            this.usuarioFinalizo = usuarioFinalizo; // Se permite asignar
null
            this.fechaFinalizada = fechaFinalizada; // Se permite asignar
null
        }
    }
}

```

### 3.2.8. Venta.cs

```

namespace Dominio
{
    //DEFINIMOS LA CLASE CON SUS ATRIBUTOS

    public class Venta : Publicacion
    {
        bool enOferta;
        public Venta() { }
        //DEFINIMOS SU CONSTRUCTOR

        public Venta(bool enOferta, string nombre, EstadosEnum.Estados
estado, DateTime fechaPublicacion, List<Articulo> articulos, Cliente?
clienteCompra, Usuario? usuarioFinalizo, DateTime? fechaFinalizada = null) :
base( nombre, estado, fechaPublicacion, articulos, clienteCompra,
usuarioFinalizo, fechaFinalizada = null)
        {
            EnOferta = enOferta;
        }

        //DEFINIMOS SUS PROPIEDADES
        public bool EnOferta { get => enOferta; set => enOferta = value; }
    }
}

```

### 3.2.9. Subasta.cs

```
namespace Dominio
{
    //DEFINIMOS LA CLASE CON SUS ATRIBUTOS

    public class Subasta : Publicacion
    {
        List<Oferta> ofertas;

        public Subasta() { }

        //DEFINIMOS SU CONSTRUCTOR

        public Subasta(List<Oferta> ofertas, string nombre,
            EstadosEnum.Estados estado, DateTime fechaPublicacion, List<Articulo>
            articulos, Cliente? clienteCompra, Usuario? usuarioFinalizo, DateTime?
            fechaFinalizada = null) : base(nombre, estado, fechaPublicacion, articulos,
            clienteCompra, usuarioFinalizo, fechaFinalizada = null)
        {
            Ofertas = ofertas;
        }

        //DEFINIMOS SUS PROPIEDADES
        public List<Oferta> Ofertas { get => ofertas; set => ofertas =
value; }
    }
}
```

### 3.2.10. EstadosEnum.cs

```
namespace Dominio
{
    // Clase que contiene una enumeración para los estados
    public class EstadosEnum
    {
        // Enumeración que define los diferentes estados posible, asignados
        con sus respectivos valores numericos
        public enum Estados { ABIERTA = 1, CERRADA = 2, CANCELADA = 3 };
    }
}
```

### 3.2.11. Oferta.cs

```
namespace Dominio
{
    //DEFINIMOS LA CLASE CON SUS ATRIBUTOS
    public class Oferta
    {
        int id;
        static int ultimoId;
```



```

    Cliente cliente;
    int monto;
    DateTime fechaOferta;

    //DEFINIMOS SUS PROPIEDADES
    public int Id { get => id; set => id = value; }
    public Cliente Cliente { get => cliente; set => cliente = value; }
    public int Monto { get => monto; set => monto = value; }
    public DateTime FechaOferta { get => fechaOferta; set => fechaOferta
= value; }

    public Oferta () { }

    //DEFINIMOS SU CONSTRUCTOR
    public Oferta(Cliente cliente, int monto, DateTime fechaOferta)
    {
        this.id = ++ultimoId;
        this.cliente = cliente;
        this.monto = monto;
        this.fechaOferta = fechaOferta;
    }
}
}
}

```

#### 4. Evidencias de ejecución de consultas a la IA (precargas)

Necesito que me armes una precarga de 10 Clientes y una de 2 Administradores siguiendo los siguientes constructores para cada caso: public Administrador (string nombre, string apellido, string mail, string contrasenia) : base (nombre, apellido, mail, contrasenia) { } public Cliente(int saldoDisponible, string nombre, string apellido, string mail, string contrasenia) : base(nombre, apellido, mail, contrasenia) { this.saldoDisponible = saldoDisponible; }

ChatGPT Plus

ChatGPT

Claro, aquí tienes un ejemplo de cómo podrías crear una precarga de 10 clientes y 2 administradores utilizando los constructores que proporcionaste. He añadido algunos valores de ejemplo para los campos.

##### **Precarga de Clientes**

```

csharp
Copiar código
List<Cliente> clientes = new List<Cliente>
{

```

```

        new Cliente(500, "Juan", "Pérez", "juan.perez@example.com",
"contrasenia1"),
        new Cliente(1000, "María", "Gómez", "maria.gomez@example.com",
"contrasenia2"),
        new Cliente(750, "Carlos", "Lopez", "carlos.lopez@example.com",
"contrasenia3"),
        new Cliente(300, "Ana", "Martínez", "ana.martinez@example.com",
"contrasenia4"),
        new Cliente(1500, "Luis", "Fernández",
"luis.fernandez@example.com", "contrasenia5"),
        new Cliente(200, "Laura", "Hernández",
"laura.hernandez@example.com", "contrasenia6"),
        new Cliente(1200, "Jorge", "Sánchez", "jorge.sanchez@example.com",
"contrasenia7"),
        new Cliente(800, "Elena", "Torres", "elena.torres@example.com",
"contrasenia8"),
        new Cliente(400, "Pedro", "Ramírez", "pedro.ramirez@example.com",
"contrasenia9"),
        new Cliente(950, "Sofía", "Vásquez", "sofia.vasquez@example.com",
"contrasenia10")
};

```

## Precarga de Administradores

```

csharp
Copiar código
List<Administrador> administradores = new List<Administrador>
{
    new Administrador("Carlos", "Admin", "carlos.admin@example.com",
"adminpass1"),
    new Administrador("Lucía", "Admin", "lucia.admin@example.com",
"adminpass2")
};

```

## Resumen

- **Cientes:** Se crean 10 instancias de `Cliente` con diferentes saldos y datos de contacto.
- **Administradores:** Se crean 2 instancias de `Administrador` con sus respectivos datos.

Puedes ajustar los nombres, correos electrónicos y contraseñas según lo necesites. ¡Espero que esto te sirva!

4o mini

Tú dijiste:

listame 5 categorías de artículos con algun ejemplo de artículo, que se venda en una tienda de shopping.

ChatGPT Plus

ChatGPT

Claro, aquí tienes cinco categorías de artículos que podrías encontrar en una tienda de shopping, junto con un ejemplo para cada una:

1. **Ropa**
  - **Ejemplo:** Camiseta de algodón.
2. **Calzado**
  - **Ejemplo:** Zapatillas deportivas.
3. **Electrónica**
  - **Ejemplo:** Smartphone.
4. **Hogar y Decoración**
  - **Ejemplo:** Lámpara de mesa.
5. **Cosmética y Cuidado Personal**
  - **Ejemplo:** Crema hidratante.

Si necesitas más detalles sobre cada categoría o ejemplos adicionales, ¡déjame saber!

4o mini

**Tú dijiste:**

bien!!, ahora para cada categoría necesito una precarga de 10 artículos, siguiendo el siguiente constructor: `public Artículo(string nombre, string categoria, int precio) { this.id = ++ultimoId; this.nombre = nombre; this.categoria = categoria; this.precio = precio; }`

ChatGPT Plus

ChatGPT

Claro, aquí tienes una precarga de 10 artículos para cada categoría utilizando el constructor que proporcionaste:

## 1. Ropa

```
csharp
Copiar código
List<Articulo> ropa = new List<Articulo>
{
    new Articulo("Camiseta de algodón", "Ropa", 20),
    new Articulo("Jeans ajustados", "Ropa", 40),
    new Articulo("Chaqueta de cuero", "Ropa", 100),
    new Articulo("Vestido de verano", "Ropa", 50),
    new Articulo("Sudadera con capucha", "Ropa", 35),
    new Articulo("Camisa de manga larga", "Ropa", 30),
    new Articulo("Shorts de playa", "Ropa", 25),
    new Articulo("Falda lápiz", "Ropa", 45),
    new Articulo("Pantalones cortos", "Ropa", 30),
    new Articulo("Abrigo de invierno", "Ropa", 120)
};
```

## 2. Calzado

```

csharp
Copiar código
List<Articulo> calzado = new List<Articulo>
{
    new Articulo("Zapatillas deportivas", "Calzado", 60),
    new Articulo("Botines de cuero", "Calzado", 80),
    new Articulo("Sandalias", "Calzado", 25),
    new Articulo("Bailarinas", "Calzado", 30),
    new Articulo("Botas de lluvia", "Calzado", 50),
    new Articulo("Zapatos de vestir", "Calzado", 90),
    new Articulo("Zapatillas de running", "Calzado", 70),
    new Articulo("Zapatos casuales", "Calzado", 55),
    new Articulo("Zapatillas de skate", "Calzado", 65),
    new Articulo("Chancas", "Calzado", 15)
};

```

### 3. Electrónica

```

csharp
Copiar código
List<Articulo> electronica = new List<Articulo>
{
    new Articulo("Smartphone", "Electrónica", 300),
    new Articulo("Tableta", "Electrónica", 250),
    new Articulo("Laptop", "Electrónica", 800),
    new Articulo("Auriculares inalámbricos", "Electrónica", 100),
    new Articulo("Televisor 4K", "Electrónica", 600),
    new Articulo("Reloj inteligente", "Electrónica", 200),
    new Articulo("Altavoz Bluetooth", "Electrónica", 80),
    new Articulo("Cámara digital", "Electrónica", 400),
    new Articulo("Disco duro externo", "Electrónica", 120),
    new Articulo("Consola de videojuegos", "Electrónica", 500)
};

```

### 4. Hogar y Decoración

```

csharp
Copiar código
List<Articulo> hogarDecoracion = new List<Articulo>
{
    new Articulo("Lámpara de mesa", "Hogar y Decoración", 40),
    new Articulo("Cojín decorativo", "Hogar y Decoración", 15),
    new Articulo("Espejo de pared", "Hogar y Decoración", 50),
    new Articulo("Cuadro artístico", "Hogar y Decoración", 75),
    new Articulo("Cortinas", "Hogar y Decoración", 30),
    new Articulo("Mesa de centro", "Hogar y Decoración", 150),
    new Articulo("Sillón", "Hogar y Decoración", 200),
    new Articulo("Reloj de pared", "Hogar y Decoración", 25),
    new Articulo("Jarrón decorativo", "Hogar y Decoración", 20),
    new Articulo("Alfombra", "Hogar y Decoración", 100)
};

```

### 5. Cosmética y Cuidado Personal

```

csharp
Copiar código

```

```

List<Articulo> cosmeticaCuidado = new List<Articulo>
{
    new Articulo("Crema hidratante", "Cosmética y Cuidado Personal",
30),
    new Articulo("Champú", "Cosmética y Cuidado Personal", 15),
    new Articulo("Jabón exfoliante", "Cosmética y Cuidado Personal",
10),
    new Articulo("Perfume", "Cosmética y Cuidado Personal", 50),
    new Articulo("Desodorante", "Cosmética y Cuidado Personal", 8),
    new Articulo("Maquillaje", "Cosmética y Cuidado Personal", 25),
    new Articulo("Crema para manos", "Cosmética y Cuidado Personal",
12),
    new Articulo("Acondicionador", "Cosmética y Cuidado Personal", 15),
    new Articulo("Gel de ducha", "Cosmética y Cuidado Personal", 12),
    new Articulo("Espejo de maquillaje", "Cosmética y Cuidado Personal",
20)
};

```

## Resumen

He creado listas de artículos para cada una de las cinco categorías, cada una con 10 artículos. Puedes ajustar los nombres y precios según lo necesites. ¡Espero que esto te sea útil!

Como puedo hacer para validar un entero, para verificar si el numero entero que paso como parametro es un valor válido de una enumeracion?

ChatGPT dijo:

ChatGPT

Para validar si un número entero es un valor válido de una enumeración en C#, puedes usar el método `Enum.IsDefined`. Este método verifica si un valor específico (en este caso, un entero) corresponde a un valor definido en la enumeración.

Sintaxis para validar un entero con una enumeración:

```

csharp
Copiar código
if (Enum.IsDefined(typeof(TuEnumeracion), valorEntero))
{
    // El valor es válido en la enumeración
}
else
{
    // El valor no es válido en la enumeración
}

```