

Universidad ORT Uruguay  
Facultad de Ingeniería  
Escuela de Tecnología

## OBLIGATORIO PROGRAMACIÓN 1

### DOCUMENTO DE ANÁLISIS

Matías Pietrafesa – 318802

Emiliano Mandacen – 331839

M2E

Docente: Liliana Pino

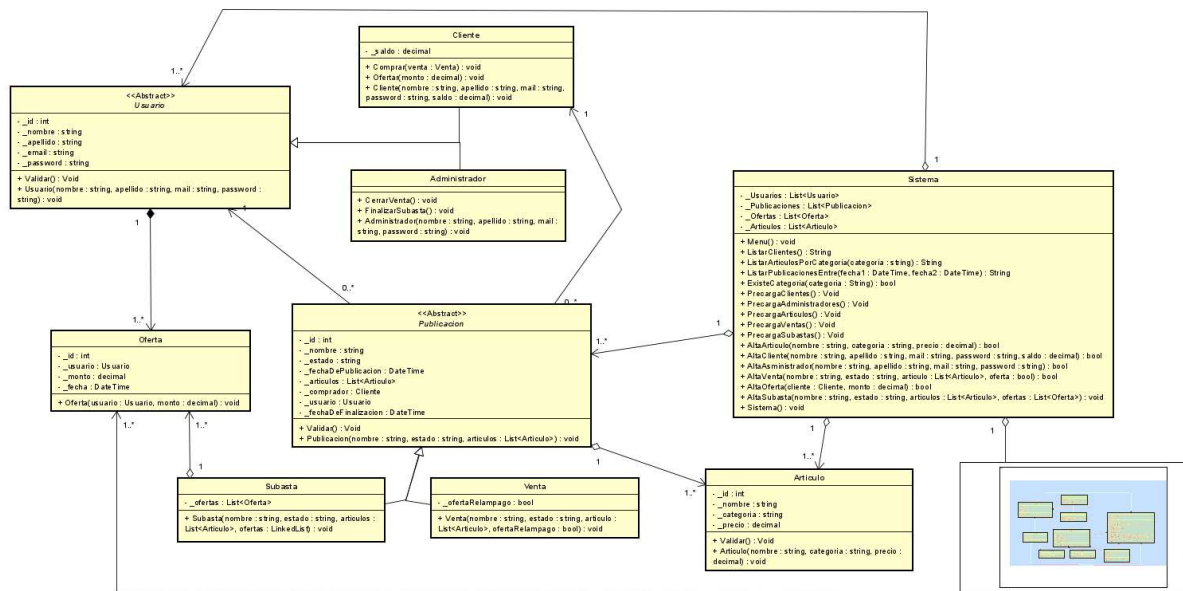
Analista en Tecnologías de la Información

Fecha de entrega del documento (10-10-2024)

## Índice


Índice .....	2
Diagrama de clases UML .....	3
Evidencia de los prompts realizados: .....	4
Codigo.....	5
//Program .....	5
//Venta.cs .....	9
//Usuario .....	10
//Subasta.....	13
//Sistema .....	15
//Publicacion .....	25
//Oferta .....	28
//Cliente .....	29
//Articulo .....	31
//Administrador.....	33
//IValidate .....	34

## Diagrama de clases UML



## Evidencia de los prompts realizados:

Precarga los datos  
Usuarios: 10 clientes, y 2 administradores.  
Artículos: 50 artículos.  
Publicaciones: 10 ventas y 10 subastas, con sus artículos (en estado ABIERTA) 2 de ellas deberán tener ofertas.

 Aquí tienes el código de precarga actualizado según las especificaciones:



```
csharp Copiar código

public Sistema()
{
    _usuarios = new List<Usuario>();
    _articulos = new List<Articulo>();
    _publicaciones = new List<Publicacion>();
    _ofertas = new List<Oferta>();
    _administradores = new List<Administrador>();

    PrecargarClientes();
    PrecargarAdministradores();
    PrecargarArticulos();
    PrecargarVentas();
    PrecargarSubastas();
}

// Precarga de clientes
private void PrecargarClientes()
{
    string[] nombresClientes = { "Juan", "Ana", "Carlos", "Laura", "Luis", "Marta", "Jorge"
    string[] apellidosClientes = { "García", "Pérez", "López", "Martínez", "Hernández", "D

    for (int i = 0; i < 10; i++)
    {
        // ...
    }
}
```

 Envía un mensaje a ChatGPT 

ChatGPT puede cometer errores. Comprueba la información importante.

## Codigo

//Program

namespace ObligatorioTiendaOnline

{

internal class Program

{

private static LogicaVentas.Sistema sistema = new LogicaVentas.Sistema();

static void Main(string[] args)

{

try

{

Menu();

}

catch (Exception ex)

{

Console.WriteLine(ex.Message);

}

}

public static void Menu()

{

int opcion = -1;

do

{

Console.Clear();

Console.WriteLine("=== Bienvenido al Menu ===\n");

```

Console.WriteLine("1. Listar Clientes");

Console.WriteLine("2. Listar articulos de una categoria");

Console.WriteLine("3. Dar de alta un nuevo articulo");

Console.WriteLine("4. Listar publicaciones entre dos fechas");

Console.WriteLine("0. Salir");


Console.Write("\nSeleccione una opcion: ");


if (!int.TryParse(Console.ReadLine(), out opcion))
{
    Console.WriteLine("\nOpcion no valida.");
    opcion = -1;
    Console.ReadKey();
}
try
{
    switch (opcion)
    {
        case 1:
            Console.WriteLine(sistema.ListarClientes());
            Console.ReadKey();
            break;
        case 2:
            Console.Write("Ingrese la categoria: ");
            string categoria = Console.ReadLine();
            if (!string.IsNullOrEmpty(categoria) &&
sistema.ExisteCategoria(categoria))
            {

```

```

        Console.WriteLine(sistema.ListarArticulosPorCategoria(categoria));
    }
    else
    {
        Console.WriteLine("La categoria ingresada no existe.");
    }

    Console.ReadKey();

    break;
case 3:
    MenuAltaArticulo();

    break;
case 4:
    Console.Write("Ingrese la primer fecha (dd/mm/yyyy): ");
    if (!DateTime.TryParse(Console.ReadLine(), out DateTime fecha1))
    {
        throw new Exception("La fecha ingresada no es valida");
    }

    Console.Write("Ingrese la segunda fecha (dd/mm/yyyy): ");
    if (!DateTime.TryParse(Console.ReadLine(), out DateTime fecha2))
    {
        throw new Exception("La fecha ingresada no es valida");
    }

    Console.WriteLine(sistema.ListarPublicacionesEntre(fecha1, fecha2));

    Console.ReadKey();

    break;
default:
    break;

```

```

        }
    }
    catch (Exception ex)
    {
        Console.WriteLine(ex.Message);
        Console.ReadKey();
    }

} while (opcion != 0);
}

public static void MenuAltaArticulo()
{
    Console.Clear();
    Console.WriteLine("=== Alta de Articulo ===\n");

    Console.Write("Ingrese el nombre del articulo: ");
    string nombre = Console.ReadLine();
    if (string.IsNullOrEmpty(nombre))
    {
        throw new Exception("El nombre no puede estar vacio.");
    }

    Console.Write("Ingrese la categoria del articulo: ");
    string categoria = Console.ReadLine();
    if (string.IsNullOrEmpty(categoria))

```



```

        {
            throw new Exception("La categoria no puede estar vacio.");
        }

        Console.Write("Ingrese el precio del articulo: ");
        if (!decimal.TryParse(Console.ReadLine(), out decimal precio))
        {
            throw new Exception("El precio ingresado para el articulo no es valido.");
        }

        if (!sistema.AltaArticulo(nombre, categoria, precio))
        {
            throw new Exception("No se pudo registrar el articulo.");
        }
    }
}

```

```

//Venta.cs
using System;

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

namespace LogicaVentas
{

```

```

public class Venta : Publicacion
{
    private bool _ofertaRelampago;

    public Venta(string nombre, string estado, List<Articulo> articulos , bool
ofertaRelampago)
        : base(nombre, estado, articulos)
    {
        _ofertaRelampago = ofertaRelampago;
    }

    public override string ToString()
    {
        return $"({Id}) - {Nombre} - {Estado} - {FechaPublicacion.ToShortDateString()}";
    }
}

```

```
//Usuario
```

```

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

```

```
namespace LogicaVentas
```

```

{
    public abstract class Usuario : IValidate
    {
        private int _id;

        private static int s_ultId = 1;

        private string _nombre;

        private string _apellido;

        private string _email;

        private string _password;


        public Usuario(string nombre, string apellido, string mail, string password)
        {
            _id = s_ultId++;

            _nombre = nombre;

            _apellido = apellido;

            _email = mail;

            _password = password;

            Validar();
        }


        internal int Id
        {
            get { return _id; }
        }


        internal string Nombre
        {

```

```
    get { return _nombre; }  
}
```

```
internal string Apellido  
{  
    get { return _apellido; }  
}
```

```
internal string Mail  
{  
    get { return _email; }  
}
```

```
public void Validar()  
{  
    if (string.IsNullOrEmpty(_nombre))  
    {  
        throw new Exception("El nombre ingresado no es valido.");  
    }  
    if (string.IsNullOrEmpty(_apellido))  
    {  
        throw new Exception("El apellido ingresado no es valido.");  
    }  
    if (string.IsNullOrEmpty(_email))  
    {  
        throw new Exception("El email ingresado no es valido.");  
    }  
}
```

```

        if (string.IsNullOrEmpty(_password))
        {
            throw new Exception("El password ingresado no es valido.");
        }
    }
}
}

```

//Subasta

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Security.Cryptography;
using System.Text;
using System.Threading.Tasks;

```

namespace LogicaVentas

```

{
    internal class Subasta : Publicacion
    {
        private List<Oferta> _ofertas;

        public Subasta(string nombre, string estado, List<Articulo> articulos, List<Oferta> ofertas)
            : base(nombre, estado, articulos)
        {
            _ofertas = ofertas;
        }
    }
}

```

```

}

public override string ToString()
{
    //string texto = "";

    //if (Estado.Trim().ToUpper() == "ABIERTA")
    //{
        // texto = $"({Id}) - {Nombre} - Abierta\n{ListarArticulos}{ListarOfertas}";
    //} else if (Estado.Trim().ToUpper() == "CERRADA")
    //{
        // texto = $"({Id}) - {Nombre} - {Estado} -
{FechaFinalizacion}\n{ListarArticulos}Ganador: {Cliente} - Cerrado por:
{Usuario}{ListarOfertas}";
    //}

    //return texto;

    return $"({Id}) - {Nombre} - {Estado} - {FechaPublicacion.ToShortDateString()}";
}

//private void ListarArticulos()
//{
// Console.WriteLine("Articulos:");
// foreach(Articulo a in Articulos)
// {
// Console.WriteLine("\t" + a.ToString());
// }
//}

//private void ListarOfertas()

```

```

    //{
    //  Console.WriteLine("Ofertas:");
    //  foreach (Oferta o in _ofertas)
    //  {
    //    Console.WriteLine("\t" + o.ToString());
    //  }
    //}
}

//Sistema

using System;
using System.Collections.Generic;
using System.Linq;
using System.Runtime.CompilerServices;
using System.Text;
using System.Threading.Tasks;
using static System.Runtime.InteropServices.JavaScript.JSType;

namespace LogicaVentas
{
    public class Sistema
    {
        private List<Usuario> _usuarios;
        private List<Articulo> _articulos;
        private List<Publicacion> _publicaciones;
        private List<Oferta> _ofertas;
    }
}

```

```
private List<Administrador> _administradores;
```

```
public Sistema()
```

```
{
```

```
    _usuarios = new List<Usuario>();
```

```
    _articulos = new List<Articulo>();
```

```
    _publicaciones = new List<Publicacion>();
```

```
    _ofertas = new List<Oferta>();
```

```
    _administradores = new List<Administrador>();
```

```
    PrecargarClientes();
```

```
    PrecargarAdministradores();
```

```
    PrecargarArticulos();
```

```
    PrecargarVentas();
```

```
    PrecargarSubastas();
```

```
}
```

```
private void PrecargarClientes()
```

```
{
```

```
    string[] nombresClientes = ["Juan", "Ana", "Carlos", "Laura", "Luis", "Marta",  
    "Jorge", "Sofía", "Pablo", "Clara"];
```

```
    string[] apellidosClientes = ["García", "Pérez", "López", "Martínez", "Hernández",  
    "Díaz", "González", "Sánchez", "Romero", "Torres"];
```

```
    for (int i = 0; i < 10; i++)
```

```
    {
```



```

        AltaCliente(nombresClientes[i], apellidosClientes[i],
"${nombresClientes[i].ToLower()}@gmail.com", $"password{i + 1}", 100m);
    }
}

```

```

private void PrecargarAdministradores()
{
    AltaAdministrador("Roberto", "Martínez", "roberto@ejemplo.com",
"passwordAdmin1");

    AltaAdministrador("Elena", "González", "elena@ejemplo.com",
"passwordAdmin2");
}

```

```

private void PrecargarArticulos()
{
    string[] nombresArticulos =
    [
        "Smartphone X1", "Laptop Pro", "Auriculares Bluetooth", "Camara DSLR", "Reloj
Inteligente",
        "Tablet Ultra", "Parlante Portatil", "Teclado Mecanico", "Monitor 4K", "Cargador
Solar",
        "Camara de Seguridad", "Impresora Inalambrica", "Consola de Videojuegos",
"Gafas de Realidad Virtual",
        "Mochila Antirrobo", "Bateria Externa", "Drone FPV", "Accesorios para
Videojuegos", "Funda para Laptop",
        "Hub USB", "Mouse Ergonomico", "Proyector Mini", "Silla Gaming", "Auriculares
Gaming",
        "Estacion de Acoplamiento", "Camara Web HD", "Pantalla Portatil", "Soporte
para Laptop",
        "Lampara LED", "Funda de Telefono", "Camara Instantanea", "Altavoz
Inteligente", "Enfriador de Aire",

```

```

        "Sistemas de Sonido", "Camaras de Accion", "Cables de Carga", "Kit de
Herramientas", "Funda para Camara",

        "Bolsa para Portatil", "Soporte de Telefono", "Accesorios de Fotografia",
"Soporte de Pared",

        "Mando a Distancia", "Luces LED", "Repetidor de Señal", "Disco Duro Externo",
"Adaptador de Viaje",

        "Camara Instantanea 2.0", "Smartphone X2", "Laptop ProMax Plus"

```

```

];

```

```

        string[] categorias = ["Electronica", "Hogar", "Ropa", "Deportes", "Libros",
"Juguetes"];

```

```

        for (int i = 0; i < 50; i++)
        {
            AltaArticulo(nombresArticulos[i], categorias[i % categorias.Length], 50 + i * 10);
        }
    }

```

```

private void PrecargarVentas()
{
    for (int i = 1; i <= 10; i++)
    {
        List<Articulo> articulosVenta = new List<Articulo> { _articulos[(i - 1) * 2],
_articulos[(i - 1) * 2 + 1] };

        AltaVenta($"Venta {i}", "ABIERTA", articulosVenta, i == 1 || i == 2);
    }
}

```

```

private void PrecargarSubastas()

```

```

{
    for(int i = 11; i <= 20; i++)
    {
        List<Articulo> articulosSubasta = new List<Articulo> { _articulos[(i - 11) * 2],
        _articulos[(i - 11) * 2 + 1] };

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

        if (i == 11 || i == 12)
        {
            AltaOferta((Cliente)_usuarios[0], 200m);
            AltaOferta((Cliente)_usuarios[1], 220m);
            ofertas.Add(_ofertas[0]);
            ofertas.Add(_ofertas[1]);
        }

        AltaSubasta($"Subasta {i - 10}", "ABIERTA", articulosSubasta, ofertas);
    }
}

```

```

public string ListarClientes()
{
    string lista = "";
    bool hayClientes = false;
    foreach (Usuario u in _usuarios)
    {
        if (u.GetType() == typeof(Cliente))

```

```

    {
        hayClientes = true;

        Cliente clie = (Cliente)u;

        lista += clie.ToString() + "\n";
    }
}

if (!hayClientes)
{
    throw new Exception("No se encontraron usuarios de tipo cliente.");
}

return lista;
}

```

```

public string ListarArticulosPorCategoria(string categoria)
{
    string lista = "";
    foreach (Articulo a in _articulos)
    {
        if (a.Categoria.Trim().ToUpper() == categoria.Trim().ToUpper())
        {
            lista += a.ToString() + "\n";
        }
    }

    return lista;
}

```

```

public bool AltaArticulo(string nombre, string categoria, decimal precio)

```

```

{
    try
    {
        Artículo art = new Artículo(nombre, categoria, precio);
        _articulos.Add(art);
        return true;
    }
    catch (Exception ex)
    {
        Console.WriteLine(ex.Message);
    }
    return false;
}

```

```

public bool AltaCliente(string nombre, string apellido, string mail, string password,
decimal saldo)

```

```

{
    try
    {
        Cliente cli = new Cliente(nombre, apellido, mail, password, saldo);
        _usuarios.Add(cli);
        return true;
    }
    catch (Exception ex)
    {
        Console.WriteLine(ex.Message);
    }
    return false;
}

```

```
}
```

```
public bool AltaAdministrador(string nombre, string apellido, string mail, string password)
```

```
{
```

```
    try
```

```
    {
```

```
        Administrador ad = new Administrador(nombre, apellido, mail, password);
```

```
        _administradores.Add(ad);
```

```
        return true;
```

```
    }
```

```
    catch (Exception ex)
```

```
    {
```

```
        Console.WriteLine(ex.Message);
```

```
    }
```

```
    return false;
```

```
}
```

```
public bool AltaVenta(string nombre, string estado, List<Articulo> articulos, bool oferta)
```

```
{
```

```
    try
```

```
    {
```

```
        Venta ven = new Venta(nombre, estado, articulos, oferta);
```

```
        _publicaciones.Add(ven);
```

```
        return true;
```

```
    }
```

```
    catch (Exception ex)
```

```

{
    Console.WriteLine(ex.Message);
}
return false;
}

```

```

public bool AltaOferta(Cliente cliente, decimal monto)

```

```

{
    try
    {
        Oferta offer = new Oferta(cliente, monto);
        _ofertas.Add(offer);
        return true;
    }
    catch (Exception ex)
    {
        Console.WriteLine(ex.Message);
    }
    return false;
}

```

```

//($"Subasta {i - 10}", "ABIERTA", DateTime.Now.AddDays(7), articulosSubasta,
(Cliente)_usuarios[i - 11], _administradores[1], ofertas);

```

```

public bool AltaSubasta(string nombre, string estado, List<Articulo> articulos,
List<Oferta> ofertas)

```

```

{
    try
    {

```

```

        Subasta subasta = new Subasta(nombre, estado, articulos, ofertas);
        _publicaciones.Add(subasta);

        return true;
    }

    catch (Exception ex)
    {
        Console.WriteLine(ex.Message);
    }

    return false;
}

```

```

public string ListarPublicacionesEntre(DateTime fecha1, DateTime fecha2)
{
    string lista = "";
    bool hayPublicaciones = false;
    foreach (Publicacion p in _publicaciones)
    {
        if (p.FechaPublicacion > fecha1 && p.FechaPublicacion < fecha2)
        {
            hayPublicaciones = true;
            lista += p.ToString() + "\n";
        }
    }

    if (!hayPublicaciones)
    {
        throw new Exception("No hay publicaciones registradas entre las fechas ingresadas.");
    }
}

```



```

        return lista;
    }

    public bool ExisteCategoria(string categoria)
    {
        foreach (Articulo a in _articulos)
        {
            if (a.Categoria.Trim().ToUpper() == categoria.Trim().ToUpper())
            {
                return true;
            }
        }
        return false;
    }
}

```

//Publicacion

```

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

```

```

namespace LogicaVentas
{
    public abstract class Publicacion : IValidate
    {
        private int _id;

        private static int s_ultId = 1;

        private string _nombre;

        private string _estado;

        private DateTime _fechaPublicacion;

        private DateTime _fechaFinalizacion;

        private List<Articulo> _articulos;

        private Cliente _cliente;

        private Usuario _usuario;

        public Publicacion(string nombre, string estado, List<Articulo> articulos)
        {
            _id = s_ultId++;

            _nombre = nombre;

            _estado = estado;

            _fechaPublicacion = DateTime.Now;

            _articulos = articulos;

            Validar();
        }

        public int Id
        {
            get { return _id; }
        }
    }
}

```

```

    }

    public string Nombre { get { return _nombre; } }
    public string Estado { get { return _estado; } }
    public DateTime FechaFinalizacion { get { return _fechaFinalizacion; } }
    public List<Articulo> Articulos { get { return _articulos; } }
    public Cliente Cliente { get { return _cliente; } }
    public Usuario Usuario { get { return _usuario; } }
    public DateTime FechaPublicacion
    {
        get { return _fechaPublicacion; }
    }

    public void Validar()
    {
        if (_nombre == null)
        {
            throw new Exception("El nombre no puede ser nulo");
        }

        if (_estado.Trim().ToUpper() != "ABIERTA" && _estado.Trim().ToUpper() !=
"CERRADA" && _estado.Trim().ToUpper() != "CANCELADA")
        {
            throw new Exception("El estado no puede ser diferente a: abierta, cerrada o
cancelada");
        }

        if (_articulos == null)
        {
            throw new Exception("La lista de articulos no puede ser nula");
        }
    }

```

```

    }
}

public override string ToString()
{
    return $"({_id}) - {_nombre} - {_estado} - {_fechaPublicacion.ToShortDateString}";
}
}
}

```

//Oferta

```

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

```

namespace LogicaVentas

```

{
    public class Oferta
    {
        private int _id;
        private static int s_ultId = 1;
        private Usuario _usuario;
        private decimal _monto;
        private DateTime _fecha;
    }
}

```

```

    public Oferta(Usuario usuario, decimal monto)
    {
        _id = s_ultId++;
        _usuario = usuario;
        _monto = monto;
        _fecha = DateTime.Now;
    }

    public override string ToString()
    {
        return $"({_id}) - {_usuario} - ${_monto} - {_fecha.ToShortDateString}";
    }
}

```

```
//Cliente
```

```

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

```

```
namespace LogicaVentas
```

```

{
    public class Cliente : Usuario
    {
        private decimal _saldo;

        public Cliente(string nombre, string apellido, string email, string password, decimal
saldo)
            : base (nombre, apellido, email, password)
        {
            _saldo = saldo;
        }

        public void Comprar(Venta venta)
        {

        }

        public void Ofertar(decimal monto)
        {

        }

        public override string ToString()
        {
            return $"({Id}) - {Apellido}, {Nombre} - {Mail} - ${_saldo}";
        }
    }
}

```

```

//Articulo

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

namespace LogicaVentas
{

    public class Articulo : IValidate
    {
        private int _id;
        private static int s_ultId = 1;
        private string _nombre;
        private string _categoria;
        private decimal _precio;

        public Articulo(string nombre, string categoria, decimal precio)
        {
            _id = s_ultId++;
            _nombre = nombre;
            _categoria = categoria;
            _precio = precio;
        }
    }
}

```

```

        Validar();
    }

    public void Validar()
    {
        if (_precio < 0)
        {
            throw new Exception("El precio no puede ser negativo");
        }
        if (_nombre == null)
        {
            throw new Exception("El nombre no puede ser nulo");
        }
        if (_categoria == null)
        {
            throw new Exception("La categoria no puede ser nula");
        }
    }

    public string Categoria
    {
        get { return _categoria; }
    }

    public override string ToString()
    {

```



```

        return $"({_id}) - {_nombre} - {_categoria} - ${_precio}";
    }

}

}

//Administrador

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

namespace LogicaVentas
{
    internal class Administrador : Usuario
    {

        public Administrador(string nombre, string apellido, string email, string password)
            : base(nombre, apellido, email, password) { }

        public void CerrarVenta()
        {

        }

    }
}

```

```

        public void FinalizarSubasta()
        {

        }

    }
}

//IValidate

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

namespace LogicaNegocio.Interfaces
{
    internal interface IValidate
    {
        void Validar();
    }
}

```