Universidad ORT Uruguay
Facultad de Ingeniería
Escuela de Tecnología
OBLIGATORIO PROGRAMACIÓN 2, parte2

Mauricio Varela

271171

Matias Alvarez

286704

Docente: Luis Dentone Analista programador / Analista en tecnologías de la información

UML

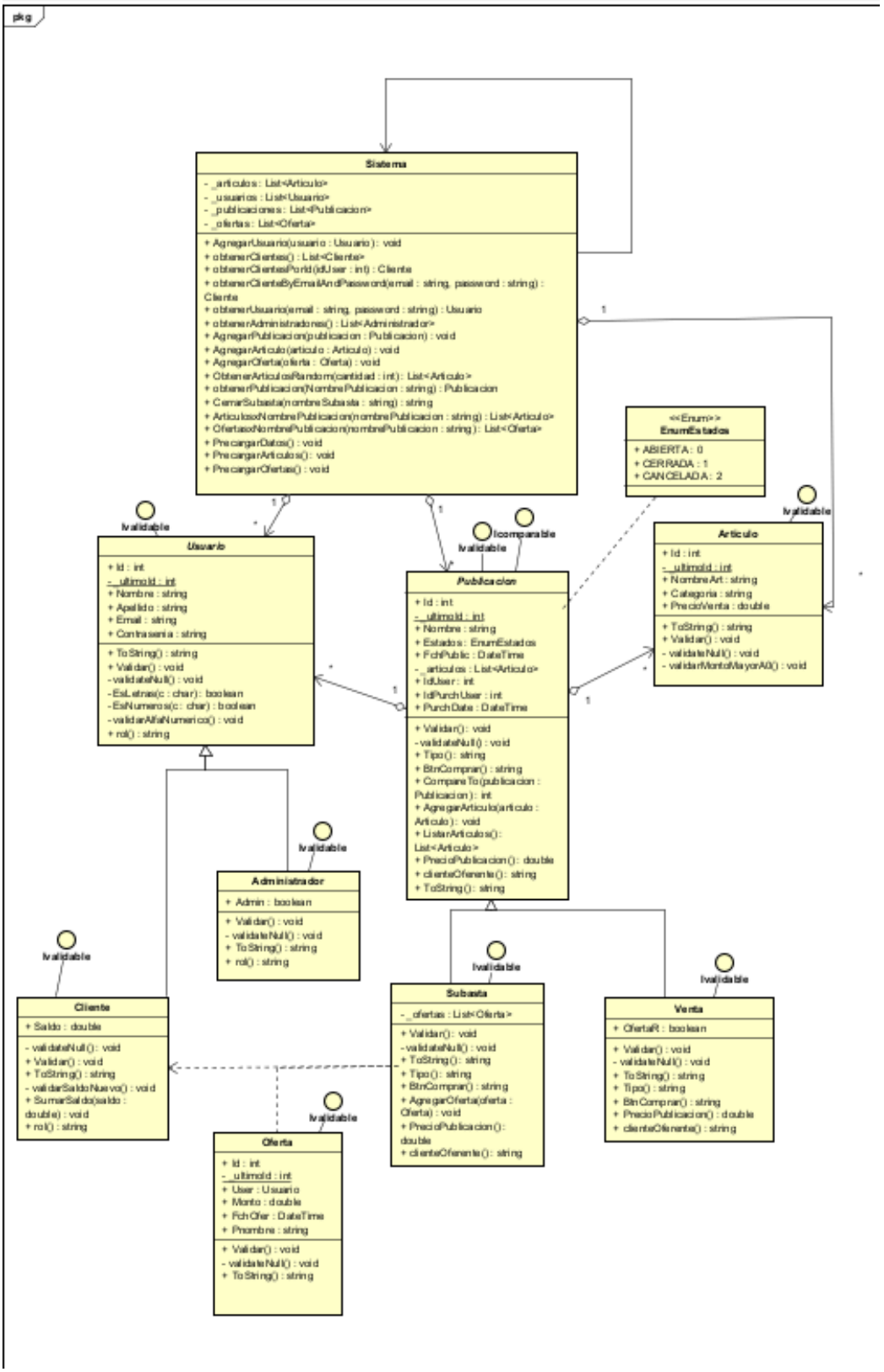

pkg
Sistema
- _articulos : List<Articulo>
- _usuarios : List<Usuario>
- _publicaciones : List<Publicacion>
- _ofertas : List<Oferta>
+ AgregarUsuario(usuario : Usuario) : void
+ obtenerClientes() : List<Cliente>
+ obtenerClientesPorId(idUser : int) : Cliente
+ obtenerClienteByEmailAndPassword(email : string, password : string) : Cliente
+ obtenerUsuario(email : string, password : string) : Usuario
+ obtenerAdministradores() : List<Administrador>
+ AgregarPublicacion(publicacion : Publicacion) : void
+ AgregarArticulo(articulo : Articulo) : void
+ AgregarOferta(oferta : Oferta) : void
+ ObtenerArticulosRandom(cantidad : int) : List<Articulo>
+ obtenerPublicacion(NombrePublicacion : string) : Publicacion
+ CerrarSubasta(nombreSubasta : string) : string
+ ArticulosxNombrePublicacion(nombrePublicacion : string) : List<Articulo>
+ OfertasxNombrePublicacion(nombrePublicacion : string) : List<Oferta>
+ PrecargarDatos() : void
+ PrecargarArticulos() : void
+ PrecargarOfertas() : void
«Enum»
EnumEstados
+ ABIERTA : 0
+ CERRADA : 1
+ CANCELADA : 2
IValidable
Usuario
+ Id : int
- _ultimoId : int
+ Nombre : string
+ Apellido : string
+ Email : string
+ Contrasenia : string
+ ToString() : string
+ Validar() : void
- validateNull() : void
-EsLetras(c : char) : boolean
-EsNumeros(c:: char) : boolean
-validarAlfaNumerico() : void
+ rol() : string
IComparable
Publicacion
+ Id : int
- _ultimoId : int
+ Nombre : string
+ Estados : EnumEstados
+ FchPublic : DateTime
- _articulos : List<Articulo>
+ IdUser : int
+ IdPurchUser : int
+ PurchDate : DateTime
+ Validar() : void
- validateNull() : void
+ Tipo() : string
+ BtnComprar() : string
+ CompareTo(publicacion : Publicacion) : int
+ AgregarArticulo(articulo : Articulo) : void
+ ListarArticulos() : List<Articulo>
+ PrecioPublicacion() : double
+ clienteOferente() : string
+ ToString() : string
Articulo
+ Id : int
- _ultimoId : int
+ NombreArt : string
+ Categoria : string
+ PrecioVenta : double
+ ToString() : string
+ Validar() : void
- validateNull() : void
- validarMontoMayorA0() : void
Administrador
+ Admin : boolean
+ Validar() : void
- validateNull() : void
+ ToString() : string
+ rol() : string
Cliente
+ Saldo : double
- validateNull() : void
+ Validar() : void
+ ToString() : string
- validarSaldoNuevo() : void
+ SumarSaldo(saldo : double) : void
+ rol() : string
Subasta
- _ofertas : List<Oferta>
+ Validar() : void
- validateNull() : void
+ ToString() : string
+ Tipo() : string
+ BtnComprar() : string
+ AgregarOferta(oferta : Oferta) : void
+ PrecioPublicacion() : double
+ clienteOferente() : string
Venta
+ OfertaR : boolean
+ Validar() : void
- validateNull() : void
+ ToString() : string
+ Tipo() : string
+ BtnComprar() : string
+ PrecioPublicacion() : double
+ clienteOferente() : string
Oferta
+ Id : int
- _ultimoId : int
+ User : Usuario
+ Monto : double
+ FchOfer : DateTime
+ Pnombre : string
+ Validar() : void
- validateNull() : void
+ ToString() : string

Tabla con información de lo precargado

Usuarios

Tipo	Nombre	Apellido	Correo	Contraseña	Monto
Cliente	Juan	Pérez	juan.perez@mail.com	password1	1000
Cliente	Ana	Gomez	ana.gomez@mail.com	password2	1500
Cliente	Carlos	Lopez	carlos.lopez@mail.com	password3	2000
Cliente	Maria	Fernandez	maria.fernandez@mail.com	password4	2500
Cliente	Mauricio	Varela	mauri.sape@mail.com	password5	3000
Cliente	Matias	Alvarez	mati.programer@mail.com	password6	3500
Cliente	Pedro	Martínez	pedro.martinez@mail.com	password7	4000
Cliente	Luis	Gómez	luis.gomez@mail.com	password8	4500
Cliente	Laura	Fernández	laura.fernandez@mail.com	password9	5000
Cliente	Jorge	Díaz	jorge.diaz@mail.com	password10	5500
Administrador	Marta	Suarez	marta.suarez@mail.com	admin1	
Administrador	Luis	Ramirez	luis.ramirez@mail.com	admin2	

Artículos

Nombre	Precio	Categoría
Laptop	1200	Electrónica
Smartphone	800	Electrónica
Televisor	1000	Electrónica
Cámara	500	Fotografía
Micrófono	150	Audio
Auriculares	90	Audio
Teclado	50	Periféricos
Mouse	30	Periféricos
Monitor	200	Periféricos
Impresora	120	Oficina
Mesa	250	Muebles
Silla	120	Muebles
Lámpara	45	Iluminación
Reloj	90	Accesorios
Pulsera	65	Joyería
Collar	150	Joyería
Anillo	300	Joyería
Chaqueta	100	Ropa
Zapatos	80	Ropa

Cartera	50	Accesorios
Mochila	90	Accesorios
Libro	20	Libros
Revista	6	Libros
Cuaderno	4	Papelería
Bolígrafo	80	Papelería

Ventilador	80	Electrodomésticos
Plancha	46	Electrodomésticos
Tostadora	30	Electrodomésticos
Cafetera	60	Electrodomésticos
Refrigerador	900	Electrodomésticos
Estufa	500	Electrodomésticos
Licuadora	35	Electrodomésticos
Cámara de seguridad	200	Seguridad
Candado	13	Seguridad
Alarma	50	Seguridad
Bicicleta	400	Deportes
Balón	20	Deportes
Raqueta	80	Deportes
Guantes	15	Deportes

Camiseta deportiva	30	Deportes
Pantalones	40	Ropa
Vestido	26	Ropa
Gafas de sol	16	Accesorios
Sombrero	200	Accesorios
Maleta	10	Viaje
Paraguas	40	Accesorios
Batería portátil	50	Electrónica
Altavoz Bluetooth	300	Audio
Escoba	150	hogar
Consola de videojuegos	300	Electrónica

Ofertas

Monto	Fecha Realizada	Nombre Usuario
150.20	DateTime.Now	Luxury
120.10	DateTime.Now	Black Friday

Subastas

id	nombre	estado	Artículos	Ofertas
1	Luxury	ABIERTA	ObtenerArtxCat("Joyería")	ofertasxPublicacion("Luxury")
2	Holiday Deals	ABIERTA	ObtenerArtxCat("Viaje")	ofertasxPublicacion("sin ofertas")
3	Black Friday	ABIERTA	ObtenerArtxCat("Accesorios")	ofertasxPublicacion("Black Friday")
4	Black Friday	ABIERTA	ObtenerArtxCat("Accesorios")	ofertasxPublicacion("sin ofertas")
5	Cyber Monday	ABIERTA	ObtenerArtxCat("Periféricos")	ofertasxPublicacion("sin ofertas")
6	Navidad 2024	ABIERTA	ObtenerArtxCat("Papelería")	ofertasxPublicacion("sin ofertas")
7	Año Nuevo 2025	ABIERTA	ObtenerArtxCat("Muebles")	ofertasxPublicacion("sin ofertas")
8	Verano 2025	ABIERTA	ObtenerArtxCat("Ropa")	ofertasxPublicacion("sin ofertas")
9	Primavera 2025	ABIERTA	ObtenerArtxCat("Ropa")	ofertasxPublicacion("sin ofertas")
10	San Valentín 2025	ABIERTA	ObtenerArtxCat("Joyería")	ofertasxPublicacion("sin ofertas")

Ventas

ID	Nombre	Estado	fecha Publicación	Artículos	Oferta relámpago
1	Electro Party	ABIERTA	DateTime.Now	ObtenerArtxCat("Electróni	true

				ca")	
2	Sport Sale	ABIERTA	DateTime.Now	ObtenerArtxCat("Deportes")	true
3	Mega Sale	ABIERTA	DateTime.Now	ObtenerArtxCat("ropa")	true
4	Tech Expo	ABIERTA	DateTime.Now	ObtenerArtxCat("Periféricos")	false
5	Gadget Fest	ABIERTA	DateTime.Now	ObtenerArtxCat("Audio")	true
6	Book Sales	ABIERTA	DateTime.Now	ObtenerArtxCat("Libros")	false
7	Work World	ABIERTA	DateTime.Now	ObtenerArtxCat("Oficina")	false
8	Mayor Tranquilidad	ABIERTA	DateTime.Now	ObtenerArtxCat("Seguridad")	false
9	Luxury	ABIERTA	DateTime.Now	ObtenerArtxCat("Joyería")	false
10	Black Friday	ABIERTA	DateTime.Now	ObtenerArtxCat("Viaje")	false

Código comentado

```
**********************************
Archivo: Sistema.cs
Carpeta:
C:\Users\Ricio\Desktop\ort\p2\obligatorioVersionFinal\Obligatorio1\Dominio\Sistema.cs
**********************************
using Dominio.Entidades;
namespace Dominio
{
   public class Sistema
   {
      private List<Articulo> _articulos = new List<Articulo>();
      private List<Usuario> _usuarios = new List<Usuario>();
      private List<Publicacion> _publicaciones = new List<Publicacion>();
      private List<Oferta> _ofertas = new List<Oferta>();

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

      public Sistema()
      {
         PrecargarDatos();
         PrecargarArticulos();
         PrecargarOfertas();
      }
      public enum EnumEstados
      {
         ABIERTA,
         CERRADA,
         CANCELADA
      }
      public IEnumerable<Articulo> Articulos
      {
         get { return _articulos; }
      }
      public IEnumerable<Publicacion> Publicaciones
      {
         get { return _publicaciones; }
      }
```

```
        public void AgregarUsuario(Usuario usuario)
        {
            if (usuario == null) throw new Exception("Error en la carga de Usuario!");
            usuario.Validar();
            _usuarios.Add(usuario);
        }
        public IEnumerable<Cliente> obtenerClientes()
        {
            List<Cliente> aux = new List<Cliente>();

            foreach (Usuario user in _usuarios)
            {
                if (user is Cliente)
                {
                    Cliente cliente = (Cliente)user;
                    aux.Add(cliente);
                }
            }
            return aux;
        }

        public Cliente obtenerClientesPorId(int idUser)
        {
            List<Cliente> aux = new List<Cliente>();

            foreach (Usuario user in _usuarios)
            {
                if (user is Cliente)
                {
                    Cliente cliente = (Cliente)user;
                    if (cliente.Id == idUser)
                    {
                        return cliente;
                    }
                }
            }
            return null;
        }

        public Cliente obtenerClienteByEmailAndPassword(string email, string password)
        {
            foreach (Usuario unCliente in obtenerClientes())
            {
                if (email != null && password != null && unCliente.Email == email.ToLower() &&
unCliente.Contrasenia == password.ToLower())
                {
                    Cliente cliente = (Cliente)unCliente;
                    return cliente;
```

```
        }
    }
    return null;
}
public Usuario obtenerUsuario(string email, string password)
{
    foreach (Usuario unS in _usuarios)
    {
        if (unS == null)
        {
            throw new Exception("Credenciales no validas");
        }
        if (unS.Email.ToLower() == email.ToLower() && unS.Contrasenia == password)
        {
            return unS;

        }
    }
    return null;
}
public IEnumerable<Administrador> obtenerAdministradores()
{
    List<Administrador> aux = new List<Administrador>();

    foreach (Usuario user in _usuarios)
    {
        if (user is Administrador)
        {
            Administrador admin = (Administrador)user;
            aux.Add(admin);
        }
    }
    return aux;
}
public void AgregarPublicacion(Publicacion publicacion)
{
    if (publicacion == null) throw new Exception("Error en la carga de publicacion!");
    publicacion.Validar();
    _publicaciones.Add(publicacion);
}
public void AgregarArticulo(Articulo articulo)
{
    if (articulo == null) throw new Exception("Error en la carga de Articulo!");
    articulo.Validar();
    _articulos.Add(articulo);
}
public void AgregarOferta(Oferta oferta)
{
```

```
            if (oferta == null)
                throw new Exception("Error en la carga de oferta!");
            oferta.Validar();
            _ofertas.Add(oferta);
        }
        public IEnumerable<Articulo> ObtenerArticulosRandom(int cantidad)
        {
            List<Articulo> aux = new List<Articulo>();
            Random rand = new Random();
            int totalArticulos = _articulos.Count;

            if (cantidad > totalArticulos)
            {
                cantidad = totalArticulos;
            }

            List<int> indicesSeleccionados = new List<int>();

            while (aux.Count < cantidad)
            {
                int index = rand.Next(totalArticulos);
                if (!indicesSeleccionados.Contains(index))
                {
                    aux.Add(_articulos[index]);
                    indicesSeleccionados.Add(index);
                }
            }

            return aux;
        }
        public Publicacion obtenerPublicacion(string NombrePublicacion)
        {
            foreach (var item in _publicaciones)
            {
                if (NombrePublicacion != null && item.Nombre.ToLower() != null &&
item.Nombre.ToLower() == NombrePublicacion.ToLower())
                {
                    return item;
                }
            }
            return null;
        }
        public string CerrarSubasta(string nombreSubasta)
        {
            Publicacion unaS = obtenerPublicacion(nombreSubasta);
            if (unaS != null)
            {
                unaS.Estados = EnumEstados.CERRADA;
```

```
            return "Subasta finalizada con exito!!";
        }
        return "Error al finalizar la subasta";
    }
    public IEnumerable<Articulo> ArticulosxNombrePublicacion(string nombrePublicacion)
    {
        List<Articulo> aux = new List<Articulo>();
        foreach (var item in _publicaciones)
        {
            if (item.Nombre.ToLower() == nombrePublicacion.ToLower())
            {
                foreach (var art in _articulos)
                {
                    aux.Add(art);
                }
            }
        }
        return aux;
    }
    public IEnumerable<Oferta> OfertasxNombrePublicacion(string nombrePublicacion)
    {
        List<Oferta> aux = new List<Oferta>();
        foreach (Publicacion item in _publicaciones)
        {
            foreach (Oferta ofe in _ofertas)
            {
                if (nombrePublicacion != null && item.Nombre == nombrePublicacion)
                {
                    aux.Add(ofe);
                }
            }
        }
        return aux;
    }

    public void PrecargarDatos()
    {
        #region clientes
        //Precarga de  10 clientes
        AgregarUsuario(new Cliente("Mauricio", "Varela", "mauri.sape@mail.com",
"password1", 1000));
        AgregarUsuario(new Cliente("Matias", "Alvarez", "mati.programer@mail.com",
"password2", 1500));
        AgregarUsuario(new Cliente("Carlos", "Lopez", "carlos.lopez@mail.com",
"password3", 2000));
        AgregarUsuario(new Cliente("María", "García", "maria.garcia@mail.com",
"password1", 1500));
```

```
        AgregarUsuario(new Cliente("Juan", "Pérez", "juan.perez@mail.com", "password2",
1000));
        AgregarUsuario(new Cliente("Ana", "Sánchez", "ana.sanchez@mail.com",
"password4", 2500));
        AgregarUsuario(new Cliente("Pedro", "Martínez", "pedro.martinez@mail.com",
"password5", 3000));
        AgregarUsuario(new Cliente("Luis", "Gómez", "luis.gomez@mail.com", "password6",
1200));
        AgregarUsuario(new Cliente("Laura", "Fernández", "laura.fernandez@mail.com",
"password7", 1800));
        AgregarUsuario(new Cliente("Jorge", "Díaz", "jorge.diaz@mail.com", "password8",
2200));
        //AgregarUsuario(null); prueba de precarga nula
        #endregion

        #region admin
        //Precarga de 2 Administradores
        AgregarUsuario(new Administrador("Marta", "Suarez", "marta.suarez@mail.com",
"administrator1"));
        AgregarUsuario(new Administrador("Luis", "Ramirez", "luis.ramirez@mail.com",
"administrator2"));
        #endregion

        #region Articulos
        //Precarga de 50 articulos

        //promt en chatgpt utilizado
        //    crea 50 precargas en codigo c# para una clase articulo que tiene el siguiente
constructor         public Articulo(string nombre, string categoria, int precioVenta)
        //{
        //        Id = _ultimoId++;
        //        NombreArt = nombre;
        //        Categoria = categoria;
        //        PrecioVenta = precioVenta;
        //    }
        AgregarArticulo(new Articulo("Laptop", "Electrónica", 1200));
        AgregarArticulo(new Articulo("Smartphone", "Electrónica", 800));
        AgregarArticulo(new Articulo("Televisor", "Electrónica", 1000));
        AgregarArticulo(new Articulo("Cámara", "Fotografía", 500));
        AgregarArticulo(new Articulo("Micrófono", "Audio", 150));
        AgregarArticulo(new Articulo("Auriculares", "Audio", 90));
        AgregarArticulo(new Articulo("Teclado", "Periféricos", 50));
        AgregarArticulo(new Articulo("Mouse", "Periféricos", 30));
        AgregarArticulo(new Articulo("Monitor", "Periféricos", 200));
        AgregarArticulo(new Articulo("Impresora", "Oficina", 120));
        AgregarArticulo(new Articulo("Mesa", "Muebles", 250));
        AgregarArticulo(new Articulo("Silla", "Muebles", 120));
        AgregarArticulo(new Articulo("Lámpara", "Iluminación", 45));
```

```
        AgregarArticulo(new Articulo("Reloj", "Accesorios", 90));
        AgregarArticulo(new Articulo("Pulsera", "Joyería", 65));
        AgregarArticulo(new Articulo("Collar", "Joyería", 150));
        AgregarArticulo(new Articulo("Anillo", "Joyería", 300));
        AgregarArticulo(new Articulo("Chaqueta", "Ropa", 100));
        AgregarArticulo(new Articulo("Zapatos", "Ropa", 80));
        AgregarArticulo(new Articulo("Cartera", "Accesorios", 50));
        AgregarArticulo(new Articulo("Mochila", "Accesorios", 90));
        AgregarArticulo(new Articulo("Libro", "Libros", 20));
        AgregarArticulo(new Articulo("Revista", "Libros", 6));
        AgregarArticulo(new Articulo("Cuaderno", "Papelería", 4));
        AgregarArticulo(new Articulo("Bolígrafo", "Papelería", 2));
        AgregarArticulo(new Articulo("Ventilador", "Electrodomésticos", 80));
        AgregarArticulo(new Articulo("Plancha", "Electrodomésticos", 46));
        AgregarArticulo(new Articulo("Tostadora", "Electrodomésticos", 30));
        AgregarArticulo(new Articulo("Cafetera", "Electrodomésticos", 60));
        AgregarArticulo(new Articulo("Refrigerador", "Electrodomésticos", 900));
        AgregarArticulo(new Articulo("Estufa", "Electrodomésticos", 500));
        AgregarArticulo(new Articulo("Licuadora", "Electrodomésticos", 35));
        AgregarArticulo(new Articulo("Cámara de seguridad", "Seguridad", 200));
        AgregarArticulo(new Articulo("Candado", "Seguridad", 13));
        AgregarArticulo(new Articulo("Alarma", "Seguridad", 50));
        AgregarArticulo(new Articulo("Bicicleta", "Deportes", 400));
        AgregarArticulo(new Articulo("Balón", "Deportes", 20));
        AgregarArticulo(new Articulo("Raqueta", "Deportes", 80));
        AgregarArticulo(new Articulo("Guantes", "Deportes", 15));
        AgregarArticulo(new Articulo("Camiseta deportiva", "Ropa", 30));
        AgregarArticulo(new Articulo("Pantalones", "Ropa", 40));
        AgregarArticulo(new Articulo("Vestido", "Ropa", 70));
        AgregarArticulo(new Articulo("Gafas de sol", "Accesorios", 26));
        AgregarArticulo(new Articulo("Sombrero", "Accesorios", 16));
        AgregarArticulo(new Articulo("Maleta", "Viaje", 200));
        AgregarArticulo(new Articulo("Paraguas", "Accesorios", 10));
        AgregarArticulo(new Articulo("Batería portátil", "Electrónica", 40));
        AgregarArticulo(new Articulo("Altavoz Bluetooth", "Audio", 50));
        AgregarArticulo(new Articulo("Consola de videojuegos", "Electrónica", 300));
        //AgregarArticulo(null); //prueba de precarga nula
        #endregion
        #region Ofertas
        //Precargas de Ofertas
        AgregarOferta(new Oferta(0, obtenerClientesPorId(0), 150.20, new DateTime(2024,
10, 05, 00, 00, 00), "Luxury"));
        AgregarOferta(new Oferta(2, obtenerClientesPorId(2), 120.10, new DateTime(2024,
10, 04, 00, 00, 00), "Black Friday"));
        #endregion

        #region Ventas
        //************Precarga Publicaciones ****************
```

```
        //Precarga 10 ventas
        AgregarPublicacion(new Venta("Electro Party", EnumEstados.ABIERTA, new
DateTime(2024, 10, 05, 00, 00, 00), 0, 0, new DateTime(2024, 10, 05, 00, 00, 00), false));
        AgregarPublicacion(new Venta("Sport Sale", EnumEstados.ABIERTA, new
DateTime(2024, 10, 05, 00, 00, 00), 0, 0, new DateTime(2024, 10, 05, 00, 00, 00), true));
        AgregarPublicacion(new Venta("Mega Sale", EnumEstados.ABIERTA, new
DateTime(2024, 10, 06, 00, 00, 00), 0, 0, new DateTime(2024, 10, 06, 00, 00, 00), false));
        AgregarPublicacion(new Venta("Tech Expo", EnumEstados.CERRADA, new
DateTime(2024, 10, 07, 00, 00, 00), 0, 0, new DateTime(2024, 10, 07, 00, 00, 00), true));
        AgregarPublicacion(new Venta("Gadget Fest", EnumEstados.ABIERTA, new
DateTime(2024, 10, 08, 00, 00, 00), 0, 0, new DateTime(2024, 10, 08, 00, 00, 00), false));
        AgregarPublicacion(new Venta("Book Sales", EnumEstados.CANCELADA, new
DateTime(2024, 10, 09, 00, 00, 00), 0, 0, new DateTime(2024, 10, 09, 00, 00, 00), true));
        AgregarPublicacion(new Venta("Work World", EnumEstados.CANCELADA, new
DateTime(2024, 10, 10, 00, 00, 00), 0, 0, new DateTime(2024, 10, 10, 00, 00, 00), false));
        AgregarPublicacion(new Venta("Mayor Tranquilidad", EnumEstados.ABIERTA, new
DateTime(2024, 10, 11, 00, 00, 00), 0, 0, new DateTime(2024, 10, 11, 00, 00, 00), true));
        AgregarPublicacion(new Venta("Big Sale", EnumEstados.ABIERTA, new
DateTime(2024, 10, 12, 00, 00, 00), 0, 0, new DateTime(2024, 10, 12, 00, 00, 00), false));
        AgregarPublicacion(new Venta("Dia del Hogar", EnumEstados.ABIERTA, new
DateTime(2024, 10, 13, 00, 00, 00), 0, 0, new DateTime(2024, 10, 13, 00, 00, 00), true));
        AgregarPublicacion(new Venta("PlayGround", EnumEstados.ABIERTA, new
DateTime(2024, 10, 14, 00, 00, 00), 0, 0, new DateTime(2024, 10, 14, 00, 00, 00), true));
        #endregion

        #region Subastas
        //Precarga 10 subastas
        AgregarPublicacion(new Subasta("Luxury", EnumEstados.ABIERTA, new
DateTime(2024, 10, 01, 00, 00, 00), 0, 0, new DateTime(2024, 10, 05, 00, 00, 00)));
        AgregarPublicacion(new Subasta("Holiday Deals", EnumEstados.ABIERTA, new
DateTime(2024, 10, 02, 00, 00, 00), 0, 0, new DateTime(2024, 10, 05, 00, 00, 00)));
        AgregarPublicacion(new Subasta("Black Friday", EnumEstados.ABIERTA, new
DateTime(2024, 10, 03, 00, 00, 00), 0, 0, new DateTime(2024, 10, 05, 00, 00, 00)));
        AgregarPublicacion(new Subasta("PlayGround", EnumEstados.ABIERTA, new
DateTime(2024, 10, 03, 00, 00, 00), 0, 0, new DateTime(2024, 10, 08, 00, 00, 00)));
        AgregarPublicacion(new Subasta("Cyber Monday", EnumEstados.ABIERTA, new
DateTime(2024, 11, 27, 00, 00, 00), 0, 0, new DateTime(2024, 11, 30, 00, 00, 00)));
        AgregarPublicacion(new Subasta("Navidad 2024", EnumEstados.ABIERTA, new
DateTime(2024, 12, 20, 00, 00, 00), 0, 0, new DateTime(2024, 12, 25, 00, 00, 00)));
        AgregarPublicacion(new Subasta("Año Nuevo 2025", EnumEstados.ABIERTA, new
DateTime(2024, 12, 31, 00, 00, 00), 0, 0, new DateTime(2025, 01, 01, 00, 00, 00)));
        AgregarPublicacion(new Subasta("Verano 2025", EnumEstados.CERRADA, new
DateTime(2025, 01, 15, 00, 00, 00), 0, 0, new DateTime(2025, 02, 15, 00, 00, 00)));
        AgregarPublicacion(new Subasta("Primavera 2025", EnumEstados.ABIERTA, new
DateTime(2025, 03, 20, 00, 00, 00), 0, 0, new DateTime(2025, 04, 10, 00, 00, 00)));
        AgregarPublicacion(new Subasta("San Valentín 2025", EnumEstados.ABIERTA, new
DateTime(2025, 02, 10, 00, 00, 00), 0, 0, new DateTime(2025, 02, 14, 00, 00, 00)));
        #endregion
```

```
    _publicaciones.Sort();
}

private void PrecargarArticulos()
{
    Publicacion p1 = obtenerPublicacion("Electro Party");
    foreach (Articulo item in ObtenerArticulosRandom(5))
    {
        p1.AgregarArticulo(item);
    }

    Publicacion p2 = obtenerPublicacion("Sport Sale");
    foreach (Articulo item in ObtenerArticulosRandom(5))
    {
        p2.AgregarArticulo(item);
    }

    Publicacion p3 = obtenerPublicacion("Mega Sale");
    foreach (Articulo item in ObtenerArticulosRandom(5))
    {
        p3.AgregarArticulo(item);
    }

    Publicacion p4 = obtenerPublicacion("Tech Expo");
    foreach (Articulo item in ObtenerArticulosRandom(5))
    {
        p4.AgregarArticulo(item);
    }

    Publicacion p5 = obtenerPublicacion("Gadget Fest");
    foreach (Articulo item in ObtenerArticulosRandom(5))
    {
        p5.AgregarArticulo(item);
    }

    Publicacion p6 = obtenerPublicacion("Book Sales");
    foreach (Articulo item in ObtenerArticulosRandom(5))
    {
        p6.AgregarArticulo(item);
    }

    Publicacion p7 = obtenerPublicacion("Work World");
    foreach (Articulo item in ObtenerArticulosRandom(5))
    {
        p7.AgregarArticulo(item);
    }

    Publicacion p8 = obtenerPublicacion("Mayor Tranquilidad");
```

```
foreach (Articulo item in ObtenerArticulosRandom(5))
{
    p8.AgregarArticulo(item);
}

Publicacion p9 = obtenerPublicacion("Luxury");
foreach (Articulo item in ObtenerArticulosRandom(5))
{
    p9.AgregarArticulo(item);
}

Publicacion p10 = obtenerPublicacion("Holiday Deals");
foreach (Articulo item in ObtenerArticulosRandom(5))
{
    p10.AgregarArticulo(item);
}

Publicacion p11 = obtenerPublicacion("Black Friday");
foreach (Articulo item in ObtenerArticulosRandom(5))
{
    p11.AgregarArticulo(item);
}

Publicacion p12 = obtenerPublicacion("Cyber Monday");
foreach (Articulo item in ObtenerArticulosRandom(5))
{
    p12.AgregarArticulo(item);
}

Publicacion p13 = obtenerPublicacion("Navidad 2024");
foreach (Articulo item in ObtenerArticulosRandom(5))
{
    p12.AgregarArticulo(item);
}

Publicacion p14 = obtenerPublicacion("Año Nuevo 2025");
foreach (Articulo item in ObtenerArticulosRandom(5))
{
    p14.AgregarArticulo(item);
}

Publicacion p15 = obtenerPublicacion("Verano 2025");
foreach (Articulo item in ObtenerArticulosRandom(5))
{
    p15.AgregarArticulo(item);
}

Publicacion p16 = obtenerPublicacion("Primavera 2025");
```

```
    foreach (Articulo item in ObtenerArticulosRandom(5))
    {
        p16.AgregarArticulo(item);
    }

    Publicacion p17 = obtenerPublicacion("San Valentín 2025");
    foreach (Articulo item in ObtenerArticulosRandom(5))
    {
        p17.AgregarArticulo(item);
    }

    Publicacion p18 = obtenerPublicacion("Big Sale");
    foreach (Articulo item in ObtenerArticulosRandom(5))
    {
        p18.AgregarArticulo(item);
    }

    Publicacion p19 = obtenerPublicacion("Dia del Hogar");
    foreach (Articulo item in ObtenerArticulosRandom(5))
    {
        p19.AgregarArticulo(item);
    }

    Publicacion p20 = obtenerPublicacion("PlayGround");
    foreach (Articulo item in ObtenerArticulosRandom(5))
    {
        p20.AgregarArticulo(item);
    }

    Publicacion p21 = obtenerPublicacion("Navidad 2024");
    foreach (Articulo item in ObtenerArticulosRandom(10))
    {
        p21.AgregarArticulo(item);
    }
}
private void PrecargarOfertas()
{
    Publicacion o1 = obtenerPublicacion("Electro Party");
    if (o1.Tipo() == "Subasta")
    {
        Subasta subasta = (Subasta)o1;
        foreach (Oferta unaoferta in _ofertas)
        {
            if (unaoferta.Pnombre == "Electro Party")
                subasta.AgregarOferta(unaoferta);
        }
    }
```

```
Publicacion o2 = obtenerPublicacion("Luxury");
if (o2.Tipo() == "Subasta")
{
   Subasta subasta = (Subasta)o2;
   foreach (Oferta unaoferta in _ofertas)
   {
      if (unaoferta.Pnombre == "Luxury")
         subasta.AgregarOferta(unaoferta);
   }
}

Publicacion o3 = obtenerPublicacion("Holiday Deals");
if (o3.Tipo() == "Subasta")
{
   Subasta subasta = (Subasta)o3;
   foreach (Oferta unaoferta in _ofertas)
   {
      if (unaoferta.Pnombre == "Holiday Deals")
         subasta.AgregarOferta(unaoferta);
   }
}
Publicacion o4 = obtenerPublicacion("Black Friday");
if (o4.Tipo() == "Subasta")
{
   Subasta subasta = (Subasta)o4;
   foreach (Oferta unaoferta in _ofertas)
   {
      if (unaoferta.Pnombre == "Black Friday")
         subasta.AgregarOferta(unaoferta);
   }
}

Publicacion o5 = obtenerPublicacion("Cyber Monday");
if (o5.Tipo() == "Subasta")
{
   Subasta subasta = (Subasta)o5;
   foreach (Oferta unaoferta in _ofertas)
   {
      if (unaoferta.Pnombre == "Cyber Monday")
         subasta.AgregarOferta(unaoferta);
   }
}

Publicacion o6 = obtenerPublicacion("Navidad 2024");
if (o6.Tipo() == "Subasta")
{
   Subasta subasta = (Subasta)o6;
   foreach (Oferta unaoferta in _ofertas)
```

```
    {
        if (unaoferta.Pnombre == "Navidad 2024")
            subasta.AgregarOferta(unaoferta);
    }
}

Publicacion o7 = obtenerPublicacion("Año Nuevo 2025");
if (o7.Tipo() == "Subasta")
{
    Subasta subasta = (Subasta)o7;
    foreach (Oferta unaoferta in _ofertas)
    {
        if (unaoferta.Pnombre == "Año Nuevo 2025")
            subasta.AgregarOferta(unaoferta);
    }
}

Publicacion o8 = obtenerPublicacion("Verano 2025");
if (o8.Tipo() == "Subasta")
{
    Subasta subasta = (Subasta)o8;
    foreach (Oferta unaoferta in _ofertas)
    {
        if (unaoferta.Pnombre == "Verano 2025")
            subasta.AgregarOferta(unaoferta);
    }
}

Publicacion o9 = obtenerPublicacion("Primavera 2025");
if (o9.Tipo() == "Subasta")
{
    Subasta subasta = (Subasta)o9;
    foreach (Oferta unaoferta in _ofertas)
    {
        if (unaoferta.Pnombre == "Primavera 2025")
            subasta.AgregarOferta(unaoferta);
    }
}

Publicacion o10 = obtenerPublicacion("San Valentín 2025");
if (o10.Tipo() == "Subasta")
{
    Subasta subasta = (Subasta)o10;
    foreach (Oferta unaoferta in _ofertas)
    {
        if (unaoferta.Pnombre == "San Valentín 2025")
            subasta.AgregarOferta(unaoferta);
    }
```

```
        }

      }

   }
}
```

****
Archivo: Administrador.cs
Carpeta: C:\Users\Ricio\Desktop\ort\p2\obligatorioVersionFinal\Obligatorio1\Dominio\Entidades\Administrador.cs
****

```
using Dominio.Interfaces;
namespace Dominio.Entidades
{
   public class Administrador : Usuario, IValidable
   {
      public bool Admin { get; }

      public Administrador(string nombre, string apellido, string email, string contrasenia) :
         base(nombre, apellido, email, contrasenia)
      {
         Admin = true;
      }
      public virtual void Validar()
      {
         validateNull();
         validarSaldoNuevo();
      }

      private void validateNull()
      {
         if (string.IsNullOrEmpty(base.Nombre))
         {
            throw new Exception("El nombre no puede ser vacio");
         }
      }

      public override string rol()
      {
         return "Admin";
      }
      public override void validarSaldoNuevo() {}
      public override string ToString()
      {
         string respuesta = base.ToString();
         if (Admin) respuesta += $"Administrador \n";
```

```
            return respuesta;
        }
    }
}

**********
Archivo: Articulo.cs
Carpeta:
C:\Users\Ricio\Desktop\ort\p2\obligatorioVersionFinal\Obligatorio1\Dominio\Entidades\Articul
o.cs
**********
using Dominio.Interfaces;

namespace Dominio.Entidades
{
    public class Articulo : IValidable
    {
        #region Atributos
        public int Id { get; set; }
        private static int _ultimoId;
        public string NombreArt { get; set; }
        public string Categoria { get; set; }
        public double PrecioVenta { get; set; }
        #endregion
        #region Constructor
        public Articulo() { }
        public Articulo(string nombre, string categoria, double precioVenta)
        {
            Id = _ultimoId++;
            NombreArt = nombre;
            Categoria = categoria;
            PrecioVenta = precioVenta;
        }
        #endregion
        public void Validar()
        {
            validateNull();
            validarMontoMayorA0();
        }
        private void validateNull()
        {
            if (string.IsNullOrEmpty(NombreArt))
            {
                throw new Exception("El nombre no puede ser vacio");
            }
        }
        private void validarMontoMayorA0()
        {
```

```
            if (PrecioVenta < 0)
            {
                throw new Exception("El precio debe ser mayor a Cero");
            }
        }
        public override string ToString()
        {
            string respuesta = string.Empty;
            respuesta += $"Id: {_ultimoId++} \n";
            respuesta += $"Nombre articulo: {NombreArt} \n";
            respuesta += $"Categoria: {Categoria} \n";
            respuesta += $"Precio de Venta: {PrecioVenta} \n";
            return respuesta;
        }
    }
}
```

****

Archivo: Cliente.cs

Carpeta: C:\Users\Ricio\Desktop\ort\p2\obligatorioVersionFinal\Obligatorio1\Dominio\Entidades\Cliente.cs

****

```
using Dominio.Interfaces;
namespace Dominio.Entidades
{
    public class Cliente : Usuario, IValidable
    {
        public double Saldo { get; set; }
        public Cliente() { }
        public Cliente(string nombre, string apellido, string email, string contrasenia, double saldo) :
            base(nombre, apellido, email, contrasenia)
        {
            Saldo = saldo;
        }
        public virtual void Validar()
        {
            validateNull();
            validarSaldoNuevo();
        }
        private void validateNull()
        {
            if (string.IsNullOrEmpty(base.Nombre))
            {
                throw new Exception("El nombre no puede ser vacio");
            }
        }
```

```
        public override void validarSaldoNuevo()
        {
            if(Saldo < 0) {
                throw new Exception("El saldo debe ser mayor a 0");
            }
        }

        public void SumarSaldo(double saldo)
        {
            validarSaldoNuevo();
            Saldo += saldo;
        }

                    public override string rol()
                    {
                            return "Cliente";
                    }
                    public override string ToString()
        {
            string respuesta = base.ToString();
            respuesta += $"saldo: {Saldo} \n";
            return respuesta;
        }
    }
}
```

****

Archivo: Oferta.cs

Carpeta: C:\Users\Ricio\Desktop\ort\p2\obligatorioVersionFinal\Obligatorio1\Dominio\Entidades\Oferta.cs

****

```
using Dominio.Interfaces;
namespace Dominio.Entidades
{
    public class Oferta : IValidable
    {
        public int Id { get; set; }
        public Usuario User { get; set; }
        public double Monto { get; set; }
        public DateTime FchOfer { get; set; }
        public string Pnombre { get; set; }
        private static int _ultimoId;
        public Oferta() { }
        public Oferta(int id, Usuario user, double monto, DateTime fchOfer, string pnombre)
        {
            Id = _ultimoId++;
```

```
            User = user;
            Monto = monto;
            FchOfer = fchOfer;
            Pnombre = pnombre;
        }
        public void Validar()
        {
            validateNull();
        }
        private void validateNull()
        {
            if (string.IsNullOrEmpty(Pnombre))
            {
                throw new Exception("El nombre no puede ser vacio");
            }
        }

                public override string ToString()
                {
                        string respuesta = string.Empty;
                        respuesta += $"Nombre de Publicacion: {Pnombre} \n";
                        respuesta += $"Nombre : {User.Nombre} \n";
                        respuesta += $"Importe: {Monto} \n";
                        respuesta += $"Fecha de Oferta: {FchOfer} \n";
                        return respuesta;
                }
        }
}
```

****
Archivo: Publicacion.cs
Carpeta:
C:\Users\Ricio\Desktop\ort\p2\obligatorioVersionFinal\Obligatorio1\Dominio\Entidades\Publicacion.cs
****

```
using Dominio.Interfaces;
using static Dominio.Sistema;

namespace Dominio.Entidades
{
        public abstract class Publicacion : IValidable, IComparable<Publicacion>
        {
                #region Atributos
                public int Id { get; set; }
                public string Nombre { get; set; }
                public EnumEstados Estados { get; set; }
                public DateTime FchPublic { get; set; }
```

```
private List<Articulo> _articulos = new List<Articulo>();
public int IdUser { get; set; }
public int IdPurchUser { get; set; }
public DateTime PurchDate { get; set; }
private static int _ultimoId;
#endregion
#region Constructor
public Publicacion() { }
public Publicacion(
                                    string nombre,
                                    EnumEstados estados,
                                    DateTime fchPublic,
                                    int idUser,
                                    int idPurchUser,
                                    DateTime purchDate
                                    )
{
        Id = _ultimoId++;
        Nombre = nombre;
        FchPublic = fchPublic;
        Estados = estados;
        IdUser = idUser;
        IdPurchUser = idPurchUser;
        PurchDate = purchDate;
}
#endregion

public void Validar()
{
        validateNull();
}
private void validateNull()
{
        if (string.IsNullOrEmpty(Nombre))
        {
                throw new Exception("El nombre no puede ser vacio");
        }
}
public abstract string Tipo();

public abstract string BtnComprar();

public int CompareTo(Publicacion? other)
{
        if (other == null)
                return 1;
        return FchPublic.CompareTo(other.FchPublic);
}
```

```
        public override string ToString()
        {
                string respuesta = string.Empty;
                respuesta += $"Id: {Id} \n";
                respuesta += $"Nombre de la publicacion: {Nombre} \n";
                respuesta += $"Estados Publicacion: {Estados} \n";
                respuesta += $"Fecha de Publicacion: {FchPublic:dd/MM/yyyy} \n";
   foreach (Articulo item in _articulos)
   {
      respuesta += $"{item.NombreArt}\n";
                        respuesta += $"{item.Categoria}\n";
                        respuesta += $"{item.PrecioVenta}\n";
   }
   return respuesta;
        }

public void AgregarArticulo(Articulo articulo)
{
   if (articulo == null)
   {
      throw new Exception("El Articulo es nulo");
   }
   articulo.Validar();
   _articulos.Add(articulo);
}

        public List<Articulo> ListarArticulos()
        {
                List<Articulo> aux = new List<Articulo>();
                foreach (Articulo item in _articulos)
                {
                        aux.Add(item);
                }
                return aux;

        }
        public virtual double PrecioPublicacion()
        {
                double total = 0;

                foreach (Articulo item in _articulos)
                {
                        total += item.PrecioVenta;
                }

                return total;
        }
```

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

**********
Archivo: Subasta.cs
Carpeta:
C:\Users\Ricio\Desktop\ort\p2\obligatorioVersionFinal\Obligatorio1\Dominio\Entidades\Subas
ta.cs
**********
using Dominio.Interfaces;
using static Dominio.Sistema;
namespace Dominio.Entidades
{
    public class Subasta : Publicacion, IValidable
    {
        private List<Oferta> _ofertas = new List<Oferta>();
        public List<Oferta> Ofertas
        {
            get { return _ofertas; }
        }
        public object Articulos { get; set; }
        public Subasta(
                    string nombre,
                    EnumEstados estados,
                    DateTime fchPublic,
                    int idUser,
                    int idPurchUser,
                    DateTime purchDate
                  ) : base(nombre, estados, fchPublic,
                     idUser, idPurchUser, purchDate)
        {
        }
        public virtual void Validar()
        {
            validateNull();
        }
        private void validateNull()
        {
            if (string.IsNullOrEmpty(base.Nombre))
            {
                throw new Exception("El nombre no puede ser vacio");
            }
        }
        public override string Tipo()
        {
            string salida = "Subasta";
            return salida;
```

```
}

public override string BtnComprar()
{
    string salida = "Ofertar";
    return salida;
}
public override string ToString()
{
    string respuesta = base.ToString();
    foreach (var item in _ofertas)
    {
        respuesta += $"{item.User.Nombre}\n";
        respuesta += $"{item.Monto}\n";
    }
    return respuesta;
}


public void AgregarOferta(Oferta oferta)
{
    if (oferta == null)
        throw new Exception("Error en la carga de oferta!");
    oferta.Validar();
    _ofertas.Add(oferta);

}

public override double PrecioPublicacion()
{
    double total = base.PrecioPublicacion();

    if (_ofertas.Count() > 0)
    {
        total = _ofertas[_ofertas.Count() - 1].Monto;
    }
    return total;
}

public override string clienteOferente()
{
    if (_ofertas.Count() == 0)
    {
        return null;
     }
    return _ofertas[_ofertas.Count() - 1].User.Email;
}
```

```
    }

}
**********
Archivo: Usuario.cs
Carpeta:
C:\Users\Ricio\Desktop\ort\p2\obligatorioVersionFinal\Obligatorio1\Dominio\Entidades\Usuar
io.cs
**********
using Dominio.Interfaces;

namespace Dominio.Entidades
{
    public abstract class Usuario : IValidable
    {
        private static int _ultimoId;
        public int Id { get; }
        public string Nombre { get; }
        public string Apellido { get; }
        public string Email { get; }
        public string Contrasenia { get; }
        public Usuario() { }
        public Usuario(string nombre, string apellido, string email, string contrasenia)
        {
            Id = _ultimoId++;
            Nombre = nombre;
            Apellido = apellido;
            Email = email;
            Contrasenia = contrasenia;
        }
        public void Validar()
        {
            validateNull();
            validarAlfaNumerico();
            validarSaldoNuevo();
        }
        private void validateNull()
        {
            if (string.IsNullOrEmpty(Nombre))
            {
                throw new Exception("El nombre no puede ser vacio");
            }
        }

        private bool EsLetras(char c)
        {
            if (!char.IsLetter(c))
            {
```

```
            return false;
        }
        return true;
    }

    private bool EsNumeros(char c)
    {
        if (!char.IsDigit(c))
        {
            return false;
        }
        return true;
    }

    private void validarAlfaNumerico()
    {
        bool letra = false;
        bool numero = false;
        bool salida = false;
        foreach (char c in Contrasenia)
        {
            if (EsNumeros(c))
            {
                numero = true;
            }
            if (EsLetras(c))
            {
                letra = true;
            }
        }
        salida = numero && letra;
        if (!salida)
        {
            throw new Exception("La contraseña debe ser alfanumerica.");
        }
    }
    public abstract void validarSaldoNuevo();

    public abstract string rol();
    public override string ToString()
    {
        string respuesta = string.Empty;
        respuesta += $"Nombre: {Nombre} \n";
        respuesta += $"Apellido: {Apellido} \n";
        respuesta += $"Email: {Email} \n";
        return respuesta;
    }
}
```

```
}

***********************************
Archivo: Venta.cs
Carpeta:
C:\Users\Ricio\Desktop\ort\p2\obligatorioVersionFinal\Obligatorio1\Dominio\Entidades\Venta.
cs
***********************************
using Dominio.Interfaces;
using static Dominio.Sistema;

namespace Dominio.Entidades
{
	public class Venta : Publicacion, IValidable
	{
		public bool OfertaR { get; set; }
		public object Articulos { get; private set; }
		public List<Articulo> ObtenerArtxPub { get; set; }
		public Venta(
					string nombre,
					EnumEstados estados,
					DateTime fchPublic,
					int idUser,
					int idPurchUser,
					DateTime purchDate,
					bool ofertar
			) : base(nombre, estados, fchPublic,
				idUser, idPurchUser, purchDate)
		{
			OfertaR = ofertar;

		}
		public virtual void Validar()
		{
			validateNull();
		}
		private void validateNull()
		{
			if (string.IsNullOrEmpty(base.Nombre))
			{
				throw new Exception("El nombre no puede ser vacio");
			}
		}

		public override string Tipo()
		{
			string salida = "Venta";
			return salida;
```

```
        }

        public override string BtnComprar()
        {
            string salida = "Comprar";
            return salida;
        }
        public override string ToString()
        {
            string respuesta = base.ToString();
            if (OfertaR)
            {
                respuesta += $"Oferta Relampago \n";
            }
            return respuesta;
        }

        public override double PrecioPublicacion()
        {
            double total = base.PrecioPublicacion();

            if (OfertaR)
            {
                total = Math.Round(total * 0.8);
            }
            return total;
        }

    public override string clienteOferente()
    {
            return null;
    }
  }
}
```

****

Archivo: IValidable.cs

Carpeta: C:\Users\Ricio\Desktop\ort\p2\obligatorioVersionFinal\Obligatorio1\Dominio\Interfaces\IValidable.cs

****

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

```
}

***********************************
Archivo: AdministradorController.cs
Carpeta:
C:\Users\Ricio\Desktop\ort\p2\obligatorioVersionFinal\Obligatorio1\WebApplication1\Controll
ers\AdministradorController.cs
***********************************
using Dominio;
using Microsoft.AspNetCore.Mvc;

namespace WebApplication1.Controllers
{
    public class AdministradorController : Controller
    {
        private Sistema _sistema = Sistema.Instancia;
        public IActionResult Index()
        {
            ViewBag.Administradores = _sistema.obtenerAdministradores();
            return View();
        }
    }
}

***********************************
Archivo: ClienteController.cs
Carpeta:
C:\Users\Ricio\Desktop\ort\p2\obligatorioVersionFinal\Obligatorio1\WebApplication1\Controll
ers\ClienteController.cs
***********************************
using Dominio;
using Microsoft.AspNetCore.Mvc;
using Dominio.Entidades;
using WebApplication1.Filtros;

namespace WebApplication1.Controllers
{
    [Logueado]
    public class ClienteController : Controller
    {
        private Sistema _sistema = Sistema.Instancia;
        public IActionResult Index()
        {
            ViewBag.Clientes = _sistema.obtenerClientes();
            return View();
        }
        [HttpGet]
        public IActionResult Cargarsaldo()
```

```
        {
            string email = HttpContext.Session.GetString("UserName");
            string password = HttpContext.Session.GetString("password");
            Cliente clienteACargar = _sistema.obtenerClienteByEmailAndPassword(email,
password);
            ViewBag.saldoActual = clienteACargar.Saldo;
            return View();
        }

        [HttpPost]
        public IActionResult Cargarsaldo(double saldo)
        {
            if (saldo > 0 && !double.IsNaN(saldo) && !double.IsNegative(saldo))
            {
                string email = HttpContext.Session.GetString("UserName");
                string password = HttpContext.Session.GetString("password");
                if (!string.IsNullOrEmpty(email) && !string.IsNullOrEmpty(password))
                {
                    Cliente clienteACargar = _sistema.obtenerClienteByEmailAndPassword(email,
password);
                    ViewBag.saldoActual = clienteACargar.Saldo;
                    clienteACargar.SumarSaldo(saldo);
                    HttpContext.Session.SetString("saldo", clienteACargar.Saldo.ToString("N2"));
                    ViewBag.NuevoSaldo = $"Tu nuevo saldo es ${clienteACargar.Saldo}";
                }
            }
            else
            {
                ViewBag.saldoInvalido = $"Tu saldo ingresado: {saldo} debe ser numero y mayor
a 0";
            }
            return View();
        }
    }
}
**********
Archivo: HomeController.cs
Carpeta:
C:\Users\Ricio\Desktop\ort\p2\obligatorioVersionFinal\Obligatorio1\WebApplication1\Controll
ers\HomeController.cs
**********
using Microsoft.AspNetCore.Mvc;

namespace WebApplication1.Controllers
{
    public class HomeController : Controller
    {
        public IActionResult Home()
```

```
        {
            return View();
        }

        public IActionResult IrALogin()
        {
            return RedirectToAction("Ingresar", "Login");
        }

        public IActionResult IrARegistro()
        {
            return RedirectToAction("Registrar", "Registrarse");
        }
    }
}
**********
Archivo: LoginController.cs
Carpeta:
C:\Users\Ricio\Desktop\ort\p2\obligatorioVersionFinal\Obligatorio1\WebApplication1\Controll
ers\LoginController.cs
**********
using Dominio;
using Dominio.Entidades;
using Microsoft.AspNetCore.Mvc;

namespace WebApplication1.Controllers
{
    public class LoginController : Controller
    {
        private Sistema _sistema = Sistema.Instancia;
        private int maximo_intentos = 3;

        [HttpGet]
        public IActionResult Ingresar()
        {
            if (HttpContext.Session.GetInt32("intentosFallidos") == null)
            {
                HttpContext.Session.SetInt32("intentosFallidos", 0);
            }
            return View();
        }

        [HttpPost]
        public IActionResult Ingresar(string email, string password)
        {
            try
            {
                int intentosFallidos = HttpContext.Session.GetInt32("intentosFallidos") ?? 0;
```

```
            if (intentosFallidos >= maximo_intentos)
            {
                ViewBag.Mensaje = "Has superado el límite de intentos permitidos. Por favor,
intenta más tarde.";
                ViewBag.BloquearFormulario = true;
                return View();
            }

            Usuario unS = _sistema.obtenerUsuario(email, password);
            if (unS == null)
            {
                intentosFallidos++;
                HttpContext.Session.SetInt32("intentosFallidos", intentosFallidos);

                if (intentosFallidos >= maximo_intentos)
                {
                    ViewBag.Mensaje = "Has superado el límite de intentos permitidos. Por favor,
intenta más tarde.";
                    ViewBag.BloquearFormulario = true;
                }
                else
                {
                    ViewBag.Mensaje = $"Credenciales no válidas. Te quedan {maximo_intentos
- intentosFallidos} intentos.";
                }
                return View();
            }

            HttpContext.Session.SetInt32("intentosFallidos", 0);

            if (unS.rol() == "Admin")
            {
                HttpContext.Session.SetString("rol", unS.rol());
                HttpContext.Session.SetString("UserName", unS.Email);
                HttpContext.Session.SetString("password", unS.Contrasenia);
                return Redirect("/Subasta/index");
            }

            if (unS.rol() == "Cliente")
            {
                HttpContext.Session.SetString("rol", unS.rol());
                HttpContext.Session.SetString("UserName", unS.Email);
                HttpContext.Session.SetString("password", unS.Contrasenia);
                return Redirect("/Publicacion/index");
            }
        }
        catch (Exception e)
```

```
            {
                ViewBag.Mensaje = e.Message;
            }
            return View();
        }

        public IActionResult Logout()
        {
            HttpContext.Session.Clear();
            return RedirectToAction("Ingresar");
        }
    }
}
**********
Archivo: OfertaController.cs
Carpeta:
C:\Users\Ricio\Desktop\ort\p2\obligatorioVersionFinal\Obligatorio1\WebApplication1\Controll
ers\OfertaController.cs
**********
using Dominio;
using Dominio.Entidades;
using Microsoft.AspNetCore.Http.Timeouts;
using Microsoft.AspNetCore.Mvc;
using System.Globalization;
namespace WebApplication1.Controllers
{
    public class OfertaController : Controller
    {
        private Sistema _sistema = Sistema.Instancia;
        public IActionResult Index(string nombrePublicacion, string mensaje, int tipo)
        {
            ViewBag.mensajeSalida = mensaje;
            ViewBag.tipo = tipo;
                            ViewBag.NombrePublicacion =
_sistema.obtenerPublicacion(nombrePublicacion);
                            var ofertas =
_sistema.OfertasxNombrePublicacion(nombrePublicacion)
                        .OrderByDescending(o => o.Monto)
                        .ToList();
                            ViewBag.Ofertas = ofertas;
                            ViewBag.OfertaMaxima = ofertas.FirstOrDefault();
            string email = HttpContext.Session.GetString("UserName");
            string password = HttpContext.Session.GetString("password");
            Cliente clienteACargar = _sistema.obtenerClienteByEmailAndPassword(email,
password);
            ViewBag.saldoActual = clienteACargar.Saldo;
            return View();
        }
```

```
        [HttpGet]
        public IActionResult puja()
        {
            return View();
        }

        [HttpPost]
        public IActionResult puja(Oferta oferta, string NombrePublicacion)
        {
            Publicacion unaP = _sistema.obtenerPublicacion(NombrePublicacion);
            int tipo;

            if (oferta.Monto > 0 && !double.IsNaN(oferta.Monto) &&
!double.IsNegative(oferta.Monto) && unaP.PrecioPublicacion() < oferta.Monto)
            {
                DateTime today = DateTime.Today;
                _sistema.AgregarOferta(new Oferta(0,
                                        _sistema.obtenerClientesPorId(0),
                                        oferta.Monto,
                                        new DateTime(today.Year, today.Month, today.Day, 00, 0, 0),
                                        NombrePublicacion));

                string Mensaje1 = "¡Oferta realizada con éxito!";
                tipo = 1;
                return RedirectToAction("Index", new { nombrePublicacion = NombrePublicacion,
mensaje = Mensaje1, tipo = 1});
            }
            tipo = 2;
            string Mensaje2 = "Oferta no realizada, por favor verifique el importe.";
            return RedirectToAction("Index", new { nombrePublicacion = NombrePublicacion,
mensaje = Mensaje2 , tipo=2});
        }
    }
}
```

****
Archivo: PublicacionController.cs
Carpeta:
C:\Users\Ricio\Desktop\ort\p2\obligatorioVersionFinal\Obligatorio1\WebApplication1\Controll
ers\PublicacionController.cs
****

```
using Dominio;
using Dominio.Entidades;
using Microsoft.AspNetCore.Mvc;
using WebApplication1.Filtros;

namespace WebApplication1.Controllers
```

```
{
	[Logueado]
	public class PublicacionController : Controller
	{
		private Sistema _sistema = Sistema.Instancia;
		public IActionResult Index(string mensaje)
		{
			ViewBag.mensaje = mensaje;
			ViewBag.Publicacion = _sistema.Publicaciones;
			string email = HttpContext.Session.GetString("UserName");
			string password = HttpContext.Session.GetString("password");
			Cliente clienteACargar =
_sistema.obtenerClienteByEmailAndPassword(email, password);
			ViewBag.Saldoactual = clienteACargar.Saldo;

			return View();
		}
		public IActionResult VerArticulos(string nombrePublicacion)
		{
			ViewBag.publiName =
_sistema.obtenerPublicacion(nombrePublicacion);
			ViewBag.publicacion =
_sistema.ArticulosxNombrePublicacion(nombrePublicacion);
			return View();
		}

		public IActionResult compraOferta(string nombrePublicacion)
		{
			string mensaje = "";
			Publicacion unaP = _sistema.obtenerPublicacion(nombrePublicacion);
			double precioPublicacion = unaP.PrecioPublicacion();
			if (unaP.Tipo() == "Venta")
			{
				string email = HttpContext.Session.GetString("UserName");
				string password = HttpContext.Session.GetString("password");
				Cliente cliente =
_sistema.obtenerClienteByEmailAndPassword(email, password);

				if (cliente.Saldo > precioPublicacion)
				{
					mensaje = "Compra realizada con exito!!";
					cliente.Saldo = cliente.Saldo - precioPublicacion;
					return RedirectToAction("Index", new { mensaje });
				}
				else
				{
					mensaje = "Saldo Insuficiente!!";
					return RedirectToAction("Index", new { mensaje });
```

```
                    }
                }

                return Redirect($"/oferta/Index?nombrePublicacion={unaP.Nombre}");
            }

        }
}
```

****
Archivo: RegistrarseController.cs
Carpeta: C:\Users\Ricio\Desktop\ort\p2\obligatorioVersionFinal\Obligatorio1\WebApplication1\Controllers\RegistrarseController.cs
****

```
using Dominio;
using Dominio.Entidades;
using Microsoft.AspNetCore.Mvc;

namespace WebApplication1.Controllers
{
   public class RegistrarseController : Controller
   {
      private Sistema _sistema = Sistema.Instancia;

      [HttpGet]
      public IActionResult Registrar()
      {
         return View();
      }

      [HttpPost]
      public IActionResult Registrar(string nombre, string apellido, string email, string password, double saldo)
      {
         try
         {
            Cliente nuevoCliente = new Cliente(nombre, apellido, email, password, saldo);
            _sistema.AgregarUsuario(nuevoCliente);
            ViewBag.Mensaje = $"Cliente {nombre} {apellido} registrado exitosamente.";
            return RedirectToAction("Ingresar", "Login");
         }
         catch (Exception ex)
         {
            ViewBag.Error = ex.Message;
            return View();
         }
      }
```

```
    }
}
***********************************
Archivo: SubastaController.cs
Carpeta:
C:\Users\Ricio\Desktop\ort\p2\obligatorioVersionFinal\Obligatorio1\WebApplication1\Controll
ers\SubastaController.cs
***********************************
using Dominio;
using Microsoft.AspNetCore.Mvc;
using WebApplication1.Filtros;

namespace WebApplication1.Controllers
{
    //[Admin]
    public class SubastaController : Controller
    {
        private Sistema _sistema = Sistema.Instancia;
        public IActionResult Index(string mensaje)
        {
                            ViewBag.mensaje = mensaje;
                            ViewBag.Publicacion = _sistema.Publicaciones;
            return View();
        }

        public IActionResult VerArticulos(string nombrePublicacion)
        {
                            ViewBag.publiName =
_sistema.obtenerPublicacion(nombrePublicacion);
                            ViewBag.publicacion =
_sistema.ArticulosxNombrePublicacion(nombrePublicacion);
                            return View();
        }
        public IActionResult cerrarSubasta(string nombreSubasta)
        {
            string mensaje = _sistema.CerrarSubasta(nombreSubasta);
                            ViewBag.publiName = _sistema.obtenerPublicacion(nombreSubasta);
                            ViewBag.publicacion =
_sistema.ArticulosxNombrePublicacion(nombreSubasta);
            //ViewBag.oferente =
_sistema.obteneroferente(_sistema.obtenerPublicacion(nombreSubasta));

                            return RedirectToAction("index", new { mensaje });
                    }
```

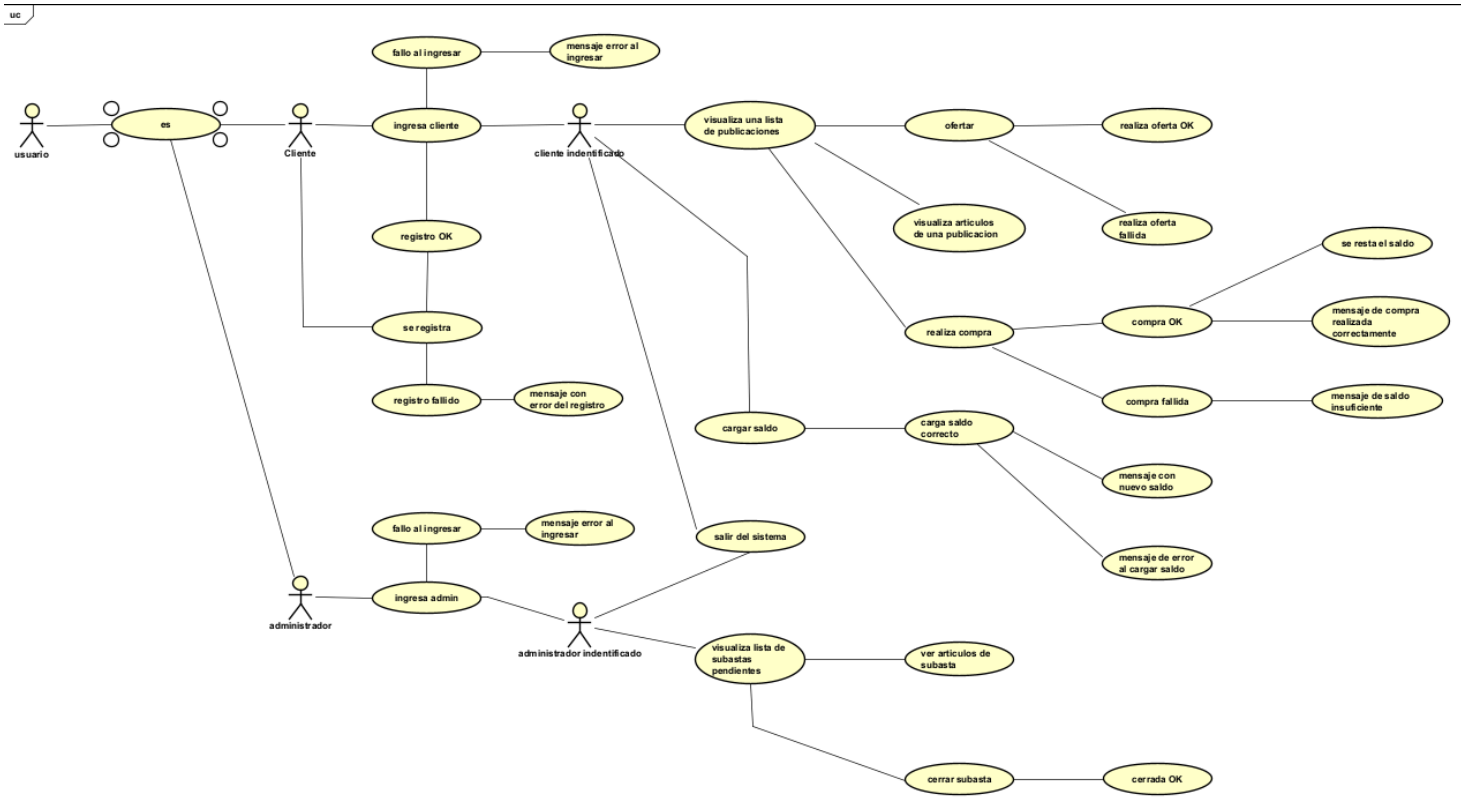
```
        }
}

***********************************
Archivo: Admin.cs
Carpeta:
C:\Users\Ricio\Desktop\ort\p2\obligatorioVersionFinal\Obligatorio1\WebApplication1\Filtros\A
dmin.cs
***********************************
using Microsoft.AspNetCore.Mvc.Filters;
using Microsoft.AspNetCore.Mvc;

namespace WebApplication1.Filtros
{
        public class Admin : Attribute, IAuthorizationFilter
        {
                public void OnAuthorization(AuthorizationFilterContext context)
                {
                        if (context.HttpContext.Session.GetString("rol") == null)
                        {
                                context.Result = new RedirectResult("/Login/Ingresar");
                        }

                        if (context.HttpContext.Session.GetString("rol") != "Admin")
                        {
                                context.Result = new RedirectResult("/Subasta/index");
                        }


                }
        }
}

***********************************
Archivo: Logueado.cs
Carpeta:
C:\Users\Ricio\Desktop\ort\p2\obligatorioVersionFinal\Obligatorio1\WebApplication1\Filtros\L
ogueado.cs
***********************************
using Microsoft.AspNetCore.Mvc.Filters;
using Microsoft.AspNetCore.Mvc;

namespace WebApplication1.Filtros
{
        public class Logueado : Attribute, IAuthorizationFilter
        {
                public void OnAuthorization(AuthorizationFilterContext context)
```

```
                {

                        if (context.HttpContext.Session.GetString("rol") == null)
                        {
                                context.Result = new RedirectResult("/Login/Ingresar");
                        }

                }
        }
}
```

****
Archivo: ErrorViewModel.cs
Carpeta: C:\Users\Ricio\Desktop\ort\p2\obligatorioVersionFinal\Obligatorio1\WebApplication1\Models\ErrorViewModel.cs
****

```
namespace WebApplication1.Models
{
    public class ErrorViewModel
    {
        public string? RequestId { get; set; }

        public bool ShowRequestId => !string.IsNullOrEmpty(RequestId);
    }
}
```

Casos de uso


uc
usuario
es
Cliente
ingresa cliente
fallo al ingresar
mensaje error al ingresar
registro OK
se registra
registro fallido
mensaje con error del registro
cliente indentificado
visualiza una lista de publicaciones
ofertar
realiza oferta OK
realiza oferta fallida
visualiza articulos de una publicacion
realiza compra
compra OK
se resta el saldo
mensaje de compra realizada correctamente
compra fallida
mensaje de saldo insuficiente
cargar saldo
carga saldo correcto
mensaje con nuevo saldo
mensaje de error al cargar saldo
salir del sistema
administrador
ingresa admin
fallo al ingresar
mensaje error al ingresar
administrador indentificado
visualiza lista de subastas pendientes
ver articulos de subasta
cerrar subasta
cerrada OK

Testing

Funcionalidad	Datos ingresados	resultado esperado	resultado obtenido
Ingresar con cliente OK	mauri.sape@mail.com password1	ingreso al sistema	ingreso al sistema
Ingresar con cliente fallido	mauri.sape@mail.com pass	mensaje con contraseña invalida	mensaje con contraseña invalida
Ingresar con adminOK	marta.suarez@mail.com administrator1	ingreso al sistema	ingreso al sistema
Ingresar con admin fallido	marta.suarez@mail.com admi21	mensaje con contraseña invalida	mensaje con contraseña invalida
Registro de un cliente OK	Pepe Prueba pepe@gmail.com pepePasswd1 500	cliente registrado correctamente	cliente registrado correctamente
Registro de un cliente fallido -> contraseña invalida	Pepe Prueba pepe@gmail.com sswd1 54	mensaje con contraseña invalida	mensaje con contraseña invalida
Registro de un cliente fallido -> monto menor a 0	Pepe Prueba pepe@gmail.com sswd1 -78	mensaje con saldo invalido	mensaje con saldo invalido
Cliente -> Ofertar con monto ok	monto	oferta realizada	oferta realizada
Cliente -> Ofertar con monto invalido	monto	mensaje de monto invalido	mensaje de monto invalido
Cliente -> Comprar con saldo ok		compra satifasctoria	compra satifasctoria
Cliente -> Comprar con saldo insuficiente		compra erronea, con mensaje de saldo insuficiente	compra erronea, con mensaje de saldo insuficiente
Cliente -> Cargar saldo ok	saldo	carga correcta	carga correcta
Cliente -> Cargar saldo negativo	saldo	mensaje de saldo invalido	mensaje de saldo invalido

Admin -> Cerrar subastas OK		cambio de estado de subasta a cerrada	cambio de estado de subasta a cerrada
Admin -> Cerrar subastas fallido		no se modifica el estado	no se modifica el estado
Logout		pantalla de inicio	pantalla de inicio

URL AZURE

https://obligatoriomatimauri-hcgrcchma9erfufp.canadacentral-01.azurewebsites.net/