Universidad ORT Uruguay
Facultad de Ingeniería

OBLIGATORIO
PROGRAMACIÓN 2

Pablo De Melo - 330897 N2B
Fabian Fernandez - 210990 N2B

Tutor/es: Santiago Baillo/Alberto Zonca

2024

Indice

Código Comentado

- Sistema

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

namespace Dominio
{
    public class Sistema
    {
        private List<Publicacion> _listaPublicaciones = new List<Publicacion>();
        private List<Articulo> _listaArticulos = new List<Articulo>();
        private List<Usuario> _listaUsuarios = new List<Usuario>();

        #region properties
        public List<Usuario> Usuario
        {
            get
            {
                return _listaUsuarios;
            }
        }
```

```
        public List<Articulo> Articulos
        {
            get
            {
                return _listaArticulos;
            }
        }
        public List<Publicacion> Publicacion
        {
            get
            {
                return _listaPublicaciones;
            }
        }
        #endregion



public Sistema()
 {
    // precargas de Publicaciones, articulos a publicaciones y ofertas con chat gpt: https://chatgpt.com/share/67071c61-c318-800a-96ec-81e337a1cd80 /
    // https://chatgpt.com/share/66f45260-8e1c-8000-96a9-b600ab75a7fb
    PrecargarArticulos();
    PrecargarUsuarios();
    PrecargarPublicaciones();
```

```
    PrecargarArticulosAPublicaciones();
    PrecargarOfertasASubastas();
}

    #region PRECARGAS

    // Precargados 50 articulos con la ayuda de ChatGPT
    private void PrecargarArticulos()
    {
        AltaArticulo(new Articulo("Televisor", "Electronica", 450.99));
        AltaArticulo(new Articulo("Laptop", "Electronica", 999.99));
        AltaArticulo(new Articulo("Smartphone", "Electronica", 699.50));
        AltaArticulo(new Articulo("Cámara", "Fotografia", 299.99));
        AltaArticulo(new Articulo("Refrigerador", "Electrodomesticos",
1200.00));
        AltaArticulo(new Articulo("Lavadora", "Electrodomesticos", 800.75));
        AltaArticulo(new Articulo("Microondas", "Cocina", 150.99));
        AltaArticulo(new Articulo("Aspiradora", "Hogar", 250.00));
        AltaArticulo(new Articulo("Monitor", "Computacion", 199.99));
        AltaArticulo(new Articulo("Impresora", "Oficina", 120.50));
        AltaArticulo(new Articulo("Tablet", "Electronica", 350.49));
        AltaArticulo(new Articulo("Parlante Bluetooth", "Electronica", 79.99));
        AltaArticulo(new Articulo("Auriculares Inalámbricos", "Electronica",
120.89));
        AltaArticulo(new Articulo("Teclado Mecánico", "Computacion", 89.99));
        AltaArticulo(new Articulo("Ratón Inalámbrico", "Computacion", 45.99));
```

AltaArticulo(new Articulo("Disco Duro Externo", "Computacion", 75.50));

AltaArticulo(new Articulo("SSD", "Computacion", 150.00));

AltaArticulo(new Articulo("Cámara de Seguridad", "Seguridad", 199.99));

AltaArticulo(new Articulo("Termómetro Digital", "Salud", 35.99));

AltaArticulo(new Articulo("Purificador de Aire", "Hogar", 299.99));

AltaArticulo(new Articulo("Freidora de Aire", "Cocina", 180.49));

AltaArticulo(new Articulo("Licuadora", "Cocina", 99.99));

AltaArticulo(new Articulo("Horno Eléctrico", "Cocina", 249.50));

AltaArticulo(new Articulo("Plancha", "Electrodomesticos", 65.99));

AltaArticulo(new Articulo("Secadora", "Electrodomesticos", 950.75));

AltaArticulo(new Articulo("Batidora", "Cocina", 55.99));

AltaArticulo(new Articulo("Cafetera", "Cocina", 120.00));

AltaArticulo(new Articulo("Reloj Inteligente", "Electronica", 199.99));

AltaArticulo(new Articulo("Proyector", "Electronica", 499.99));

AltaArticulo(new Articulo("Router Wi-Fi", "Electronica", 75.99));

AltaArticulo(new Articulo("Silla de Oficina", "Oficina", 175.50));

AltaArticulo(new Articulo("Escritorio", "Oficina", 299.99));

AltaArticulo(new Articulo("Lámpara LED", "Hogar", 29.99));

AltaArticulo(new Articulo("Aire Acondicionado", "Electrodomesticos", 899.99));

AltaArticulo(new Articulo("Tostadora", "Cocina", 45.99));

AltaArticulo(new Articulo("Ventilador", "Hogar", 69.99));

AltaArticulo(new Articulo("Humidificador", "Hogar", 85.50));

AltaArticulo(new Articulo("Calefactor", "Hogar", 120.00));

```
            AltaArticulo(new Articulo("Lava Vajillas", "Cocina", 600.75));
            AltaArticulo(new Articulo("Cepillo Eléctrico", "Salud", 45.99));
            AltaArticulo(new Articulo("Báscula Digital", "Salud", 50.99));
            AltaArticulo(new Articulo("Smartband", "Electronica", 99.99));
            AltaArticulo(new Articulo("Bicicleta Eléctrica", "Transporte", 1500.00));
            AltaArticulo(new Articulo("Scooter Eléctrico", "Transporte", 1200.00));
            AltaArticulo(new Articulo("Tijeras Eléctricas", "Jardineria", 99.99));
            AltaArticulo(new Articulo("Robot Aspiradora", "Hogar", 399.99));
            AltaArticulo(new Articulo("Papelera Inteligente", "Hogar", 45.00));
            AltaArticulo(new Articulo("Cámara Instantánea", "Fotografia", 120.00));
            AltaArticulo(new Articulo("Trípode", "Fotografia", 79.99));
        }
        private void PrecargarUsuarios()
        {

            AltaUsuario(new Administrador("Pablo", "de Melo",
"pablodemelo@gmail.com", "pablo123"));
            AltaUsuario(new Administrador("Fabian", "Fernandez",
"fabianfernandez@gmail.com", "fabianfer333"));

            //Alta de clientes con ayuda de ChatGPT
            AltaUsuario(new Cliente("Cristian", "Rodriguez",
"cristian12@gmail.com", "abc12", 23000.00));
            AltaUsuario(new Cliente("Sofia", "Martinez",
"sofia.martinez@gmail.com", "pass123", 15000.00));
            AltaUsuario(new Cliente("Lucas", "Fernandez",
"lucas.fernandez@gmail.com", "fernandez456", 18000.00));
```

```
        AltaUsuario(new Cliente("Valentina", "Gomez",
"valentina.gomez@gmail.com", "vale789", 12000.00));

        AltaUsuario(new Cliente("Mateo", "Lopez", "mateo.lopez@gmail.com",
"lopez321", 20000.00));

        AltaUsuario(new Cliente("Camila", "Garcia",
"camila.garcia@gmail.com", "cami654", 25000.00));

        AltaUsuario(new Cliente("Juan", "Perez", "juan.perez@gmail.com",
"juan987", 17000.00));

        AltaUsuario(new Cliente("Martina", "Ruiz", "martina.ruiz@gmail.com",
"ruiz111", 22000.00));

        AltaUsuario(new Cliente("Joaquin", "Mendez",
"joaquin.mendez@gmail.com", "joaquin222", 14000.00));

        AltaUsuario(new Cliente("Renata", "Silva", "renata.silva@gmail.com",
"silva333", 26000.00));

      }

      private void PrecargarPublicaciones()

{

   //Ventas

   AltaPublicacion(new Venta(true, "Venta 1", TipoEstado.ABIERTA, new
DateTime(2024, 10, 5), null, null, null));

   AltaPublicacion(new Venta(false, "Venta 2", TipoEstado.ABIERTA, new
DateTime(2024, 10, 9), null, null, null));

   AltaPublicacion(new Venta(true, "Venta 3", TipoEstado.ABIERTA, new
DateTime(2024, 9, 28), null, null, null));

   AltaPublicacion(new Venta(false, "Venta 4", TipoEstado.ABIERTA, new
DateTime(2024, 10, 1), null, null, null));

   AltaPublicacion(new Venta(true, "Venta 5", TipoEstado.ABIERTA, new
DateTime(2024, 10, 3), null, null, null));

   AltaPublicacion(new Venta(false, "Venta 6", TipoEstado.ABIERTA, new
DateTime(2024, 9, 30), null, null, null));
```

```
    AltaPublicacion(new Venta(true, "Venta 7", TipoEstado.ABIERTA, new
DateTime(2024, 10, 7), null, null, null));

    AltaPublicacion(new Venta(false, "Venta 8", TipoEstado.ABIERTA, new
DateTime(2024, 9, 29), null, null, null));

    AltaPublicacion(new Venta(true, "Venta 9", TipoEstado.ABIERTA, new
DateTime(2024, 10, 4), null, null, null));

    AltaPublicacion(new Venta(false, "Venta 10", TipoEstado.ABIERTA, new
DateTime(2024, 10, 6), null, null, null));



    //Subastas

    AltaPublicacion(new Subasta("Subasta 1", TipoEstado.ABIERTA, new
DateTime(2024, 10, 5), null, null, null));

    AltaPublicacion(new Subasta("Subasta 2", TipoEstado.ABIERTA, new
DateTime(2024, 10, 9), null, null, null));

    AltaPublicacion(new Subasta("Subasta 3", TipoEstado.ABIERTA, new
DateTime(2024, 9, 28), null, null, null));

    AltaPublicacion(new Subasta("Subasta 4", TipoEstado.ABIERTA, new
DateTime(2024, 10, 1), null, null, null));

    AltaPublicacion(new Subasta("Subasta 5", TipoEstado.ABIERTA, new
DateTime(2024, 10, 3), null, null, null));

    AltaPublicacion(new Subasta("Subasta 6", TipoEstado.ABIERTA, new
DateTime(2024, 9, 30), null, null, null));

    AltaPublicacion(new Subasta("Subasta 7", TipoEstado.ABIERTA, new
DateTime(2024, 10, 7), null, null, null));

    AltaPublicacion(new Subasta("Subasta 8", TipoEstado.ABIERTA, new
DateTime(2024, 9, 29), null, null, null));

    AltaPublicacion(new Subasta("Subasta 9", TipoEstado.ABIERTA, new
DateTime(2024, 10, 4), null, null, null));

    AltaPublicacion(new Subasta("Subasta 10", TipoEstado.ABIERTA, new
DateTime(2024, 10, 6), null, null, null));
```

```
}
private void PrecargarOfertasASubastas()
{
    // Ofertas para la subasta 11
    AgregarOfertaASubasta(11, 3, 1500, new DateTime(2024, 10, 05));
    AgregarOfertaASubasta(11, 7, 2000, new DateTime(2024, 10, 06));
    // Ofertas para la subasta 13
    AgregarOfertaASubasta(13, 4, 1800, new DateTime(2024, 10, 03));
    AgregarOfertaASubasta(13, 9, 2500, new DateTime(2024, 10, 04));



}
private void PrecargarArticulosAPublicaciones()
{
    // Artículos para las ventas y subastas
    AgregarArticuloAPublicacion(1, 1);
    AgregarArticuloAPublicacion(1, 2);

    AgregarArticuloAPublicacion(2, 3);
    AgregarArticuloAPublicacion(2, 4);

    AgregarArticuloAPublicacion(3, 5);
    AgregarArticuloAPublicacion(3, 6);
```

AgregarArticuloAPublicacion(4, 7);

AgregarArticuloAPublicacion(4, 8);

AgregarArticuloAPublicacion(5, 9);

AgregarArticuloAPublicacion(5, 10);

AgregarArticuloAPublicacion(6, 11);

AgregarArticuloAPublicacion(6, 12);

AgregarArticuloAPublicacion(7, 13);

AgregarArticuloAPublicacion(7, 14);

AgregarArticuloAPublicacion(8, 15);

AgregarArticuloAPublicacion(8, 16);

AgregarArticuloAPublicacion(9, 17);

AgregarArticuloAPublicacion(9, 18);

AgregarArticuloAPublicacion(10, 19);

AgregarArticuloAPublicacion(10, 20);

AgregarArticuloAPublicacion(11, 21);

AgregarArticuloAPublicacion(11, 22);

AgregarArticuloAPublicacion(12, 23);

AgregarArticuloAPublicacion(12, 24);

AgregarArticuloAPublicacion(13, 25);

AgregarArticuloAPublicacion(13, 26);

AgregarArticuloAPublicacion(14, 27);

AgregarArticuloAPublicacion(14, 28);

AgregarArticuloAPublicacion(15, 29);

AgregarArticuloAPublicacion(15, 30);

AgregarArticuloAPublicacion(16, 31);

AgregarArticuloAPublicacion(16, 32);

AgregarArticuloAPublicacion(17, 33);

AgregarArticuloAPublicacion(17, 34);

AgregarArticuloAPublicacion(18, 35);

AgregarArticuloAPublicacion(18, 36);

AgregarArticuloAPublicacion(19, 37);

AgregarArticuloAPublicacion(19, 38);

AgregarArticuloAPublicacion(20, 39);

```
    AgregarArticuloAPublicacion(20, 40);

}

#endregion

        #region AGREGACIONES

        public void AgregarArticuloAPublicacion(int idPubli, int idArt)
//Agregamos un articulo a la lista de articulos de una publicacion

        {

            Articulo a = ObtenerArticulosPorId(idArt); //buscamos el articulo y
vaidamos

            if (a == null) throw new Exception("El articulo no puede ser nulo");

            Publicacion p = ObtenerPublicacionPorId(idPubli); //buscamos la
publicacion y la validamos

            if (p == null) throw new Exception("La publicacion no puede ser nula");

            p.RegistrarArticulo(a);//Añadimos el articulo a la publicacion

        }



        public void AgregarOfertaASubasta(int idSub, int idClie, double monto,
DateTime fecha)

        {

            //Agregamos una oferta a una subasta, validando si el cliente nunca

            //realizo una oferta en esa subasta y si el monto ofertado es superior al
ultimo.

            Subasta s = ObtenerSubastaPorId(idSub);

            if (s == null) throw new Exception("La subasta no puede ser nula");

            OfertaSubasta ofe = new OfertaSubasta(fecha,
ObtenerClientePorId(idClie), monto);

            s.RegistrarOferta(ofe);

        }
```

```
        #endregion


        #region ALTAS

        public void AltaArticulo(Articulo articulo)

        {

            if (articulo == null) throw new Exception("El articulo no puede ser
nulo");

            articulo.Validar();

            if (_listaArticulos.Contains(articulo)) throw new Exception("El articulo
ya existe."); //agregado con el metodo equals en Articulo

            _listaArticulos.Add(articulo);

        }

        public void AltaUsuario(Usuario usuario)

        {

            if (usuario == null) throw new Exception("El Usuario no puede ser
nulo");

            usuario.Validar();

            _listaUsuarios.Add(usuario);

        }


        public void AltaPublicacion(Publicacion publi)

        {

            if (publi == null) throw new Exception("La publicacion no puede ser
nula");

            publi.Validar();

            _listaPublicaciones.Add(publi);

        }
```

```
        #endregion


        public List<Articulo> ArticulosPorCategoria(string categoria)

        {

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

            foreach (Articulo a in _listaArticulos)

            {

                // Pasamos los dos strings a minusculas y los comparamos. si son
iguales lo agregamos a los buscados

                if (a.Categoria.ToLower() == categoria.ToLower()) buscados.Add(a);

            }

            return buscados;



        }


        public List<Publicacion> ListarPublicacionesEntreFechas(DateTime
fechaInicio, DateTime fechaFin)

        {

            List<Publicacion> buscados = new List<Publicacion>();

            foreach (Publicacion p in _listaPublicaciones)

            {

                //comprobamos que la fecha de publicacion de una publicacion este
dentro de las dos fechas solicitadas.

                if (p.FechaPublicacion >= fechaInicio && p.FechaPublicacion <=
fechaFin) buscados.Add(p);

            }

            return buscados;
```

```
        }
public List<Cliente> ListarClientes()
{
    List<Cliente> buscados = new List<Cliente>();
    foreach (Usuario u in _listaUsuarios)
    {
        // comprueba que Usuario u sea un Cliente y no Administrador y lo agrega
a la lista.
        if (u is Cliente cliente)
        {
            buscados.Add(u as Cliente);
        }

    }
    return buscados;
}

        #region OBTENER POR ID
        public Cliente ObtenerClientePorId(int id)
        {
            Cliente buscado = null;
            int i = 0;

            while (i < _listaUsuarios.Count && buscado == null)
            {
```

```
        // Intentamos convertir el usuario a Cliente usando 'as'
        Cliente cliente = _listaUsuarios[i] as Cliente;

        // Si la conversión fue exitosa (cliente no es null) y el ID coincide
        if (cliente != null && cliente.Id == id)
        {
            buscado = cliente;
        }

        i++;
    }

    return buscado;
}
public Administrador ObtenerAdministradorPorId(int id)
{
    Administrador buscado = null;
    int i = 0;

    while (i < _listaUsuarios.Count && buscado == null)
    {
        // Intentamos convertir el usuario a Administrador usando 'as'
        Administrador administrador = _listaUsuarios[i] as Administrador;
```

```
            // Si la conversión fue exitosa (administrador no es null) y el ID
coincide
            if (administrador != null && administrador.Id == id)
            {
                buscado = administrador;
            }

            i++;
        }

        return buscado;
    }
    public Publicacion ObtenerPublicacionPorId(int id)
    {
        Publicacion buscado = null;
        int i = 0;
        while (i < _listaPublicaciones.Count && buscado == null)
        {
            if (_listaPublicaciones[i].Id == id) buscado = _listaPublicaciones[i];
            i++;
        }
        return buscado;
    }
    public Articulo ObtenerArticulosPorId(int id)
    {
```

```
    Articulo buscado = null;
    int i = 0;
    while (i < _listaArticulos.Count && buscado == null)
    {
        if (_listaArticulos[i].Id == id) buscado = _listaArticulos[i];
        i++;
    }

    return buscado;
}

public Subasta ObtenerSubastaPorId(int id)
{
    Subasta buscado = null;
    int i = 0;
    while (i < _listaPublicaciones.Count && buscado == null)
    {
        Subasta sub = _listaPublicaciones[i] as Subasta;

        if (_listaPublicaciones[i].Id == id) buscado = sub;
        i++;
    }

    return buscado;
}
```

```
    }


}
#endregion
```

- Usuario

```
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

    {

        protected int _id;

        protected static int s_ultId = 1;

        protected string _nombre;

        protected string _apellido;

        protected string _email;

        protected string _contrasena;


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

        {

            _id = s_ultId;

            s_ultId++;

            _nombre = nombre;
```

```
            _apellido = apellido;
            _email = email;
            _contrasena = contrasena;
        }

        public void Validar()
        {
            if (string.IsNullOrEmpty(_nombre) || string.IsNullOrEmpty(_apellido) ||
                string.IsNullOrEmpty(_email) || string.IsNullOrEmpty(_contrasena))
                throw new Exception("Los campos de nombre, apellido, contraseña y
email son obligatorios.");
        }

        public override string ToString()
        {
            return $"nombre: {_nombre} - apellido: {_apellido} email: {_email}";
        }
    }
}
```

- Cliente

```
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

    {

        private double _saldo;


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

        {

            _saldo = saldo;

        }

        public int Id

        {

            get { return _id; }

        }

        public double Saldo

        {
```

```
            get
            {
                return _saldo;
            }
        }
        public void DescontarSaldo(double monto) // metodo no requerido para la primer entrega.
        {
            if (monto > _saldo)
            {
                throw new Exception("Saldo insuficiente.");
            }
            else
            {
                _saldo -= monto;
            }
        }

        public override string ToString()
        {
            return $"id: {_id} - Nombre: {_nombre} {_apellido} - Email: {_email} - saldo: ${_saldo}";
        }

        public override bool Equals(object? obj)
        {
```

```
            Cliente c = obj as Cliente;
            return c != null && this._id.Equals(c._id);
        }
    }
}
```

- Administración

```
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
    {
        public Administrador(string nombre, string apellido, string email, string contrasena):base(nombre, apellido, email, contrasena)
        {

        }
        public int Id
        {
```

```
        get { return _id; }
    }
  }
}
```

- Publicación

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

namespace Dominio
{
  public abstract class Publicacion : IValidable
  {
    protected int _id;
    protected static int s_ultId = 1;
    protected string _nombre;
    protected TipoEstado _estado;
    protected DateTime _fechaPublicacion;
    protected List<Articulo> _listaArticulos = new List<Articulo>();
    protected Cliente? _comprador;
```

protected Usuario _usuarioCierre; //modifique el usuario cierre colocando Usuario en vez de Administrador

protected DateTime? _fechaCierre;

public Publicacion(string nombre, TipoEstado estado, DateTime fechaPublicacion, Cliente? comprador, Usuario usuarioCierre, DateTime? fechaCierre)

{

_id = s_ultId;

s_ultId++;

_nombre = nombre;

_estado = estado;

_fechaPublicacion = fechaPublicacion;

_comprador = comprador;

_usuarioCierre = usuarioCierre;

_fechaCierre = fechaCierre;

}

- TipoEstado

{

get { return _estado; }

}

public DateTime FechaPublicacion

{

get { return _fechaPublicacion; }

}

```
        public int Id
        {
            get { return _id; }
        }
        public string Nombre
        {
            get { return _nombre; }
        }

        public virtual void Validar()
        {
            if (string.IsNullOrEmpty(_nombre)) throw new Exception("El nombre no puede ser vacio");
            if (_estado != TipoEstado.ABIERTA) throw new Exception("La publicación no se encuentra en estado ABIERTA.");
        }

        public void ValidarArticulosPublicacion()
        {
            if(_listaArticulos.Count == 0) throw new Exception("La publicación debe contener al menos un artículo.");
        }

        public override string ToString()
        {
```

```
            return $"ID: {_id} - Nombre: {_nombre} - Estado: {_estado} - Fecha de
Publicación: {_fechaPublicacion}";


        }


        public void RegistrarArticulo(Articulo a)

        {

            if (a == null) throw new Exception("El articulo no puede ser nulo");

            a.Validar();

            _listaArticulos.Add(a);

        }


    }

}
```

- Subasta

```
using Dominio.Interfaces;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;


namespace Dominio

{

   public class Subasta : Publicacion

   {

      private List<OfertaSubasta>? _listaOferta = new List<OfertaSubasta>();
//Agregue el "new List<OfertaSubasta>();" para crear el objeto y cuando
apliquemos add, pueda agregar las ofertas



      public Subasta(string nombre, TipoEstado estado, DateTime
fechaPublicacion, Cliente? comprador, Administrador? usuarioCierre,
DateTime? fechaCierre):base(nombre, estado, fechaPublicacion, comprador,
usuarioCierre, fechaCierre)

      {



      }

      public List<OfertaSubasta> Ofertas

      {

         get { return _listaOferta; }

      }
```

```
        public override void Validar(){
            ValidarSaldo();
            ValidarEstado();
            ValidarOfertasValidas();
        }
        #region Metodos que no son requeridos para la primer entrega. (Cerrar subasta y distintas validaciones)
        public void ValidarSaldo()
        {
            foreach (OfertaSubasta of in _listaOferta)
            {
                if (of.Cliente.Saldo < of.Monto)
                    throw new Exception($"El Cliente {of.Cliente.Id} no tiene saldo suficiente para cubrir la oferta.");

                // Descontar el saldo si es suficiente
                of.Cliente.DescontarSaldo(of.Monto);
            }
        }

        public void ValidarOfertasValidas()
        {
            foreach (OfertaSubasta of in _listaOferta)
            {
                // Validar que la oferta no sea nula
                if (of == null)
```

```
            {
                throw new Exception("La oferta no puede ser nula.");
            }

            // Validar que el monto de la oferta sea positivo
            if (of.Monto <= 0)
            {
                throw new Exception("La oferta debe tener un monto positivo.");
            }

        }
    }

    public void ValidarEstado()
    {
        if (_estado != TipoEstado.ABIERTA) throw new Exception("La subasta
no está en estado ABIERTA.");
    }

    public void Cerrar()
    {
        ValidarEstado(); // Verificar que la subasta esté ABIERTA.

        OfertaSubasta mejorOferta = null;
```

```
        // Recorrer la lista de ofertas para encontrar la mejor oferta con saldo suficiente
        foreach (OfertaSubasta of in _listaOferta)
        {
            if (of.Cliente.Saldo >= of.Monto)
            {
                if (mejorOferta == null || of.Monto > mejorOferta.Monto)
                {
                    mejorOferta = of;
                }
            }
        }

        // Validar si hay una oferta válida con saldo suficiente
        if (mejorOferta != null)
        {
            mejorOferta.Cliente.DescontarSaldo(mejorOferta.Monto);
            _comprador = mejorOferta.Cliente;
            _estado = TipoEstado.CERRADA;
            _fechaCierre = DateTime.Now;
        }
        else
        {
            throw new Exception("Ningún oferente tiene saldo suficiente para adjudicarse la subasta.");
        }
```

```
        }

        public void FinalizarSubasta(Administrador admin)
        {
            if (admin == null) throw new Exception("Solo un administrador puede
cerrar la subasta.");

            Cerrar(); // Cierra la subasta si las condiciones son correctas.
        }
        #endregion

        public void RegistrarOferta(OfertaSubasta ofe)
        {

            foreach (OfertaSubasta o in _listaOferta) //comprobamos que un cliente
realize unicamente una oferta
            {
                if (o.Cliente.Equals(ofe.Cliente)) throw new Exception("El cliente ya
realizo una oferta en esta publicacion.");
            }

            //comprobamos que un cliente realize una oferta con el monto mas alto
al anterior
            double ultMonto = 0;
            if (_listaOferta.Count > 0) ultMonto = _listaOferta[(_listaOferta.Count -
1)].Monto;
            if (ofe == null) throw new Exception("La oferta no puede ser nula");
```

```
            if (ofe.Monto < ultMonto) throw new Exception("La oferta no puede ser
menor a la oferta anterior");

            ofe.Validar();

            _listaOferta.Add(ofe);

        }


    }

}
```

- Venta

```
using Dominio.Interfaces;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using static System.Runtime.InteropServices.JavaScript.JSType;


namespace Dominio

{

    public class Venta : Publicacion

    {

        private bool _ofertaRelampago;


        public Venta(bool ofertaRelampago, string nombre, TipoEstado estado,
DateTime fechaPublicacion, Cliente? comprador, Cliente? usuarioCierre,
DateTime? fechaCierre) :base(nombre, estado, fechaPublicacion, comprador,
usuarioCierre, fechaCierre)

        {

            _ofertaRelampago = ofertaRelampago;

        }


        public string TieneOfertaRelampago() //retornar si la venta tien oferta
relampago

        {

            if (_ofertaRelampago) return "Si";
```

```
            else return "No";
        }
    }
}
```

- OfertaSubasta

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

namespace Dominio
{
    public class OfertaSubasta : IValidable
    {
        private int _id;
        private int s_ultId = 1;
        private Cliente _cliente;
        private double _monto;
        private DateTime _fecha;

        public OfertaSubasta(DateTime fecha, Cliente cliente, double monto)
        {
```

```
            _id = s_ultId;
            s_ultId++;
            _cliente = cliente;
            _monto = monto;
            _fecha = fecha;
        }

        public Cliente Cliente
        {
            get { return _cliente; }
        }

        public double Monto
        {
            get { return _monto; }
        }

        public void Validar()
        {
            if (_monto <= 0) throw new Exception("El monto no puede ser negativo
o cero");
        }

        public override string ToString()
        {
```

```
            return $"id: {_id} cliente: {_cliente} - monto: {_monto} - fecha {_fecha}";
        }
    }
}
```

- Articulo

```
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 _precioVenta;

        public Articulo(string nombre, string categoria, double precioVenta)
        {
            _id = s_ultId;
```

```
    s_ultId++;
    _nombre = nombre;
    _categoria = categoria;
    _precioVenta = precioVenta;
}
public double PrecioVenta
{
    get { return _precioVenta; }
}
public int Id
{
    get { return _id; }
}
public string Nombre
{
    get { return _nombre; }
}

public string Categoria
{
    get { return _categoria; }
}

public void Validar()
{
```

```
        if (string.IsNullOrEmpty(_nombre)) throw new Exception("Por favor! Ingrese el nombre del articulo!");

        if (string.IsNullOrEmpty(_categoria)) throw new Exception("Por favor! Ingrese una categoria!");

        if (_precioVenta <= 0) throw new Exception("El precio debe ser mayor a 0");

    }



    public override string ToString()

    {

        return $"Articulo: {_nombre} - Categoria: {_categoria} - Precio: ${_precioVenta}";

    }



    public override bool Equals(object? obj)

    {

        Articulo a = obj as Articulo;

        return a != null && this._id.Equals(a._id);

    }



  }

}
```

- Interfaz IValidable

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

namespace Dominio.Interfaces
{
    internal interface IValidable
    {
        public void Validar();
    }
}
```

- Program

```
using Dominio;

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":

                  ListadoClientes();

                  break;

               case "2":
```

```
                ListarArticulosPorNombreCategoria();
                break;
            case "3":
                ListarPublicacionesEntreFechas();
                break;
            case "4":
                AltaArticulo();
                break;
            case "0":
                Console.WriteLine("Salir ...");
                break;
            default:
                MostrarError("Debe ingresar una opcion valida");
                PressToContinue();
                break;
        }
    }
}

#region METODOS DE MENU

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

```
        CambioDeColor("**************************************************",
ConsoleColor.Yellow);

        CambioDeColor("*                MENU                  *",
ConsoleColor.Yellow);

        CambioDeColor("**************************************************",
ConsoleColor.Yellow);

        CambioDeColor("*                                      *",
ConsoleColor.Yellow);

        CambioDeColor("* 1 - Listado de clientes              *",
ConsoleColor.Yellow);

        CambioDeColor("* 2 - Listado de articulos por nombre de categoria *",
ConsoleColor.Yellow);

        CambioDeColor("* 3 - Listado de publicaciones por fecha    *",
ConsoleColor.Yellow);

        CambioDeColor("* 4 - Alta de Articulos                *",
ConsoleColor.Yellow);

        CambioDeColor("* 0 - Salir                            *",
ConsoleColor.Yellow);

        CambioDeColor("**************************************************",
ConsoleColor.Yellow);

    }


    static void ListadoClientes()

{

   Console.Clear();

   CambioDeColor("Listado de Clientes", ConsoleColor.Yellow);

   Console.WriteLine();

   List<Cliente> listarClie = miSistema.ListarClientes();
```

```
    if (listarClie.Count == 0)
    {
        MostrarError($"No hay clientes registrados en el sistema");
    }
    else
    {
        foreach (Cliente c in listarClie)
        {
            Console.WriteLine(c);
        }
    }
    PressToContinue();
}



        static void ListarArticulosPorNombreCategoria()
        {
            Console.Clear();
            CambioDeColor("LISTA DE ARTICULOS POR CATEGORIA",
ConsoleColor.Yellow);
            Console.WriteLine();


            string pedirCategoria = PedirPalabras("Ingrese la categoria para
buscar: ");
            Console.WriteLine();
```

```
        try
        {
        List<Articulo> articulosBuscados =
miSistema.ArticulosPorCategoria(pedirCategoria);
        if(articulosBuscados.Count == 0)
        {
            MostrarError("No existen articulos con dicha categoria!");
        }
        else
        {
            foreach(Articulo a in articulosBuscados)
            {
                Console.WriteLine(a);
            }
        }
        }
        catch(Exception ex)
        {
            MostrarError(ex.Message);
        }

        PressToContinue();
    }
    static void AltaArticulo()
    {
```

```
Console.Clear();
CambioDeColor("ALTA DE ARTICULOS", ConsoleColor.Yellow);
Console.WriteLine();

string nombre = PedirPalabras("Ingrese el nombre de un articulo: ");
string categoria = PedirPalabras("Ingrese una categoria: ");
bool exito = false;
double precioVenta = 0;

do
{
   Console.Write("Ingrese un precio de venta: ");
   exito = double.TryParse(Console.ReadLine(), out precioVenta);
   if (!exito) MostrarError("ERROR: Debe ingresar solo números.");
}
while (!exito);

try
{
   Articulo miArticulo = new Articulo(nombre, categoria, precioVenta);
   miSistema.AltaArticulo(miArticulo);
   MostrarExito("Articulo dado de alta con exito!");
}
catch (Exception ex)
{
```

```
            MostrarError(ex.Message);
        }

        PressToContinue();

    }
    static void ListarPublicacionesEntreFechas()
    {
        Console.Clear();
        CambioDeColor("LISTAR PUBLICACIONES ENTRE DOS FECHAS",
ConsoleColor.Yellow);
        Console.WriteLine();

        DateTime fechaInicio;
        DateTime fechaFin;

        while (true)
        {
            fechaInicio = PedirFecha("Ingrese la fecha de inicio");
            fechaFin = PedirFecha("Ingrese la fecha de fin");

            // Validar que la fecha de inicio no sea mayor que la fecha de fin
            if (fechaInicio > fechaFin)
            {
                MostrarError($"La fecha de inicio no puede ser mayor que la fecha
de fin. Intente de nuevo.");
```

```
            }
            else
            {
                break; // Salir del bucle si las fechas son válidas
            }
        }

        List<Publicacion> listarPub =
miSistema.ListarPublicacionesEntreFechas(fechaInicio, fechaFin);

        if (listarPub.Count == 0)
        {
            MostrarError($"No existen publicaciones entre las fechas
{fechaInicio} y {fechaFin}");
        }
        else
        {
            foreach (Publicacion p in listarPub)
            {
                Console.WriteLine(p);
            }
        }

        PressToContinue();
    }
```

```
#endregion

#region METODOS ADICIONALES

static string PedirPalabras(string mensaje)
{
    Console.Write(mensaje);
    string datos = Console.ReadLine();
    return datos;
}
static void PressToContinue()
{
    Console.WriteLine();
    Console.WriteLine("Presione una tecla para continuar ...");
    Console.ReadKey();
}
static DateTime PedirFecha(string mensaje)
{
    bool exito = false;
    DateTime fecha = new DateTime();

    while (!exito)
    {
        Console.Write($"{mensaje} [MM/dd/yyyy]:");
```

```
            exito = DateTime.TryParse(Console.ReadLine(), out fecha);


            if (!exito)

            {

                MostrarError("ERROR: La fecha no respeta el formato
MM/dd/yyyy");

            }

        }


        return fecha;

    }


    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 CambioDeColor(string mensaje, ConsoleColor color)
        {
            Console.ForegroundColor = color;
            Console.WriteLine(mensaje);
            Console.ForegroundColor = ConsoleColor.Gray;
        }

        #endregion
    }
}
```

Tabla de Precarga

- PrecargarArticulos()

string nombre	**string categoria**	**double precioVenta**
Televisor	Electronica	450,99
Laptop	Electronica	999,99
Smartphone	Electronica	699,5
Cámara	Fotografia	299,99
Refrigerador	Electrodomesticos	1200
Lavadora	Electrodomesticos	800,75
Microondas	Cocina	150,99
Aspiradora	Hogar	250
Monitor	Computacion	199,99
Impresora	Oficina	120,5
Tablet	Electronica	350,49
Parlante Bluetooth	Electronica	79,99
Auriculares Inalámbricos	Electronica	120,89
Teclado Mecánico	Computacion	89,99
Ratón Inalámbrico	Computacion	45,99
Disco Duro Externo	Computacion	75,5
SSD	Computacion	150
Cámara de Seguridad	Seguridad	199,99
Termómetro Digital	Salud	35,99
Purificador de Aire	Hogar	299,99
Freidora de Aire	Cocina	180,49
Licuadora	Cocina	99,99
Horno Eléctrico	Cocina	249,5
Plancha	Electrodomesticos	65,99
Secadora	Electrodomesticos	950,75
Batidora	Cocina	55,99
Cafetera	Cocina	120
Reloj Inteligente	Electronica	199,99
Proyector	Electronica	499,99
Router Wi-Fi	Electronica	75,99
Silla de Oficina	Oficina	175,5
Escritorio	Oficina	299,99
Lámpara LED	Hogar	29,99
Aire Acondicionado	Electrodomesticos	899,99
Tostadora	Cocina	45,99

Ventilador	Hogar	69,99
Humidificador	Hogar	85,5
Calefactor	Hogar	120
Lava Vajillas	Cocina	600,75
Cepillo Eléctrico	Salud	45,99
Báscula Digital	Salud	50,99
Smartband	Electronica	99,99
Bicicleta Eléctrica	Transporte	1500
Scooter Eléctrico	Transporte	1200
Tijeras Eléctricas	Jardineria	99,99
Robot Aspiradora	Hogar	399,99
Papelera Inteligente	Hogar	45
Cámara Instantánea	Fotografia	120
Trípode	Fotografia	79,99

- PrecargarUsuarios()

Administradores

string nombre	**string apellido**	**string email**	**string contrasena**
Pablo	de Melo	pablodemelo@gmail.com	pablo123
Fabian	Fernandez	fabianfernandez@gmail.com	fabianfer333

Clientes

string nombre	**string apellido**	**string email**	**string contrasena**	**double saldo**
Cristian	Rodriguez	cristian12@gmail.com	abc12	23000
Sofia	Martinez	sofia.martinez@gmail.com	pass123	15000
Lucas	Fernandez	lucas.fernandez@gmail.com	fernandez456	18000
Valentina	Gomez	valentina.gomez@gmail.com	vale789	12000
Mateo	Lopez	mateo.lopez@gmail.com	lopez321	20000
Camila	Garcia	camila.garcia@gmail.com	cami654	25000
Juan	Perez	juan.perez@gmail.com	juan987	17000
Martina	Ruiz	martina.ruiz@gmail.com	ruiz111	22000
Joaquin	Mendez	joaquin.mendez@gmail.com	joaquin222	14000
Renata	Silva	renata.silva@gmail.com	silva333	26000

- PrecargarPublicaciones()

Venta

bool ofertaRelampago	string nombre	TipoEstado estado	DateTime fechaPublicacion	Cliente? comprador	Cliente? usuarioCierre	DateTime? fechaCierre
true	Venta 1	ABIERTA	2024-10-5	null	null	null
false	Venta 2	ABIERTA	2024-10-9	null	null	null
true	Venta 3	ABIERTA	2024-9-28	null	null	null
false	Venta 4	ABIERTA	2024-10-1	null	null	null
true	Venta 5	ABIERTA	2024-10-3	null	null	null
false	Venta 6	ABIERTA	2024-9-30	null	null	null
true	Venta 7	ABIERTA	2024-10-7	null	null	null
false	Venta 8	ABIERTA	2024-9-29	null	null	null
true	Venta 9	ABIERTA	2024-10-4	null	null	null
false	Venta 10	ABIERTA	2024-10-6	null	null	null

Subasta

string nombre	TipoEstado estado	DateTime fechaPublicacion	Cliente? comprador	Cliente? usuarioCierre	DateTime? fechaCierre
Subasta 1	ABIERTA	2024-10-5	null	null	null
Subasta 2	ABIERTA	2024-10-9	null	null	null
Subasta 3	ABIERTA	2024-9-28	null	null	null
Subasta 4	ABIERTA	2024-10-1	null	null	null
Subasta 5	ABIERTA	2024-10-3	null	null	null
Subasta 6	ABIERTA	2024-9-30	null	null	null
Subasta 7	ABIERTA	2024-10-7	null	null	null
Subasta 8	ABIERTA	2024-9-29	null	null	null
Subasta 9	ABIERTA	2024-10-4	null	null	null
Subasta 10	ABIERTA	2024-10-6	null	null	null

- PrecargarOfertasASubastas()

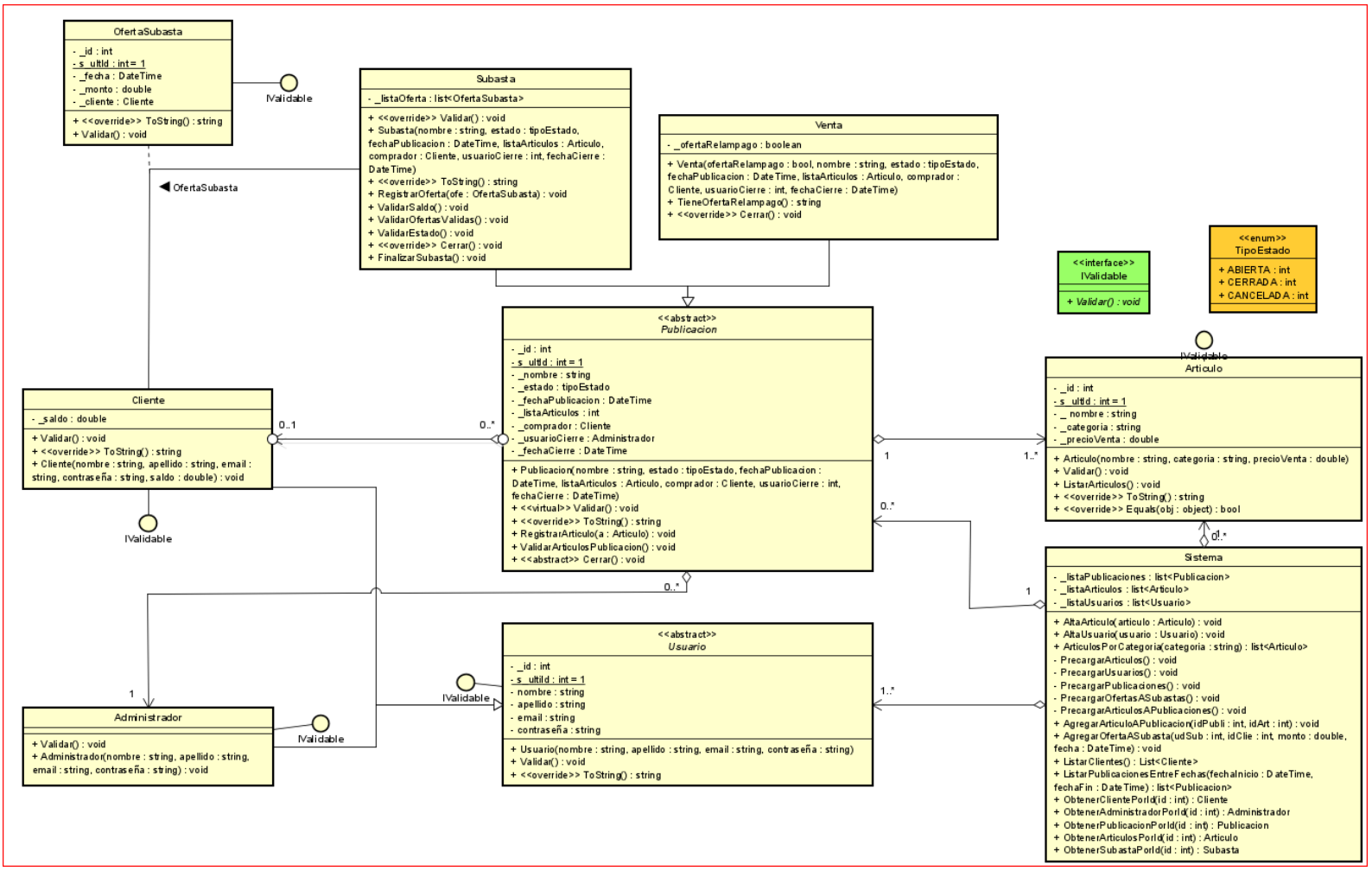
int idSub	int idClie	double monto	DateTime fecha
11	3	1500	2024-10-05
11	7	2000	2024-10-06
13	4	1800	2024-10-03
13	9	2500	2024-10-04

- PrecargarArticulosAPublicaciones()

int idPubli	int idArt
1	1
1	2
2	3
2	4
3	5
3	6
4	7
4	8
5	9
5	10
6	11
6	12
7	13
7	14
8	15
8	16
9	17
9	18
10	19
10	20
11	21
11	22
12	23
12	24
13	25
13	26
14	27
14	28
15	29
15	30
16	31
16	32

17	33
17	34
18	35
18	36
19	37
19	38
20	39
20	40

Diagrama de clases UML dominio



Precarga de Entidades

- **https://chatgpt.com/share/67071c61-c318-800a-96ec-81e337a1cd80**
- **https://chatgpt.com/share/66f45260-8e1c-8000-96a9-b600ab75a7fb**

ANEXOS

Anexo 1

El presente Obligatorio presenta una plataforma en C# para la gestión de ventas y subastas de artículos, operada por clientes y administradores. Los clientes pueden adquirir artículos a través de compras directas o mediante subastas, donde realizan una única oferta. Los administradores son responsables de cerrar las subastas y adjudicar el artículo al cliente con la mejor oferta.

El sistema incluye la validación de saldo de los clientes al momento de ofertar o realizar compras, asegurando que los fondos sean suficientes para completar las transacciones. La plataforma organiza las publicaciones, permitiendo a los clientes participar en ventas de artículos listados con precios fijos o en subastas donde el precio final es determinado por la oferta más alta. Las publicaciones tienen un estado (ABIERTA, CERRADA o CANCELADA), y el cierre de las mismas están bajo la supervisión de los administradores.

Las clases desarrolladas hasta el momento permiten gestionar usuarios, clientes, artículos y las ofertas en subastas. Además, el sistema controla la relación entre publicaciones y artículos, garantizando que un artículo no pueda estar en dos publicaciones al mismo tiempo.

A pesar de los avances, aún quedan por implementar los métodos de cálculo de precios de las publicaciones y los métodos de finalización de las mismas. Estas funcionalidades serán agregadas en futuras etapas para mejorar el funcionamiento general de la plataforma.

Palabras claves

- publicaciones
- subasta
- usuario
- administrador
- cliente
- venta
- validar
- id
- nombre
- lista
- articulo

- alta
- precarga
- categoría
- fecha
- obtener
- monto
- cierre
- comprador
- relampago
- oferta
- tostring
- equals
- enum

Anexo 2

Nosotros, Pablo de Melo y Fabian Fernandez, declaramos que el trabajo que se presenta en esta obra es de nuestra propia mano. Podemos asegurar que:

- La obra fue producida en su totalidad mientras realizábamos el obligatorio asignado.

- Cuando hemos consultado el trabajo publicado por otros, lo hemos atribuido con claridad.

- Cuando hemos citado obras de otros, hemos indicado las fuentes. Con excepción de estas citas, la obra es enteramente nuestra.

- En la obra, hemos acusado recibo de las ayudas recibidas por medio de ChatGPT.

- Cuando la obra se basa en trabajo realizado conjuntamente con otros, hemos explicado claramente qué fue contribuido por otros, y que fue contribuido por nosotros.

- Ninguna parte de este trabajo ha sido publicada previamente a su entrega, excepto donde se han realizado las aclaraciones correspondientes.