Universidad ORT Uruguay

Facultad de Ingeniería

Escuela de Tecnología

OBLIGATORIO – PROGRAMACIÓN 2

DOCUMENTACION.

Luis Ignacio López Perdomo – 278179

Ana Karelina Fabra – 222487

Grupo: N2D

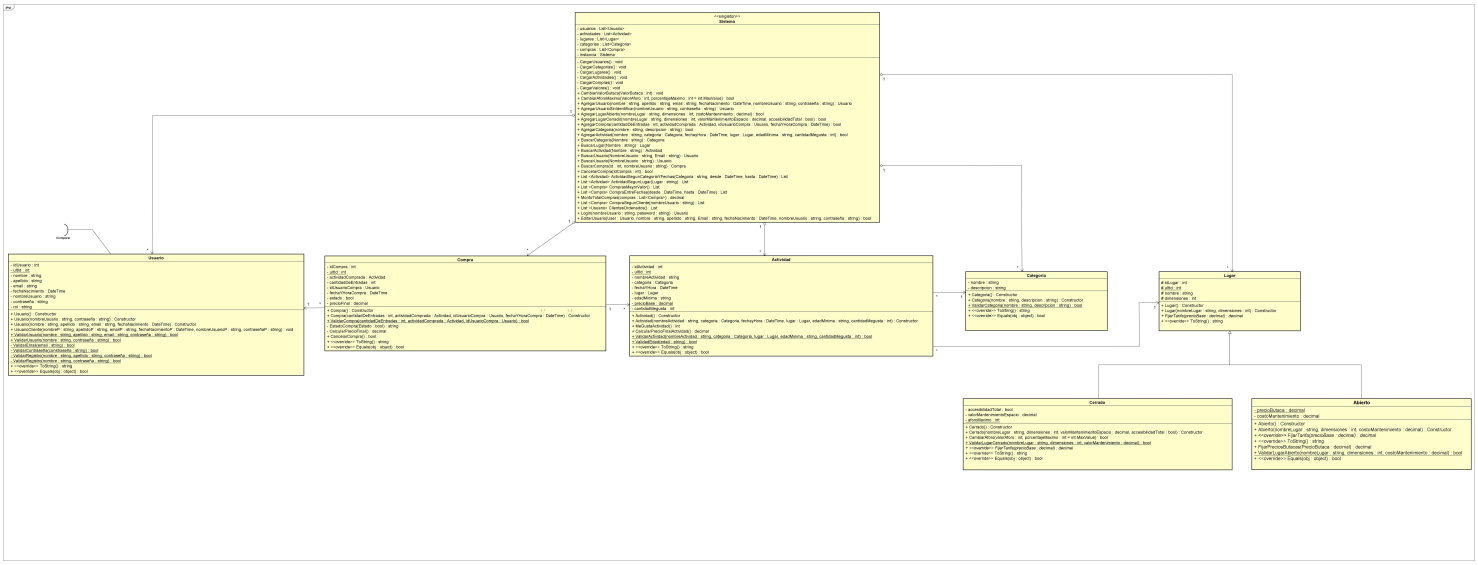
Docente: Luis Dentone

Analista en Tecnologías de la Información

Fecha de entrega del documento: 25-11-2021

Indicie

1. DIAGRAMA DE CLASES



2. TESTING

Usuario Sin Identificar			
Funcionalidad	***Datos Ingresados***	***Resultado Esperado***	***Resultado Obtenido***
Login	Usuarios y/ contraseña vacíos	Permanecer en el Login	Las credenciales son incorrectas o se encuentran vacías. ¡Ingrese los datos correctamente y vuelva a intentarlo!
	Usuarios y/ contraseña Incorrectos	Permanecer en el Login	Las credenciales son incorrectas o se encuentran vacías. ¡Ingrese los datos correctamente y vuelva a intentarlo!
	Usuarios y/ contraseña Correctos	Ingresar a la pagina	Ver el menú adjudicado al rol correspondiente
Registro	Campo nombre vacío o con un largo menor a 2 letras	Error de Registro	¡No se ha podido registrar, ingrese los datos correctamente!
	Campo Apellido vacío o con un largo menor a 2 letras	Error de Registro	¡No se ha podido registrar, ingrese los datos correctamente!
	Contraseña de un largo menor a 6 dígitos, con una mayúscula, una minúscula y un digito (formato correcto).	Error de Registro	¡No se ha podido registrar, ingrese los datos correctamente!
	Contraseña de un largo mayor a 6 dígitos, con el formato correcto, pero sin una mayúscula.	Error de Registro	¡No se ha podido registrar, ingrese los datos correctamente!
	Contraseña de un largo mayor a 6 dígitos, con el formato correcto, pero sin una minúscula.	Error de Registro	¡No se ha podido registrar, ingrese los datos correctamente!

	Contraseña de un largo mayor a 6 dígitos, con el formato correcto, pero sin un Digito	Error de Registro	¡No se ha podido registrar, ingrese los datos correctamente!
	Nombre de usuario repetido	Error de Registro	¡No se ha podido registrar, ingrese los datos correctamente!
	Email repetido	Error de Registro	¡No se ha podido registrar, ingrese los datos correctamente!
	El nombre y apellido tienen un largo mínimo de 2 caracteres. La contraseña tiene un largo mínimo de 6 caracteres y contiene una mayúscula, una minúscula y un dígito. El nombre de usuario y el email son únicos.	Se registro Correctamente	¡Usted se registró correctamente como Cliente!

Usuario Cliente			
Funcionalidad	***Datos Ingresados***	***Resultado Esperado***	***Resultado Obtenido***
Visualización de Actividades Con opción a compra	Click en el botón comprar	Redirección a comprar Actividad para ingresar números de entradas	La compra se ha realizado con éxito
Comprar Entradas de una Actividad	Seleccionar Actividad y Cantidad de Entradas	Compra de Actividad, seleccionado la misma y la cantidad de entradas	La compra se ha realizado con éxito
Ver todas las compras del usuario logueado con Costo final	Ver actividades del usuario logueado	Ver las actividades	Se muestran las Actividades Correctamente
Cancelación de la compra	Ingreso de numeración de la compra incorrecto	No se muestra ninguna compra	No se ha podido encontrar ninguna compra con ese número, ingréselo correctamente y vuelva a intentarlo.
	Ingreso de numeración de la compra correctamente con fecha mayor a 24 hr de efectuada	No se puede efectuar la cancelación	La cancelación solo se puede realizar si está comprendida dentro del plazo de 24 horas antes de efectuarla. Plazo máximo excedido, no se ha podido llevar a cabo la cancelación.
	Ingreso de numeración de la compra correctamente con fecha menor a 24 hr de efectuada	La cancelación se puede realizar	La compra se cancelo correctamente
Dar me gusta a una Actividad	Ver todas las actividades y poder darle like sin un límite determinado	Que el contador de likes suba	El contador de likes sube al hacer click en el botón correspondiente

Usuario Operador			
Funcionalidad	***Datos Ingresados***	***Resultado Esperado***	***Resultado Obtenido***
Compra entre fechas	Ingreso de fechas vacías	No mostrar ninguna tabla	No se han podido encontrar compras entre el rango de fechas dado
	Ingreso de fechas que no tienen compras realizadas	No mostrar ninguna tabla	No se han podido encontrar compras entre el rango de fechas dado
	Ingreso de fechas correctas	Mostrar la tabla de las compras de Actividades realizadas entre las fechas comprendidas con el monto total recaudado	Se muestran las compras de las actividades con el monto total recaudado
Lista de Usuarios Ordenados		Muestra la tabla con los usuarios ordenados ascendentemente. Por apellido y en caso de que el apellido sea el mismo, usa el mismo criterio con los nombres	Los usuarios se muestran ordenados correctamente
Lista de Actividades por Lugar	Seleccionar el lugar	Tabla de las actividades que se realizan en ese lugar	Se muestra la tabla correctamente
Actividades por Categoría y Fecha	Seleccionar una categoría y un rango de fechas	Muestra la tabla de las actividades que son de esa categoría en el rango de fechas establecido	Se muestra la tabla correctamente
	Categoría seleccionada sin fechas ingresadas	No mostrar ninguna tabla	No se encontró ninguna actividad en el rango de fechas dados
Listas de compras de Mayor precio		Muestra las compras de mayor valor, en caso de tener el mismo valor, muestra todas las compras con ese importe.	Se muestra correctamente la tabla

Vista de Roles				
Rol	Menús que deberían ver	Vista Correcta	Credencial	Acceso a las vistas de los controladores por url
Usuario Sin Loguear	Login.	Si		NO
Usuario sin Identificar	Lista de Actividades, Registro, Salir.	SI	Lucho / Lucho1234	NO
Usuario Cliente	Compra de Actividades, Compras de Entradas, Compras Personales, Cancelar Compra, Like de Actividades, Salir.	SI	Josefina / Josefina1234	NO
Usuario Operador	Compra entre fechas, Lista de Usuarios, Buscar Actividad por Lugar, Actividad por Categoría y Fechas, Compras Mayores, Salir.	SI	Santiago / Santiago1234	NO

Polimorfismo						
Precio de la Actividad	Tipo de Lugar	Condición	Precio	Cantidad de Entradas	Condición de Entradas	Precio Final Actividad
150	Abierto	Si la dimensión es mayor a 1 se recarga un 10 %	150 + 15 = 165	4	Si compran 5 o menos entradas, el precio es el mismo	660
150	Abierto	Si la dimensión es 1 permanece igual	150	6	Si compran más de 5 entradas se descuenta un 5%	900 - 45 = 855
150	Cerrado	Si el Aforo es menor a 50 se recarga un 30 %	150 + 45 = 195	6	Si compran más de 5 entradas se descuenta un 5%	1170 - 58,5 = 1111,5
150	Cerrado	Si el Aforo se encuentra entre 50-70 se recarga un 15 %	150 + 22,5 = 172,5	6	Si compran más de 5 entradas se descuenta un 5%	1035 - 51,75 = 983,25
150	Cerrado	Si el Aforo es mayor a 70 permanece igual	150	4	Si compran 5 o menos entradas, el precio es el mismo	600

Clases de Dominio

Clase Abierto

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

namespace Dominio
{
    public class Abierto : Lugar
    {
        private static decimal precioButaca=0;
        private decimal costoMantenimiento=0;
        public decimal PrecioButaca
        {
            get { return precioButaca;}
        }
        public Abierto() { }
        public Abierto(string nombreLugar, int dimensiones, decimal
        costoMantenimiento) : base(nombreLugar, dimensiones)
        {
            this.idLugar = ultId;
            this.costoMantenimiento = costoMantenimiento;
        }
        public override decimal FijarTarifa(decimal precioBase) //Metodo Polimorfico
        {
            decimal recargo = 1;
                if (this.dimensiones > 1)
                {
                recargo += 0.10m;
                }
            return recargo * precioBase;
        }
        public decimal FijarPreciosButacas(decimal PrecioButaca) {
            return precioButaca = PrecioButaca;
        }
        public override string ToString()
        {
            string Datos;
            Datos = base.ToString();
            Datos += "Tipo de Lugar: Abierto \n";
            Datos += "Precio de la Butaca : " + precioButaca + "\n";
            return Datos;
        }
        public static bool ValidarLugarAbierto(string nombreLugar, int dimensiones,
        decimal costoMantenimiento)
        {
         return nombreLugar.Trim() != "" && dimensiones > 0 && costoMantenimiento > 0;
        }
        public override bool Equals(object obj)
        {
            Abierto unLA = obj as Abierto;
            return unLA != null && idLugar == unLA.IdLugar;
        }
    }
}
```

Clase Cerrado

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

namespace Dominio
{
    public class Cerrado : Lugar
    {
        private bool accesibilidadTotal;
        private decimal valorMantenimientoEspacio=0;
        static int aforoMaximo;

        public bool AccesibilidadTotal
        {
            get { return accesibilidadTotal;}
        }
        public decimal ValorMantenimientoEspacio
        {
            get { return valorMantenimientoEspacio;}
        }
        public static int AforoMaximo
        {
            get { return aforoMaximo;}
        }

        public Cerrado() { }
        public Cerrado(string nombreLugar, int dimensiones, decimal
        valorMantenimientoEspacio,bool accesibilidadTotal) : base(nombreLugar,
        dimensiones)
        {
            this.idLugar = ultId;
            this.accesibilidadTotal = accesibilidadTotal;
            this.valorMantenimientoEspacio = valorMantenimientoEspacio;
        }
        public override decimal FijarTarifa(decimal precioBase) //Metodo Polimorfico
        {
            decimal precioTotal = precioBase;
                if (aforoMaximo < 50)
                {
                precioTotal *= 1.30m;
                }

                if (aforoMaximo >= 50 && aforoMaximo <= 70)
                {
                precioTotal *= 1.15m;
                }
            return precioTotal;
        }
        public override string ToString()
        {
            string Datos;
                Datos =  base.ToString();
                Datos += "Tipo de Lugar: Cerrado \n";
                Datos += "Valor de Mantenimiento : " + valorMantenimientoEspacio +
"\n";
            return Datos;
        }
        public bool CambiarAforo(int valorAforo, int porcentajeMaximo= int.MaxValue)
```

```
        {
            bool estadoDeIngreso = false;
                if (valorAforo <= porcentajeMaximo) {
                    aforoMaximo = valorAforo;
                    estadoDeIngreso = true;
            }

            return estadoDeIngreso;
        }
        public int mostrarAforoMaximo()
        {
            return aforoMaximo;
        }
        public static bool ValidarLugarCerrado(string nombreLugar, int dimensiones,
        decimal valorMantenimientoEspacio)
        {
            return nombreLugar.Trim() != "" && dimensiones > 0 &&
                   valorMantenimientoEspacio > 0;
        }
        public override bool Equals(object obj)
        {
            Cerrado unLC = obj as Cerrado;
            return unLC != null && idLugar == unLC.IdLugar;
        }
    }
}
```

Clase Lugar

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

namespace Dominio
{
    public abstract class Lugar
    {
        protected int idLugar;
        protected string nombreLugar="";
        protected int dimensiones = 0;
        protected static int ultId = 0;

        public int IdLugar
        {
            get { return idLugar;}
        }
        public string NombreLugar
        {
            get { return nombreLugar;}
        }
        public int Dimensiones
        {
            get { return dimensiones;}
        }

        public Lugar() { }
        public Lugar(string nombreLugar, int dimensiones) {
            ultId++;
            this.idLugar = ultId;
            this.nombreLugar = nombreLugar;
            this.dimensiones = dimensiones;
        }
        public abstract decimal FijarTarifa(decimal precioBase); //Metodo Polimorfico
        public override string ToString()
        {
            string Datos;
                Datos = "Lugar: " + nombreLugar + "\n";
                Datos += "Dimensiones: " + dimensiones + " km2" + "\n";
            return Datos;
        }

    }
}
```

Clase Categoría

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

namespace Dominio
{
    public class Categoria
    {
        private string nombre;
        private string descripcion;

        public string Nombre
        {
            get { return nombre;}
        }
        public string Descripcion
        {
            get { return descripcion;}
        }

        public Categoria() { }
        public Categoria(string nombre, string descripcion)
        {
            this.nombre = nombre;
            this.descripcion = descripcion;
        }
        public override string ToString()
        {
            return nombre;
        }
        public static bool ValidarCategoria(string nombre, string descripcion)
        {
            return nombre.Trim() != "" && descripcion.Trim() != "";
        }
        public override bool Equals(object obj)
        {
            Categoria unaC = obj as Categoria;
            return unaC != null && nombre == unaC.Nombre;
        }
    }
}
```

Clase Usuario

```
using System;
using System.Collections.Generic;
using System.Diagnostics.CodeAnalysis;
using System.Text;

namespace Dominio
{
    public class Usuario
    {
        private int idUsuario;
        private string nombre="";
        private string apellido="";
        private string email="" ;
        private DateTime fechaNacimiento;
        private string nombreUsuario = "";
        private string contraseña = "";
        private string rol = "";
        static int ultId = 0;

        public int IdUsuario
        {
            get { return idUsuario;}
        }
        public string Nombre
        {
            get { return nombre;}
        }
        public string Apellido
        {
            get { return apellido;}
        }
        public string Email
        {
            get { return email;}
        }
        public DateTime FechaNacimiento
        {
            get { return fechaNacimiento;}
        }
        public string NombreUsuario
        {
            get { return nombreUsuario; }
        }
        public string Contraseña
        {
            get { return contraseña; }
        }
        public string Rol
        {
            get { return rol; }
        }

        public Usuario() { }
        public Usuario(string nombreUsuario, string contraseña) // Constructor usuario
sin identificacion
        {
            this.nombreUsuario = nombreUsuario;
            this.contraseña = contraseña;
```

```
    this.rol = "SinIdentificar";
}
public Usuario(string nombre, string apellido, string email, DateTime
fechaNacimiento,string nombreUsuario, string contraseña,string rol)
{
    ultId++;
    this.idUsuario = ultId;
    this.nombre = nombre;
    this.apellido = apellido;
    this.email = email;
    this.fechaNacimiento = fechaNacimiento;
    this.nombreUsuario = nombreUsuario;
    this.contraseña = contraseña;
    this.rol = rol;
}
public override string ToString()
{
    string Datos;
        Datos = "ID: " + idUsuario + "\n";
        Datos += "Nombre: " + nombre + "\n";
        Datos += "Apellido: " + apellido + "\n";
        Datos += "E-mail: " + email + "\n";
        Datos += "Fecha de Nacimiento:  " +
                 fechaNacimiento.ToShortDateString() + "\n";
    return Datos;
}
public static bool ValidarUsuario(string nombre, string apellido, string
email,string contraseña)
{
    return nombre?.Trim() != null && apellido?.Trim() != null &&
      ValidarEmail(email) && ValidarRegistro(nombre, apellido, contraseña);
}
public static bool ValidarUsuario(string nombre,string contraseña)
{
    return nombre?.Trim() != null && ValidarRegistro(nombre, contraseña);
}
private static bool ValidarEmail(string email)
{
        bool esValido;
        try {
            var DirEmail = new System.Net.Mail.MailAddress(email);
            esValido = DirEmail.Address == email;
        } catch {
            esValido = false;
        }
        return esValido;
}
private static bool ValidarRegistro(string nombre,string apellido,string
contraseña)
{
    bool esValido=true;
    if(nombre?.Length <2 || apellido?.Length <2 || contraseña?.Length < 6)
    {
        esValido = false;
    }
    if (!ValidarContraseña(contraseña))
    {
        esValido = false;
    }
    return esValido;
}
```

```
        private static bool ValidarRegistro(string nombre, string contraseña)
        {
            bool esValido = true;
            if (nombre?.Length < 2 || contraseña.Length < 6)
            {
                esValido = false;
            }
            if (!ValidarContraseña(contraseña))
            {
                esValido = false;
            }
            return esValido;
        }
        private static bool ValidarContraseña(string contraseña)
        {
            bool esValido = false;
            int mayusculas = 0;
            int minusculas = 0;
            int digitos = 0;
            for (int i = 0; i < contraseña?.Length; i++)
            {
                char letra = contraseña[i];
                if (Char.IsDigit(letra))
                {
                    digitos++;
                }
                if (Char.IsLower(letra))
                {
                    minusculas++;
                }
                if (Char.IsUpper(letra))
                {
                    mayusculas++;
                }
            }

            if (mayusculas >= 1 && minusculas >= 1 && digitos >= 1)
            {
                esValido = true;
            }
            return esValido;
        }
        public void UsuarioCliente(string nombreP, string apellidoP, string emailP,
        DateTime fechaNacimientoP, string nombreUsuarioP, string contraseñaP)
        {
            nombre = nombreP;
            apellido = apellidoP;
            email = emailP;
            fechaNacimiento = fechaNacimientoP;
            nombreUsuario = nombreUsuarioP;
            contraseña = contraseñaP;
            rol = "Cliente";
        }//Metodo para modificar un usuario sin identificacion
        public override bool Equals(object obj)
        {
            Usuario unU = obj as Usuario;
            return unU != null && nombreUsuario == unU.NombreUsuario && email ==
                    unU.Email;
        }
    }
}
```

Clase Compra

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

namespace Dominio
{
    public class Compra
    {
        private int idCompra;
        private Actividad actividadComprada = null;
        private int cantidadDeEntradas=0;
        private Usuario idUsuarioCompra=null;
        private DateTime fechaYHoraCompra;
        private bool estado= true;
        private decimal precioFinal =0;
        static int ultId = 0;

        public int IdCompra
        {
            get { return idCompra;}
        }
        public Actividad ActividadComprada
        {
            get { return actividadComprada;}
        }
        public int CantidadDeEntradas
        {
            get { return cantidadDeEntradas;}
        }
        public Usuario IdUsuarioCompra
        {
            get { return idUsuarioCompra;}
        }
        public DateTime FechaYHoraCompra
        {
            get { return fechaYHoraCompra;}
        }
        public bool Estado
        {
            get { return estado;}
        }
        public decimal PrecioFinal
        {
            get { return precioFinal;}
        }

        public Compra() { }
        public Compra(int cantidadDeEntradas, Actividad actividadComprada, Usuario
        idUsuarioCompra, DateTime fechaYHoraCompra)
        {
            ultId++;
            this.idCompra = ultId;
            this.actividadComprada = actividadComprada;
            this.cantidadDeEntradas = cantidadDeEntradas;
            this.idUsuarioCompra = idUsuarioCompra;
            this.fechaYHoraCompra = fechaYHoraCompra;
            this.precioFinal = CalcularPrecioFinal();
        }
```

```
        public static bool ValidarCompra(int cantidadDeEntradas, Actividad
        actividadComprada, Usuario idUsuarioCompra)
        {
            return cantidadDeEntradas > 0 && actividadComprada != null &&
                    idUsuarioCompra != null;
        }
        public override string ToString()
        {
            string Datos;
                Datos = "ID: " + idCompra + "\n";
                Datos += "Actividad: " + actividadComprada.NombreActividad + "\n";
                Datos += "cantidadDeEntradas: " + cantidadDeEntradas + "\n";
                Datos += "Usuario ID: " + idUsuarioCompra.IdUsuario + "\n";
                Datos += "Fecha de Compra:  " + fechaYHoraCompra.ToShortDateString() +
                         "\n";
                Datos += "Estado De Compra: " + EstadoCompra(estado) + "\n";
                Datos += "Precio Final: " + precioFinal + "\n";
            return Datos;
        }
        private string EstadoCompra(bool Estado)
        {
            return Estado == true ? "Activa" : "Cancelada";
        }
        public bool CancelarCompra()
        {
            return estado = false;
        }
        private decimal CalcularPrecioFinal() // Metodo Polimorfico
        {
            decimal precioFinal = actividadComprada.CalcularPrecioFinalActividad() *
                    cantidadDeEntradas;
                if (cantidadDeEntradas > 5)
                {
                    precioFinal *= 0.95m;
                }
                return precioFinal;
        }
        public override bool Equals(object obj)
        {
            Compra unaC = obj as Compra;
            return unaC != null && idCompra == unaC.IdCompra;
        }
    }
}
```

Clase Actividad

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

namespace Dominio
{
    public class Actividad
    {
        private int idActividad;
        private string nombreActividad="";
        private Categoria categoria=null;
        private DateTime fechaYHora;
        private Lugar lugar=null;
        private string edadMinima="";
        private static decimal precioBase=150;
        private int cantidadMegusta=0;
        private static int ultId=0;
        public int IdActividad
        {
            get { return idActividad;}
        }
        public string NombreActividad
        {
            get { return nombreActividad;}
        }
        public Categoria Categoria
        {
            get { return categoria;}
        }
        public DateTime FechaYHora
        {
            get { return fechaYHora;}
        }
        public Lugar Lugar
        {
            get { return lugar;}
        }
        public string EdadMinima
        {
            get { return edadMinima;}
        }
        public decimal PrecioBase
        {
            get { return precioBase;}
        }
        public int CantidadMegusta
        {
            get { return cantidadMegusta;}
        }
        public Actividad() { }
        public Actividad(string nombre, Categoria categoria, DateTime fechaYHora,
        Lugar lugar, string edadMinima,int cantidadMegusta) {
            ultId++;
            this.idActividad = ultId;
            this.nombreActividad = nombre;
            this.categoria = categoria;
            this.fechaYHora = fechaYHora;
            this.lugar = lugar;
```

```
            this.edadMinima = edadMinima;
            this.cantidadMegusta = cantidadMegusta;
        }

        public static bool ValidarActividad(string nombre, Categoria categoria, Lugar
        lugar, string edadMinima, int cantidadMegusta)
        {
            return nombre.Trim() != "" && categoria != null && lugar != null &&
                    edadMinima !="" && precioBase > 0 && cantidadMegusta >= 0;
        }
        public static bool ValidarEdad(string edad)
        {
            return edad.ToLower() == "p" || edad.ToLower() == "c13" || edad.ToLower()
                    == "c16" || edad.ToLower() == "c18";
        }

        public decimal CalcularPrecioFinalActividad() //Metodo Polimorfico
        {
            return lugar.FijarTarifa(precioBase);
        }
        public int MeGustaActividad()
        {
            return cantidadMegusta++;
        }
        public override string ToString()
        {
            string Datos;
            Datos = "Identificador de Actividad: " + idActividad + "\n";
            Datos += "Nombre de la actividad: " + nombreActividad + "\n";
            Datos += "Categoria: " + categoria + "\n";
            Datos += "Fecha y Hora: " + fechaYHora + "\n";
            Datos += "Edad Minima: " + edadMinima + "\n";
            Datos += "Cantidad de Megusta: " + cantidadMegusta + "\n";
            Datos += lugar;
            return Datos;
        }
        public override bool Equals(object obj)
        {
            Actividad unaA = obj as Actividad;
            return unaA != null && idActividad == unaA.IdActividad;
        }
    }
}
```

Clase Sistema

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

namespace Dominio
{
    public class Sistema
    {
        private Sistema() { }
        private static Sistema instancia = null;
        public static Sistema GetInstancia()
        {
            if (instancia == null)
            {
                instancia = new Sistema();
                instancia.CargarValores();
                instancia.CargarUsuarios();
                instancia.CargarLugares();
                instancia.CargarCategorias();
                instancia.CargarActividades();
                instancia.CargarCompras();
            }
            return instancia;
        }

        private List<Usuario> usuarios = new List<Usuario>();
        private List<Actividad> actividades = new List<Actividad>();
        private List<Lugar> lugares = new List<Lugar>();
        private List<Categoria> categorias = new List<Categoria>();
        private List<Compra> compras = new List<Compra>();
        public List<Usuario> Usuarios
        {
            get { return usuarios; }
        }
        public List<Actividad> Actividades
        {
            get { return actividades;}
        }
        public List<Lugar> Lugares
        {
            get { return lugares;}
        }
        public List<Categoria> Categorias
        {
            get { return categorias;}
        }
        public List<Compra> Compras
        {
            get { return compras;}
        }
```

```
private void CargarUsuarios()
{
    //Usuarios Sin Identificacion
    AgregarUsuarioSinIdentificar("Lucho", "Lucho1234");
    AgregarUsuarioSinIdentificar("Gaston", "Gaston1234");
    AgregarUsuarioSinIdentificar("Sol", "Sol1234");
    //Usuarios Cliente
    AgregarUsuario("CFlorencia", "Pereira", "jose6.p@hotmail.com", new
     DateTime(1994, 06, 02), "Josefina6", "Josefina1234", "Cliente"); //Testeo
    Orden Lista
    AgregarUsuario("BAgustina", "Pereira", "jose7.p@hotmail.com", new
    DateTime(1994, 06, 02), "Josefina7", "Josefina1234", "Cliente");  //Testeo
    Orden Lista
    AgregarUsuario("APedro", "Pereira", "jose8.p@hotmail.com", new
    DateTime(1994, 06, 02), "Josefina8", "Josefina1234", "Cliente");
    //Testeo Orden Lista
    AgregarUsuario("Josefina", "Pereira", "jose.p@hotmail.com", new
    DateTime(1994, 06, 02), "Josefina", "Josefina1234", "Cliente");
    AgregarUsuario("Alvaro", "Fernandez", "alvaro.fer25@hotmail.com", new
    DateTime(1986, 10, 25), "Alvaro", "Alvaro1234", "Cliente");
    AgregarUsuario("Ana", "Mendez", "anamendez1995@hotmail.com", new
    DateTime(1995, 03, 12), "Ana", "Ana1234", "Cliente");
    AgregarUsuario("Federico", "Gonzales", "fede_12@hotmail.com", new
    DateTime(1990, 04, 16), "Federico", "Federico1234", "Cliente");
    //Usuario Operador
    AgregarUsuario("Santiago", "Unty", "SantiagoUn@hotmail.com", new
    DateTime(1990, 04, 16), "Santiago", "Santiago1234", "Operador");
    AgregarUsuario("Nicolas", "Soler", "NicoSoler@hotmail.com", new
    DateTime(1990, 04, 16), "Nicolas", "Nicolas1234", "Operador");
    AgregarUsuario("Lucia", "Arrospide", "LuchiArrospide@hotmail.com", new
    DateTime(1990, 04, 16), "Lucia", "Lucia1234", "Operador");
    //Errores de Usuario
    AgregarUsuario(" ", "Ente", "luca.s@hotmail.com", new DateTime(1994, 04,
    03),"Lucas", "Lucas1234", "SinIdentificar");   // Error Nombre Vacio
    AgregarUsuario("cris", "  ", "crito@hotmail.com", new DateTime(1990, 10,
    24), "Cris", "Cris1234", "SinIdentificar");   // Error Apellido Vacio
    AgregarUsuario("lolo", "pons", "crito hotmail.com", new DateTime(1896, 11,
    14), "lolo", "lolo1234", "SinIdentificar");   // Error Email Incorrecto
    AgregarUsuario("Josefina12", "Pereira12", "jose.p@hotmail.com", new
    DateTime(1994, 06, 02), "Josefina244", "Josefina1234", "SinIdentificar");
    // Error Email repetido
    AgregarUsuario("Josefina1", "Pereira1", "jose.p123@hotmail.com", new
    DateTime(1994, 06, 02),"Josefina", "Josefina1234", "SinIdentificar"); //
    Error Nombre de Usuario repetido
    AgregarUsuario("lolo1", "pons", "crito1@hotmail.com", new DateTime(1896,
    11, 14), "lolo1", "lolo1234", "SinIdentificar");   // Error Contraseña sin
    mayuscula
    AgregarUsuario("lolo2", "pons", "crito2@hotmail.com", new DateTime(1896,
    11, 14), "lolo2", "LOLO1234", "SinIdentificar");   // Error Contraseña sin
    minuscula
    AgregarUsuario("lolo3", "pons", "crito3@hotmail.com", new DateTime(1896,
    11, 14), "lolo3", "Lolololololo", "SinIdentificar");   // Error Contraseña
    sin digito
    AgregarUsuario("lolo4", "pons", "crito4@hotmail.com", new DateTime(1896,
    11, 14), "lolo4", "Lol12", "SinIdentificar");   // Error Contraseña 5
    caracteres
    AgregarUsuario("lolo5", "p", "crito5@hotmail.com", new DateTime(1896, 11,
    14), "lolo5", "Lolo1234", "SinIdentificar");  // Error Apellido 1 caracter
    AgregarUsuario("l", "pons", "crito4@hotmail.com", new DateTime(1896, 11,
    14), "lolo4", "lolo1234", "SinIdentificar");   // Error Nombre 1 caracter
}
```

```
        private void CargarCategorias()
        {
            AgregarCategoria("Cine", "Las mejores peliculas");
            AgregarCategoria("Teatro", "Obras imperdibles");
            AgregarCategoria("Concierto", "Shows imperdibles");
            AgregarCategoria("Feria Gastronomica", "Comidas Gourmet de primer nivel");
            AgregarCategoria("Feria Tecnologica", "Las inovaciones mas asombrosas");
            //Errores Categoria
            AgregarCategoria("sky", "    "); //Error descripcion vacia
            AgregarCategoria("    ", "Asombroso");// Error nombre vacio;
        }

        private void CargarLugares()
        {
            // Lugares Abiertos
            AgregarLugarAbierto("Prado", 20, 11000);
            AgregarLugarAbierto("Parque Rodo", 30, 5000);
            AgregarLugarAbierto("Parque Rivera", 40, 12000);
            AgregarLugarAbierto("Parque Roosvelt", 60, 9000);
            AgregarLugarAbierto("Plaza Virgilio", 28, 15000);
            AgregarLugarAbierto("Plaza Cagancha", 1 , 15000);
            AgregarLugarAbierto("Plaza Conaprole", 1, 15000);
            // Errores Lugar Abierto
            AgregarLugarAbierto("  ", 28, 15000); // Error Nombre vacio
            AgregarLugarAbierto("Plaza Virgilio", 0, 9000); //Error dimension 0
            AgregarLugarAbierto("Plaza Virgilio", 0, 0); //Error precioMantenimiento 0
            //Lugares Cerrados
            AgregarLugarCerrado("World Trade Center", 45, 5000, true);
            AgregarLugarCerrado("Movie Center", 24, 3500, true);
            AgregarLugarCerrado("Casino Carrasco", 30, 7000, false);
            AgregarLugarCerrado("Sodre", 50, 5500, false);
            AgregarLugarCerrado("Casino Conrad", 45, 10000, true);
            // Errores Lugar Cerrado
            AgregarLugarCerrado("    ", 45, 10000, true);    // Error Nombre vacio
            AgregarLugarCerrado("Casino Conrad", 0, 10000, true);  // Error dimension
0
            AgregarLugarCerrado("Casino Conrad", 45, 0, true);  // Error Valor
mantenimiento 0
        }

private void CargarActividades()
        {
            AgregarActividad("Plaza Conaprole", BuscarCategoria("Feria Tecnologica"),
DateTime.Now, BuscarLugar("Plaza Conaprole"), "P", 12); // Actividad con Lugar Abierto
1km
            AgregarActividad("Plaza Cagancha", BuscarCategoria("Feria Tecnologica"),
new DateTime(2021, 05, 04, 12, 25, 00), BuscarLugar("Plaza Cagancha"), "P", 12);   //
Actividad con Lugar Abierto 1km
            AgregarActividad("Prado", BuscarCategoria("Concierto"), new DateTime(2021,
09, 06, 17, 25, 00), BuscarLugar("Prado"), "P", 50);
            AgregarActividad("Prado1", BuscarCategoria("Concierto"), new
DateTime(2021, 09, 06, 17, 25, 00), BuscarLugar("Prado"), "P", 50);
            AgregarActividad("Prado2", BuscarCategoria("Concierto"), new
DateTime(2021, 09, 06, 17, 25, 00), BuscarLugar("Prado"), "P", 50);
            AgregarActividad("Parque Rodo", BuscarCategoria("Feria Gastronomica"), new
DateTime(2021, 05, 15, 16, 30, 15), BuscarLugar("Parque Rodo"), "C13", 26);
            AgregarActividad("Parque Rodo1", BuscarCategoria("Feria Gastronomica"),
new DateTime(2021, 05, 15, 16, 30, 15), BuscarLugar("Parque Rodo"), "C13", 26);
            AgregarActividad("Parque Rivera", BuscarCategoria("Concierto"), new
DateTime(2021, 09, 06, 13, 14, 15), BuscarLugar("Parque Rivera"), "C16", 35);
```

```
        AgregarActividad("Parque Roosvelt", BuscarCategoria("Feria Gastronomica"),
        new DateTime(2021, 07, 06, 14, 27, 03), BuscarLugar("Parque Roosvelt"),
        "P", 58);
        AgregarActividad("Plaza Virgilio", BuscarCategoria("Feria Gastronomica"),
        new DateTime(2021, 11, 25, 17, 25, 36), BuscarLugar("Plaza Virgilio"),
        "C18", 165);
        AgregarActividad("World Trade Center", BuscarCategoria("Feria
        Tecnologica"), new DateTime(2021, 09, 06, 20, 39, 54), BuscarLugar("World
        Trade Center"), "C16", 45);
        AgregarActividad("Movie Center", BuscarCategoria("Cine"), new
        DateTime(2021, 09, 06, 11, 25, 17), BuscarLugar("Movie Center"), "P", 83);
        AgregarActividad("Casino Carrasco", BuscarCategoria("Teatro"), new
        DateTime(2021, 09, 06, 21, 45, 36), BuscarLugar("Casino Carrasco"), "C18",
        36);
        AgregarActividad("Sodre", BuscarCategoria("Concierto"), new DateTime(2021,
        09, 06, 18, 56, 14), BuscarLugar("Sodre"), "P", 300);
        AgregarActividad("Casino Conrad", BuscarCategoria("Feria Gastronomica"),
        new DateTime(2021, 01, 02, 12, 25, 22), BuscarLugar("Casino Conrad"),
        "C18", 235);
        //Errores Actividades
          AgregarActividad("Prado", BuscarCategoria("Concierto"), new
          DateTime(2021, 09, 06), BuscarLugar("Prado"), "P", 50); // Error
          Actividad Repetida
        AgregarActividad("", BuscarCategoria("Concierto"), new DateTime(2021, 09,
        06), BuscarLugar("Prado"), "P", 51); //Error Actividad Nombre Vacio
        AgregarActividad("Prado", BuscarCategoria(" "), new DateTime(2021, 09,
        06), BuscarLugar("Prado"), "P", 52);//Error Actividad Categoria null
        AgregarActividad("Prado1", BuscarCategoria("Concierto"), new
        DateTime(2021, 09, 06), BuscarLugar(" "), "P", 53);//Error Actividad Lugar
        null
        AgregarActividad("Prado2", BuscarCategoria("Concierto"), new
        DateTime(2021, 09, 06), BuscarLugar("Prado"), "As", 54);//Error Actividad
        edadMinima Incorrecta
        AgregarActividad("Prado3", BuscarCategoria("Concierto"), new
        DateTime(2021, 09, 06), BuscarLugar("Prado"), " ", 55);//Error Actividad
        edadMinima Vacia
    }

    private void CargarCompras()
    {
         AgregarCompra(2, BuscarActividad("Prado"), BuscarUsuario("Josefina",
        "jose.p@hotmail.com"),new DateTime(2021, 09, 06,16,30,00));
        AgregarCompra(5, BuscarActividad("Parque Rodo"), BuscarUsuario("Alvaro",
        "alvaro.fer25@hotmail.com"), new DateTime(2021, 03, 05,17,25,00));
        AgregarCompra(3, BuscarActividad("Parque Rivera"), BuscarUsuario("Ana",
        "anamendez1995@hotmail.com"), new DateTime(2021, 09, 24,08,14,25));
        AgregarCompra(1, BuscarActividad("Parque Roosvelt"),
        BuscarUsuario("Federico", "fede_12@hotmail.com"), new DateTime(2021, 02,
        19,20,36,22));
        AgregarCompra(4, BuscarActividad("World Trade Center"),
        BuscarUsuario("Alvaro", "alvaro.fer25@hotmail.com"), new DateTime(2021,
        01, 04,21,45,00));
        AgregarCompra(6, BuscarActividad("Movie Center"),
        BuscarUsuario("Josefina", "jose.p@hotmail.com"), new DateTime(2021, 09,
        05,13,25,36));

        AgregarCompra(7, BuscarActividad("Casino Conrad"), BuscarUsuario("Ana",
        "anamendez1995@hotmail.com"), new DateTime(2021, 07, 30, 23 ,00, 00));
        AgregarCompra(7, BuscarActividad("Casino Conrad"), BuscarUsuario("Ana",
        "anamendez1995@hotmail.com"), new DateTime(2021, 10, 30, 23, 56, 22));
```

```
            AgregarCompra(7, BuscarActividad("Casino Conrad"), BuscarUsuario("Ana",
            "anamendez1995@hotmail.com"), new DateTime(2021, 10, 31, 23, 00, 22));
            //Errores Compra
                        AgregarCompra(2, BuscarActividad("Prado12"),
              BuscarUsuario("Josefina124", "jose.p@hotmail.com"), new DateTime(2021,
              09, 06)); //Error Actividad vacia o incorrecta
            AgregarCompra(2, BuscarActividad("Prado"), BuscarUsuario("JosefinaASD",
            "jose.p@hotmail.com"), new DateTime(2021, 09, 06)); //Error Usuario vacio
            o incorrecto
            AgregarCompra(0, BuscarActividad("Prado"), BuscarUsuario("Josefina123",
            "jose.p@hotmail.com"), new DateTime(2021, 09, 06)); //Error 0 cantidad de
            entradas
        }

private void CargarValores()
        {
            CambiarValorButaca(150);
            CambiarAforoMaximo(49);
        }
        public void CambiarValorButaca(int ValorButaca)
        {
            Abierto CambiarPrecio = new Abierto();
            CambiarPrecio.FijarPreciosButacas(ValorButaca);
        }
        public bool CambiarAforoMaximo(int valorAforo, int PorcentajeMaximo =
       int.MaxValue)
        {
           Cerrado CambiarAforo = new Cerrado();
           return CambiarAforo.CambiarAforo(valorAforo, PorcentajeMaximo);
        }
        public Usuario AgregarUsuario(string nombre, string apellido, string email,
       DateTime fechaNacimiento,string nombreUsuario, string contraseña,string rol)
        {
            Usuario User = null;
                if (Usuario.ValidarUsuario(nombre, apellido, email, contraseña) &&
       BuscarUsuario(nombreUsuario, email) == null)
                {
                    User= new Usuario(nombre, apellido, email, fechaNacimiento,
       nombreUsuario, contraseña, rol);
                     usuarios.Add(User);
                }
            return User;
        }
        public Usuario AgregarUsuarioSinIdentificar(string nombreUsuario, string
       contraseña)
        {
            Usuario User = null;
            if (Usuario.ValidarUsuario(nombreUsuario, contraseña) &&
       BuscarUsuario(nombreUsuario) == null)
            {
                User = new Usuario(nombreUsuario, contraseña);
                usuarios.Add(User);
            }
            return User;
        }
        public bool AgregarLugarAbierto(string nombreLugar, int dimensiones,decimal
       costoMantenimiento)
        {
            bool SeAgrego = false;
                if (Abierto.ValidarLugarAbierto(nombreLugar, dimensiones,
       costoMantenimiento) && BuscarLugar(nombreLugar)==null ) {
```

```
            Abierto Abierto = new Abierto(nombreLugar, dimensiones,
costoMantenimiento);
            lugares.Add(Abierto);
            SeAgrego = true;
        }
    return SeAgrego;
 }
 public bool AgregarLugarCerrado(string nombreLugar, int dimensiones, decimal
valorMantenimientoEspacio, bool accesibilidadTotal)
 {
    bool SeAgrego = false;
        if (Cerrado.ValidarLugarCerrado(nombreLugar, dimensiones,
valorMantenimientoEspacio) && BuscarLugar(nombreLugar)==null) {
            Cerrado Cerrado = new Cerrado(nombreLugar, dimensiones,
valorMantenimientoEspacio, accesibilidadTotal);
            lugares.Add(Cerrado);
            SeAgrego = true;
        }
    return SeAgrego;
 }
 public bool AgregarCompra(int cantidadDeEntradas, Actividad actividadComprada,
Usuario idUsuarioCompra, DateTime fechaYHoraCompra)
 {
    bool SeAgrego = false;
        if (Compra.ValidarCompra(cantidadDeEntradas, actividadComprada,
idUsuarioCompra) ) {
            Compra nuevaCompra = new Compra(cantidadDeEntradas,
actividadComprada, idUsuarioCompra,fechaYHoraCompra);
            compras.Add(nuevaCompra);
            SeAgrego = true;
        }
    return SeAgrego;
 }
 public bool AgregarCategoria(string nombre, string descripcion)
 {
    bool SeAgrego = false;
        if (Categoria.ValidarCategoria(nombre, descripcion) &&
BuscarCategoria(nombre)==null) {
            Categoria Cat = new Categoria(nombre, descripcion);
            categorias.Add(Cat);
            SeAgrego = true;
        }
    return SeAgrego;
 }
 public bool AgregarActividad(string nombre, Categoria categoria, DateTime
fechaYHora, Lugar lugar, string edadMinima, int cantidadMegusta)
 {
    bool SeAgrego = false;
    if (Actividad.ValidarActividad(nombre, categoria, lugar, edadMinima,
cantidadMegusta) && Actividad.ValidarEdad(edadMinima) &&
BuscarActividad(nombre)==null) {
        Actividad NuevaActividad = new Actividad(nombre, categoria,
fechaYHora, lugar, edadMinima, cantidadMegusta);
        actividades.Add(NuevaActividad);
        SeAgrego = true;
    }
    return SeAgrego;
 }
```

```
public Categoria BuscarCategoria(string Nombre)
        {
            Categoria nuevaCat = null;
            int i = 0;
                while (nuevaCat==null && i < categorias.Count)
                {
                    if (categorias[i].Nombre == Nombre) {
                        nuevaCat = categorias[i];
                    }
                i++;
                }
            return nuevaCat;
        }
        public Lugar BuscarLugar(string Nombre)
        {
            Lugar nuevoLugar = null;
            int i = 0;
                while (nuevoLugar==null && i < lugares.Count) {
                    if (lugares[i].NombreLugar == Nombre) {
                        nuevoLugar = lugares[i];
                    }
                i++;
                }
            return nuevoLugar;
        }
        public Actividad BuscarActividad(string Nombre)
        {
            Actividad nuevaActividad = null;
            int i = 0;
                while (nuevaActividad==null && i < actividades.Count) {
                    if (actividades[i].NombreActividad == Nombre) {
                        nuevaActividad = actividades[i];
                    }
                    i++;
                }
            return nuevaActividad;
        }
        public Usuario BuscarUsuario(string NombreUsuario,string Email)
        {
            Usuario nuevoUsuario = null;
            int i = 0;
                while (nuevoUsuario == null && i < usuarios.Count) {
                    if (usuarios[i].NombreUsuario == NombreUsuario ||
          usuarios[i].Email == Email) {
                        nuevoUsuario = usuarios[i];
                    }
                    i++;
                }
            return nuevoUsuario;
        }
        public Usuario BuscarUsuario(string NombreUsuario)
        {
            Usuario nuevoUsuario = null;
            int i = 0;
            while (nuevoUsuario == null && i < usuarios.Count)
            {
                if (usuarios[i].NombreUsuario == NombreUsuario)
                {
                    nuevoUsuario = usuarios[i];
                }
                i++;
```

```
            }
            return nuevoUsuario;
        }
        public Compra BuscarCompra(int id,string nombreUsuario)
        {
            Compra nuevaCompra = null;
            int i = 0;
                while (nuevaCompra == null && i < compras.Count) {
                    if (compras[i].IdCompra == id && compras[i].Estado==true &&
      compras[i].IdUsuarioCompra.NombreUsuario == nombreUsuario) {
                        nuevaCompra = compras[i];
                    }
                    i++;
                }
            return nuevaCompra;
        }
          public bool CancelarCompra(int IdCompra)
        {
            bool realizado = false;
            foreach(Compra unaC in compras){
                var horas = (DateTime.Now -
                           unaC.ActividadComprada.FechaYHora).TotalHours;

                if(unaC.IdCompra== IdCompra && unaC.Estado==true && horas > 24) {
                    unaC.CancelarCompra();
                    realizado = true;
                }
            }
            return realizado;
        }         public List<Actividad> ActividadSegunCategoriaYFechas(string
Categoria,
      DateTime Desde, DateTime Hasta)
        {
            List<Actividad> ListaAuxiliar = new List<Actividad>();
                foreach (Actividad act in Actividades) {
                    if (act.FechaYHora >= Desde && act.FechaYHora <= Hasta &&
      act.Categoria.Nombre == Categoria) {
                        ListaAuxiliar.Add(act);
                    }
                }
            return ListaAuxiliar;
        }
        public List<Actividad> ActividadSegunLugar(string Lugar)
        {
            List<Actividad> ListaAuxiliar = new List<Actividad>();
            foreach (Actividad act in Actividades)
            {
                if (act.Lugar.NombreLugar == Lugar)
                {
                    ListaAuxiliar.Add(act);
                }
            }
            return ListaAuxiliar;
        }
        public List<Compra> ComprasMayorValor()
        {
            List<Compra> auxCompra = new List<Compra>();
            decimal mayorValor = decimal.MinValue;
            decimal precio = 0m;

            foreach (Compra unaC in Compras)
```

```
        {
            precio = unaC.PrecioFinal;

            if (precio > mayorValor)
            {
                mayorValor = precio;
                auxCompra.Clear();
                auxCompra.Add(unaC);
            }
            else if(precio == mayorValor){
                auxCompra.Add(unaC);
            }
        }
        return auxCompra;
    }
    public List<Compra> CompraEntreFechas(DateTime Desde, DateTime Hasta)
    {
        List<Compra> auxCompra = new List<Compra>();
        foreach (Compra unaC in Compras)
        {
            if (unaC.FechaYHoraCompra >= Desde && unaC.FechaYHoraCompra <= Hasta
&& unaC.Estado==true)
            {
                auxCompra.Add(unaC);
            }
        }
        return auxCompra;
    }
    public decimal MontoTotalCompras(List<Compra> compras)
    {
        decimal total = 0;

            foreach (Compra unaC in compras)
            {
                total += unaC.PrecioFinal;
            }
            return total;
    }
    public List<Compra> CompraSegunCliente(string nombreUsuario)
    {
        List<Compra> auxCompra = new List<Compra>();
        foreach (Compra unaC in Compras)
        {
            if (unaC.IdUsuarioCompra.NombreUsuario == nombreUsuario && unaC.Estado
== true)
            {
                auxCompra.Add(unaC);
            }
        }
        return auxCompra;
    }
    public List<Usuario> ClientesOrdenados()
    {
        List<Usuario> auxCompra = new List<Usuario>();
            foreach (Usuario unUser in usuarios)
            {
                if (unUser.Rol == "Cliente")
                {
                    auxCompra.Add(unUser);
                }
            }
```

```
            auxCompra.Sort(new ComparadorClienteApellido());
            return auxCompra;
        }
        public Usuario Login(string nombreUsuario, string password)
        {
            Usuario nuevoUsuario = null;
            int i = 0;
                while (nuevoUsuario == null && i < usuarios.Count)
                {
                    if (usuarios[i].NombreUsuario == nombreUsuario &&
    usuarios[i].Contraseña == password)
                    {
                        nuevoUsuario = usuarios[i];
                    }
                    i++;
                }
            return nuevoUsuario;
        }
        public bool EditarUsuario(Usuario User,string nombre, string apellido, string
    Email, DateTime fechaNacimiento, string NombreUsuario, string Contraseña)
        {
            bool correcto = false;
                if (Usuario.ValidarUsuario(nombre, apellido, Email, Contraseña) &&
    User!=null && BuscarUsuario(NombreUsuario, Email) == null)
                {
                    User.UsuarioCliente(nombre, apellido, Email, fechaNacimiento,
    NombreUsuario, Contraseña);
                    correcto = true;
                }
            return correcto;
        }
    }
}
```

URL Deploy Somee

https://www.obligatoriop2.somee.com

CONTROLADORES

Actividad Controller

```
using Microsoft.AspNetCore.Mvc;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Http;
using Dominio;

namespace WebObligatorio.Controllers
{
    public class ActividadController : Controller
    {
        Sistema s = Sistema.GetInstancia();
        public IActionResult Index()
        {
            return View();
        }
        public IActionResult ListarActividades()
        {
            if (HttpContext.Session.GetString("LogRol") == "SinIdentificar")
            {
                ViewBag.ListaAct = s.Actividades;
                return View();
            } else
            {
                return RedirectToAction("Login", "Login");
            }
        }

        public IActionResult SegunCatFecha()
        {
            if (HttpContext.Session.GetString("LogRol") == "Operador")
            {
                ViewBag.Categorias = s.Categorias;
                return View();
            } else
            {
                return RedirectToAction("Login", "Login");
            }
        }
        [HttpPost]
               public IActionResult SegunCatFecha(string nombreCat,DateTime
              desde,DateTime hasta)
        {
            ViewBag.Categorias = s.Categorias;
            ViewBag.unaA = s.ActividadSegunCategoriaYFechas(nombreCat, desde, hasta);
            if (ViewBag.unaA.Count == 0)
            {
                ViewBag.Respuesta = "No se encontro una actividad en el rango de
                    fechas dados";
            }
            return View();
        }
        public IActionResult SegunLugar()
        {
            if (HttpContext.Session.GetString("LogRol") == "Operador")
```

```
            {
                ViewBag.unL = s.Lugares;
                return View();
            } else
            {
                return RedirectToAction("Login", "Login");
            }
        }
        [HttpPost]
        public IActionResult SegunLugar(string nombreLugar)
        {
            ViewBag.unL = s.Lugares;
            ViewBag.unaA = s.ActividadSegunLugar(nombreLugar);
            if (ViewBag.unaA.Count == 0)
            {
                ViewBag.Respuesta = "No se encontro una actividad en el Lugar
              mencionado";
            }
            return View();
        }
        public IActionResult ListActCompra()
        {
            if (HttpContext.Session.GetString("LogRol") == "Cliente")
            {
                ViewBag.ListaAct = s.Actividades;
                return View();
            } else
            {
                return RedirectToAction("Login", "Login");
            }
        }
        public IActionResult ListActMeGusta()
        {
            if (HttpContext.Session.GetString("LogRol") == "Cliente")
            {
                ViewBag.ListaActividades = s.Actividades;
                return View();
            }
            else
            {
                return RedirectToAction("Login", "Login");
            }
        }
        public IActionResult MeGustaActividad(string nombreActividad)
        {
            if (HttpContext.Session.GetString("LogRol") == "Cliente")
            {
                Actividad unaA = s.BuscarActividad(nombreActividad);
                unaA.MeGustaActividad();
                return RedirectToAction("ListActMeGusta", "Actividad");
            }else
            {
                return RedirectToAction("Login", "Login");
            }
        }
    }
}
```

Compra Controller

```
using Microsoft.AspNetCore.Mvc;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Http;
using Dominio;

namespace WebObligatorio.Controllers
{
    public class CompraController : Controller
    {
        Sistema s = Sistema.GetInstancia();
        public IActionResult ComprasMayores()
        {
            if (HttpContext.Session.GetString("LogRol") == "Operador")
            {
                ViewBag.unaC = s.ComprasMayorValor();
                return View();
            }
            else
            {
                return RedirectToAction("Login","Login");
            }
        }
        public IActionResult ComprasEntreFechas()
        {
            if (HttpContext.Session.GetString("LogRol") == "Operador")
            {
                return View();
            } else
            {
                return RedirectToAction("Login", "Login");
            }
        }
        [HttpPost]
        public IActionResult ComprasEntreFechas(DateTime desde, DateTime hasta)
        {
            ViewBag.unaC = s.CompraEntreFechas(desde, hasta);
            ViewBag.Total = s.MontoTotalCompras(s.CompraEntreFechas(desde, hasta));
            if(ViewBag.unaC.Count == 0)
            {
                ViewBag.Respuesta = "No se han podido encontrar compras entre el rango
              de fechas dado";
            }
            return View();
        }
        public IActionResult CompraSegunCliente()
        {
            if (HttpContext.Session.GetString("LogRol") == "Cliente")
            {
                string nombreUserLog = HttpContext.Session.GetString("LogNombre");
                ViewBag.CompraCliente = s.CompraSegunCliente(nombreUserLog);
                return View();
            }else
            {
                return RedirectToAction("Login", "Login");
            }
```

```
}
public IActionResult BuscarCompra()
{
    if (HttpContext.Session.GetString("LogRol") == "Cliente")
    {
        return View();
    } else
    {
        return RedirectToAction("Login", "Login");
    }
}
[HttpPost]
public IActionResult BuscarCompra(int id)
{
    string nombreUserLog = HttpContext.Session.GetString("LogNombre");
    ViewBag.unaC = s.BuscarCompra(id,nombreUserLog);
    if(ViewBag.unaC == null)
    {
        ViewBag.Resultado= "No se ah podido encontrar ninguna compra con ese
      numero, ingreselo correctamente y vuelva a intentarlo";
    }
    return View();
}
public IActionResult CancelarCompra(int id)
{
    if (HttpContext.Session.GetString("LogRol") == "Cliente")
    {
        bool cancelarCompra = s.CancelarCompra(id);
        if (cancelarCompra)
        {
            ViewBag.ResultadoOk = "La compra se cancelo correctamente";
        }
        else
        {
            ViewBag.ResultadoError = "La cancelacion solo se puede realizar si
      esta comprendida dentro del plazo de 24 horas antes de la realizacion de
      la Actividad.." +
      "Plazo maximo exedido, no se ah podido llevar a cabo la cancelacion";
        }
        return View();
    } else
    {
        return RedirectToAction("Login", "Login");
    }
}
public IActionResult ComprarActividad()
{
    if (HttpContext.Session.GetString("LogRol") == "Cliente")
    {
        return View();
    } else
    {
        return RedirectToAction("Login","Login");
    }
}
[HttpPost]
public IActionResult ComprarActividad(string NombreAct,int cantidadEntradas)
{
    string nombreUserLog = HttpContext.Session.GetString("LogNombre");
    Actividad unaA = s.BuscarActividad(NombreAct);
    Usuario unUser = s.BuscarUsuario(nombreUserLog);
```

```
            bool unaC = s.AgregarCompra(cantidadEntradas, unaA, unUser, DateTime.Now);
            if (unaC)
            {
                ViewBag.ResultadoOk = "La compra se ah realizado con exito";
            }
            else
            {
                ViewBag.ResultadoError = "La compra no se ah podido realizar,
              verifique si ingreso los datos correctamente";
            }
            return View();
        }
        public IActionResult ComprarEntradas()
        {
            if (HttpContext.Session.GetString("LogRol") == "Cliente")
            {
                ViewBag.Act = s.Actividades;
                ViewBag.User = s.Usuarios;
                return View();
            }else {
                return RedirectToAction("Login", "Login");
            }
        }
        [HttpPost]
        public IActionResult ComprarEntradas(string nombreActividad, int
              cantidadEntradas)
        {
            ViewBag.Act = s.Actividades;
            ViewBag.User = s.Usuarios;
            string nombreUserLog = HttpContext.Session.GetString("LogNombre");
            Actividad unaA = s.BuscarActividad(nombreActividad);
            Usuario unUser = s.BuscarUsuario(nombreUserLog);
            bool unaC = s.AgregarCompra(cantidadEntradas, unaA, unUser, DateTime.Now);
            if (unaC)
            {
                ViewBag.ResultadoOk = "La compra se ah realizado con exito";
            }
            else
            {
                ViewBag.ResultadoError = "La compra no se ah podido realizar";
            }
            return View();
        }

    }
}
```

Login Controller

```
using Microsoft.AspNetCore.Mvc;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using System.Web;
using Microsoft.AspNetCore.Http;
using Dominio;

namespace WebObligatorio.Controllers
{
    public class LoginController : Controller
    {
        Sistema s = Sistema.GetInstancia();
        public IActionResult Login()
        {
            return View();
        }
        [HttpPost]
        public IActionResult Login(string nombreDeUsuario,string password)
        {
            Usuario unU = s.Login(nombreDeUsuario, password);

            if (unU != null)
            {
                HttpContext.Session.SetString("LogNombre", unU.NombreUsuario);
                HttpContext.Session.SetString("LogRol", unU.Rol);
            } else
            {
                ViewBag.Error = "Las credenciales son incorrectas o se encuentran
              vacias.";
                ViewBag.Error1 = "¡Ingrese los datos correctamente y vuelva a
              intentarlo!";
            }

                if(HttpContext.Session.GetString("LogRol") == "SinIdentificar")
                {
                    return RedirectToAction("Bienvenida", "Usuario");
                } else if (HttpContext.Session.GetString("LogRol") == "Cliente")
                {
                    return RedirectToAction("Bienvenida", "Usuario");
                }
                else if(HttpContext.Session.GetString("LogRol") == "Operador")
                {
                    return RedirectToAction("Bienvenida", "Usuario");
                } else
                {
                    return View();
                }
        }
        public IActionResult LogOut()
        {
            HttpContext.Session.Clear();
            return RedirectToAction("Login","Login");
        }
    }
}
```

Usuario Controller

```
using Microsoft.AspNetCore.Mvc;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Http;
using Dominio;

namespace WebObligatorio.Controllers
{
    public class UsuarioController : Controller
    {
        Sistema s = Sistema.GetInstancia();
        public IActionResult Index()
        {
            return View();
        }

        public IActionResult AltaUsuario()
        {
            if (HttpContext.Session.GetString("LogRol") == "SinIdentificar")
            {
                return View(new Usuario());
            } else
            {
               return RedirectToAction("Login", "Login");
            }
        }
        [HttpPost]
        public IActionResult AltaUsuario(string nombre, string apellido, string email,
       DateTime fechaNacimiento,string nombreUsuario, string contraseña)
        {
            string nombreUsuarioLog = HttpContext.Session.GetString("LogNombre");
            Usuario usuarioEditado = s.BuscarUsuario(nombreUsuarioLog);
            bool editado = false;
                 editado = s.EditarUsuario(usuarioEditado, nombre,
                    apellido,email,fechaNacimiento, nombreUsuario, contraseña);

            if (editado == true)
                {
                ViewBag.ResultadoOk = "¡Usted se registro correctamente como
                    Cliente!";
                }
                else
                {
                ViewBag.nombre = nombre;
                ViewBag.apellido = apellido;
                ViewBag.email = email;
                ViewBag.fechaNacimiento = fechaNacimiento.ToString("yyyy-MM-dd");
                ViewBag.nombreUsuario = nombreUsuario;
                ViewBag.ResultadoError = "¡No se ah podido registrar, ingrese los
                    datos correctamente!";
                }
            return View();
        }
        public IActionResult AltaUsuarioSinIdentificar()
        {
            return View();
```

```
        }
        [HttpPost]
        public IActionResult AltaUsuarioSinIdentificar(string nombreUsuario, string
              contraseña)
        {
            Usuario usuarioEditado = s.AgregarUsuarioSinIdentificar(nombreUsuario,
              contraseña);

                if (usuarioEditado != null)
                {
                    ViewBag.ResultadoOk = "¡Usted se registro correctamente!";
                }
                else
                {
                    ViewBag.nombreUsuario = nombreUsuario;
                    ViewBag.ResultadoError = "¡No se ah podido registrar, ingrese los
              datos correctamente!. La contraseña debe tener una minuscula, una
              mayuscula y un digito como minimo.";
                }
            return View();
        }
        public IActionResult ListaUsuario()
        {
            if (HttpContext.Session.GetString("LogRol") == "Operador")
            {
                ViewBag.ListaUsuarios = s.ClientesOrdenados();
                return View();
            } else
            {
                return RedirectToAction("Login", "Login");
            }
        }
        public IActionResult Bienvenida()
        {


            if (HttpContext.Session.GetString("LogRol") == "SinIdentificar" ||
                     HttpContext.Session.GetString("LogRol") == "Cliente" ||
                HttpContext.Session.GetString("LogRol") == "Operador")
            {
                string nombreUsuarioLog = HttpContext.Session.GetString("LogNombre");
                Usuario User = s.BuscarUsuario(nombreUsuarioLog);
                return View(User);
            }
            else
            {
                return RedirectToAction("Login", "Login");
            }
        }

    }
}
```

VISTAS

Actividad

ListActCompra

```
@using Dominio;

@*
    For more information on enabling MVC for empty projects, visit
https://go.microsoft.com/fwlink/?LinkID=397860
*@
@{
}

<h1>Lista de Actividades</h1>

        <table class="table table-dark table-striped">
            <thead>
                <tr>
                    <th scope="col">Id</th>
                    <th scope="col">Nombre</th>
                    <th scope="col">Categoria</th>
                    <th scope="col">Fecha y Hora</th>
                    <th scope="col">Lugar</th>
                    <th scope="col">Edad Minima</th>
                    <th scope="col">Precio</th>
                    <th scope="col">Cantidad de Likes</th>
                </tr>
            </thead>
            <tbody>
                @foreach (Actividad unaA in @ViewBag.ListaAct)
                {
                    <tr>
                        <th scope="row">@unaA.IdActividad</th>
                        <td>@unaA.NombreActividad</td>
                        <td>@unaA.Categoria</td>
                        <td>@unaA.FechaYHora</td>
                        <td>@unaA.Lugar.NombreLugar</td>
                        <td>@unaA.EdadMinima</td>
                        <td>@unaA.CalcularPrecioFinalActividad()</td>
                        <td>@unaA.CantidadMegusta</td>
                        <td> <a
href="/Compra/ComprarActividad/?nombreAct=@unaA.NombreActividad"> Comprar </a>

                    </tr>
                }
            </tbody>
        </table>
```

ListActMegusta

```
@using Dominio;
@*
    For more information on enabling MVC for empty projects, visit
https://go.microsoft.com/fwlink/?LinkID=397860
*@
@{
}

<h1>Lista de Actividades</h1>

@if(ViewBag.ListaActividades != null && ViewBag.ListaActividades.Count !=0) {
<table class="table table-dark table-striped">
    <thead>
        <tr>
            <th scope="col">Id</th>
            <th scope="col">Nombre</th>
            <th scope="col">Categoria</th>
            <th scope="col">Fecha y Hora</th>
            <th scope="col">Lugar</th>
            <th scope="col">Edad Minima</th>
            <th scope="col">Precio</th>
            <th scope="col">Cantidad de Likes</th>
        </tr>
    </thead>
    <tbody>
        @foreach (Actividad unaA in @ViewBag.ListaActividades)
        {
            <tr>
                <th scope="row">@unaA.IdActividad</th>
                <td>@unaA.NombreActividad</td>
                <td>@unaA.Categoria</td>
                <td>@unaA.FechaYHora</td>
                <td>@unaA.Lugar.NombreLugar</td>
                <td>@unaA.EdadMinima</td>
                <td>@unaA.CalcularPrecioFinalActividad()</td>
                <td>@unaA.CantidadMegusta</td>
                <td> <a
href="/Actividad/MeGustaActividad/?nombreActividad=@unaA.NombreActividad"> <img
src="~/img/Megusta.png" width="40"> </a>

            </tr>
        }
    </tbody>
</table>
}
```

ListarActividades

```
@using Dominio;
@*
    For more information on enabling MVC for empty projects, visit
https://go.microsoft.com/fwlink/?LinkID=397860
*@
@{
}

<h1>Lista de Actividades</h1>

        <table class="table table-dark table-striped">
            <thead>
                <tr>
                    <th scope="col">Id</th>
                    <th scope="col">Nombre</th>
                    <th scope="col">Categoria</th>
                    <th scope="col">Fecha y Hora</th>
                    <th scope="col">Lugar</th>
                    <th scope="col">Edad Minima</th>
                    <th scope="col">Precio</th>
                    <th scope="col">Cantidad de Likes</th>
                </tr>
            </thead>
            <tbody>

                @foreach (Actividad unaA in @ViewBag.ListaAct)
                {
                    <tr>
                        <th scope="row">@unaA.IdActividad</th>
                        <td>@unaA.NombreActividad</td>
                        <td>@unaA.Categoria</td>
                        <td>@unaA.FechaYHora</td>
                        <td>@unaA.Lugar.NombreLugar</td>
                        <td>@unaA.EdadMinima</td>
                        <td>@unaA.CalcularPrecioFinalActividad()</td>
                        <td>@unaA.CantidadMegusta</td>
                    </tr>

                }
            </tbody>
        </table>
```

SegunCatFecha

```
@using Dominio;

@*
    For more information on enabling MVC for empty projects, visit
https://go.microsoft.com/fwlink/?LinkID=397860
*@
@{
}

<h1> Lista de Actividades por Categoria y Fecha</h1><br><br>
<div class="listafechas">
    <form method="post" class="formulario">
        <h3> Categoria </h3>
        <label> Seleccione la Categoria</label>
        <select name=nombreCat>
            @foreach (Categoria unaC in ViewBag.Categorias)
            {
                <option> @unaC.Nombre </option>
            }
        </select> <br> <br>
        <h3> Fechas </h3><br>
        <label> Desde</label>
        <input type="datetime-local" name="desde"> <br> <br>
        <label> Hasta </label>
        <input type="datetime-local" name="hasta"> <br> <br>
        <input type="submit" value="Buscar">
    </form>
</div><br><br>

@if (ViewBag.unaA != null && ViewBag.unaA.Count != 0)
{
    <table class="table table-dark table-striped">
        <thead>
            <tr>
                <th scope="col">Id</th>
                <th scope="col">Nombre</th>
                <th scope="col">Categoria</th>
                <th scope="col">Fecha y Hora</th>
                <th scope="col">Lugar</th>
                <th scope="col">Edad Minima</th>
                <th scope="col">Precio</th>
                <th scope="col">Cantidad de Likes</th>
            </tr>
        </thead>
        <tbody>
            @foreach (Actividad unaA in @ViewBag.unaA)
            {
                <tr>
                    <th scope="row">@unaA.IdActividad</th>
                    <td>@unaA.NombreActividad</td>
                    <td>@unaA.Categoria</td>
                    <td>@unaA.FechaYHora</td>
                    <td>@unaA.Lugar.NombreLugar</td>
                    <td>@unaA.EdadMinima</td>
                    <td>@unaA.CalcularPrecioFinalActividad()</td>
                    <td>@unaA.CantidadMegusta</td>
                </tr>
            }
        </tbody>
```

```
    </table>
}

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

SegunLugar

```
@using Dominio;
@*
    For more information on enabling MVC for empty projects, visit
https://go.microsoft.com/fwlink/?LinkID=397860
*@
@{
}
<h1>Lista de Actividades por Lugar</h1>
<br><br>
<form method="post" class="formulario">
    <label> Seleccione el Lugar</label>
    <br>
    <select name=nombreLugar>
        @foreach (Lugar unLugar in ViewBag.UnL)
        {
            <option> @unLugar.NombreLugar </option>
        }
    </select>  <br> <br>
    <input type="submit" value="Buscar">
</form><br><br>

@if(ViewBag.unaA != null && ViewBag.unaA.Count != 0) {
    <table class="table table-dark table-striped">
        <thead>
            <tr>
                <th scope="col">Id</th>
                <th scope="col">Nombre</th>
                <th scope="col">Categoria</th>
                <th scope="col">Fecha y Hora</th>
                <th scope="col">Lugar</th>
                <th scope="col">Edad Minima</th>
                <th scope="col">Precio</th>
                <th scope="col">Cantidad de Likes</th>
            </tr>
        </thead>
        <tbody>
            @foreach (Actividad unaA in @ViewBag.unaA)
            {
                <tr>
                    <th scope="row">@unaA.IdActividad</th>
                    <td>@unaA.NombreActividad</td>
                    <td>@unaA.Categoria</td>
                    <td>@unaA.FechaYHora</td>
                    <td>@unaA.Lugar.NombreLugar</td>
                    <td>@unaA.EdadMinima</td>
                    <td>@unaA.CalcularPrecioFinalActividad()</td>
                    <td>@unaA.CantidadMegusta</td>
                </tr>
            }
        </tbody>
    </table>
}
@if (ViewBag.Respuesta != null)
{
<div class="alert alert-danger" role="alert">
    @ViewBag.Respuesta
</div>
        }
```

Compra

BuscarCompra

```
@using Dominio;
@model Compra
@*
    For more information on enabling MVC for empty projects, visit
https://go.microsoft.com/fwlink/?LinkID=397860
*@
@{
}

<h1> Buscar Compra</h1>
<br>
<br>
<form method="post" class="formulario">
    <label> Ingrese el ID de la Compra</label>
    <br>
    <input type="number" name="id">
    <br>
    <br>
    <input type="submit" value="Buscar">
</form>
<br>
<br>

@if (ViewBag.unaC != null)
{
    <h2> Compra </h2>
    <p>ID Compra: @ViewBag.unaC.IdCompra</p>
    <p>Cantidad de Entradas: @ViewBag.unaC.CantidadDeEntradas</p>
    <p>Fecha y Hora: @ViewBag.unaC.FechaYHoraCompra</p>
    <p>Nombre de Usuario:  @ViewBag.unaC.IdUsuarioCompra.Nombre</p>
    <p>Precio Total: @ViewBag.unaC.PrecioFinal</p>
    <br>

    <a href="/Compra/CancelarCompra/@ViewBag.unaC.IdCompra"> Cancelar Compra </a>

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

CancelarCompra

```
@using Dominio;

@*
    For more information on enabling MVC for empty projects, visit
https://go.microsoft.com/fwlink/?LinkID=397860
*@
@{
}
@*<br><br>*@

<h1>Estado de Cancelacion</h1>

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

@if(@ViewBag.ResultadoError != null)
 {
    <div class="alert alert-danger" role="alert">
        @ViewBag.ResultadoError
    </div>
 }  <br>
    <a href="/Compra/BuscarCompra"> Volver </a>
```

ComprarActividad

```
@using Dominio;

@*
    For more information on enabling MVC for empty projects, visit
https://go.microsoft.com/fwlink/?LinkID=397860
*@
@{
}
<h1> Comprar Actividad</h1>

<form method="post" class="formulario">
    <label> Cantidad de Entradas</label> <br>
    <input type="number" name="cantidadEntradas"><br> <br>
    <input type="submit" value="Comprar">
</form>

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

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

ComprarEntradas

```
@using Dominio;
@*
    For more information on enabling MVC for empty projects, visit
https://go.microsoft.com/fwlink/?LinkID=397860
*@
@{
}

<h1> Compra de Entradas</h1>

<form method="post" class="formulario">
    <label> Seleccione la Actividad</label> <br>
    <select name="nombreActividad">
        @foreach (Actividad unaA in ViewBag.Act)
        {
            <option>@unaA.NombreActividad </option>
        }
    </select><br><br>
    <label> Cantidad de Entradas</label> <br>
    <input type="number" name="cantidadEntradas"><br> <br>
    <input type="submit" value="Comprar">
</form>

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

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

ComprasSegunCliente

```
@using Dominio;
@*
    For more information on enabling MVC for empty projects, visit
https://go.microsoft.com/fwlink/?LinkID=397860
*@
@{
}

<h1> Lista de Compras Cliente</h1>

@if(@ViewBag.CompraCliente !=null && @ViewBag.CompraCliente.Count != 0) {
    <table class="table table-dark table-striped">
        <thead>
             <tr>
                <th scope="col">Id</th>
                <th scope="col">Actividad</th>
                <th scope="col">Fecha de Actividad</th>
                <th scope="col">Cantidad de Entradas</th>
                <th scope="col">Usuario</th>
                <th scope="col">Fecha y Hora</th>
                <th scope="col">Precio</th>
            </tr>
        </thead>
        <tbody>
            @foreach (Compra unaC in @ViewBag.CompraCliente)
            {
                <tr>
                    <th scope="row">@unaC.IdCompra</th>
                    <td>@unaC.ActividadComprada.NombreActividad</td>
                    <td>@unaC.ActividadComprada.FechaYHora</td>
                    <td>@unaC.CantidadDeEntradas</td>
                    <td>@unaC.IdUsuarioCompra.Nombre</td>
                    <td>@unaC.FechaYHoraCompra</td>
                    <td>@unaC.PrecioFinal</td>
                    <td> <a href="/Compra/CancelarCompra/@unaC.IdCompra"> Cancelar
Compra </a> </td>
                </tr>
            }
        </tbody>
    </table>
} else
{
    <p> Usted no ah realizado compras a la fecha </p>
}
```

ComprasEntreFechas

```
@using Dominio;
@*
    For more information on enabling MVC for empty projects, visit
https://go.microsoft.com/fwlink/?LinkID=397860
*@
@{
}

<h1>Lista de Compras Entre Fechas</h1>
<br><br>
<div class="listafechas">
    <form method="post" class="formulario">
        <h3> Seleccione el rango de fecha</h3>
        <label> Desde</label>
        <input type="datetime-local" name="desde">
        <br> <br>
        <label> Hasta </label>
        <input type="datetime-local" name="hasta"><br><br>
        <input type="submit" value="Buscar">
    </form> <br> <br>
</div>
@if (ViewBag.unaC != null && @ViewBag.unaC.Count != 0)
{
    <table class="table table-dark table-striped">
        <thead>
            <tr>
                <th scope="col">Id</th>
                <th scope="col">Actividad</th>
                <th scope="col">Cantidad de Entradas</th>
                <th scope="col">Usuario</th>
                <th scope="col">Fecha y Hora</th>
                <th scope="col">Precio</th>
            </tr>
        </thead>
        <tbody>
            @foreach (Compra unaC in @ViewBag.unaC)
            {
                <tr>
                    <th scope="row">@unaC.IdCompra</th>
                    <td>@unaC.ActividadComprada.NombreActividad</td>
                    <td>@unaC.CantidadDeEntradas</td>
                    <td>@unaC.IdUsuarioCompra.Nombre</td>
                    <td>@unaC.FechaYHoraCompra</td>
                    <td>@unaC.PrecioFinal</td>
                </tr>
            }
        </tbody>
    </table>
    <p>Precio Total Recaudado:@ViewBag.Total</p>
}
else
{
    @if(@ViewBag.Respuesta !=null )
    {
    <div class="alert alert-danger" role="alert">
        @ViewBag.Respuesta
    </div>
```

```
    }
}
```

ComprasMayores

```
@using Dominio;
@*
    For more information on enabling MVC for empty projects, visit
https://go.microsoft.com/fwlink/?LinkID=397860
*@
@{
}

<h1> Lista de Compras de Mayor Precio</h1>

    <table class="table table-dark table-striped">
        <thead>
            <tr>
                <th scope="col">Id</th>
                <th scope="col">Actividad</th>
                <th scope="col">Cantidad de Entradas</th>
                <th scope="col">Usuario</th>
                <th scope="col">Fecha y Hora</th>
                <th scope="col">Precio</th>
            </tr>
        </thead>
        <tbody>
            @foreach (Compra unaC in @ViewBag.unaC)
            {
                <tr>
                    <th scope="row">@unaC.IdCompra</th>
                    <td>@unaC.ActividadComprada.NombreActividad</td>
                    <td>@unaC.CantidadDeEntradas</td>
                    <td>@unaC.IdUsuarioCompra.NombreUsuario</td>
                    <td>@unaC.FechaYHoraCompra</td>
                    <td>@unaC.PrecioFinal</td>
                </tr>
            }
        </tbody>
    </table>
```

Login

```
@using Dominio;

@*
    For more information on enabling MVC for empty projects, visit
https://go.microsoft.com/fwlink/?LinkID=397860
*@
@{
}


<div class="centrar">

    <form method="post" class="text-center">
        <div class="mb-3">
            <label for="nombreUsuario" class="form-label">Nombre de Usuario</label>
            <input type="text" name="nombreDeUsuario" class="form-control"
id="nombreUsuario">
        </div>
        <div class="mb-3">
            <label for="password" class="form-label">Password</label>
            <input type="password" name="password" class="form-control" id="password">
        </div>
        <button type="submit" class="btn btn-danger">Ingresar</button>
        <a href="/Usuario/AltaUsuarioSinIdentificar" class="btn btn-danger"> Registrar
</a>
    </form>

    @if (@ViewBag.Error != null && @ViewBag.Error1 != null)
    {
        <div class="alert alert-danger" role="alert">
            @ViewBag.Error <br>
            @ViewBag.Error1
        </div>
    }

</div>
```

Usuario

Alta Usuario

```
@using Dominio;
@model Usuario
@*
    For more information on enabling MVC for empty projects, visit
https://go.microsoft.com/fwlink/?LinkID=397860
*@
@{
}

<form method="post" class="formulario" id="registro">
    <label for="">Nombre </label>
    <input type="text" name="nombre" value="@ViewBag.nombre"><br>
    <label for="">Apellido</label>
    <input type="text" name="apellido" value="@ViewBag.apellido"><br>
    <label for="">E-Mail </label>
    <input type="text" name="email" value="@ViewBag.email"><br>
    <label for="">Fecha de Nacimiento</label>
    <input type="date" name="fechaNacimiento" value="@ViewBag.fechaNacimiento"
required><br>
    <label for="">Nombre de Usuario </label>
    <input type="text" name="nombreUsuario" value="@ViewBag.nombreUsuario"><br>
    <label for="">Contraseña </label>
    <input type="password" name="contraseña"><br>
    <input type="submit" value="Registrar">
</form>


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

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

AltaUsuarioSinIdentificar

```
@using Dominio;
@model Usuario

@*
    For more information on enabling MVC for empty projects, visit
https://go.microsoft.com/fwlink/?LinkID=397860
*@
@{
}

<form method="post" class="formulario" id="registro">
    <label for="">Nombre de Usuario </label>
    <input type="text" name="nombreUsuario" value="@ViewBag.nombreUsuario"><br>
    <label for="">Contraseña </label>
    <input type="password" name="contraseña"><br>
    <input type="submit" value="Registrar">
    <a href="/Login/Login"> Volver </a>
</form>


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

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

Bienvenida

```
@using Dominio;
@model Usuario
@*
    For more information on enabling MVC for empty projects, visit
https://go.microsoft.com/fwlink/?LinkID=397860
*@
@{
}


<div class="formulario">

    <h1> ¡Bienvenido/a @Model.NombreUsuario ! </h1>

</div>
```

ListaUsuario

```
@using Dominio;
@*
    For more information on enabling MVC for empty projects, visit
https://go.microsoft.com/fwlink/?LinkID=397860
*@
@{
}
<h1> Lista de Usuarios Clientes Ordenados</h1>
<table class="table table-dark table-striped">
    <thead>
        <tr>
            <th scope="col">Id</th>
            <th scope="col">Nombre</th>
            <th scope="col">Usuario</th>
            <th scope="col">Apellido</th>
            <th scope="col">Email</th>
            <th scope="col">FechaNacimiento</th>
        </tr>
    </thead>
    <tbody>
        @foreach (Usuario unU in ViewBag.ListaUsuarios)
        {
        <tr>
            <th scope="row">@unU.IdUsuario</th>
            <td>@unU.Nombre</td>
            <td>@unU.NombreUsuario</td>
            <td>@unU.Apellido</td>
            <td>@unU.Email</td>
            <td>@unU.FechaNacimiento</td>
        </tr>
        }
    </tbody>
</table>
```

Aclaración

Se hizo dos tipos de registros, el primer registro que se encuentra en el login, esta echo solo para registrarse con un usuario y una contraseña sin ningún otro dato, cuenta con las validaciones correspondientes, una vez registrado se puede acceder a la vista de los usuarios sin identificación y al segundo registro que va a tomar ese mismo usuario y lo va a registrar como cliente agregando los datos que demanda ese registro, siguiendo las validaciones pertinentes también.