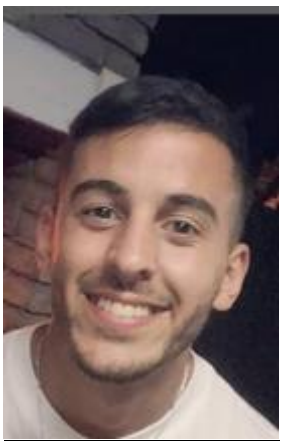
Universidad ORT Uruguay Facultad de Ingeniería Escuela de Tecnología

OBLIGATORIO PROGRAMACIÓN 2 DOCUMENTACION

Federico Gallo- 210578

Marcio Huertas – 279268



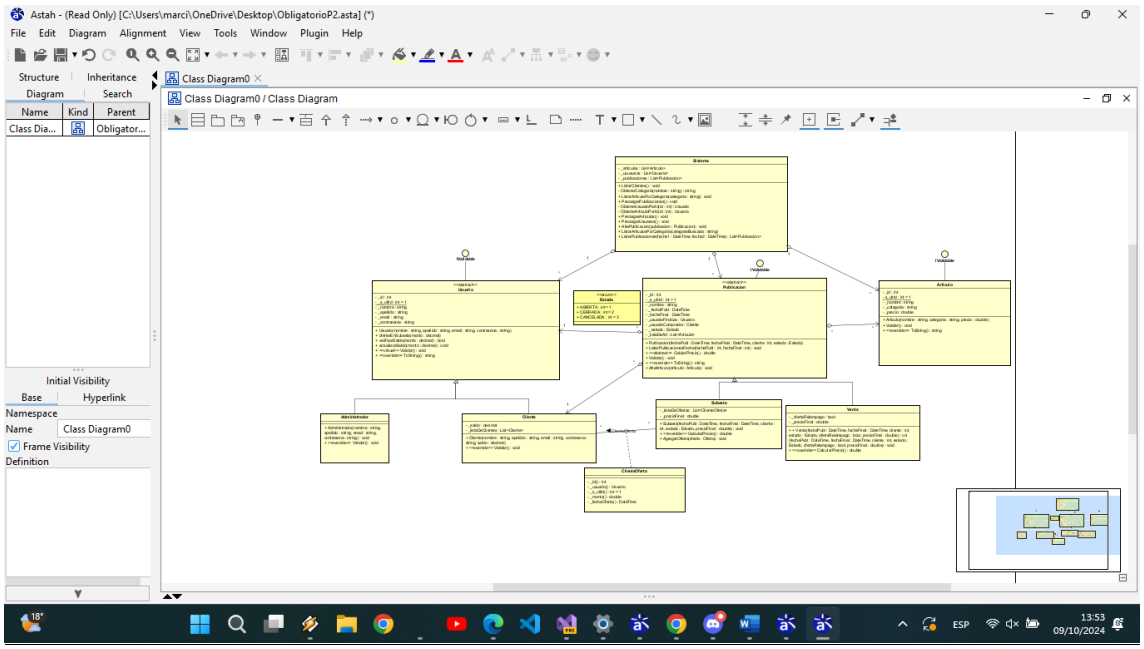
N2B – Remoto

Docente: Santiago Baillo

Analista en Tecnologías de la información 28/11/2024

Contenido

Diagrama de clases completo del Dominio del problema.



Código fuente consola:

Program.cs

```
namespace Obligatoriop2
{
    public class Program
    {
        public static void Main(string[] args)
        {
            var builder = WebApplication.CreateBuilder(args);

            // Add services to the container.
            builder.Services.AddControllersWithViews();

            //PARA USAR VARIABLES DE SESSION: PONER ESTA LINEA ANTES DE var app = builder.Build();
            builder.Services.AddSession();

            var app = builder.Build();

            // Configure the HTTP request pipeline.
            if (!app.Environment.IsDevelopment())
            {
                app.UseExceptionHandler("/Home/Error");
            }
            app.UseStaticFiles();
```

```
        app.UseRouting();

        app.UseAuthorization();

        app.MapControllerRoute(
            name: "default",
            pattern: "{controller=Home}/{action=Index}/{id?}");

        //PARA USAR VARIABLES DE SESSION: PONER ESTA LINEA ANTES DE app.Run();
        app.UseSession();
        app.Run();
    }
  }
}
```

Layout

```
<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="utf-8" />

    <meta name="viewport" content="width=device-width, initial-scale=1.0" />

    <title>@ViewData["Title"] - Obligatoriop2</title>

    <link rel="stylesheet" href="~/lib/bootstrap/dist/css/bootstrap.min.css" />

    <link rel="stylesheet" href="~/css/site.css" asp-append-version="true" />

    <link rel="stylesheet" href="~/Obligatoriop2.styles.css" asp-append-version="true" />

</head>

<body>

    <header>

        <nav class="navbar navbar-expand-sm navbar-toggleable-sm navbar-light bg-white border-bottom box-shadow mb-3">

            <div class="container-fluid">

                <a class="navbar-brand" asp-area="" asp-controller="Home" asp-action="Index">Obligatoriop2</a>

                <button class="navbar-toggler" type="button" data-bs-toggle="collapse" data-bs-target=".navbar-collapse" aria-controls="navbarSupportedContent"

                        aria-expanded="false" aria-label="Toggle navigation">

                    <span class="navbar-toggler-icon"></span>

                </button>

                <div class="navbar-collapse collapse d-sm-inline-flex justify-content-between">

                    <ul class="navbar-nav flex-grow-1">
```

```
            <li class="nav-item">
                <a class="nav-link text-dark" asp-area="" asp-controller="Home" asp-action="Index">Home</a>
            </li>
            <li class="nav-item">
                <a class="nav-link text-dark" asp-area="" asp-controller="Home" asp-action="Privacy">Privacy</a>
            </li>

            @if (Context.Session.GetString("rol") != null)
            {
                <!-- Menú para Administrador -->
                @if (Context.Session.GetString("rol") == "Admin")
                {
                <li class="nav-item">
                    <a class="nav-link text-dark" asp-area="" asp-controller="Publicaciones" asp-action="ListadoSubastas">Subastas</a>
                </li>

                }

                <!-- Menú para Cliente -->
                @if (Context.Session.GetString("rol") == "Cliente")
                {
                <li class="nav-item">
                    <a class="nav-link text-dark" asp-area="" asp-controller="Publicaciones" asp-action="Listado">Publicaciones</a>
                </li>
                <li class="nav-item">
                    <a class="nav-link text-dark" asp-area="" asp-controller="Usuarios" asp-action="CargarSaldoBilletera">Billetera</a>
```

```
                </li>

                }

                <li class="nav-item">
                    <a class="nav-link text-dark" asp-area="" asp-controller="Usuarios" asp-action="Logout">Salir</a>
                </li>
                }
                else
                {
                <li class="nav-item">
                    <a class="nav-link text-dark" asp-area="" asp-controller="Usuarios" asp-action="Login">Login</a>
                </li>
                <li class="nav-item">
                    <a class="nav-link text-dark" asp-area="" asp-controller="Usuarios" asp-action="Registro">Registro</a>
                </li>
                }

            </ul>
        </div>
    </div>
</nav>
</header>
```

```
    <div class="container">
        <main role="main" class="pb-3">
            @RenderBody()
        </main>
    </div>

    <footer class="border-top footer text-muted">
        <div class="container">
            &copy; 2024 - Obligatoriop2 - <a asp-area="" asp-controller="Home" asp-action="Privacy">Privacy</a>
        </div>
    </footer>
    <script src="~/lib/jquery/dist/jquery.min.js"></script>
    <script src="~/lib/bootstrap/dist/js/bootstrap.bundle.min.js"></script>
    <script src="~/js/site.js" asp-append-version="true"></script>
    @await RenderSectionAsync("Scripts", required: false)
</body>
</html>
```

Administrador

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


namespace Dominio
{
   public class Administrador : Usuario, IValidable
   {


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


      }


      public override string ToString()
      {
         return $"{Nombre}"; // CORREGIR
      }
      public override void Validar()
      {
         base.Validar();
      }
```

```
        public override string Rol()
        {
            return "Admin";
        }
    }
}
```

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;
      public string _nombre { get; set; }
      private string _categoria;
      private decimal _precio;

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

```
        get { return _categoria; }


    }




    public decimal Precio
    {
        get { return _precio; }


    }


    public int Id
    {
        get { return _id; }


    }
    public void Validar()
    {
        if (string.IsNullOrEmpty(_nombre)) throw new ArgumentNullException("El nombre no puede estar vacio");
        if (string.IsNullOrEmpty(_categoria)) throw new ArgumentNullException("la categoria puede estar vacio");   //VALIDACIONES BASICAS.
        if (_precio < 0) throw new ArgumentOutOfRangeException("El precio no puede ser negativo");



    }
    public override string ToString()
```

```
    {
        return $"Nombre: {_nombre}, Categoría: {_categoria}, Precio: {_precio}";
    }
  }

}
```

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, IValidable
   {
      public decimal _saldo { get; set; }


      // Constructor que llama al constructor de la clase base (Usuario)
      public Cliente(string nombre, string apellido, string email, string contrasena,
decimal saldo)
       : base(nombre, apellido, email, contrasena)
      {
         _saldo = saldo;
      }


      // Constructor sin saldo (inicializa el saldo en 0)
      public Cliente(string nombre, string apellido, string email, string contrasena)
         : base(nombre, apellido, email, contrasena)
      {
         _saldo = 0;
```

```
    }

    public void AgregarSaldo(decimal nuevoSaldo)
    {
        if (nuevoSaldo < 0) throw new Exception("El saldo tiene que ser mayor que cero");
        _saldo += nuevoSaldo;
    }

    public decimal Saldo { get { return _saldo; } set { _saldo = value; } }

    public override string ToString()
    {
        return $"{Nombre}, Saldo: {_saldo}";
    }
    public override void Validar()
    {
        base.Validar();

    }

    public override string Rol()
    {
        return "Cliente";
```

```
    }
  }
}
```

Estado

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

namespace Dominio
{
   public enum Estado
   {
      ABIERTA = 1,
      CERRADA = 2,
      CANCELADA = 3

   }
}
```

Oferta

```
using Dominio.Interfaces;
using System;

namespace Dominio
{
   public class Oferta : IValidable, IComparable<Oferta>
   {
      private int _id;
      private static int s_ultId = 1;

      private Cliente _cliente;
      private decimal _monto;
      private DateTime _fechaOferta;

      public Oferta(Cliente cliente, decimal monto, DateTime fechaOferta)
      {
         _id = s_ultId;
         s_ultId++;
         _cliente = cliente;
         _monto = monto;
         _fechaOferta = fechaOferta;
      }

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

```
public Cliente Cliente
{
   get { return _cliente; }
}

public decimal Monto
{
   get { return _monto; }
}

public bool TieneCliente(Cliente cli)
{
   return _cliente.Equals(cli);
}

public void Validar()
{
   if (_cliente == null)
      throw new Exception("El cliente no puede ser nulo.");
   if (_monto < 0)
      throw new Exception("El monto debe ser mayor a 0.");
}
public string TipoPublicacion()
{
   return "Oferta";
}
public int CompareTo(Oferta other)
```

```
    {
        if (other == null) return 1;

        // Ordenar por monto descendente
        return other.Monto.CompareTo(this.Monto);
    }
  }
}
```

Publicacion

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

namespace Dominio
{
   public abstract class Publicacion : IValidable
   {
      private int _id;
      private static int s_ultId = 1;
      private string _nombre;
      private DateTime _fechaPublic;
      private DateTime _fechaFinaliz;
      private Usuario _usuarioFinaliza;
      private Cliente _usuarioComprador;
      private List<Articulo> _articulos = new List<Articulo>();
      private Estado _estado;

      public Publicacion( string name, DateTime fechaPublic, DateTime fechaFinaliz,
Cliente cliente, List<Articulo> articulos, Estado estado)
      {
         _nombre = name;
         _id = s_ultId;
         s_ultId++;
```

```
            _fechaPublic = fechaPublic;
            _fechaFinaliz = fechaFinaliz;
            _usuarioFinaliza = cliente;
            _articulos = articulos;
            _estado = estado;
        }


        public Estado Estado { get { return _estado; } set { _estado = value; } }


        public int Id
            { get { return _id; } }


        public Usuario UsuarioFinaliza { get { return _usuarioFinaliza; } set {
_usuarioFinaliza = value; } }


        //public decimal PrecioFinal
        //{
        //    get { return _precioFinal; }
        //    set { _precioFinal = value; }
        //}

        public string Nombre
        {
            get { return _nombre; }
```

```
    }

    public List<Articulo> Articulos
    { get { return _articulos; } }

    public DateTime FechaPublic
    {
       get { return _fechaPublic; }
    }

    public DateTime FechaFinaliz { get { return _fechaFinaliz; } set { _fechaFinaliz =
value; } }
    public virtual void Validar()
    {
       if (string.IsNullOrEmpty(_nombre))
          throw new ArgumentNullException("El nombre no puede ser vacio");

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



    }
    public void AgregarOferta( Oferta oferta)
    {


    }

    public abstract string tipoPublicacion();
```

```
        public abstract decimal CalcularPrecio();

        public override string ToString()

        {

          return $"Publicación: Id: {_id}Nombre: {_nombre},  Estado: {_estado}, 
Artículos: {Articulos.Count} Precio: {CalcularPrecio}";

        }

        public void AgregarArticulo(Articulo articulo)

        {


            //validarrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrr

            _articulos.Add(articulo);

        }




        //public int CompareTo(Subasta other)

        //{

        //    return this._fechaPublic.CompareTo(other._fechaPublic);

        //}



        public abstract void FinalizarPublicacion(Publicacion p, Usuario usuarioFinal, 
DateTime fechaFinalizacion);




    }

}
```

Sistema

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

namespace Dominio
{
  public class Sistema
  {

    private static Sistema s_instancia;
    //LISTAS GENERALES DE SISTEMA
    private List<Articulo> _articulos = new List<Articulo>();
    private List<Oferta> _ofertas = new List<Oferta>();
    private List<Usuario> _usuarios = new List<Usuario>();
    private List<Publicacion> _publicaciones = new List<Publicacion>();




    private Sistema()
    {
      PrecargarUsuarios();
      PrecargarArticulos();
      PrecargarPublicaciones();
```

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

public List<Usuario> Usuarios
{
   get { return _usuarios; }
}

public List<Articulo> Articulos
{
   get { return _articulos; }
}

public List<Publicacion> Publicaciones
{
   get { return _publicaciones; }
}
```

```
public Usuario ObtenerUsuarioPorId(int id)
{
    Usuario buscada = null;
    int i = 0;
    while (i < _usuarios.Count && buscada == null)
    {
        if (_usuarios[i].Id == id) buscada = _usuarios[i];
        i++;
    }

    return buscada;
}
public Articulo ObtenerArticuloPorId(int id)
{
    Articulo buscada = null;
    int i = 0;
    while (i < _articulos.Count && buscada == null)
    {
        if (_articulos[i].Id == id) buscada = _articulos[i];
        i++;
    }

    return buscada;
}
```

```
public Publicacion ObtenerPublicacionPorId(int id)
{
    Publicacion buscada = null;
    int i = 0;
    while (i < _publicaciones.Count && buscada == null)
    {
        if (_publicaciones[i].Id == id) buscada = _publicaciones[i];
        i++;
    }

    return buscada;
}



//precarga de articulos
private void PrecargarArticulos()
{
    AltaArticulo(new Articulo("Pelota futbol", "deporte", 100));
    AltaArticulo(new Articulo("Pelota futbol", "Deporte", 100));
    AltaArticulo(new Articulo("Raqueta de tenis", "Deporte", 150));
    AltaArticulo(new Articulo("Bicicleta", "Deporte", 300));
    AltaArticulo(new Articulo("Pesas 10kg", "Deporte", 80));
    AltaArticulo(new Articulo("Balón de básquet", "Deporte", 120));

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

```
        AltaArticulo(new Articulo("Laptop", "Tecnología", 1000));
        AltaArticulo(new Articulo("Tablet", "Tecnología", 300));
//precargas de articulos
        AltaArticulo(new Articulo("Auriculares inalámbricos", "Tecnología", 200));
        AltaArticulo(new Articulo("Monitor 24 pulgadas", "Tecnología", 250));

        AltaArticulo(new Articulo("Sofá", "Hogar", 700));
        AltaArticulo(new Articulo("Mesa de comedor", "Hogar", 400));
        AltaArticulo(new Articulo("Lámpara de pie", "Hogar", 100));
        AltaArticulo(new Articulo("Refrigerador", "Hogar", 1200));
        AltaArticulo(new Articulo("Microondas", "Hogar", 150));

        AltaArticulo(new Articulo("Muñeca", "Juguetería", 30));
        AltaArticulo(new Articulo("Coche de juguete", "Juguetería", 40));
        AltaArticulo(new Articulo("Rompecabezas 1000 piezas", "Juguetería", 25));
        AltaArticulo(new Articulo("Videojuego", "Juguetería", 60));
        AltaArticulo(new Articulo("Juguete de construcción", "Juguetería", 90));

        AltaArticulo(new Articulo("Reloj inteligente", "Tecnología", 300));
        AltaArticulo(new Articulo("Cámara fotográfica", "Tecnología", 700));
        AltaArticulo(new Articulo("Teclado mecánico", "Tecnología", 120));
        AltaArticulo(new Articulo("Mouse inalámbrico", "Tecnología", 80));
        AltaArticulo(new Articulo("Impresora", "Tecnología", 200));

        AltaArticulo(new Articulo("Cama", "Hogar", 500));
        AltaArticulo(new Articulo("Silla de oficina", "Hogar", 150));
        AltaArticulo(new Articulo("Aspiradora", "Hogar", 180));
        AltaArticulo(new Articulo("Cafetera", "Hogar", 90));
```

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

    AltaArticulo(new Articulo("Patinete", "Deporte", 100));
    AltaArticulo(new Articulo("Casco de bicicleta", "Deporte", 50));
    AltaArticulo(new Articulo("Zapatillas deportivas", "Deporte", 120));
    AltaArticulo(new Articulo("Cinta para correr", "Deporte", 800));
    AltaArticulo(new Articulo("Guantes de boxeo", "Deporte", 60));

    AltaArticulo(new Articulo("Drone", "Tecnología", 500));
    AltaArticulo(new Articulo("Consola de videojuegos", "Tecnología", 600));
    AltaArticulo(new Articulo("Smart TV", "Tecnología", 900));
    AltaArticulo(new Articulo("Cargador portátil", "Tecnología", 40));
    AltaArticulo(new Articulo("Router WiFi", "Tecnología", 120));

    AltaArticulo(new Articulo("Silla de comedor", "Hogar", 100));
    AltaArticulo(new Articulo("Ventilador", "Hogar", 60));
    AltaArticulo(new Articulo("Estufa", "Hogar", 200));
    AltaArticulo(new Articulo("Planchita de ropa", "Hogar", 70));
    AltaArticulo(new Articulo("Cortinas", "Hogar", 40));

    AltaArticulo(new Articulo("Juego de mesa", "Juguetería", 50));
    AltaArticulo(new Articulo("Pelota de playa", "Juguetería", 15));
    AltaArticulo(new Articulo("Peluche", "Juguetería", 25));
    AltaArticulo(new Articulo("Lego", "Juguetería", 100));
    AltaArticulo(new Articulo("Trompo", "Juguetería", 5));
}
```

```
    //precarga de usuarios(clientes y administradores)
    private void PrecargarUsuarios()
    {
        AltaUsuario(new Cliente("Marcio", "Pérez", "marcio@example.com",
"password123", 10000));
        AltaUsuario(new Cliente("Federico", "Cuello", "fede@example.com",
"fede123", 3000));
        AltaUsuario(new Cliente("Carlos", "Gómez", "carlos@example.com",
"passCarlos", 700));
        AltaUsuario(new Cliente("Sofia", "Rodríguez", "sofia@example.com",
"passSofia", 200));
        AltaUsuario(new Cliente("Diego", "Martínez", "diego@example.com",
"passDiego", 600)); //Precarga de usuarios
        AltaUsuario(new Cliente("Laura", "Fernández", "laura@example.com",
"passLaura", 400));
        AltaUsuario(new Cliente("Pablo", "Sánchez", "pablo@example.com",
"passPablo", 350));
        AltaUsuario(new Cliente("Lucía", "Hernández", "lucia@example.com",
"passLucia", 800));
        AltaUsuario(new Cliente("Javier", "Ramírez", "javier@example.com",
"passJavier", 150));
        AltaUsuario(new Cliente("Valentina", "Ruiz", "valentina@example.com",
"passValentina", 550));
        AltaUsuario(new Administrador("Marcio", "Huertas",
"marciohuertasrial1995@outlook.com", "marcio"));
        AltaUsuario(new Administrador("Fede", "Gallo", "fede@outlook.com",
"fede"));

    }
```

```
    public void PrecargarPublicaciones()
    {
        List<Articulo> listaArticulos1 = new List<Articulo> { _articulos[1], _articulos[2] };
        List<Articulo> listaArticulos4 = new List<Articulo> { _articulos[1], _articulos[2] };
        List<Articulo> listaArticulos5 = new List<Articulo> { _articulos[1], _articulos[2] };
        List<Articulo> listaArticulos6 = new List<Articulo> { _articulos[1], _articulos[2] };
        List<Articulo> listaArticulos7 = new List<Articulo> { _articulos[1], _articulos[2] };
        List<Articulo> listaArticulos8 = new List<Articulo> { _articulos[1], _articulos[2] };

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

        Cliente cliente1 = (Cliente)ObtenerUsuarioPorId(1);
        Administrador admin1 = (Administrador)ObtenerUsuarioPorId(12);

        if (cliente1 == null || admin1 == null)
            throw new Exception("Cliente o administrador no encontrado");

        // Precargar 8 publicaciones de venta
        AltaPublicacion(new Venta("Electrodomesticos", new DateTime(2024, 10, 12), new DateTime(2024, 12, 12), (Cliente)ObtenerUsuarioPorId(2), listaArticulos1, Estado.ABIERTA, false));
        AltaPublicacion(new Venta("Juegos recreativos", new DateTime(2024, 11, 01), new DateTime(2024, 12, 25), cliente1, listaArticulos4, Estado.ABIERTA, false));
```

AltaPublicacion(new Venta("Ropa antigua", new DateTime(2024, 09, 15), new DateTime(2024, 11, 30), (Cliente)ObtenerUsuarioPorId(3), listaArticulos8, Estado.ABIERTA, false));

AltaPublicacion(new Venta("Ludos retro", new DateTime(2024, 08, 20), new DateTime(2024, 10, 20), cliente1, listaArticulos1, Estado.ABIERTA, false));

AltaPublicacion(new Venta("Gamer pc y mas", new DateTime(2024, 12, 01), new DateTime(2025, 01, 01), (Cliente)ObtenerUsuarioPorId(4), listaArticulos5, Estado.ABIERTA, false));

AltaPublicacion(new Venta("Jueto de baño", new DateTime(2024, 07, 10), new DateTime(2024, 09, 10), cliente1, listaArticulos6, Estado.ABIERTA, false));

AltaPublicacion(new Venta("Juego de pesas", new DateTime(2024, 06, 05), new DateTime(2024, 08, 05), (Cliente)ObtenerUsuarioPorId(5), listaArticulos1, Estado.ABIERTA, false));

AltaPublicacion(new Venta("Instrumentos", new DateTime(2024, 05, 15), new DateTime(2024, 07, 15), cliente1, listaArticulos8, Estado.ABIERTA, false));

AltaPublicacion(new Venta("CD`s antiguos", new DateTime(2024, 05, 15), new DateTime(2024, 07, 15), cliente1, listaArticulos7, Estado.ABIERTA, false));

AltaPublicacion(new Venta("Kit basket", new DateTime(2024, 05, 15), new DateTime(2024, 07, 15), cliente1, listaArticulos5, Estado.ABIERTA, false));

// Precargar subasta con ofertas como ejemplo adicional

Subasta subasta1 = new Subasta("Kit Montaña", new DateTime(2024, 05, 10), new DateTime(2025, 01, 01), cliente1, listaArticulos1, Estado.ABIERTA, 0, admin1, listaOfertas);

AltaPublicacion(subasta1);

// Agregar ofertas a la primera subasta

AgregarOfertaAUnaSubasta(cliente1.Id, subasta1.Id, 500, new DateTime(2023, 05, 11));

AgregarOfertaAUnaSubasta(2, subasta1.Id, 600, new DateTime(2023, 06, 12));

// Precargar una nueva subasta con ofertas

```
        List<Articulo> listaArticulos2 = new List<Articulo> { _articulos[3], _articulos[4] };

        Subasta subasta2 = new Subasta("Pack Hogar", new DateTime(2024, 06, 01), new DateTime(2025, 02, 01), cliente1, listaArticulos2, Estado.ABIERTA, 0, admin1, new List<Oferta>());

        AltaPublicacion(subasta2);

        // Agregar ofertas a la nueva subasta

        AgregarOfertaAUnaSubasta(cliente1.Id, subasta2.Id, 800, new DateTime(2024, 06, 15));

        AgregarOfertaAUnaSubasta(3, subasta2.Id, 900, new DateTime(2024, 06, 20));

        AgregarOfertaAUnaSubasta(4, subasta2.Id, 1000, new DateTime(2024, 07, 01));

        List<Articulo> listaArticulos3 = new List<Articulo> { _articulos[3], _articulos[4] };

        // Creación de las subastas

        Subasta subasta3 = new Subasta("Juegos de mesa", new DateTime(2024, 05, 10), new DateTime(2025, 01, 01), cliente1, listaArticulos2, Estado.ABIERTA, 0, admin1, new List<Oferta>());

        AltaPublicacion(subasta3);

        Subasta subasta4 = new Subasta("Pack Acuatico", new DateTime(2024, 06, 01), new DateTime(2025, 02, 01), cliente1, listaArticulos2, Estado.ABIERTA, 0, admin1, new List<Oferta>());

        AltaPublicacion(subasta4);

        Subasta subasta5 = new Subasta("Colección de Arte", new DateTime(2024, 07, 01), new DateTime(2025, 03, 01), cliente1, listaArticulos2, Estado.ABIERTA, 0, admin1, new List<Oferta>());
```

```
        AltaPublicacion(subasta5);


        Subasta subasta6 = new Subasta("Instrumentos Musicales", new
DateTime(2024, 08, 01), new DateTime(2025, 04, 01), cliente1, listaArticulos2,
Estado.ABIERTA, 0, admin1, new List<Oferta>());

        AltaPublicacion(subasta6);


        Subasta subasta7 = new Subasta("Electrodomésticos", new DateTime(2024,
09, 01), new DateTime(2025, 05, 01), cliente1, listaArticulos2, Estado.ABIERTA, 0,
admin1, new List<Oferta>());

        AltaPublicacion(subasta7);


        Subasta subasta8 = new Subasta("Antigüedades", new DateTime(2024, 10,
01), new DateTime(2025, 06, 01), cliente1, listaArticulos2, Estado.ABIERTA, 0,
admin1, new List<Oferta>());

        AltaPublicacion(subasta8);


        Subasta subasta9 = new Subasta("Equipos de Camping", new
DateTime(2024, 11, 01), new DateTime(2025, 07, 01), cliente1, listaArticulos2,
Estado.ABIERTA, 0, admin1, new List<Oferta>());

        AltaPublicacion(subasta9);


        Subasta subasta10 = new Subasta("Colección de Libros", new
DateTime(2024, 12, 01), new DateTime(2025, 08, 01), cliente1, listaArticulos2,
Estado.ABIERTA, 0, admin1, new List<Oferta>());

        AltaPublicacion(subasta10);




    }
```

```
    public void AgregarOfertaAUnaSubasta(int idCliente, int idSubasta, decimal
monto, DateTime fecha)
    {
        Cliente clientebuscado = (Cliente)ObtenerUsuarioPorId(idCliente);
        if (clientebuscado == null)
            throw new Exception("El id del usuario no existe");

        Publicacion subastaBuscada = ObtenerPublicacionPorId(idSubasta);
        if (subastaBuscada == null)
            throw new Exception("El id de la subasta no existe");

        if (!(subastaBuscada is Subasta subasta))
            throw new Exception("La publicación no es una subasta");

        Oferta oferta = new Oferta(clientebuscado, monto, fecha);

        // Agregar la oferta a la subasta
        subasta.AgregarOferta(oferta);
    }
```

```
//Metodo que devuelve una lista de Clientes
public List<Cliente> ListarClientes()
{
   List<Cliente> clientesFiltrados = new List<Cliente>();

   foreach (Usuario usuario in _usuarios)
   {
      if (usuario is Cliente cliente)
      {
         clientesFiltrados.Add(cliente);
      }
   }
```

```
        return clientesFiltrados;
    }


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

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

    public void AltaVenta(Venta publicacion)
    {
        if (publicacion == null) throw new Exception("La publicacion no puede ser
nuka");
        publicacion.Validar();
        _publicaciones.Add(publicacion);
    }

    public void AltaPublicacion(Publicacion publicacion)
    {
```

```
        if (publicacion == null) throw new Exception("La publicacion no puede ser nula");
        publicacion.Validar();
        _publicaciones.Add(publicacion);
    }

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

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

        return articulosFiltrados;
```

```
    }

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

      foreach (Publicacion p in _publicaciones)
      {

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




      }



      return publicaciones;
    }

    //Metodo que agrega un articulo a una publicacion
    public void AgregarArticuloAPublicacion(int IdArticulo, int idPublicacion)
    {
      Articulo articuloBuscado = ObtenerArticuloPorId(IdArticulo);
      if (articuloBuscado == null) throw new Exception("El articulo no puedr ser
vacio");
      Publicacion publicacionBuscada = ObtenerPublicacionPorId(idPublicacion);
```

```
        if (publicacionBuscada == null) throw new Exception("La publicacion no
puede ser vacio");
        publicacionBuscada.AgregarArticulo(articuloBuscado);



    }









    //Login
    public Usuario Login(string email, string pass)
    {
        Usuario usuarioBuscado = null;
        int i = 0;
        while (usuarioBuscado == null && i < _usuarios.Count)
        {
            if (_usuarios[i].Email == email && _usuarios[i].Contrasena == pass)
usuarioBuscado = _usuarios[i];
            i++;
        }
```

```
        return usuarioBuscado;
    }



    //Lista Subastas
    public List<Subasta> ListadoSubastas()
    {
        // Lista para almacenar subastas encontradas
        List<Subasta> subastasEncontradas = new List<Subasta>();

        // Filtrar solo las publicaciones que sean del tipo Subasta
        foreach (var publicacion in _publicaciones)
        {
            if (publicacion is Subasta subasta)
            {
                subastasEncontradas.Add(subasta);
            }
        }

        // Ordenar por fecha de publicación (asume que Subasta tiene un atributo FechaPublicacion)
        subastasEncontradas.Sort();
        return subastasEncontradas;
    }
```

```
        //Metodo que carga la billetera del cliente
        public void CargarSaldoEnBilletera(int idCliente, decimal nuevoSaldo)
        {
            Usuario clienteBuscado = ObtenerUsuarioPorId(idCliente);

            if (clienteBuscado == null)
                throw new Exception("El cliente no se encontró");

            if (clienteBuscado is Cliente cliente)
            {
                cliente.AgregarSaldo(nuevoSaldo);
            }
            else
            {
                throw new Exception("El usuario encontrado no es un cliente");
            }
        }


    }
}
```

Subasta

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

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

      //Propiedades solo de lectura para acceder a los atributos
      public List<Oferta> Ofertas
      {
         get { return _ofertas; }
      }

      public decimal PrecioFinal
      {
         get { return _precioFinal; }
      }

      public Cliente Cliente
      {
         get { return _cliente; }
```

```
    }

    public Administrador Administrador
    {
        get { return _usuario; }
    }

    // Constructor de la clase Subasta
    public Subasta(string nombre, DateTime fechaPublicacion, DateTime
fechaFinalizacion, Cliente cliente, List<Articulo> articulos, Estado estado, decimal
precioFinal, Administrador admin, List<Oferta> ofertas)
    : base(nombre, fechaPublicacion, fechaFinalizacion, cliente, articulos, estado)
    {
        _precioFinal = precioFinal;
        _usuario = admin;
        _ofertas = ofertas ?? new List<Oferta>();
    }



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

        bool existeCliente = false;
        foreach (var ofe in _ofertas)
        {
            if (ofe.TieneCliente(nuevaOferta.Cliente))
```

```
            {
                existeCliente = true;
            }
            break;
        }
        if (!existeCliente)

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

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



        // Si la nueva oferta es válida, agregarla a la lista
        _ofertas.Add(nuevaOferta);
        _precioFinal = nuevaOferta.Monto;

    }
```

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


    // Método ToString para faciilitar la visualización
    public override string ToString()
    {
        return $"Subasta: {Nombre}, Precio Final: {_precioFinal}, Ofertas:
{Ofertas.Count}";
    }
    public override void Validar()
    {
        base.Validar();
    }


    public override decimal CalcularPrecio()
    {
        if (_ofertas == null || _ofertas.Count == 0) return 0;
        Oferta ultimaOferta = _ofertas[_ofertas.Count - 1];
        return ultimaOferta.Monto;
    }


    public override string tipoPublicacion()
    {
        return "Subasta";
    }
```

```
public decimal Precio { get; set; }


public int CompareTo(Subasta other)
{
   if (other == null) return 1;

   // Ordenar de mayor a menor precio
   return other.FechaPublic.CompareTo(this.FechaPublic);
}
public List<Oferta> ObtenerOfertas()
{
   // Crear una nueva lista para no modificar la lista original
   List<Oferta> ofertasOrdenadas = new List<Oferta>(_ofertas);

   // Insertion Sort: ordenar las ofertas en orden descendente por monto
   for (int i = 1; i < ofertasOrdenadas.Count; i++)
   {
      Oferta ofertaActual = ofertasOrdenadas[i];
      int j = i - 1;

      // Desplazar las ofertas que son menores que la oferta actual
      while (j >= 0 && ofertasOrdenadas[j].Monto < ofertaActual.Monto)
      {
         ofertasOrdenadas[j + 1] = ofertasOrdenadas[j];
         j--;
      }

      // Insertar la oferta actual en su posición ordenada
```

```
            ofertasOrdenadas[j + 1] = ofertaActual;
        }

        return ofertasOrdenadas;
    }


    public override void FinalizarPublicacion(Publicacion p, Usuario usuarioFinal,
DateTime fechaFinalizacion)
    {
        // Verificar que la publicación sea una subasta
        Subasta s = p as Subasta;
        if (s == null)
        {
            throw new Exception("La publicación no es una subasta.");
        }

        // Obtener las ofertas ordenadas
        List<Oferta> ofertasOrdenadas = s.ObtenerOfertas();

        // Buscar un comprador con saldo suficiente
        Cliente comprador = null;
        decimal precioFinal = 0;

        // Iterar sobre las ofertas y encontrar un comprador con saldo suficiente
        int i = 0;
        while (i < ofertasOrdenadas.Count && comprador == null)
        {
```

```
    Oferta oferta = ofertasOrdenadas[i];
    Usuario usuarioOferta = oferta.Cliente;
    Cliente cliente = usuarioOferta as Cliente;

    if (cliente != null && cliente.Saldo >= oferta.Monto)
    {
        comprador = cliente;
        precioFinal = oferta.Monto;
    }
    else
    {
        throw new Exception("Saldo insuficente ");
    }
    i++;
}
if (comprador.Saldo < precioFinal)
{
    throw new Exception("Saldo insuficiente para realziar la compra");
}
// Si no se encontró un comprador válido
if (comprador == null)
{
    p.Estado = Estado.CANCELADA;

    if (ofertasOrdenadas.Count == 0)
    {
        throw new Exception("No hay compradores en la subasta.");
    }
```

```
            else
            {
                throw new Exception("No se encontró un comprador  con saldo
suficiente.");
            }



            {
                throw new Exception("Saldo insuficiente para realziar la compra");
            }
        }

        // Realizarla transacción
        comprador.Saldo -= precioFinal;
        p.Estado = Estado.CERRADA;
        p.UsuarioFinaliza = usuarioFinal;
        p.FechaFinaliz = fechaFinalizacion;
    }


  }
}
```

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
   {
      private int id;
      private static int s_ultId = 1;
      private string _nombre;
      private string _apellido;
      private string _email;
      private string _contrasena;



      protected Usuario( string nombre, string apellido, string email, string
contrasena)
      {
         id = s_ultId;
         s_ultId++;
         _nombre = nombre;
         _apellido = apellido;
         _email = email;
         _contrasena = contrasena;
      }
```

```
public string Nombre
{
   get { return _nombre; }
   set { _nombre = value; }
}


public string Apellido
{
   get { return _apellido; }
   set { _apellido = value; }
}

public string Email
{
   get { return _email; }
   set { _email = value; }
}

public string Contrasena
{
   get { return _contrasena; }
   set { _contrasena = value; }
}
```

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









        public virtual void Validar()
        {
            if (string.IsNullOrEmpty(_nombre)) throw new ArgumentNullException("El nombre no puede ser vacio");
            if (string.IsNullOrEmpty(_apellido)) throw new ArgumentNullException("El apellido no puede ser vacio");
            if (string.IsNullOrEmpty(Contrasena)) throw new Exception("La contraseña no puede estar vacia");

            if (string.IsNullOrEmpty(_email)) throw new ArgumentNullException("El emial no puede ser vacio");
        }

        public abstract string Rol();
    }
}
```

Venta

```
using Dominio.Interfaces;
using System;
```

```
using System.Collections.Generic;

namespace Dominio
{
    public class Venta : Publicacion, IValidable
    {
        private bool _ofertaRelampago;
        //private decimal _precioFinal;
        private Usuario _comprador;

        // Atributos específicos de la clase Venta
        public bool OfertaRelampago
        {
            get { return _ofertaRelampago; }
        }

        //public decimal PrecioFinal
        //{
        //    get { return _precioFinal; }
        //    set { _precioFinal = value; }
        //}

        public Venta(string nombre, DateTime fechaPublicacion, DateTime fechaFinalizacion, Cliente cliente, List<Articulo> articulos, Estado estado, bool ofertaRelampago)
            : base(nombre, fechaPublicacion, fechaFinalizacion, cliente, articulos, estado) // Llama al constructor de Publicacion
        {
            _ofertaRelampago = ofertaRelampago;
```

```
        //_precioFinal = precioFinal;
    }

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



    public override decimal CalcularPrecio()
    {
        decimal precioFinal = 0;

        foreach (Articulo articulo in Articulos)
        {
            precioFinal += articulo.Precio;
        }

        // Aplicar el 20% de descuento si es ofertrelampago
        if (OfertaRelampago)
        {
            precioFinal *= 0.8m; // Multiplica por 0.8 para aplicar el descuento
        }



        return precioFinal;
```

```
    }



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

    public override string tipoPublicacion()
    {
        return "Venta";
    }

    public override void FinalizarPublicacion(Publicacion p, Usuario usuarioFinal,
DateTime fechaFinalizacion)



    {
        if(p.Estado != Estado.ABIERTA)
        {
            throw new Exception("L publicacion esta cerrada");

        }

        decimal precio = p.CalcularPrecio();
```

```
        Cliente cliente  = usuarioFinal as Cliente;


        if (cliente == null)
        {
            throw new Exception ("El usuario no existe en el sistema");
        }

        if (cliente.Saldo < precio)
        {
            throw new Exception("Saldo insuficiente para realziar la compra");
        }

        cliente.Saldo -= precio;
        p.Estado = Estado.CERRADA;
        p.UsuarioFinaliza = usuarioFinal;
        p.FechaFinaliz = DateTime.Now;
    }
  }



  }
```

PublicacionesController.cs

Vistas de PublicacionesController

DetallePublicacion.cshtml

```
@using Dominio;
@{
    ViewData["Title"] = "DetallePublicacion";
}

<h1>Detalle Publicación</h1>
<br />
<br />
@if (ViewBag.detallePublicacion == null)
{
    <div class="alert alert-danger">Publicación inexistente</div>
}
else
{
    Publicacion p = ViewBag.detallePublicacion;
    <p><strong>Id: @p.Id</strong></p>
    <P><strong>Nombre: @p.Nombre</strong></P>
    <P><strong>Monto: @p.CalcularPrecio()</strong></P>
    <br />
    <tbody>
        @foreach (Articulo a in p.Articulos)
        {
            <tr>
                <td><p><strong>Articulos: @a._nombre</strong></p></td>
```

```
        </tr>
    }
    </tbody>
    <br />
    <br />
    @if (ViewBag.detallePublicacion is Venta venta)
    {
        <form>
            <label>Haga click en el botón para comprar</label><br /><br />
            <input class="btn btn-success" type="submit" value="Comprar" />
        </form>
    }
    else if (ViewBag.detallePublicacion is Subasta subasta)
    {
        <form method="post" action="/publicaciones/OfertaEnSubasta">
            <input type="text" value="@ViewBag.detallePublicacion.Id" name="id"
hidden />
            <label>Monto</label>
            <br />
            <br />
            <input type="number" placeholder="Ingrese monto" name="monto" />
            <br />
            <br />
            <input class="btn btn-success" type="submit" value="Subastar" />
            <p><strong>Ingrese un precio mayor al último ingresado si va a
subastar</strong></p>

        </form>
    }
```

}

Listado.cshtml

```
@using Dominio;

@{
    ViewData["Title"] = "Listado";
}

<h1>Listado de Publicaciones</h1>

<br />

<!-- Mostrar mensaje de éxito o error -->
@if (TempData["Mensaje"] != null)
{
    <div class="alert alert-info" role="alert">
        @TempData["Mensaje"]
    </div>
}

<br />

@if (ViewBag.Listado == null || ViewBag.Listado.Count == 0)
```

```
{
   <div class="alert alert-danger">No hay agencias en el sistema</div>
}
else
{
   <table class="table table-striped">
      <thead>
         <tr>
            <th>Nombre</th>
            <th>Estado</th>
            <th>Precio</th>
            <th>Acciones</th>
         </tr>
      </thead>
      <tbody>
         @foreach (Publicacion p in ViewBag.Listado)
         {
            <tr>
               <td>@p.Nombre</td>
               <td>@p.Estado</td>
               <td>@p.CalcularPrecio()</td>
               <td>
                  @if (p.Estado == Estado.ABIERTA)
                  {
                     @if (p is Venta)
                     {
                        <!-- Botón para finalizar publicación (Comprar) -->
```

```
                <form method="post" action="/publicaciones/FinalizarPublicacion">
                    <input type="hidden" value="@p.Id" name="id" />
                    <button type="submit" class="btn btn-success">Comprar</button>
                </form>
            }
            else if (p is Subasta)
            {
                <!-- Botón para ofertar en subasta -->
            <td><a href="/publicaciones/detallePublicacion?Id=@p.Id">Detalle</a></td>
            }
        }
        </td>
    </tr>
}
</tbody>
</table>
}
```

ListadoSubastas.cshtml

```
@using Dominio;

@{
    ViewData["Title"] = "Listado de Subastas";
}
```

```
<h1>Listado de Subastas</h1>

@if (ViewBag.Exito != null)
{
    <div class="alert alert-success">@ViewBag.Exito</div>
}

@if (ViewBag.Subastas == null || ViewBag.Subastas.Count == 0)
{
    <div class="alert alert-danger">No hay subastas en el sistema.</div>
}
else
{
    <table class="table table-striped">
        <thead>
            <tr>
                <th>Nombre</th>
                <th>Fecha</th>
                <th>Estado</th>
                <th>Precio</th>
                <th>Acciones</th>
            </tr>
        </thead>
        <tbody>
            @foreach (Publicacion subasta in ViewBag.Subastas)
            {
                <tr>
```

```
            <td>@subasta.Nombre</td>
            <td>@subasta.FechaPublic.ToShortDateString()</td>
            <td>@subasta.Estado</td>
            <td>@subasta.CalcularPrecio()</td>
            <td>
                @if (subasta.Estado == Estado.ABIERTA)
                {
                    <form method="post" action="/Publicaciones/CerrarSubasta">
                        <input type="hidden" name="id" value="@subasta.Id" />
                        <button type="submit" class="btn btn-danger">Cerrar Subasta</button>
                    </form>
                }
            </td>
        </tr>
    }
    </tbody>
  </table>
}
```

UsuariosController.cs

Registro.cshtml

```
@using Dominio

@{
   ViewData["Title"] = "Registro";
}

<h1>Registro</h1>

@if (ViewBag.Error != null)
{
   <div class="alert alert-danger">@ViewBag.Error</div>
}

@if (ViewBag.Exito != null)
{
   <div class="alert alert-success">@ViewBag.Exito</div>
}

<form method="post" action="/TuControlador/Registro">
   <div class="form-group">
      <label for="nombre">Nombre</label>
      <input type="text" class="form-control" id="nombre" name="nombre"
value="@ViewBag.Nombre" required>
   </div>
   <div class="form-group">
```

```
      <label for="apellido">Apellido</label>
      <input type="text" class="form-control" id="apellido" name="apellido"
value="@ViewBag.Apellido" required>
    </div>
    <div class="form-group">
      <label for="email">Email</label>
      <input type="email" class="form-control" id="email" name="email"
value="@ViewBag.Email" required>
    </div>
    <div class="form-group">
      <label for="contrasena">Contraseña</label>
      <input type="password" class="form-control" id="contrasena"
name="contrasena" required>
    </div>
    <button type="submit" class="btn btn-primary">Registrar</button>
</form>
```

Login.cshtml

```
@{
    ViewData["Title"] = "Login";
}

<h1>Login</h1>
<br />

@if (ViewBag.Exito != null)
{
    <div class="alert alert-success">@ViewBag.Exito</div>
```

```
}

@if (ViewBag.Error != null)
{
   <div class="alert alert-danger">@ViewBag.Error</div>
}

<br />
<form method="post" action="/usuarios/login">
   <label>Email</label>
   <input type="email" placeholder="Ingrese usuario" name="email" />

   <br />
   <br />

   <label>Contraseña</label>
   <input type="password" placeholder="Ingrese contraseña" name="pass" />

   <br />
   <br />

   <input type="submit" class="btn btn-primary" value="Login" />
</form>
```

CargarSaldoBilletera.cshtml

```
@using Dominio;
@{
   ViewData["Title"] = "CambiarSaldoBilletera";
```

```
}

<h1>Cargar Saldo en Billetera</h1>

<br />
@if (ViewBag.Error != null)

{
  <div class="alert alert-danger">@ViewBag.Error</div>
}
@if (ViewBag.Exito != null)

{
  <div class="alert alert-success">@ViewBag.Exito</div>
}
<br />
<form method="post" action="/usuarios/CargarSaldoBilletera">
  <div class="form-group">
  </div>
  <br />
  <div class="form-group">
    <label class="form-label">Ingrese Monto</label>
    <input type="number" class="form-control" placeholder="Ingrese un monto" name="nuevoSaldo" />
  </div>
  <br />
  <div>
    <input class="btn btn-primary" type="submit" value="Aceptar" />
```

```
    </div>

</form>
```

Link de acceso Azure:

Link a aplicación

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

Precargas de Ventas, Subastas, Administradores, Clientes

Csharp Copiar

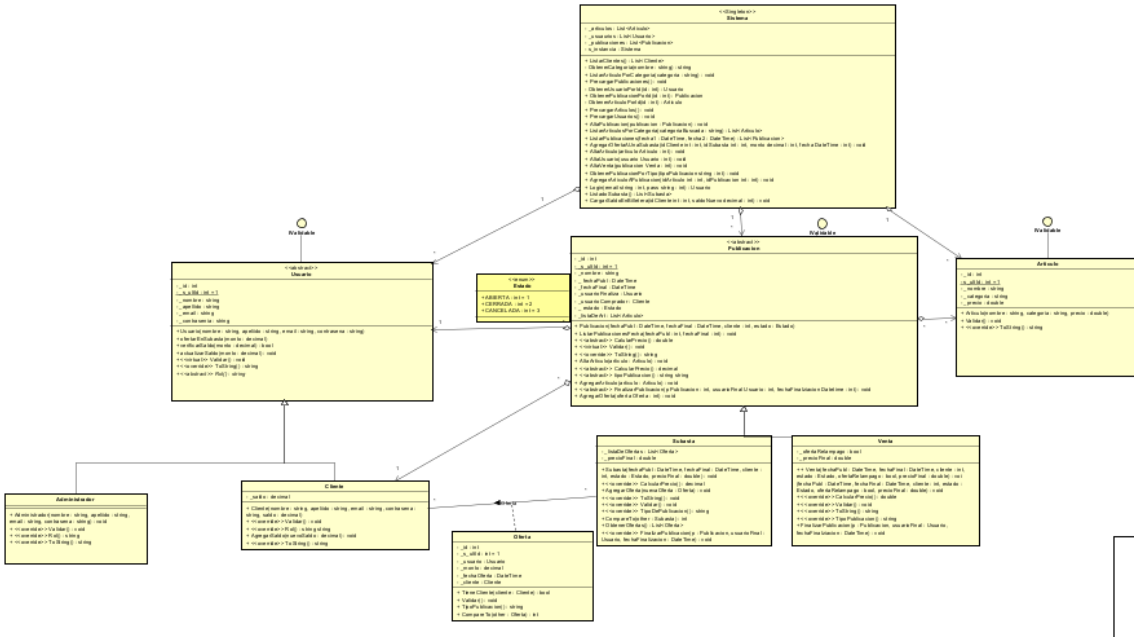
```
{
    AltaArticulo(new Articulo("Pelota futbol", "deporte", 100));
    AltaArticulo(new Articulo("Pelota futbol", "Deporte", 100));
    AltaArticulo(new Articulo("Raqueta de tenis", "Deporte", 150));
    AltaArticulo(new Articulo("Bicicleta", "Deporte", 300));
    AltaArticulo(new Articulo("Pesas 10kg", "Deporte", 80));
    AltaArticulo(new Articulo("Balón de básquet", "Deporte", 120));

    AltaArticulo(new Articulo("Teléfono móvil", "Tecnología", 500));
    AltaArticulo(new Articulo("Laptop", "Tecnología", 1000));
    AltaArticulo(new Articulo("Tablet", "Tecnología", 300));
    AltaArticulo(new Articulo("Auriculares inalámbricos", "Tecnología", 200
    AltaArticulo(new Articulo("Monitor 24 pulgadas", "Tecnología", 250));

    AltaArticulo(new Articulo("Sofá", "Hogar", 700));
    AltaArticulo(new Articulo("Mesa de comedor", "Hogar", 400));
    AltaArticulo(new Articulo("Lámpara de pie", "Hogar", 100));
    AltaArticulo(new Articulo("Refrigerador", "Hogar", 1200));
    AltaArticulo(new Articulo("Microondas", "Hogar", 150));
```

Csharp Copiar

```
//precarga de usuarios(clientes y administradores)
private void PrecargarUsuarios()
{
    AltaUsuario(new Cliente("Marcio", "Pérez", "marcio@example.com", "passw
    AltaUsuario(new Cliente("Federico", "Cuello", "fede@example.com", "fede
    AltaUsuario(new Cliente("Carlos", "Gómez", "carlos@example.com", "passC
    AltaUsuario(new Cliente("Sofia", "Rodríguez", "sofia@example.com", "pas
    AltaUsuario(new Cliente("Diego", "Martínez", "diego@example.com", "pass
    AltaUsuario(new Cliente("Laura", "Fernández", "laura@example.com", "pas
    AltaUsuario(new Cliente("Pablo", "Sánchez", "pablo@example.com", "passP
    AltaUsuario(new Cliente("Lucía", "Hernández", "lucia@example.com", "pas
    AltaUsuario(new Cliente("Javier", "Ramírez", "javier@example.com", "pas
    AltaUsuario(new Cliente("Valentina", "Ruiz", "valentina@example.com", "
    AltaUsuario(new Administrador("Marcio", "Huertas", "marciohuertasrial19
    AltaUsuario(new Administrador("Fede", "Gallo", "fede@outlook.com", "fed
}
```

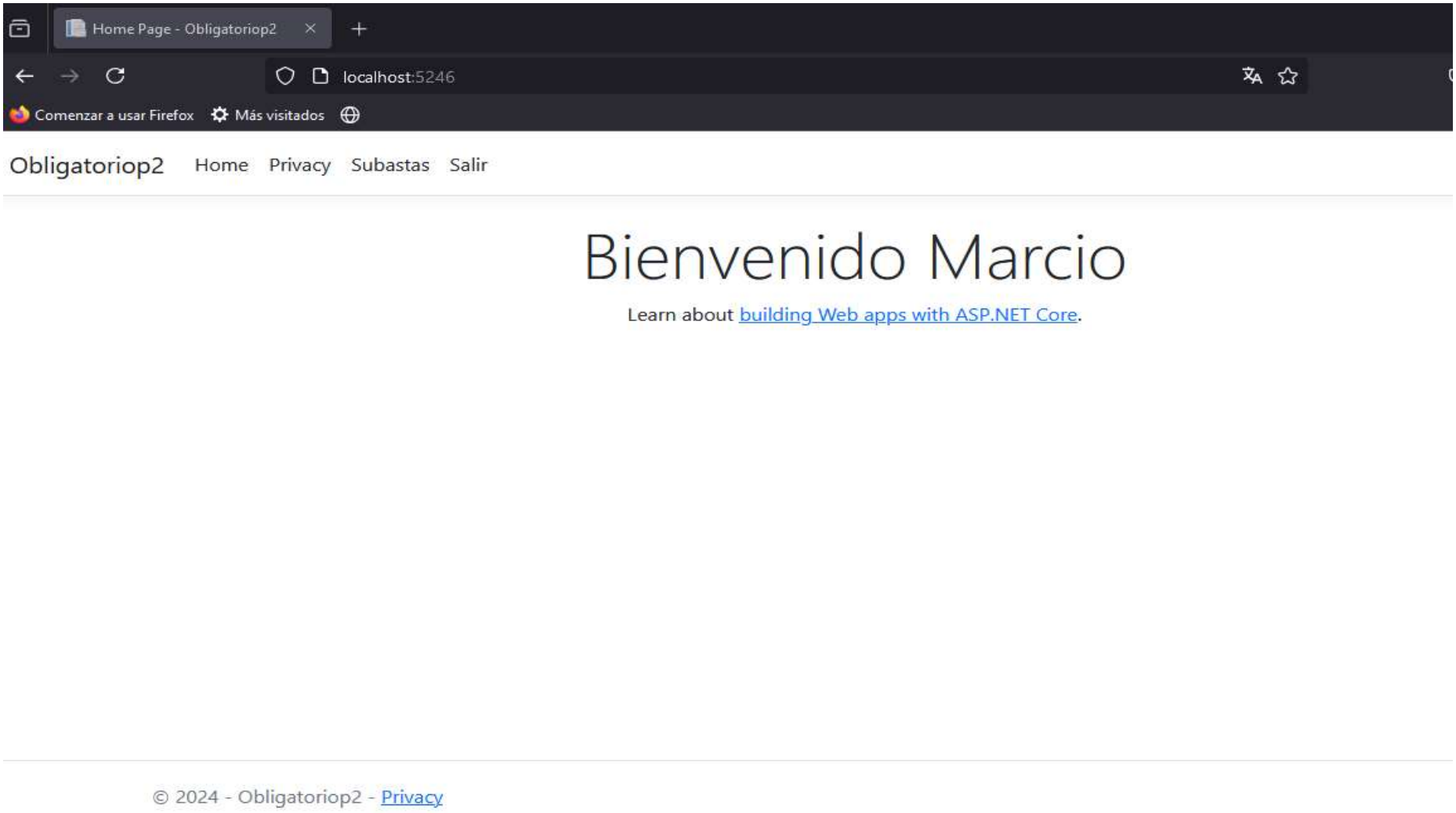
```
// Creación de las subastas
Subasta subasta3 = new Subasta("Juegos de mesa", new DateTime(2024, 05, 10)
AltaPublicacion(subasta3);

Subasta subasta4 = new Subasta("Pack Acuatico", new DateTime(2024, 06, 01),
AltaPublicacion(subasta4);

Subasta subasta5 = new Subasta("Colección de Arte", new DateTime(2024, 07, (
AltaPublicacion(subasta5);

Subasta subasta6 = new Subasta("Instrumentos Musicales", new DateTime(2024,
AltaPublicacion(subasta6);

Subasta subasta7 = new Subasta("Electrodomésticos", new DateTime(2024, 09, (
AltaPublicacion(subasta7);

Subasta subasta8 = new Subasta("Antigüedades", new DateTime(2024, 10, 01), (
AltaPublicacion(subasta8);

Subasta subasta9 = new Subasta("Equipos de Camping", new DateTime(2024, 11,
AltaPublicacion(subasta9);
```

Diagrama con clases de dominio

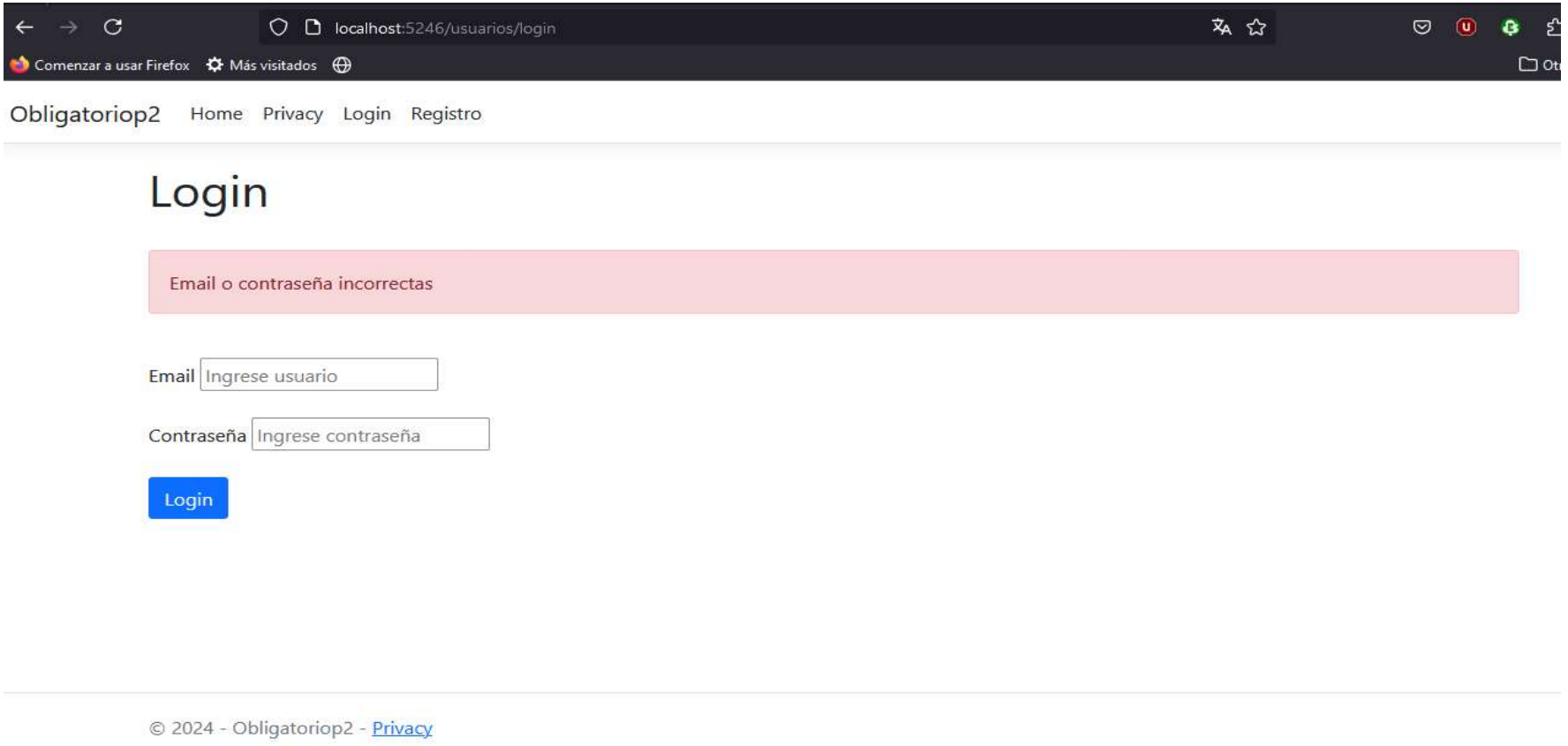


Funcionalidad	Caso de prueba	Pasos	Datos	Resultado esperado	Resultado obtenido	Captura
Acceder al sistema (Login)	Login de Administrador	1) Ir al link de login en el header y hacer click login 2) Ingresar usuario y contraseña validos 3) Hacer click en "Acceder"	usuario : marciohuertasrial1995@outlook.com contraseña : "marcio"	El usuario ingresa al sistema y en el body puede ver: 1) Mail con el que se logeo En el Header puede ver: 1) Lista de Clientes 2) Listado de Subastas 3) Logout	PASS	**Ir a Hoja Captura 1**
	Login fallido de administrador	1) Navegar a la pantalla de login 2) Ingresar usuario y contraseña invalidos 3) Hacer click en "Login"	usuario : marciohuertasrial1995@outlook.com contraseña : "marsio" (ver error en el ingreso de la contresaña)	El usuario es redirigido a la misma pantalla de Login y se despliega una alerta en rojo con un error con el texto "Usuario o contraseña incorrectas"	PASS	**Hoja Captura 2**
Filtrar búsquedas	Filtro por publicaciones	1) Ir al link de login en el header y hacer click login 2) Ingresar usuario y contraseña validos 3) Hacer click en "Acceder" 4) En el headder hacer click en Publicaciones 5) En la nueva pantalla ingresar al input de filtro 6)Escribir "Venta" o "subasta" 7)Click en Buscar	usuario : "fede@example.com" contraseña : "fede123"	El usuario una vez que hace click en buscar va a poder ver la lista filtrada de venta o de subasta, es una búsqieda de acuerdo a las pubolicaciones que hay	PASS	**Hoja Captura 3**
	Filtro por publicaciones vacía o mal escitras	1) Ir al link de login en el header y hacer click login 2) Ingresar usuario y contraseña validos 3) Hacer click en "Acceder" 4) En el headder hacer click en Publicaciones 5) En la nueva pantalla ingresar al input de filtro 6)Escribir "Benta" o "Zubasta" 7)Click en Buscar	usuario : "fede@example.com" contraseña : "fede123"	Al Cliente se le va a desplegar una alerta en la cual dice: "Debe ingresar un tipo de publicacion correcta o que no esté vacía"	PASS	**Hoja Captura 4**
Comprar	Hacer una compra	1) Ir al link de login en el header y hacer click login 2) Ingresar usuario y contraseña validos 3) Hacer click en "Acceder" 4) En el headder hacer click en Publicaciones 5) En la nueva pantalla ingresar al input de filtro 6)Escribir "Venta" 7)Click en Buscar 8)Seleccionar una venta y click en comprar	usuario : "fede@example.com" contraseña : "fede123"	El cliente una vez que haya hecho la compra va a saltar un cartel que dice: "La publicación "x" fue finalzada exitosamente"	PASS	**Hoja Captura 5**
Cargar Billetera	Cargar saldo en la billetera de un ususario	1) Ir al link de login en el header y hacer click login 2) Ingresar usuario y contraseña validos 3) Hacer click en "Acceder" 4) En el headder hacer click en "Billetera" 5) Ingresar monto in input 6)click en "Aceptar"	usuario : "fede@example.com" contraseña : "fede123"	Luego de clickear en aceptar nos va a saltar un cartel quedice: Se cambio el sando de cliente X, acaba de agregr a su saldo x monto.	PASS	**Hoja Captura 6**
Subastas	Hacer una Subasta	1) Ir al link de login en el header y hacer click login 2) Ingresar usuario y contraseña validos 3) Hacer click en "Acceder" 4) En el headder hacer click en Publicaciones 5) En la nueva pantalla ingresar al input de filtro 6)Escribir "Subasta" 7)Click en Buscar 8)Seleccionar una subasta haciendo click en detalle 9) Se nos va a abrir una nueva ventana en donde en el input tenemos que ingresar el monto superior al precio que vimos en la publicación	usuario : "fede@example.com" contraseña : "fede123"	Luego de ingresar un monto superior nos va a dirigir a la pantalla de publicaciones y vamos a ver el monto nuevo ingresado	PASS	**Hoja Captura 7**

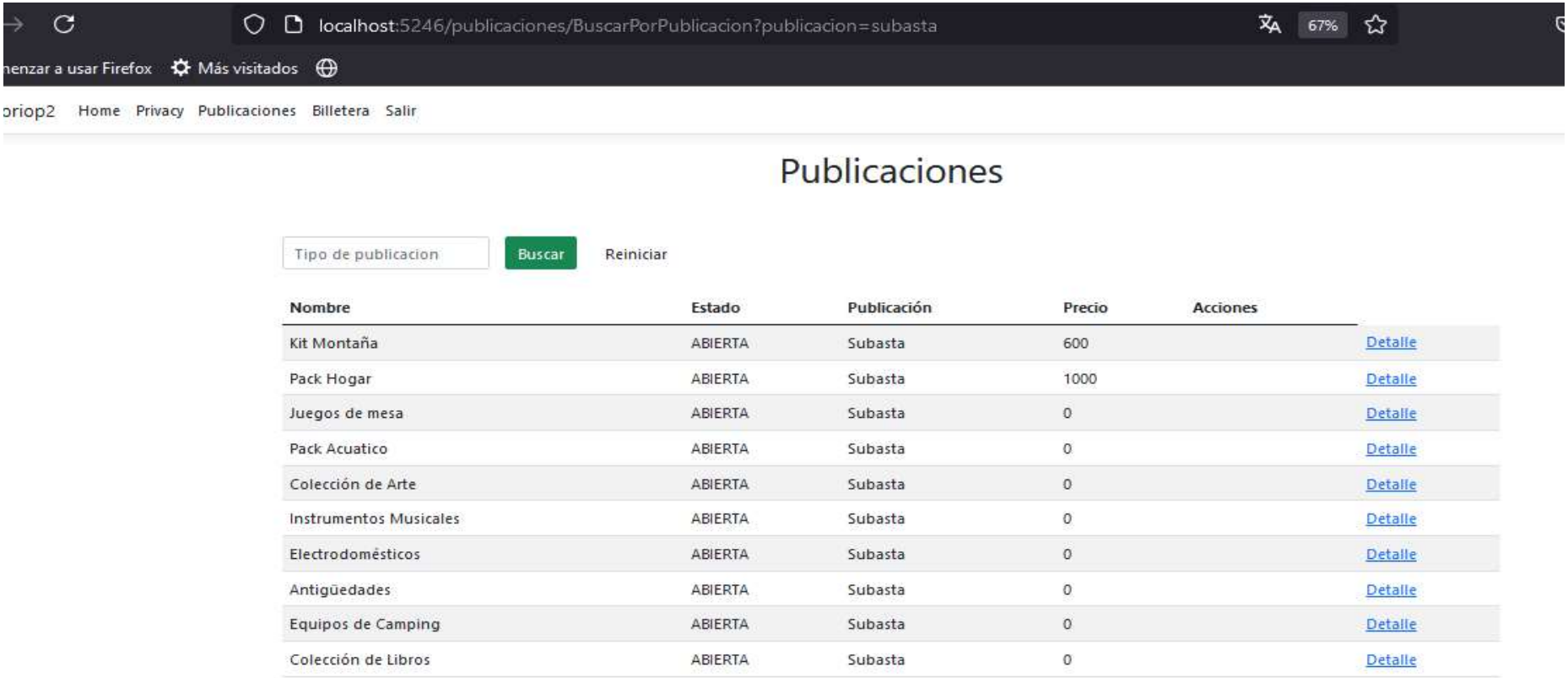
Captura 1


Home Page - Obligatoriop2
localhost:5246
Comenzar a usar Firefox
Más visitados
Obligatoriop2
Home
Privacy
Subastas
Salir
Bienvenido Marcio
Learn about building Web apps with ASP.NET Core.
© 2024 - Obligatoriop2 - Privacy

Captura 2


localhost:5246/usuarios/login
Comenzar a usar Firefox
Más visitados
Obligatoriop2
Home
Privacy
Login
Registro
Login
Email o contraseña incorrectas
Email
Ingrese usuario
Contraseña
Ingrese contraseña
Login
© 2024 - Obligatoriop2 - Privacy

Captura 3


localhost:5246/publicaciones/BuscarPorPublicacion?publicacion=subasta
67%
nenzar a usar Firefox
Más visitados
oriop2
Home
Privacy
Publicaciones
Billetera
Salir
Publicaciones
Tipo de publicacion
Buscar
Reiniciar
Nombre
Estado
Publicación
Precio
Acciones
Kit Montaña ABIERTA Subasta 600 Detalle
Pack Hogar ABIERTA Subasta 1000 Detalle
Juegos de mesa ABIERTA Subasta 0 Detalle
Pack Acuatico ABIERTA Subasta 0 Detalle
Colección de Arte ABIERTA Subasta 0 Detalle
Instrumentos Musicales ABIERTA Subasta 0 Detalle
Electrodomésticos ABIERTA Subasta 0 Detalle
Antigüedades ABIERTA Subasta 0 Detalle
Equipos de Camping ABIERTA Subasta 0 Detalle
Colección de Libros ABIERTA Subasta 0 Detalle

Captura 3


localhost:5246/publicaciones/BuscarPorPublicacion?publicacion=ventas
67%
ar a usar Firefox
Más visitados
p2
Home
Privacy
Publicaciones
Billetera
Salir
Publicaciones
ventas
Buscar
Reiniciar
ventas
No hay publicaciones de ese tipo en el sistema

Captura 5

Publicaciones

La publicación 'Electrodomesticos' fue finalizada exitosamente.

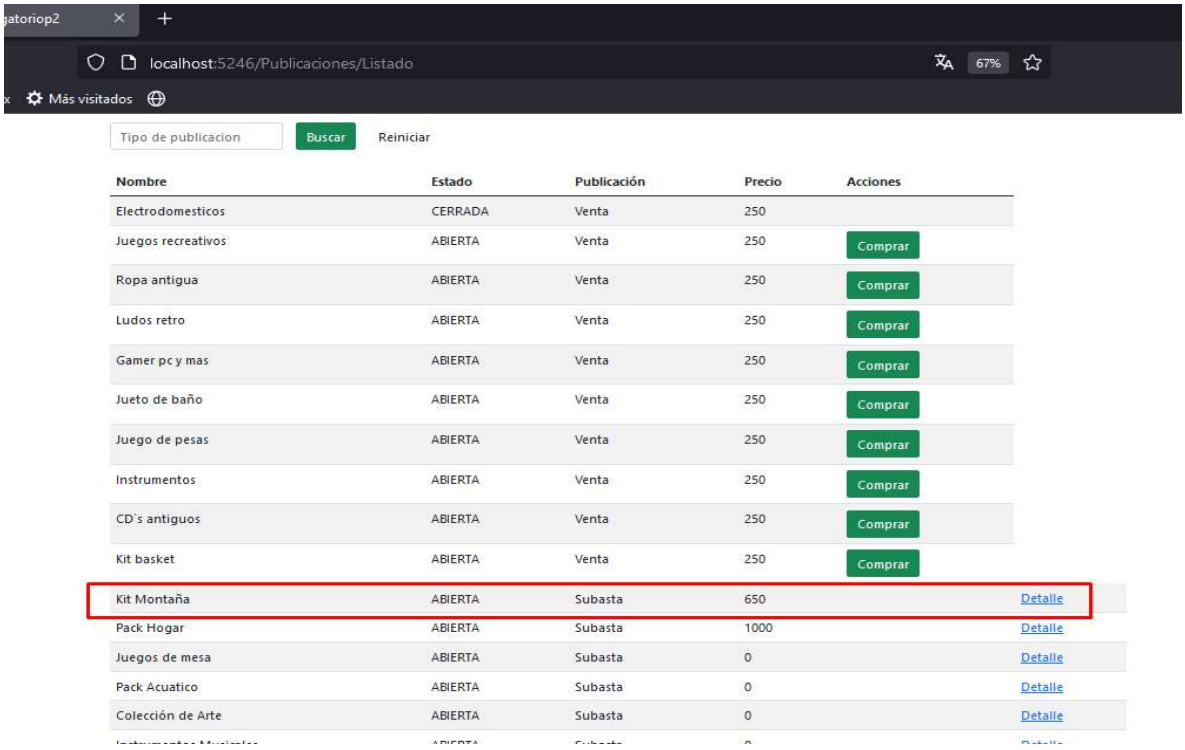
Tipo de publicacion | Buscar | Reiniciar

Nombre	Estado	Publicación	Precio	Acciones
Electrodomesticos	CERRADA	Venta	250	
Juegos recreativos	ABIERTA	Venta	250	Comprar
Ropa antigua	ABIERTA	Venta	250	Comprar
Ludos retro	ABIERTA	Venta	250	Comprar
Gamer pc y mas	ABIERTA	Venta	250	Comprar
Jueto de baño	ABIERTA	Venta	250	Comprar
Juego de pesas	ABIERTA	Venta	250	Comprar
Instrumentos	ABIERTA	Venta	250	Comprar
CD`s antiguos	ABIERTA	Venta	250	Comprar
Kit basket	ABIERTA	Venta	250	Comprar

Captura 6


localhost:5246/usuarios/CargarSaldoBilletera
67%
Más visitados
Privacy
Publicaciones
Billetera
Salir
Cargar Saldo en Billetera
Se cambio el saldo de Federico, acaba de agregar a su saldo 5000
Ingrese Monto
Ingrese un monto
Aceptar

Captura 7


gatoriop2
localhost:5246/Publicaciones/Listado
67%
Más visitados
Tipo de publicacion
Buscar
Reiniciar
Nombre
Estado
Publicación
Precio
Acciones
Electrodomesticos
CERRADA
Venta
250
Juegos recreativos
ABIERTA
Venta
250
Comprar
Ropa antigua
ABIERTA
Venta
250
Comprar
Ludos retro
ABIERTA
Venta
250
Comprar
Gamer pc y mas
ABIERTA
Venta
250
Comprar
Jueto de baño
ABIERTA
Venta
250
Comprar
Juego de pesas
ABIERTA
Venta
250
Comprar
Instrumentos
ABIERTA
Venta
250
Comprar
CD's antiguos
ABIERTA
Venta
250
Comprar
Kit basket
ABIERTA
Venta
250
Comprar
Kit Montaña
ABIERTA
Subasta
650
Detalle
Pack Hogar
ABIERTA
Subasta
1000
Detalle
Juegos de mesa
ABIERTA
Subasta
0
Detalle
Pack Acuatico
ABIERTA
Subasta
0
Detalle
Colección de Arte
ABIERTA
Subasta
0
Detalle