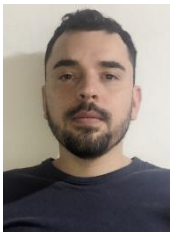
Universidad ORT Uruguay
Facultad de Ingeniería

Documento de Análisis
"Soccer Social Network"

Entregado como requisito obligatorio para el curso de Programación II

Analista en Tecnologías de la Información

Grupo N2F



Rodrigo Cristaldo - 310992

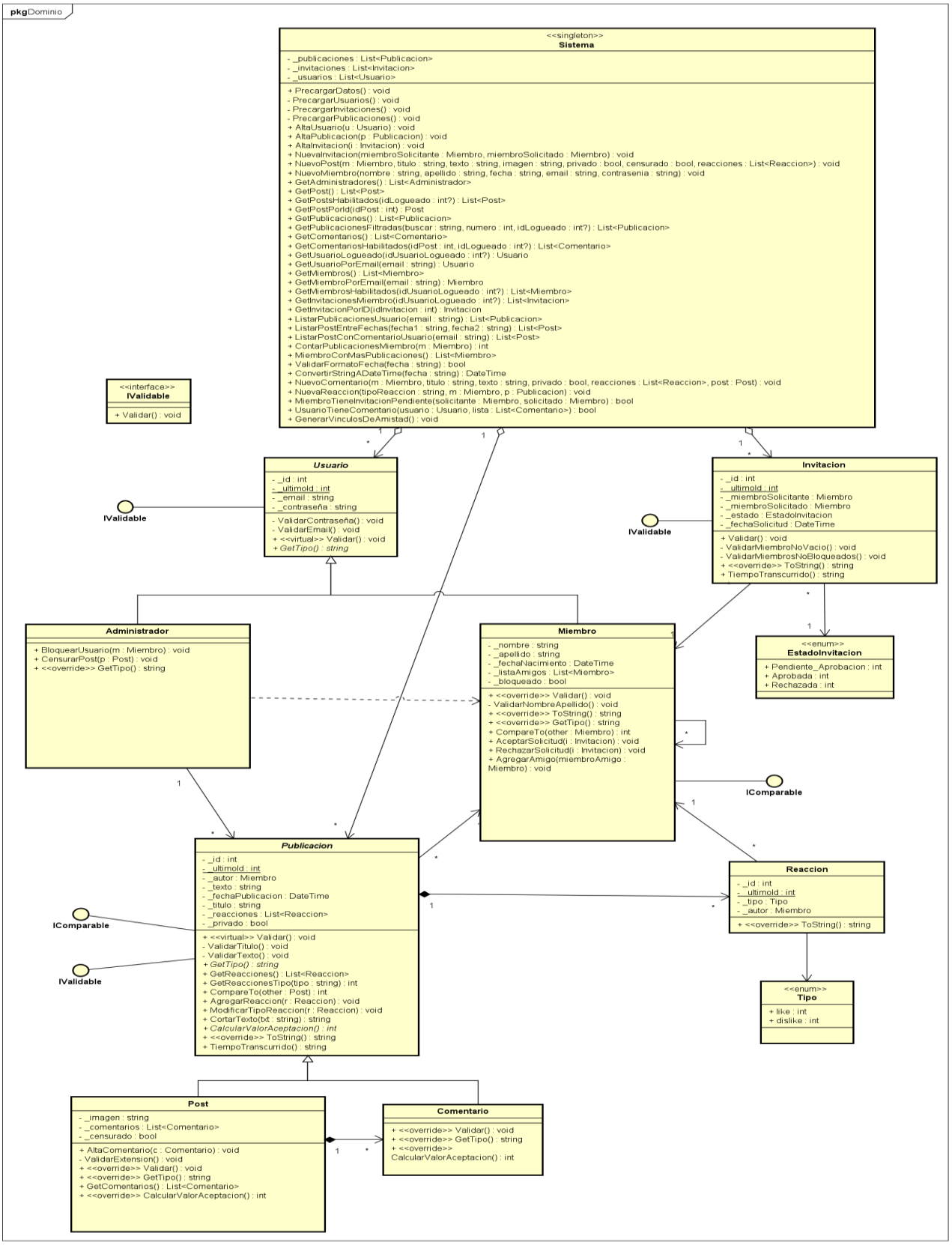
Angel Vaz – 320162

Tutor: Andrés Ureta

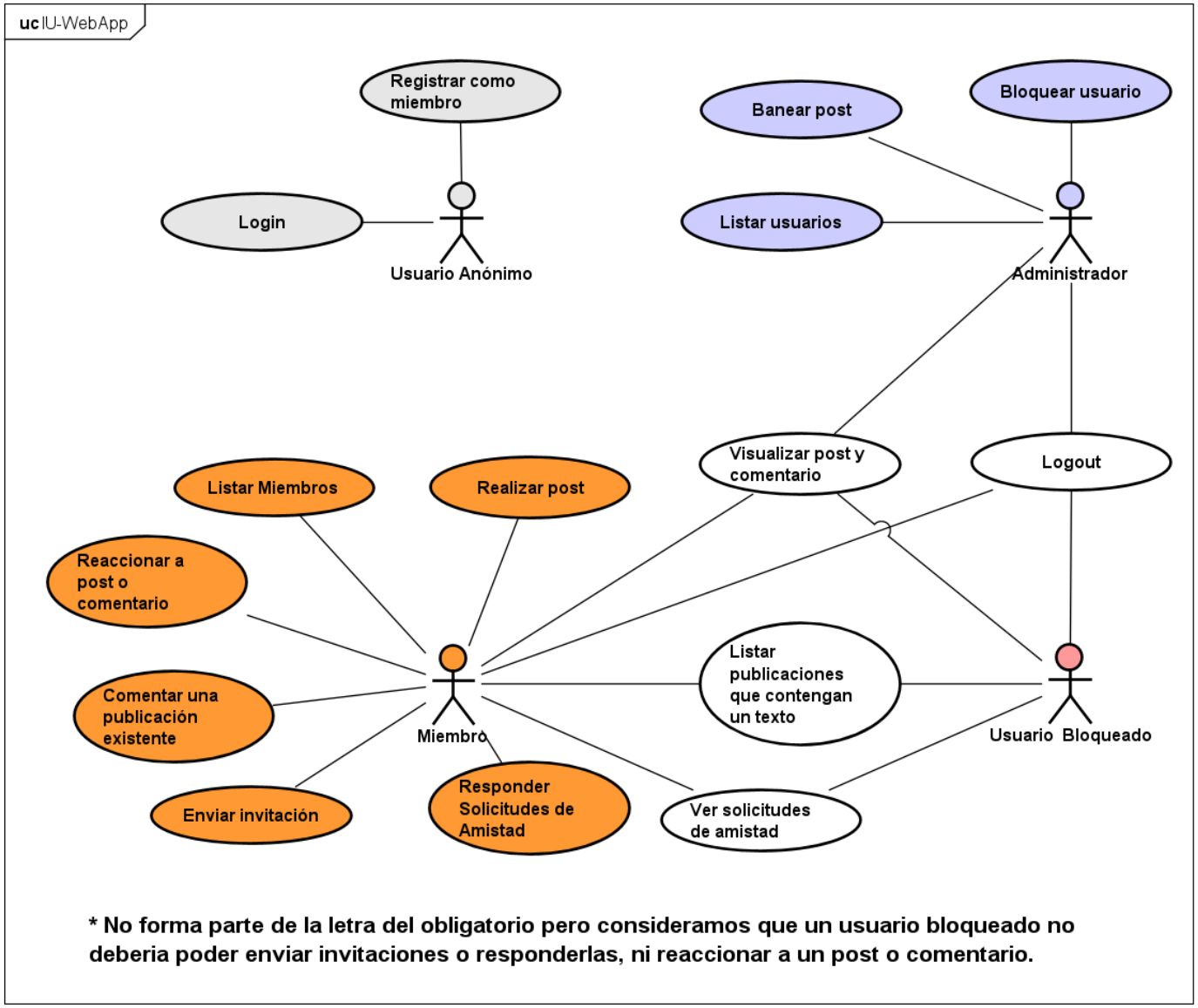
Noviembre 2023

CONTENIDO

1.1 Diagrama de clases definitivo:



1.2 Diagrama de casos de uso:



1.3 Acceso

1.3.1 Link Somee

URL : http://soccersocialnet.somee.com

1.3.2 Credenciales

Usuario	Nombre	Apellido	Fecha Nacimiento	Email	Contraseña
m1	Rodrigo	Cristaldo	13/02/1995	rodrigo@gmail.com	rC1234
m2	Angel	Vaz	03/10/2000	angel@gmail.com	aV1234
m3	Edinson	Cavani	14/02/1987	edi@gmail.com	eC1234
m4	Diego	Forlan	19/05/1979	forlan@gmail.com	dF1234
m5	Sergio	Ramos	30/03/1986	sergio@gmail.com	sR1234
m6	Robert	Lewandowski	21/08/1988	robert@gmail.com	rL1234
m7	Cristiano	Ronaldo	05/02/1985	cr7@gmail.com	cR1234
m8	Neymar	Santos	05/02/1992	neymar@gmail.com	nS1234
m9	Luis	Suarez	24/01/1987	lucho@gmail.com	lS1234
m10	Lionel	Messi	24/06/1987	messi@gmail.com	lm1234
m11	Pato	Sosa	02/06/1987	patososa@gmail.com	pS1234
admin1	-	-	-	admin1@gmail.com	Admin4321

1.4 Tabla de Testing:

1.4.1 Usuario Anónimo:

1.4.1.1 F01: Funcionalidad "Registrarse como miembro":

ID	Escenario de test	Pasos	Datos utilizados	Resultado esperado	Resultado obtenido	Estado (F=Falla, P=pasa)
F01-T01	Vista de la pestaña "Registrarse".	1-Ingreso a la página. 2-Selección de opción "Registrarse".	-	Despliegue de vista correspondiente al registro como miembro.	Despliegue correcto de la vista correspondiente al registro como miembro.	P
F01-T02	Vista de la pestaña "Registrarse".	1-Ingreso a la página. 2-Selección de opción "Registrarse". 3- Llenar los campos correspondientes.	1-Nombre. 2-Apellido. 3-Fecha de Nacimiento. 4-Email. 5-Contraseña 6-Checkbox.	Registro exitoso como miembro por parte de un usuario anónimo. Mensaje correspondiente.	Registro exitoso como miembro por parte de un usuario anónimo. Mensaje "Se ha registrado correctamente" .	P
F01-T03	Vista de la pestaña "Registrarse".	1-Ingreso a la página. 2-Selección de opción "Registrarse". 3- Llenar los campos correspondientes.	1-Nombre. 2-Apellido. 3-Fecha de Nacimiento. 4-Email. 5-Contraseña. 6-Checkbox.	Al ingresar algunos de los campos vacíos no se registra el usuario.	Al no ingresar algunos de los campos vacíos no se registra el usuario.	P

1.4.1.2 F02: Funcionalidad "Login":

ID	Escenario de test	Pasos	Datos utilizados	Resultado esperado	Resultado obtenido	Estado (F=Falla, P=pasa)
F01-T01	Vista de la pestaña "Iniciar Sesión".	1-Ingreso a la página. 2-Selección de opción "Iniciar Sesión".	-	Despliegue exitoso de la vista "Iniciar Sesión".	Despliegue exitoso de la vista "Iniciar Sesión".	P
F01-T02	Vista de la pestaña "Iniciar Sesión".	1-Ingreso a la página. 2-Selección de opción "Iniciar Sesión".	1-Email. 2-Contraseña	Mensaje especifico de error si ingresamos un email que no esté en el sistema. No inicia sesión.	Mensaje "El correo electrónico 'X' no está en el sistema". No se inicia sesión.	P

		3- Llenar los campos correspondientes.				
F01-T03	Vista de la pestaña “Iniciar Sesión”.	1-Ingreso a la página. 2-Selección de opción “Iniciar Sesión” 3- Llenar los campos correspondientes.	1-Email 2-Contraseña	Despliegue de la vista de inicio correspondiente a Miembro o Administrador según los datos ingresados.	Despliegue exitoso de la vista de inicio correspondiente a Miembro o Administrador según datos ingresados.	P

1.4.2 Usuario Administrador:

1.4.2.1 F01 Funcionalidad “Bloquear Usuario”:

ID	Escenario de test	Pasos	Datos utilizados	Resultado esperado	Resultado obtenido	Estado (F=Falla, P=pasa)
F01-T01	Vista de la pestaña “Miembros”	1-Ingreso a la página. 2-Selección de opción “Iniciar sesión”. 3- Llenar los campos correspondientes a un administrador. 4- Ingresar a la pestaña Miembros. 5-Presionar botón ‘Bloquear’ correspondiente a un miembro.	1-Email 2-Contraseña	Al presionar el botón de Bloquear correspondiente a un miembro, se despliega una nueva vista para confirmar la elección.	Al presionar el botón de Bloquear correspondiente a un miembro, se despliega exitosamente una nueva vista para confirmar la elección.	P
F01-T02	Vista de la pestaña “Miembros”	1-Ingreso a la pagina 2-Selección de opción “Iniciar sesión” 3- Llenar los campos correspondientes a un administrador. 4- Ingresar a la pestaña Miembros. 5-Presionar el botón Bloquear	1-Email 2-Contraseña	Al bloquear un miembro, este no podrá postear ni responder a otros posts.	Al bloquear un miembro, este no podrá postear ni responder a otros posts.	P
F01-T03	Vista de la pestaña “Miembros”	1-Ingreso a la pagina 2-Selección de opción “Iniciar sesión” 3- Llenar los campos correspondientes a un administrador.	1-Email 2-Contraseña	Al bloquear un miembro, en la lista de miembros se actualiza el botón por un texto que contiene la palabra “Bloqueado”.	Al bloquear un miembro, en la lista de miembros se actualiza el botón por un texto que contiene la palabra “Bloqueado”.	P

		4- Ingresar a la pestaña Miembros. 5-Presionar el botón Bloquear				

1.4.2.2 F02 Funcionalidad "Banear Post":

ID	Escenario de test	Pasos	Datos utilizados	Resultado esperado	Resultado obtenido	Estado (F=Falla, P=pasa)
F01-T01	Vista de la pestaña "Publicaciones"	1-Ingreso a la pagina 2-Selección de opción "Iniciar sesión" 3- Llenar los campos correspondientes a un administrador. 4- Ingresar a la pestaña Publicaciones.	1-Email 2-Contraseña	Al Banear un post, este deja de ser visible en la aplicación y solo lo verá el administrador.	Al Banear un post, deja de ser visible en la aplicación y solo lo ve el administrador.	P
F01-T02	Vista de la pestaña "Publicaciones"	1-Ingreso a la pagina 2-Selección de opción "Iniciar sesión" 3- Llenar los campos correspondientes a un administrador. 4- Ingresar a la pestaña Publicaciones.	1-Email 2-Contraseña	Al presionar el botón para banear un post, se despliega una nueva vista pidiendo confirmación.	Al presionar el botón para banear un post, se despliega una nueva vista pidiendo confirmación.	P
F01-T03	Vista de la pestaña "Publicaciones"	1-Ingreso a la pagina 2- Selección de opción "Iniciar Sesión" 3- Llenar los campos correspondientes a un administrador. 4- Ingresar a la pestaña Publicaciones.	1-Email 2-Contraseña	Luego de banear un post, el botón se reemplaza por un texto que indica que el mismo se encuentra baneado.	Luego de banear un post, el botón se reemplaza por un texto que indica que el mismo se encuentra baneado.	P

1.4.3 Usuario Miembro:

1.4.3.1 F01 Funcionalidad "Realizar post":

ID	Escenario de test	Pasos	Datos utilizados	Resultado esperado	Resultado obtenido	Estado (F=Falla, P=pasa)
F01-T01	Vista de la pestaña "Publicar"	1-Ingreso a la pagina 2-Selección de opción "Iniciar Sesión" 3- Llenar los campos correspondientes a un miembro 4- Ingresar a la pestaña Publicar 5-Llenar datos correspondientes	1-Email 2-Contraseña 3-Titulo 4-Contenido 5-Imagen 6-Checkbox	Creación de un nuevo post del miembro logueado con los datos suministrados.	Creación de un nuevo post del miembro logueado con los datos suministrados.	P
F01-T02	Vista de la pestaña "Publicar"	1-Ingreso a la pagina 2-Selección de opción "Iniciar Sesión" 3- Llenar los campos correspondientes a un miembro 4- Ingresar a la pestaña Publicar 5-Llenar datos correspondientes	1-Email 2-Contenido	Creación de un post Privado o Público según la selección dentro del checkbox.	Creación de un post Privado o Público según la selección dentro del checkbox.	P
F01-T03	Vista de la pestaña "Publicar"	1-Ingreso a la pagina 2-Selección de opción "Iniciar Sesión" 3- Llenar los campos correspondientes a un miembro 4- Ingresar a la pestaña Publicar 5-Llenar datos correspondientes	1-Email 2-Contenido	Mensaje al usuario de "Posteo Realizado!" luego de llenar los campos correctamente y presionar el botón "Postear"	Mensaje al usuario de "Posteo Realizado!" luego de llenar los campos correctamente y presionar el botón "Postear"	P

1.4.3.2 F02 Funcionalidad "Dado un texto y un número, listar post y/o comentarios que contengan ese texto":

ID	Escenario de test	Pasos	Datos utilizados	Resultado esperado	Resultado obtenido	Estado (F=Falla, P=pasa)
F01-T01	Vista de la pestaña "Buscar"	1-Ingreso a la pagina 2-Selección de opción "Iniciar Sesión" 3- Llenar los campos correspondientes a un miembro 4- Ingresar a la pestaña Buscar 5-Llenar datos correspondientes	1-Email 2-Contenido 3-Texto 4-Valor de Aceptación	Dentro de la misma vista, despliegue exitoso de todos los posts y/o comentarios correspondientes que contengan el texto y un VA mayor al ingresado.	Dentro de la misma vista, despliegue exitoso de todos los posts y/o comentarios correspondientes que contengan el texto y un VA mayor al ingresado.	P
F01-T02	Vista de la pestaña "Buscar"	1-Ingreso a la pagina 2-Selección de opción "Iniciar Sesión" 3- Llenar los campos correspondientes a un miembro 4- Ingresar a la pestaña Buscar 5-Llenar datos correspondientes	1-Email 2-Contenido 3-Texto 4-Valor de Aceptación	Ingreso vacío de algunos de los campos muestra mensaje de error y no permite listar los posts.	Ingreso vacío de algunos de los campos muestra mensaje de error y no permite listar los posts.	P
F01-T03	Vista de la pestaña "Buscar"	1-Ingreso a la pagina 2-Selección de opción "Iniciar Sesión" 3- Llenar los campos correspondientes a un miembro 4- Ingresar a la pestaña Buscar 5-Llenar datos correspondientes	1-Email 2-Contenido 3-Texto 4-Valor de Aceptación	Si no se encuentra un post que contengan los datos ingresados, no muestra ningún post	Si no se encuentra un post que contengan los datos ingresados, no muestra ningún post	P

1.5 Código fuente:

1.5.1 Dominio:

Sistema:

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Globalization;
using System.Linq;
using System.Linq.Expressions;
using System.Text;
using System.Threading.Tasks;
using static System.Net.Mime.MediaTypeNames;
using static System.Runtime.InteropServices.JavaScript.JSType;

namespace Dominio
{
public class Sistema
{
    private List<Publicacion> _publicaciones = new List<Publicacion>();
    private List<Invitacion> _invitaciones = new List<Invitacion>();
    private List<Usuario> _usuarios = new List<Usuario>();

    #region SINGLETON

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

    #endregion
    private Sistema()
    {
        PrecargarDatos();
    }
```

```
    #region PRECARGAS
    //Precargamos todos los datos necesarios separandolos en 3 metodos privados para una mayor
    legibilidad, evitar perdida de datos y seguridad de los mismos.
    public void PrecargarDatos()
    {
        try
        {
            PrecargarUsuarios();
            PrecargarPublicaciones();
            PrecargarInvitaciones();
        }
        catch (Exception e)
        {
            throw new Exception(e.Message);
        }
    }

    //Precarga de Usuarios (miembro y administrador), utilizamos metodo AgregarUsuario para
    dar alta a nuestra lista de Usuarios del Sistema.
    private void PrecargarUsuarios()
{
    try
    {
        NuevoMiembro("Rodrigo", "Cristaldo", "13/02/1995", "rodrigo@gmail.com", "rC1234");
        NuevoMiembro("Angel", "Vaz", "02/07/1997", "angel@gmail.com", "aV1234");
        NuevoMiembro("Edinson", "Cavani", "14/02/1987", "edi@gmail.com", "eC1234");
        NuevoMiembro("Diego", "Forlan", "19/05/1979", "forlan@gmail.com", "dF1234");
        NuevoMiembro("Sergio", "Ramos", "30/03/1986", "sergio@gmail.com", "sR1234");
        NuevoMiembro("Robert", "Lewandowski", "21/08/1988", "robert@gmail.com", "rL1234");
        NuevoMiembro("Cristiano", "Ronaldo", "05/02/1985", "cr7@gmail.com", "cR1234");
        NuevoMiembro("Neymar", "Santos", "05/02/1992", "neymar@gmail.com", "nS1234");
        NuevoMiembro("Luis", "Suarez", "24/01/1987", "lucho@gmail.com", "lS1234");
        NuevoMiembro("Lionel", "Messi", "24/06/1987", "messi@gmail.com", "lM1234");
        NuevoMiembro("Pato", "Sosa", "02/06/1987", "patososa@gmail.com", "pS1234");

        Administrador admin1 = new Administrador("admin1@gmail.com", "Admin4321");
        AltaUsuario(admin1);
    }
    catch (Exception e)
    {
        //Mostramos un mensaje de error en caso de que por alguna razon no se den de alta.
```

```
            throw new Exception(e.Message);
        }
    }
    //Precarga de las Publicaciones (post,comentario) y Reacciones a las mismas, utilizando los
        metodos de dar alta a nuestras listas de sistema.
    private void PrecargarPublicaciones()
    {
        try
        {
            // Creamos 5 Publicaciones de tipo Post
            Post po1 = new Post("balonDeOro.JPG", new List<Comentario>(), false,
        GetMiembroPorEmail("rodrigo@gmail.com"), new DateTime(2023, 09, 28), "Rivales", "Lionel
        Messi a un paso de un nuevo balón de oro, se alargará la diferencia con Cristiano
        Ronaldo?", true, new List<Reaccion>());
            Post po2 = new Post("messiEnUsa.jpg", new List<Comentario>(), false,
        GetMiembroPorEmail("messi@gmail.com"), new DateTime(2023, 09, 20), "El GOAT en USA",
        "Luego de un mal sabor de boca en PSG se ve un brillante desempeño de Lionel Messi en el
        Inter de Miami", true, new List<Reaccion>());
            Post po3 = new Post("fichajesEuropa.png", new List<Comentario>(), false,
        GetMiembroPorEmail("lucho@gmail.com"), new DateTime(2023, 09, 25), "Mercado de fichajes",
        "Los clubes de la elite europea realizan movimientos estratégicos", true, new
        List<Reaccion>());
            Post po4 = new Post("UCL2023.jpg", new List<Comentario>(), false,
        GetMiembroPorEmail("angel@gmail.com"), new DateTime(2023, 09, 15), "UEFA Champions
        League", "Clubes poderosos compiten por el título europeo", true, new List<Reaccion>());
            Post po5 = new Post("var.png", new List<Comentario>(), false,
        GetMiembroPorEmail("robert@gmail.com"), new DateTime(2023, 09, 05), "VAR", "2023: El VAR
        continúa transformando el fútbol moderno", true, new List<Reaccion>());

            // Damos de alta a los Posts a nuestra lista de Publicaciones del Sistema
            AltaPublicacion(po1);
            AltaPublicacion(po2);
            AltaPublicacion(po3);
            AltaPublicacion(po4);
            AltaPublicacion(po5);

        //Nuevos Post Obligatorio2

NuevoPost(GetMiembroPorEmail("neymar@gmail.com"), "Gol Sorprendente", "¡Increíble momento en
        el campo!", "neymar_gol.jpg", false, false, new List<Reaccion>());
NuevoPost(GetMiembroPorEmail("robert@gmail.com"), "Jugada Maestra", "Deslumbrante habilidad en
        el partido", "robert_jugada.jpg", false, false, new List<Reaccion>());
NuevoPost(GetMiembroPorEmail("sergio@gmail.com"), "Victoria Asombrosa", "Celebrando una
        victoria épica", "sergio_victoria.jpg", false, false, new List<Reaccion>());
NuevoPost(GetMiembroPorEmail("cr7@gmail.com"), "Clásico Intenso", "Momentos emocionantes en el
        clásico", "cr7_clasico.jpg", false, false, new List<Reaccion>());
NuevoPost(GetMiembroPorEmail("patososa@gmail.com"), "Regate Asombroso", "¡No podrás creer este
        regate!", "patososa_regate.jpg", false, false, new List<Reaccion>());
```

```
NuevoPost(GetMiembroPorEmail("forlan@gmail.com"), "Gol de Chilena", "La chilena que dejó a
    todos boquiabiertos", "forlan_chilena.jpg", false, false, new List<Reaccion>());

NuevoPost(GetMiembroPorEmail("neymar@gmail.com"), "Remate Preciso", "Precisión en cada disparo
    al arco", "neymar_remate.jpg", false, false, new List<Reaccion>());

NuevoPost(GetMiembroPorEmail("robert@gmail.com"), "Duelo Intenso", "Partido emocionante hasta
    el último minuto", "robert_duelo.jpg", false, false, new List<Reaccion>());

NuevoPost(GetMiembroPorEmail("sergio@gmail.com"), "Fiesta en el Estadio", "La afición haciendo
    vibrar el estadio", "sergio_fiesta.jpg", false, false, new List<Reaccion>());

NuevoPost(GetMiembroPorEmail("cr7@gmail.com"), "Debut Triunfal", "Un debut soñado en el mundo
    del fútbol", "cr7_debut.jpg", false, false, new List<Reaccion>());


    // Creamos 15 Publicaciones de tipo Comentario, 3 para cada Publicacion de tipo Post
    existente


    Comentario co1 = new Comentario(GetMiembroPorEmail("angel@gmail.com"), "Grandes", "Sus
    habilidades en el campo son asombrosas y verlos competir por este premio es simplemente
    épico.", false, new List<Reaccion>());

    Comentario co2 = new Comentario(GetMiembroPorEmail("lucho@gmail.com"), "Como dos
    fieras", "La eterna pulseada por el Balón de Oro!", false, new List<Reaccion>());

    Comentario co3 = new Comentario(GetMiembroPorEmail("neymar@gmail.com"), "Show", "No
    hay duda, Messi y Ronaldo saben cómo mantenernos en vilo. Este duelo por el Balón de Oro
    es un espectáculo en sí mismo.", false, new List<Reaccion>());


    Comentario co4 = new Comentario(GetMiembroPorEmail("cr7@gmail.com"), "Competencia",
    "Un fuerte abrazo para Leo en esta nueva etapa en Inter de Miami. La competencia siempre
    eleva nuestro juego y estoy seguro de que la vas a romper en la MLS", false, new
    List<Reaccion>());

    Comentario co5 = new Comentario(GetMiembroPorEmail("sergio@gmail.com"), "Que sigan los
    goles", "Estoy seguro de que vas a dejar huella en la MLS. Que sigan los éxitos y los
    goles, Leo", false, new List<Reaccion>());

    Comentario co6 = new Comentario(GetMiembroPorEmail("edi@gmail.com"), "Retos y
    Triunfos", "Que sigan los retos y los triunfos, admiración por tu talento", false, new
    List<Reaccion>());


    Comentario co7 = new Comentario(GetMiembroPorEmail("rodrigo@gmail.com"), "Partidazos",
    "Los movimientos del mercado de fichaje que estamos viendo son verdaderamente
    impresionantes. Prepárense para una temporada cargada de sorpresas y partidazos! ",
    false, new List<Reaccion>());

    Comentario co8 = new Comentario(GetMiembroPorEmail("forlan@gmail.com"), "Elite", "Es
    fascinante ver cómo los clubes de la élite están planificando sus futuras temporadas.",
    false, new List<Reaccion>());

    Comentario co9 = new Comentario(GetMiembroPorEmail("neymar@gmail.com"), "Espectáculo",
    "Un espectáculo deportivo sin igual. ", false, new List<Reaccion>());


    Comentario co10 = new Comentario(GetMiembroPorEmail("rodrigo@gmail.com"), "Momentos
    mágicos", "¡Que vivan los momentos mágicos de la Champions!", false, new
    List<Reaccion>());

    Comentario co11 = new Comentario(GetMiembroPorEmail("lucho@gmail.com"), "Nada se
    compara", "La lucha por el título europeo es un verdadero espectáculo, No hay nada como
    la Champions!", false, new List<Reaccion>());
```

```
    Comentario co12 = new Comentario(GetMiembroPorEmail("neymar@gmail.com"), "Futuro", "La Champions de este año es el escenario donde las futuras leyendas del fútbol están forjando su destino", false, new List<Reaccion>());


    Comentario co13 = new Comentario(GetMiembroPorEmail("cr7@gmail.com"), "Defectos del VAR", "Afecta la espontaneidad y la emoción del juego. Las celebraciones intensas tras un gol se ven frenadas por la revisión constante", false, new List<Reaccion>());
    Comentario co14 = new Comentario(GetMiembroPorEmail("forlan@gmail.com"), "Clave", "La clave está en perfeccionar su uso para evitar malentendidos y mantener el juego justo y emocionante.", false, new List<Reaccion>());
    Comentario co15 = new Comentario(GetMiembroPorEmail("edi@gmail.com"), "Progreso", "Bienvenido sea el progreso tecnológico en nuestro deporte favorito", false, new List<Reaccion>());


    //Asociamos los comentarios a cada publicacion.


    po1.AltaComentario(co1);
    AltaPublicacion(co1);
    po1.AltaComentario(co2);
    AltaPublicacion(co2);
    po1.AltaComentario(co3);
    AltaPublicacion(co3);


    po2.AltaComentario(co4);
    AltaPublicacion(co4);
    po2.AltaComentario(co5);
    AltaPublicacion(co5);
    po2.AltaComentario(co6);
    AltaPublicacion(co6);


    po3.AltaComentario(co7);
    AltaPublicacion(co7);
    po3.AltaComentario(co8);
    AltaPublicacion(co8);
    po3.AltaComentario(co9);
    AltaPublicacion(co9);


    po4.AltaComentario(co10);
    AltaPublicacion(co10);
    po4.AltaComentario(co11);
    AltaPublicacion(co11);
    po4.AltaComentario(co12);
    AltaPublicacion(co12);
```

```
        po5.AltaComentario(co13);
        AltaPublicacion(co13);
        po5.AltaComentario(co14);
        AltaPublicacion(co14);
        po5.AltaComentario(co15);
        AltaPublicacion(co15);

    //Nuevos Comentarios Obligatorio2

// Comentarios para la publicación de Neymar
NuevoComentario(GetMiembroPorEmail("lucho@gmail.com"), "Increíble", "Excelente gol, Neymar!",
    false, new List<Reaccion>(), GetPostPorId(6));
NuevoComentario(GetMiembroPorEmail("angel@gmail.com"), "Asombroso", "¡Qué habilidad
    increíble!", false, new List<Reaccion>(), GetPostPorId(6));
NuevoComentario(GetMiembroPorEmail("messi@gmail.com"), "Sorprendente", "Neymar siempre
    sorprendiéndonos", false, new List<Reaccion>(), GetPostPorId(6));

// Comentarios para la publicación de Robert
NuevoComentario(GetMiembroPorEmail("sergio@gmail.com"), "Increíble", "Impresionante jugada,
    Robert!", false, new List<Reaccion>(), GetPostPorId(7));
NuevoComentario(GetMiembroPorEmail("rodrigo@gmail.com"), "Asombroso", "¡Esa habilidad es de
    otro nivel!", false, new List<Reaccion>(), GetPostPorId(7));
NuevoComentario(GetMiembroPorEmail("cr7@gmail.com"), "Impresionante", "Robert, siempre
    dejándonos sin palabras", false, new List<Reaccion>(), GetPostPorId(7));

// Comentarios para la publicación de Sergio
NuevoComentario(GetMiembroPorEmail("patososa@gmail.com"), "Felicidades", "¡Felicidades por la
    victoria, Sergio!", false, new List<Reaccion>(), GetPostPorId(8));
NuevoComentario(GetMiembroPorEmail("lucho@gmail.com"), "Emocionante", "Esa celebración lo dice
    todo", false, new List<Reaccion>(), GetPostPorId(8));
NuevoComentario(GetMiembroPorEmail("forlan@gmail.com"), "Líder", "Sergio, un líder en el
    campo", false, new List<Reaccion>(), GetPostPorId(8));

// Comentarios para la publicación de CR7
NuevoComentario(GetMiembroPorEmail("neymar@gmail.com"), "Épico", "Ese clásico estuvo
    increíble, CR7!", false, new List<Reaccion>(), GetPostPorId(9));
NuevoComentario(GetMiembroPorEmail("lucho@gmail.com"), "Épico", "Momentos épicos en cada
    partido", false, new List<Reaccion>(), GetPostPorId(9));
NuevoComentario(GetMiembroPorEmail("messi@gmail.com"), "El Mejor", "CR7, el mejor jugador del
    mundo", false, new List<Reaccion>(), GetPostPorId(9));

// Comentarios para la publicación de Patososa
NuevoComentario(GetMiembroPorEmail("forlan@gmail.com"), "Asombroso", "Regate asombroso,
    Patososa!", false, new List<Reaccion>(), GetPostPorId(10));
NuevoComentario(GetMiembroPorEmail("sergio@gmail.com"), "Único", "¡Esa habilidad en el campo
    es única!", false, new List<Reaccion>(), GetPostPorId(10));
```

```
NuevoComentario(GetMiembroPorEmail("neymar@gmail.com"), "Rey del Regate", "Patososa, el rey
    del regate", false, new List<Reaccion>(), GetPostPorId(10));


// Comentarios para la publicación de Forlán

NuevoComentario(GetMiembroPorEmail("lucho@gmail.com"), "Impresionante", "Chilena
    impresionante, Forlán!", false, new List<Reaccion>(), GetPostPorId(11));

NuevoComentario(GetMiembroPorEmail("robert@gmail.com"), "Demostrando Calidad", "Forlán,
    siempre demostrando su calidad", false, new List<Reaccion>(), GetPostPorId(11));

NuevoComentario(GetMiembroPorEmail("sergio@gmail.com"), "Ícono", "Forlán, un ícono del
    fútbol", false, new List<Reaccion>(), GetPostPorId(11));


// Comentarios para la publicación de Neymar

NuevoComentario(GetMiembroPorEmail("patososa@gmail.com"), "Preciso", "Precisión en el remate,
    Neymar!", false, new List<Reaccion>(), GetPostPorId(12));

NuevoComentario(GetMiembroPorEmail("forlan@gmail.com"), "Sin Aliento", "Neymar, siempre
    dejándonos sin aliento", false, new List<Reaccion>(), GetPostPorId(12));

NuevoComentario(GetMiembroPorEmail("rodrigo@gmail.com"), "Maestro", "Neymar, un maestro del
    fútbol", false, new List<Reaccion>(), GetPostPorId(12));


// Comentarios para la publicación de Robert

NuevoComentario(GetMiembroPorEmail("messi@gmail.com"), "Intenso", "Duelo intenso, Robert!",
    false, new List<Reaccion>(), GetPostPorId(13));

NuevoComentario(GetMiembroPorEmail("neymar@gmail.com"), "Entregando Todo", "Robert, siempre
    entregando todo en el campo", false, new List<Reaccion>(), GetPostPorId(13));

NuevoComentario(GetMiembroPorEmail("sergio@gmail.com"), "Gran Partido", "Gran partido,
    Robert!", false, new List<Reaccion>(), GetPostPorId(13));


// Comentarios para la publicación de Sergio

NuevoComentario(GetMiembroPorEmail("robert@gmail.com"), "Presente", "La afición siempre
    presente, Sergio!", false, new List<Reaccion>(), GetPostPorId(14));

NuevoComentario(GetMiembroPorEmail("lucho@gmail.com"), "El Alma", "Sergio, el alma del
    equipo", false, new List<Reaccion>(), GetPostPorId(14));

NuevoComentario(GetMiembroPorEmail("patososa@gmail.com"), "Líder", "Sergio, un verdadero
    líder", false, new List<Reaccion>(), GetPostPorId(14));


// Comentarios para la publicación de CR7

NuevoComentario(GetMiembroPorEmail("rodrigo@gmail.com"), "Debut Soñado", "Debut soñado, CR7!",
    false, new List<Reaccion>(), GetPostPorId(15));

NuevoComentario(GetMiembroPorEmail("sergio@gmail.com"), "Marcando la Diferencia", "CR7,
    siempre marcando la diferencia", false, new List<Reaccion>(), GetPostPorId(15));

NuevoComentario(GetMiembroPorEmail("forlan@gmail.com"), "Inolvidable", "Esa primera vez en el
    campo, inolvidable", false, new List<Reaccion>(), GetPostPorId(15));



    //Precarga de Reacciones a Publicaciones


    Reaccion re1 = new Reaccion(Tipo.like, GetMiembroPorEmail("angel@gmail.com"));
```

```
        Reaccion re2 = new Reaccion(Tipo.dislike, GetMiembroPorEmail("rodrigo@gmail.com"));
        Reaccion re3 = new Reaccion(Tipo.like, GetMiembroPorEmail("lucho@gmail.com"));

        Reaccion re4 = new Reaccion(Tipo.like, GetMiembroPorEmail("neymar@gmail.com"));
        Reaccion re5 = new Reaccion(Tipo.like, GetMiembroPorEmail("messi@gmail.com"));
        Reaccion re6 = new Reaccion(Tipo.dislike, GetMiembroPorEmail("edi@gmail.com"));

        Reaccion re7 = new Reaccion(Tipo.dislike, GetMiembroPorEmail("angel@gmail.com"));
        Reaccion re8 = new Reaccion(Tipo.like, GetMiembroPorEmail("rodrigo@gmail.com"));
        Reaccion re9 = new Reaccion(Tipo.like, GetMiembroPorEmail("lucho@gmail.com"));

        Reaccion re10 = new Reaccion(Tipo.like, GetMiembroPorEmail("neymar@gmail.com"));
        Reaccion re11 = new Reaccion(Tipo.dislike, GetMiembroPorEmail("messi@gmail.com"));
        Reaccion re12 = new Reaccion(Tipo.like, GetMiembroPorEmail("edi@gmail.com"));

        //Agregamos 3 reacciones a las lista de reacciones de 2 publicaciones de tipo Post y 2
    publicaciones de tipo Comentario mediante un metodo.

        po2.AgregarReaccion(re1);
        po2.AgregarReaccion(re2);
        po2.AgregarReaccion(re3);

        po3.AgregarReaccion(re4);
        po3.AgregarReaccion(re5);
        po3.AgregarReaccion(re6);

        co9.AgregarReaccion(re7);
        co9.AgregarReaccion(re8);
        co9.AgregarReaccion(re9);

        co13.AgregarReaccion(re10);
        co13.AgregarReaccion(re11);
        co13.AgregarReaccion(re12);

    //Nuevas Reacciones Obligatorio2

// Reacciones de tipo "like"
NuevaReaccion("like", GetMiembroPorEmail("messi@gmail.com"), GetPublicacionPorId(1));
NuevaReaccion("like", GetMiembroPorEmail("lucho@gmail.com"), GetPublicacionPorId(2));
NuevaReaccion("like", GetMiembroPorEmail("angel@gmail.com"), GetPublicacionPorId(3));
NuevaReaccion("like", GetMiembroPorEmail("rodrigo@gmail.com"), GetPublicacionPorId(4));
NuevaReaccion("like", GetMiembroPorEmail("angel@gmail.com"), GetPublicacionPorId(5));
```

```
NuevaReaccion("like", GetMiembroPorEmail("sergio@gmail.com"), GetPublicacionPorId(6));
NuevaReaccion("like", GetMiembroPorEmail("robert@gmail.com"), GetPublicacionPorId(7));
NuevaReaccion("like", GetMiembroPorEmail("rodrigo@gmail.com"), GetPublicacionPorId(8));
NuevaReaccion("like", GetMiembroPorEmail("forlan@gmail.com"), GetPublicacionPorId(9));
NuevaReaccion("like", GetMiembroPorEmail("patososa@gmail.com"), GetPublicacionPorId(10));
NuevaReaccion("like", GetMiembroPorEmail("lucho@gmail.com"), GetPublicacionPorId(11));
NuevaReaccion("like", GetMiembroPorEmail("sergio@gmail.com"), GetPublicacionPorId(12));
NuevaReaccion("like", GetMiembroPorEmail("neymar@gmail.com"), GetPublicacionPorId(13));
NuevaReaccion("like", GetMiembroPorEmail("messi@gmail.com"), GetPublicacionPorId(14));
NuevaReaccion("like", GetMiembroPorEmail("robert@gmail.com"), GetPublicacionPorId(15));
NuevaReaccion("like", GetMiembroPorEmail("rodrigo@gmail.com"), GetPublicacionPorId(16));
NuevaReaccion("like", GetMiembroPorEmail("cr7@gmail.com"), GetPublicacionPorId(17));
NuevaReaccion("like", GetMiembroPorEmail("angel@gmail.com"), GetPublicacionPorId(18));
NuevaReaccion("like", GetMiembroPorEmail("sergio@gmail.com"), GetPublicacionPorId(19));
NuevaReaccion("like", GetMiembroPorEmail("forlan@gmail.com"), GetPublicacionPorId(20));
NuevaReaccion("like", GetMiembroPorEmail("patososa@gmail.com"), GetPublicacionPorId(21));
NuevaReaccion("like", GetMiembroPorEmail("neymar@gmail.com"), GetPublicacionPorId(22));
NuevaReaccion("like", GetMiembroPorEmail("lucho@gmail.com"), GetPublicacionPorId(23));
NuevaReaccion("like", GetMiembroPorEmail("robert@gmail.com"), GetPublicacionPorId(24));
NuevaReaccion("like", GetMiembroPorEmail("messi@gmail.com"), GetPublicacionPorId(25));
NuevaReaccion("like", GetMiembroPorEmail("angel@gmail.com"), GetPublicacionPorId(26));
NuevaReaccion("like", GetMiembroPorEmail("messi@gmail.com"), GetPublicacionPorId(27));
NuevaReaccion("like", GetMiembroPorEmail("robert@gmail.com"), GetPublicacionPorId(28));
NuevaReaccion("like", GetMiembroPorEmail("rodrigo@gmail.com"), GetPublicacionPorId(29));
NuevaReaccion("like", GetMiembroPorEmail("forlan@gmail.com"), GetPublicacionPorId(30));
NuevaReaccion("like", GetMiembroPorEmail("patososa@gmail.com"), GetPublicacionPorId(31));
NuevaReaccion("like", GetMiembroPorEmail("lucho@gmail.com"), GetPublicacionPorId(32));
NuevaReaccion("like", GetMiembroPorEmail("sergio@gmail.com"), GetPublicacionPorId(33));
NuevaReaccion("like", GetMiembroPorEmail("neymar@gmail.com"), GetPublicacionPorId(34));
NuevaReaccion("like", GetMiembroPorEmail("messi@gmail.com"), GetPublicacionPorId(35));
NuevaReaccion("like", GetMiembroPorEmail("robert@gmail.com"), GetPublicacionPorId(36));
NuevaReaccion("like", GetMiembroPorEmail("rodrigo@gmail.com"), GetPublicacionPorId(37));
NuevaReaccion("like", GetMiembroPorEmail("cr7@gmail.com"), GetPublicacionPorId(38));
NuevaReaccion("like", GetMiembroPorEmail("angel@gmail.com"), GetPublicacionPorId(39));
NuevaReaccion("like", GetMiembroPorEmail("sergio@gmail.com"), GetPublicacionPorId(40));
NuevaReaccion("like", GetMiembroPorEmail("forlan@gmail.com"), GetPublicacionPorId(41));
NuevaReaccion("like", GetMiembroPorEmail("patososa@gmail.com"), GetPublicacionPorId(42));
NuevaReaccion("like", GetMiembroPorEmail("lucho@gmail.com"), GetPublicacionPorId(43));
NuevaReaccion("like", GetMiembroPorEmail("neymar@gmail.com"), GetPublicacionPorId(44));
NuevaReaccion("like", GetMiembroPorEmail("messi@gmail.com"), GetPublicacionPorId(45));
NuevaReaccion("like", GetMiembroPorEmail("robert@gmail.com"), GetPublicacionPorId(46));
NuevaReaccion("like", GetMiembroPorEmail("rodrigo@gmail.com"), GetPublicacionPorId(47));
```

```
NuevaReaccion("like", GetMiembroPorEmail("cr7@gmail.com"), GetPublicacionPorId(48));
NuevaReaccion("like", GetMiembroPorEmail("angel@gmail.com"), GetPublicacionPorId(49));
NuevaReaccion("like", GetMiembroPorEmail("sergio@gmail.com"), GetPublicacionPorId(50));
NuevaReaccion("like", GetMiembroPorEmail("forlan@gmail.com"), GetPublicacionPorId(51));
NuevaReaccion("like", GetMiembroPorEmail("patososa@gmail.com"), GetPublicacionPorId(52));
NuevaReaccion("like", GetMiembroPorEmail("lucho@gmail.com"), GetPublicacionPorId(53));
NuevaReaccion("like", GetMiembroPorEmail("neymar@gmail.com"), GetPublicacionPorId(54));
NuevaReaccion("like", GetMiembroPorEmail("messi@gmail.com"), GetPublicacionPorId(55));
NuevaReaccion("like", GetMiembroPorEmail("robert@gmail.com"), GetPublicacionPorId(56));
NuevaReaccion("like", GetMiembroPorEmail("rodrigo@gmail.com"), GetPublicacionPorId(57));
NuevaReaccion("like", GetMiembroPorEmail("cr7@gmail.com"), GetPublicacionPorId(58));
NuevaReaccion("like", GetMiembroPorEmail("angel@gmail.com"), GetPublicacionPorId(59));
NuevaReaccion("like", GetMiembroPorEmail("sergio@gmail.com"), GetPublicacionPorId(60));

// Reacciones de tipo "dislike"

NuevaReaccion("dislike", GetMiembroPorEmail("messi@gmail.com"), GetPublicacionPorId(6));
NuevaReaccion("dislike", GetMiembroPorEmail("rodrigo@gmail.com"), GetPublicacionPorId(7));
NuevaReaccion("dislike", GetMiembroPorEmail("angel@gmail.com"), GetPublicacionPorId(8));
NuevaReaccion("dislike", GetMiembroPorEmail("patososa@gmail.com"), GetPublicacionPorId(9));
NuevaReaccion("dislike", GetMiembroPorEmail("lucho@gmail.com"), GetPublicacionPorId(10));
NuevaReaccion("dislike", GetMiembroPorEmail("sergio@gmail.com"), GetPublicacionPorId(11));
NuevaReaccion("dislike", GetMiembroPorEmail("neymar@gmail.com"), GetPublicacionPorId(12));
NuevaReaccion("dislike", GetMiembroPorEmail("messi@gmail.com"), GetPublicacionPorId(13));
NuevaReaccion("dislike", GetMiembroPorEmail("robert@gmail.com"), GetPublicacionPorId(14));
NuevaReaccion("dislike", GetMiembroPorEmail("rodrigo@gmail.com"), GetPublicacionPorId(15));
NuevaReaccion("dislike", GetMiembroPorEmail("cr7@gmail.com"), GetPublicacionPorId(16));
NuevaReaccion("dislike", GetMiembroPorEmail("angel@gmail.com"), GetPublicacionPorId(17));
NuevaReaccion("dislike", GetMiembroPorEmail("sergio@gmail.com"), GetPublicacionPorId(18));
NuevaReaccion("dislike", GetMiembroPorEmail("forlan@gmail.com"), GetPublicacionPorId(19));
NuevaReaccion("dislike", GetMiembroPorEmail("patososa@gmail.com"), GetPublicacionPorId(20));
NuevaReaccion("dislike", GetMiembroPorEmail("neymar@gmail.com"), GetPublicacionPorId(21));
NuevaReaccion("dislike", GetMiembroPorEmail("lucho@gmail.com"), GetPublicacionPorId(22));
NuevaReaccion("dislike", GetMiembroPorEmail("robert@gmail.com"), GetPublicacionPorId(23));
NuevaReaccion("dislike", GetMiembroPorEmail("messi@gmail.com"), GetPublicacionPorId(24));
NuevaReaccion("dislike", GetMiembroPorEmail("rodrigo@gmail.com"), GetPublicacionPorId(25));



    }
    catch (Exception e)
    {
```

```
        //Mostramos un mensaje en caso de error
        throw new Exception(e.Message);
    }
}

//Precargamos las invitaciones con sus respectivas altas.
private void PrecargarInvitaciones()
{
    try
    {
        //Creamos invitaciones en sus 3 estados posibles
        Invitacion i1 = new Invitacion(GetMiembroPorEmail("rodrigo@gmail.com"),
    GetMiembroPorEmail("angel@gmail.com"), EstadoInvitacion.Aprobada);
        Invitacion i2 = new Invitacion(GetMiembroPorEmail("rodrigo@gmail.com"),
    GetMiembroPorEmail("messi@gmail.com"), EstadoInvitacion.Aprobada);
        Invitacion i3 = new Invitacion(GetMiembroPorEmail("rodrigo@gmail.com"),
    GetMiembroPorEmail("lucho@gmail.com"), EstadoInvitacion.Aprobada);
        Invitacion i4 = new Invitacion(GetMiembroPorEmail("messi@gmail.com"),
    GetMiembroPorEmail("lucho@gmail.com"), EstadoInvitacion.Aprobada);
        Invitacion i5 = new Invitacion(GetMiembroPorEmail("angel@gmail.com"),
    GetMiembroPorEmail("lucho@gmail.com"), EstadoInvitacion.Aprobada);
        Invitacion i6 = new Invitacion(GetMiembroPorEmail("angel@gmail.com"),
    GetMiembroPorEmail("robert@gmail.com"), EstadoInvitacion.Pendiente_Aprobacion);
        Invitacion i7 = new Invitacion(GetMiembroPorEmail("robert@gmail.com"),
    GetMiembroPorEmail("lucho@gmail.com"), EstadoInvitacion.Pendiente_Aprobacion);
        Invitacion i8 = new Invitacion(GetMiembroPorEmail("lucho@gmail.com"),
    GetMiembroPorEmail("neymar@gmail.com"), EstadoInvitacion.Rechazada);
        Invitacion i9 = new Invitacion(GetMiembroPorEmail("robert@gmail.com"),
    GetMiembroPorEmail("neymar@gmail.com"), EstadoInvitacion.Rechazada);
        Invitacion i10 = new Invitacion(GetMiembroPorEmail("messi@gmail.com"),
    GetMiembroPorEmail("robert@gmail.com"), EstadoInvitacion.Pendiente_Aprobacion);

        //Damos alta a las invitaciones a la lista de Invitaciones del Sistema
        AltaInvitacion(i1);
        AltaInvitacion(i2);
        AltaInvitacion(i3);
        AltaInvitacion(i4);
        AltaInvitacion(i5);
        AltaInvitacion(i6);
        AltaInvitacion(i7);
        AltaInvitacion(i8);
        AltaInvitacion(i9);
        AltaInvitacion(i10);

      // Invitaciones utilizando GetMiembroPorEmail
```

```
        NuevaInvitacion(GetMiembroPorEmail("neymar@gmail.com"),
        GetMiembroPorEmail("messi@gmail.com"));

        NuevaInvitacion(GetMiembroPorEmail("angel@gmail.com"),
        GetMiembroPorEmail("messi@gmail.com"));

        NuevaInvitacion(GetMiembroPorEmail("cr7@gmail.com"),
        GetMiembroPorEmail("robert@gmail.com"));

        NuevaInvitacion(GetMiembroPorEmail("sergio@gmail.com"),
        GetMiembroPorEmail("rodrigo@gmail.com"));

        NuevaInvitacion(GetMiembroPorEmail("forlan@gmail.com"),
        GetMiembroPorEmail("patososa@gmail.com"));

        NuevaInvitacion(GetMiembroPorEmail("lucho@gmail.com"),
        GetMiembroPorEmail("sergio@gmail.com"));

        NuevaInvitacion(GetMiembroPorEmail("messi@gmail.com"),
        GetMiembroPorEmail("forlan@gmail.com"));

        NuevaInvitacion(GetMiembroPorEmail("robert@gmail.com"),
        GetMiembroPorEmail("neymar@gmail.com"));

        NuevaInvitacion(GetMiembroPorEmail("rodrigo@gmail.com"),
        GetMiembroPorEmail("cr7@gmail.com"));

        NuevaInvitacion(GetMiembroPorEmail("patososa@gmail.com"),
        GetMiembroPorEmail("angel@gmail.com"));



            //Mediante este metodo generamos el vinculo de amistad en los casos en que el estado
        de la invitacion sea 'Aprobada'.
            GenerarVinculosDeAmistad();
        }
        catch (Exception e)
        {
            //Mostramos un mensaje en caso de error.
            throw new Exception(e.Message);
        }
    }


    #region ALTAS


    //Desarrollamos los metodos de altas para (Usuario, Publicacion, Invitacion). Consta de
        agregar el objeto a la lista correspondiente del sistema luego de sortear validaciones
        especificas para cada uno.


    public void AltaUsuario(Usuario u)
    {
        try
        {
            if (u != null)
            {
                //Validamos los datos.
```

```
            u.Validar();
            //Agregamos a la lista de usuarios del sistema
            _usuarios.Add(u);
        }
    }
    catch (Exception e)
    {
        //Mensaje de error en caso que falle.
        throw new Exception(e.Message);
    }
}

public void AltaPublicacion(Publicacion p)
{
    try
    {
        if (p != null)
        {
            //Validamos los datos.
            p.Validar();
            //Agregamos a la lista de publicaciones del sistema.
            _publicaciones.Add(p);
        }
    }
    catch (Exception e)
    {
        //Mensaje de error en caso que falle.
        throw new Exception(e.Message);
    }
}

public void AltaInvitacion(Invitacion i)
{
    try
    {
        if (i != null)
        {
            //Validamos los datos.
            i.Validar();
            //Agregamos a la lista de publicaciones del sistema.
            _invitaciones.Add(i);
        }
```

```
    }
    catch (Exception e)
    {
        //Mensaje de error en caso que falle.
        throw new Exception(e.Message);
    }
}

//Metodo para crear un nuevo miembro.
public void NuevoMiembro(string nombre, string apellido, string fecha, string email, string
    contrasenia)
{
    try
    {
        foreach (Usuario u in _usuarios)
        {
            if (u.Email == email)
            {
                //Mensaje si existe otro usuario con el mismo email.
                throw new Exception("Ya existe un usuario con ese Email en el sistema");
            }
        }

        if (string.IsNullOrEmpty(nombre))
        {
            //Mensaje de error en caso de que el nombre este vacio.

            throw new Exception("El nombre no puede estar vacio");
        }

        if (string.IsNullOrEmpty(apellido))
        {
            //Mensaje de error en caso de que el apellido este vacio.

            throw new Exception("El apellido no puede estar vacio");

        }

        if (string.IsNullOrEmpty(fecha))
        {
            //Mensaje de error en caso de que la fecha este vacia.
```

```
            throw new Exception("La fecha de nacimiento no puede ser vacia");
        }

        if (!ValidarFormatoFecha(fecha))
        {
            //Mensaje de error en caso de que la fecha no sea del formato indicado .

            throw new Exception("La fecha de nacimiento debe tener formato DD/MM/AAAA");
        }

        if (string.IsNullOrEmpty(email))
        {
            //Mensaje de error en caso de que el Email este vacio.
            throw new Exception("El mail no puede ser vacio");
        }
        if (string.IsNullOrEmpty(contrasenia))
        {
            //Mensaje de error en caso de que la contraseña este vacia.
            throw new Exception("La contrasenia no puede ser vacia");
        }
        //Aqui se crea el nuevo miembro.
        DateTime fechaForm = ConvertirStringADateTime(fecha);
        Miembro m1 = new Miembro(nombre, apellido, fechaForm, false, email, contrasenia);
        AltaUsuario(m1);
    }
    catch (Exception e)
    {
        //Mostramos los errores si es que surgen.
        throw new Exception(e.Message);
    }

}

public void NuevoComentario(Miembro m, string titulo, string texto, bool privado,
    List<Reaccion> reacciones, Post post)
{
    try
    {
        if (m == null)
        {
            //Mensaje de error en caso de que la fecha este vacia.
```

```
            throw new Exception("Debe iniciar sesión para poder comentar");
        }

        if (m.Bloqueado)
        {
            //Mensaje de error en caso de que la fecha este vacia.

            throw new Exception("No puede comentar, está bloqueado");
        }

        if (post.Censurado)
        {
            //Mensaje de error en caso de que la fecha este vacia.

            throw new Exception("Este Post no se puede comentar ya que ha sido baneado");
        }

        if (string.IsNullOrEmpty(titulo))
        {
            //Mensaje de error en caso de que el nombre este vacio.

            throw new Exception("El titulo no puede estar vacio");
        }

        if (string.IsNullOrEmpty(texto))
        {
            //Mensaje de error en caso de que el apellido este vacio.

            throw new Exception("El comentario no puede estar vacio");

        }

        //Aqui se crea el nuevo comentario.
        Comentario c = new Comentario(m, titulo, texto, privado, reacciones);
        post.AltaComentario(c);
        AltaPublicacion(c);
    }
    catch (Exception e)
    {
        //Mostramos los errores si es que surgen.
        throw new Exception(e.Message);
    }
```

```
}

public void NuevoPost(Miembro m, string titulo, string texto, string imagen, bool privado,
    bool censurado, List<Reaccion> reacciones)
{
    try
    {
        if (m == null)
        {
            //Mensaje de error en caso de que la fecha este vacia.

            throw new Exception("Debe iniciar sesión para poder publicar");
        }

        if (m.Bloqueado)
        {
            //Mensaje de error en caso de que la fecha este vacia.

            throw new Exception("No puede publicar, está bloqueado");
        }

        if (string.IsNullOrEmpty(titulo))
        {
            //Mensaje de error en caso de que el nombre este vacio.

            throw new Exception("El titulo no puede estar vacio");
        }

        if (string.IsNullOrEmpty(texto))
        {
            //Mensaje de error en caso de que el apellido este vacio.

            throw new Exception("El contenido no puede estar vacio");

        }

        if (string.IsNullOrEmpty(imagen))
        {
            //Mensaje de error en caso de que el apellido este vacio.

            throw new Exception("La imagen no puede estar vacia");
```

```
        }

        //Aqui se crea el nuevo comentario.
        Post p = new Post(imagen, new List<Comentario>(), censurado, m, DateTime.Now, titulo, texto, privado, reacciones);
        AltaPublicacion(p);
    }
    catch (Exception e)
    {
        //Mostramos los errores si es que surgen.
        throw new Exception(e.Message);
    }
}

public void NuevaInvitacion(Miembro miembroSolicitante, Miembro miembroSolicitado)
{
    try
    {
        if (miembroSolicitante == null)
        {
            //Mensaje de error en caso de que el miembro solicitante no este en el sistema // no esté logueado

            throw new Exception("Debe iniciar sesión para poder enviar solicitudes");
        }

        if (miembroSolicitado == null)
        {
            //Mensaje de error en caso de que el miembro solicitado no este en el sistema

            throw new Exception("Miembro solicitado no está en el sistema");
        }

        if (miembroSolicitante.Bloqueado)
        {
            //Mensaje de error en caso de que el miembro solicitante este bloqueado

            throw new Exception("No puede enviar solicitudes, está bloqueado");
        }

        if (miembroSolicitado.Bloqueado)
        {
```

```
            //Mensaje de error en caso de que el miembro solicitado este bloqueado


            throw new Exception("El destinatario no puede recibir solicitudes, está
    bloqueado");
        }


        //Aqui se crea la nueva invitacion.
        Invitacion i = new Invitacion(miembroSolicitante, miembroSolicitado,
    EstadoInvitacion.Pendiente_Aprobacion);
        AltaInvitacion(i);
    }
    catch (Exception e)
    {
        //Mostramos los errores si es que surgen.
        throw new Exception(e.Message);
    }
}


public void NuevaReaccion(string tipoReaccion, Miembro m, Publicacion p)
{
    try
    {
        if (string.IsNullOrWhiteSpace(tipoReaccion))
        {
            //Mensaje de error en caso de que el miembro solicitante no este en el sistema //
    no esté logueado


            throw new Exception("Seleccione una reaccion valida");
        }


        if (m == null)
        {
            //Mensaje de error en caso de que el miembro solicitado no este en el sistema


            throw new Exception("El miembro seleccionado no está en el sistema");
        }


        if (m.Bloqueado)
        {
            //Mensaje de error en caso de que el miembro solicitante este bloqueado


            throw new Exception("No puede reaccionar, está bloqueado");
        }
```

```
        if (p == null)
        {
            //Mensaje de error en caso de que el miembro solicitado no este en el sistema

            throw new Exception("La publicacion seleccionada no está en el sistema");
        }

        if (p.GetTipo() == "Post")
        {
            Post post = (Post)p;
            if (post.Censurado)
            {
                //Mensaje de error en caso de que el miembro solicitado este bloqueado

                throw new Exception("El post no puede ser reaccionado, está baneado");
            }
        }

        Tipo reac;
        if (tipoReaccion == "like")
        {
            reac = Tipo.like;
        }
        else
        {
            reac = Tipo.dislike;
        }
        //Aqui se crea la nueva reacción.
        Reaccion r = new Reaccion(reac, m);
        p.AgregarReaccion(r);
    }
    catch (Exception e)
    {
        //Mostramos los errores si es que surgen.
        throw new Exception(e.Message);
    }
}

//Metodo para generar vinculos de amistad.
public void GenerarVinculosDeAmistad()
{
```

```
    foreach (Invitacion i in _invitaciones)
    {
        //Verificamos las invitaciones entre 2 miembros que sean 'Aprobada'.
        if (i.Estado == EstadoInvitacion.Aprobada)
        {
            //Agregamos a un miembro (Solicitante, Solicitado) a la lista de amigos del otro.
    Asi creamos el vinculo de amistad
            i.MiembroSolicitado.AgregarAmigo(i.MiembroSolicitante);
            i.MiembroSolicitante.AgregarAmigo(i.MiembroSolicitado);
        }
    }
}

#endregion ALTAS

#region GETS

//Con los metodos Get tenemos acceso a informacion especifica del sistema (listas).
public List<Miembro> GetMiembros()
{
    //Obtenemos la lista de miembros del sistema.
    List<Miembro> miembros = new List<Miembro>();
    foreach (Usuario u in _usuarios)
    {
        //Verificamos si el usuario es un Miembro.
        if (u != null && u.GetTipo() == "Miembro")
        {
            Miembro miembroAux = u as Miembro;
            miembros.Add(miembroAux);
        }
    }
    miembros.Sort();
    return miembros;
}

public List<Administrador> GetAdministradores()
{
    //Obtenemos la lista de administradores del sistema.
    List<Administrador> administradores = new List<Administrador>();
    foreach (Usuario u in _usuarios)
    {
        //Verificamos si el usuario es un Administrador
```

```
        if (u != null && u.GetTipo() == "Administrador")
        {
            Administrador adminAux = u as Administrador;
            administradores.Add(adminAux);
        }
    }
    return administradores;
}

public List<Publicacion> GetPublicaciones()
{
    //Obtenemos la lista de todas las publicaciones en el sistema
    return _publicaciones;
}

public List<Post> GetPosts()
{
    //Obtenemos la lista de posts del sistema.
    List<Post> posts = new List<Post>();
    foreach (Publicacion p in _publicaciones)
    {
        //Verificamos si la publicacion es un Post.
        if (p != null && p.GetTipo() == "Post")
        {
            Post postAux = p as Post;
            posts.Add(postAux);
        }
    }
    return posts;
}

public List<Comentario> GetComentarios()
{
    //Obtener la lista de comentarios del sistema.
    List<Comentario> comentarios = new List<Comentario>();
    foreach (Publicacion p in _publicaciones)
    {
        //Verificamos si la publicaciones es un Comentario
        if (p != null && p.GetTipo() == "Comentario")
        {
            Comentario comentarioAux = p as Comentario;
            comentarios.Add(comentarioAux);
```

```
        }
    }
    return comentarios;
}

//Metodo provado para obtener un Miembro por su Email
public Miembro GetMiembroPorEmail(string Email)
{
    foreach (Miembro usuario in GetMiembros())
    {
        //Buscar un Miembro por su Email.
        if (usuario.Email.Equals(Email))
        {
            return usuario;
        }
    }
    //Lanzamos una excepcion si el Email no esta en el sistema
    throw new Exception($"El correo electronico '{Email}' no esta en el sistema");
}

public Usuario GetUsuarioPorEmail(string Email)
{
    foreach (Usuario usuario in _usuarios)
    {
        //Buscar un Miembro por su Email.
        if (usuario.Email.Equals(Email))
        {
            return usuario;
        }
    }
    //Lanzamos una excepcion si el Email no esta en el sistema
    throw new Exception($"El correo electronico '{Email}' no esta en el sistema");
}

public Usuario GetUsuarioLogueado(int? idUsuarioLogueado)
{
    foreach (Usuario u in _usuarios)
    {
        if (u.Id.Equals(idUsuarioLogueado))
        {
            return u;
        }
```

```
    }
    return null;
}

//Listamos los posts que estén habilitados para un usuario del que conocemos el id
public List<Post> GetPostsHabilitados(int? idLogueado)
{
    List<Post> listaRet = new List<Post>();
    if (idLogueado != null)
    {
        foreach (Post p in GetPosts())
        {
            if (!p.Privado || p.Autor.Id.Equals(idLogueado) ||
    p.Autor.ListaAmigos.Contains(GetUsuarioLogueado(idLogueado)))
            {
                if (!p.Censurado)
                {
                    listaRet.Add(p);
                }
            }
        }
    }
    return listaRet;
}

// Buscamos un post en nuestros Posts de la lista de publicaciones por su id
public Post GetPostPorId(int idPost)
{
    foreach (Post p in GetPosts())
    {
        if (p.Id.Equals(idPost))
        {
            return p;
        }
    }
    return null;
}

//Listamos los comentarios de un post del que conocemos el id, que estén habilitados para un
    usuario del que conocemos el id
public List<Comentario> GetComentariosHabilitados(int idPost, int? idLogueado)
{
```

```
    List<Comentario> listaRet = new List<Comentario>();
    if (idLogueado != null)
    {
        Post p = GetPostPorId(idPost);
        Usuario u = GetUsuarioLogueado(idLogueado);
        Miembro m = GetMiembroPorEmail(u.Email);
        foreach (Comentario c in p.GetComentarios())
        {
            if (!c.Privado || c.Autor.Id.Equals(idLogueado) ||
    c.Autor.ListaAmigos.Contains(m))
            {
                listaRet.Add(c);
            }
        }
    }
    return listaRet;
}

public Publicacion GetPublicacionPorId(int idPublicacion)
{
    foreach (Publicacion p in _publicaciones)
    {
        if (p.Id.Equals(idPublicacion))
        {
            return p;
        }
    }
    return null;
}

public List<Publicacion> GetPublicacionesHabilitadas(int? idLogueado)
{
    List<Publicacion> listaRet = new List<Publicacion>();
    if (idLogueado != null)
    {
        Usuario u = GetUsuarioLogueado(idLogueado);
        Miembro m = GetMiembroPorEmail(u.Email);
        foreach (Publicacion p in GetPublicaciones())
        {
            if (!p.Privado || p.Autor.Id.Equals(idLogueado) ||
    p.Autor.ListaAmigos.Contains(m))
            {
```

```
                if (p.GetTipo() == "Post")
                {
                    Post po = (Post)p;
                    if (!po.Censurado)
                    {
                        listaRet.Add(po);
                    }
                }
                else
                {
                    listaRet.Add(p);
                }
            }
        }
    }
    return listaRet;
}

public List<Publicacion> GetPublicacionesFiltradas(string buscar, int numero, int? idLogueado)
{
    List<Publicacion> listaHabilitada = GetPublicacionesHabilitadas(idLogueado);
    List<Publicacion> listaRet = new List<Publicacion>();
    foreach (Publicacion p in listaHabilitada)
    {
        if (p.CalcularValorAceptacion() > numero)
        {
            if (p.Texto.IndexOf(buscar, StringComparison.OrdinalIgnoreCase) >= 0 ||
    p.Titulo.IndexOf(buscar, StringComparison.OrdinalIgnoreCase) >= 0)
            {
                listaRet.Add(p);
            }
        }
    }
    return listaRet;
}

public List<Invitacion> GetInvitacionesMiembro(int? idUsuarioLogueado)
{
    List<Invitacion> listaRet = new List<Invitacion>();
    foreach (Invitacion i in _invitaciones)
    {
```

```
        if (i.MiembroSolicitado.Id.Equals(idUsuarioLogueado) && i.Estado ==
    EstadoInvitacion.Pendiente_Aprobacion)
        {
            listaRet.Add(i);
        }
    }
    return listaRet;
}

public Invitacion GetInvitacionPorID(int idInvitacion)
{
    foreach (Invitacion i in _invitaciones)
    {
        if (i.Id.Equals(idInvitacion))
        {
            return i;
        }
    }
    return null;
}

public List<Miembro> GetMiembrosHabilitados(int? idUsuarioLogueado)
{
    List<Miembro> listaRet = new List<Miembro>();
    Usuario u = GetUsuarioLogueado(idUsuarioLogueado);
    Miembro miembro = (Miembro)u;
    foreach (Miembro m in GetMiembros())
    {
        if (!m.Bloqueado && !miembro.ListaAmigos.Contains(m) && m.Id != miembro.Id)
        {
            if (!MiembroTieneInvitacionPendiente(miembro, m))
            {
                listaRet.Add(m);
            }
        }
    }
    return listaRet;
}

#endregion

#region IUConsola
```

```
//Aqui estan los metodos que llamamos en program.

// Método que lista las publicaciones de un usuario dado su correo electrónico
public List<Publicacion> ListarPublicacionesUsuario(string Email)
{
    try
    {
        //Verificamos si el Email esta vacio.
        if (string.IsNullOrEmpty(Email))
        {
            throw new Exception("El mail no puede ser vacio");
        }

        //Verificamos si el Email esta en la lista de usuarios del sistema.
        bool correoEncontrado = false;
        foreach (Usuario u in _usuarios)
        {
            if (u.Email == Email)
            {
                correoEncontrado = true;
                break;

            }

        }
        //Si el Email esta en el sistema, obtenemos las publicaciones asociadas al usuario.
        if (correoEncontrado)
        {
            List<Publicacion> listaPublicaciones = new List<Publicacion>();
            foreach (Publicacion pub in _publicaciones)
            {
                if (pub.Autor.Email == Email)
                {

                    listaPublicaciones.Add(pub);
                }

            }
            return listaPublicaciones;
        }
        else
```

```
        {
            throw new Exception("El correo no se encuentra en nuestra base de datos");
        }
    }
    catch (Exception e)
    {
        throw new Exception(e.Message);
    }


}

// Método que lista los posts que contienen comentarios de un usuario
public List<Post> ListarPostConComentarioUsuario(string Email)
{
    // verificar de hacer metodo extra 'existe usuario'

    try
    {
        //Verificamos si el Email esta vacio.
        if (string.IsNullOrEmpty(Email))
        {
            throw new Exception("El mail no puede ser vacio");
        }
        //Obtenemos el usuario con el Email proporcionado.
        List<Post> listaPost = new List<Post>();
        Miembro usuario = GetMiembroPorEmail(Email);

        //Recorremos las publicaciones y agregamos los post que tienen comentarios del
    usuario.
        foreach (Publicacion pubCaso2 in _publicaciones)
        {
            if (pubCaso2.GetTipo() == "Post")
            {
                Post postAux = pubCaso2 as Post;
                if (UsuarioTieneComentario(usuario, postAux.GetComentarios()))
                {
                    listaPost.Add(postAux);
                }
            }
        }
```

```
        return listaPost;
    }
    catch (Exception e)
    {
        throw new Exception(e.Message);
        throw;
    }
}

// Método que verifica si un usuario tiene al menos un comentario en una lista de comentarios
public bool UsuarioTieneComentario(Usuario usuario, List<Comentario> lista)
{

    foreach (Comentario c in lista)
    {
        if (c.Autor == usuario)
        {
            return true;
        }
    }
    return false;

}

// Método que lista los posts que fueron publicados entre dos fechas
public List<Post> ListarPostEntreFechas(string fecha1, string fecha2)
{
    List<Post> listaPost = new List<Post>();

    //Convertimos las fechas al formato DateTime.
    DateTime fecha1Form = ConvertirStringADateTime(fecha1);
    DateTime fecha2Form = ConvertirStringADateTime(fecha2);

    //Verificamos la diferencia entre fechas para determinar la fecha mas reciente
    TimeSpan diferencia = fecha1Form - fecha2Form;
    DateTime mayor = fecha1Form;

    if (diferencia.TotalDays < 0)
    {
        mayor = fecha2Form;
        fecha2Form = fecha1Form;
    }
```

```
    //Filtramos los posts que fueron publicados entre las fechas proporcionadas
    foreach (Publicacion p in _publicaciones)
    {
        if (p.GetTipo() == "Post")
        {
            Post postAux = p as Post;
            if (postAux.FechaPublicacion >= fecha2Form && postAux.FechaPublicacion <= mayor)
            {
                listaPost.Add(postAux);
            }
        }
    }
    //Ordenamos la lista de forma descendente
    listaPost.Sort();
    //listaPost.Reverse();
    return listaPost;
}

// Método que lista los miembros con más publicaciones.
public List<Miembro> MiembroConMasPublicaciones()
{
    List<Miembro> listaRet = new List<Miembro>();
    int mayor = -1;
    //Iteramos sobre todas las publicaciones y contamos las publicaciones de cada Miembro
    llamando un metodo.
    foreach (Publicacion p in GetPublicaciones())
    {
        int aux = ContarPublicacionesDeMiembro(p.Autor);
        if (aux > mayor)
        {
            listaRet.Clear();
            listaRet.Add(p.Autor);
            mayor = aux;
        }
        else if (aux == mayor)
        {
            //Si el usuario aun no está en la lista lo agrega
            if (!listaRet.Contains(p.Autor))
            {
                listaRet.Add(p.Autor);
            }
```

```
        }
    }

    return listaRet;
}

// Método que cuenta las publicaciones de un miembro dado
public int ContarPublicacionesDeMiembro(Miembro m)
{
    int cantidad = 0;

    // Contamos las publicaciones asociadas a un miembro específico
    foreach (Publicacion p in GetPublicaciones())
    {
        if (p.Autor == m)
        {
            cantidad++;
        }
    }
    return cantidad;
}

public DateTime ConvertirStringADateTime(string fecha)
{
    try
    {
        if (ValidarFormatoFecha(fecha))
        {
            DateTime fechaForm = DateTime.ParseExact(fecha, "dd/MM/yyyy",
    CultureInfo.InvariantCulture);
            return fechaForm;
        }
        else
        {
            throw new Exception("La fecha debe tener formato DD/MM/AAAA");
        }
    }
    catch (Exception e)
    {
        throw new Exception("La fecha debe tener dormato DD/MM/AAAA");
    }
}
```

```
public bool ValidarFormatoFecha(string fecha)
{
    return DateTime.TryParseExact(fecha, "dd/MM/yyyy", CultureInfo.InvariantCulture,
    DateTimeStyles.None, out DateTime fechaForm);
}

#endregion

#region IUWebApp

public bool MiembroTieneInvitacionPendiente(Miembro solicitante, Miembro solicitado)
{
    foreach (Invitacion i in _invitaciones)
    {
        if (i.MiembroSolicitado.Id.Equals(solicitado.Id) &&
    i.MiembroSolicitante.Id.Equals(solicitante.Id))
        {
            if (i.Estado == EstadoInvitacion.Pendiente_Aprobacion)
            {
                return true;
            }
        }
    }
    return false;
}

#endregion

}
}
```

Usuario:

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

namespace Dominio
{
    // La clase Usuario implementa la interfaz IValidable.
```

```
public abstract class Usuario : IValidable
{
    public int Id { get; set; }
    public static int UltimoId { get; set; } = 1;
    public string Email { get; set; }
    public string Contrasenia { get; set; }

    //Creamos los constructores para usuario
    public Usuario()
    {
        Id = UltimoId;
        UltimoId++;
    }

    public Usuario(string email, string contrasenia)
    {
        Id = UltimoId;
        UltimoId++;
        Email = email;
        Contrasenia = contrasenia;
    }

    //Validamos los metodos privados
    public virtual void Validar()
    {
        ValidarContrasenia();
        ValidarEmail();
    }


    private void ValidarContrasenia()
    {
        //Verificamos que la contrasenia no sea nula.
        if (string.IsNullOrEmpty(Contrasenia))
        {
            throw new Exception("La contraseña no puede ser vacia");
        }
        //Segunda verificacion, la contrasenia no puede ser menor que 5
        if (Contrasenia.Length < 5)
        {
            throw new Exception("La contraseña debe tener al menos 5 caracteres");
        }
```

```
        }

        private void ValidarEmail()
        {
            //Verificamos si el Email es nulo.
            if (string.IsNullOrEmpty(Email))
            {
                throw new Exception("El mail no debe estar vacio");
            }
        }

        // Metodo firmado, aplicamos su funcionalidad en los hijos (Administrador y Miembro)
        public abstract string GetTipo();

    }
}
```

Administrador:

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

namespace Dominio
{
    //La clase Administrador hereda de la clase Padre Usuario
    public class Administrador : Usuario
    {
        //Constructores de Administrador.
        public Administrador()
        {

        }

        public Administrador(string email, string contrasenia) : base(email, contrasenia)
        {

        }

        //Metodo para bloquear a un miembro.
        public void BloquearUsuario(Miembro m)
```

```
        {
            //Verificamos si no es nulo y lo marcamos como bloqueado.
            if (m != null)
            {
                m.Bloqueado = true;
            }
        }

        //Metodo para censurar un post
        public void CensurarPost(Post p)
        {
            //Verificamos si el post no es nulo y lo marcamos como censurado.
            if (p != null)
            {
                p.Censurado = true;
            }
        }

        //Metodo que devuelve el tipo de usuario (Administrador)
        public override string GetTipo()
        {
            return "Administrador";
        }

    }
}
```

Miembro:

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

namespace Dominio
{
    // La clase Miembro hereda de la clase base Usuario e implementa la interfaz IValidable
    public class Miembro : Usuario, IValidable, IComparable<Miembro>
    {
        public string Nombre { get; set; }
        public string Apellido { get; set; }
```

```
    public DateTime FechaNacimiento { get; set; }
    public List<Miembro> ListaAmigos { get; } = new List<Miembro>();
    public bool Bloqueado { get; set; }

    //Creamos los constructores.
    public Miembro()
    {

    }

    public Miembro(string nombre, string apellido, DateTime fechaNacimiento, bool
bloqueado, string email, string contrasenia) : base(email, contrasenia)
    {
        Nombre = nombre;
        Apellido = apellido;
        FechaNacimiento = fechaNacimiento;
        ListaAmigos = new List<Miembro>(); //Creamos una nueva lista de amigos cuando se
crea el Miembro
        Bloqueado = bloqueado;
    }

    #region VALIDACIONES

    //Implementacion del metodo validar de la interfaz Ivalidable
    public override void Validar()
    {
        base.Validar();// Validacion de la clase base Usuario
        ValidarNombreApellido();
    }

    // Método privado para validar que el nombre y el apellido no estén vacíos
    private void ValidarNombreApellido()
    {
        //Verificamos si es nulo el nombre.
        if (string.IsNullOrEmpty(Nombre))
        {
            throw new Exception("El nombre no puede estar vacio");
        }
        //Verificamos si es nulo el apellido.
        if (string.IsNullOrEmpty(Apellido))
        {
            throw new Exception("El apellido no puede estar vacio");
```

```
        }
    }

    #endregion

    //Metodo para agregar un amigo a la lista de amigos del miembro
    public void AgregarAmigo(Miembro miembroAmigo)
    {
        //Verificamos si el miembro no esta ya en la lista de amigos
        if (!ListaAmigos.Contains(miembroAmigo))
        {
            //Verificamos si ninguno de los miembros esta bloqueado para agregar un amigo.
            if (!Bloqueado && !miembroAmigo.Bloqueado)
            {
                ListaAmigos.Add(miembroAmigo);
            }
            else if(!Bloqueado)
            {
                //Lanzamos una excepcion si uno de los miembros esta bloqueado.
                throw new Exception("No se puede aceptar la solicitud, el usuario
seleccionado está bloqueado");
            }
            else if (!miembroAmigo.Bloqueado)
            {
                //Lanzamos una excepcion si ambos miembros están bloqueados.
                throw new Exception("No se puede aceptar la solicitud, usted está
bloqueado");
            }
            else
            {
                //Lanzamos una excepcion si ambos miembros están bloqueados.
                throw new Exception("No se puede aceptar la solicitud, los miembros no
pueden estar bloqueados");
            }
        }
        else
        {
            //Lanzamos una excepcion si el miembro ya es amigo del actual.
            throw new Exception($"{Nombre} ya es amigo de {miembroAmigo.Nombre}");
        }
    }

    //Metodo que devuelve el tipo de usuario('Miembro').
```

```
    public override string GetTipo()
    {
        return "Miembro";
    }

    // Método que sobrescribe el método ToString con un mensaje personalizado.
    public override string ToString()
    {
        return $"Nombre: {Nombre}, Apellido: {Apellido}, Fecha Nacimiento:
{FechaNacimiento}, Email: {Email}.";
    }

    public int CompareTo(Miembro other)
    {
        string nombreCompleto = Apellido + " " + Nombre;
        string otherNombreCompleto = other.Apellido + " " + other.Nombre;
        return nombreCompleto.CompareTo(otherNombreCompleto);
    }

    #region UtilizableAFuturo

    //public List<Miembro> GetAmigosMiembro()

    //EVALUAR SI ES NECESARIO
    //{
    //    return ListaAmigos;
    //}

    public void AceptarSolicitud(Invitacion i)
    {
        if (i.Estado == EstadoInvitacion.Pendiente_Aprobacion)
        {
            i.Estado = EstadoInvitacion.Aprobada;
            AgregarAmigo(i.MiembroSolicitante);
            i.MiembroSolicitante.AgregarAmigo(this); // i.MiembroSolicitado es una
alternativa al this
        }
    }

    public void RechazarSolicitud(Invitacion i)
    {
        if (i.Estado == EstadoInvitacion.Pendiente_Aprobacion)
```

```
            {
                i.Estado = EstadoInvitacion.Rechazada;
            }
        }


        #endregion

    }
}
```

Invitacion:

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

namespace Dominio
{
    // En esta clase utilizamos la interface IValidable
    public class Invitacion : IValidable
    {
        public int Id { get; set; }
        public static int UltimoId { get; set; } = 1;

        public Miembro MiembroSolicitante { get; set; }
        public Miembro MiembroSolicitado { get; set; }
        public EstadoInvitacion Estado { get; set; }
        public DateTime FechaSolicitud { get; set; }


        //Creamos los contructores
        public Invitacion()
        {
            Id = UltimoId;
            UltimoId++;
        }

        public Invitacion(Miembro miembroSolicitante, Miembro miembroSolicitado,
    EstadoInvitacion estado)
```

```
    {
        Id = UltimoId;
        UltimoId++;
        MiembroSolicitante = miembroSolicitante;
        MiembroSolicitado = miembroSolicitado;
        Estado = estado;
        FechaSolicitud = DateTime.Now; //La fecha de la invitacion siempre va a ser su
fecha de creación
    }

    #region VALIDACIONES

    // Implementación del método de la interfaz IValidable
    public void Validar()
    {
        ValidarMiembroNoVacio();
        ValidarMiembrosNoBloqueados();
    }

    //posibilidad de separar validar miembro bloqueado por solicitante y solicitado o
dejarlos en un metodo

    //Metodo privado para validar que los miembros no esten bloqueados
    private void ValidarMiembrosNoBloqueados()
    {
        if (MiembroSolicitante.Bloqueado || MiembroSolicitado.Bloqueado)
        {
            throw new Exception("Los miembros no pueden estar bloqueados por el
administrador para enviar o recibir invitaciones");
        }
    }

    // Método privado para validar que los miembros no sean nulos
    private void ValidarMiembroNoVacio()
    {
        if (MiembroSolicitante == null || MiembroSolicitado == null)
        {
            throw new Exception($"Ambos miembros deben existir en el sistema");
        }
    }

    #endregion
```

```
        //Metodo ToString para mostrar un mensaje personalizado
        public override string ToString()
        {
            return "Invitación";
        }

        public string TiempoTranscurrido() //Calcula el tiempo que pasó desde la invitacion a
    la hora actual
        {
            TimeSpan diferencia = DateTime.Now - FechaSolicitud;
            double segundos = Math.Round(diferencia.TotalSeconds);
            string ret = segundos.ToString();
            if (segundos > 59)
            {
                double minutos = Math.Round(diferencia.TotalMinutes);
                if (minutos >= 1 && minutos <= 59)
                {
                    ret = minutos.ToString();
                    return $"Hace {ret} minutos";
                }
                else if (minutos > 59 && minutos < 1440)
                {
                    double horas = Math.Round(diferencia.TotalHours);
                    ret = horas.ToString();
                    return $"Hace {ret} horas";
                }
                else if (minutos >= 1440)
                {
                    double dias = Math.Round(diferencia.TotalDays);
                    ret = dias.ToString();
                    return $"Hace {ret} dias";
                }
            }
            return $"Hace {ret} segundos";
        }

    }
}
```

Publicacion:

```
using System;
```

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

namespace Dominio
{

    // La clase abstracta Publicacion implementa las interfaces IValidable e IComparable<Post>
    public abstract class Publicacion : IValidable, IComparable<Post>
    {
        public int Id { get; set; }

        public static int UltimoId { get; set; } = 1;

        public Miembro Autor { get; set; }

        public DateTime FechaPublicacion { get; set; }

        public string Texto { get; set; } // Tomamos como texto al contenido de las
    publicaciones

        public string Titulo { get; set; }

        public bool Privado { get; set; }

        //Protejemos la lista, haciéndola privada y sin set
        private List<Reaccion> _reacciones { get; } = new List<Reaccion>();



        //Creacion de constructores
        public Publicacion()
        {
            Id = UltimoId;
            UltimoId++;
        }

        public Publicacion(Miembro autor, string titulo, string texto, bool privado,
    List<Reaccion> reacciones)
        {
            Id = UltimoId;
            UltimoId++;
```

```
        Autor = autor;
        FechaPublicacion = DateTime.Now;
        Titulo = titulo;
        Privado = privado;
        _reacciones = reacciones;
        Texto = texto;
    }


    // Método para agregar una reacción a la publicación.
    public void AgregarReaccion(Reaccion r)
    {
        // Evaluar si es necesaria
        // Evaluamos si el usuario ya reaccionó y actualizamos la reacción existente o
agregamos una nueva
        bool usuarioYaReacciono = false;

        foreach (Reaccion reaccionExistente in _reacciones)
        {
            if (reaccionExistente.Autor == r.Autor)
            {
                usuarioYaReacciono = true;
                if (reaccionExistente.Tipo.Equals(r.Tipo))
                {
                    _reacciones.Remove(reaccionExistente);
                }
                else
                {
                    reaccionExistente.Tipo = r.Tipo;
                }
                break;
            }
        }
        if (!usuarioYaReacciono)
        {
            _reacciones.Add(r);
        }
    }

    public List<Reaccion> GetReacciones()
    {
        return _reacciones;
```

```
}

public int GetReaccionesTipo(string tipo)
{
    int cantidadReacciones = 0;
    foreach (Reaccion r in _reacciones)
    {
        if (r.ToString() == tipo)
        {
            cantidadReacciones++;
        }
    }
    return cantidadReacciones;
}

#region VALIDACIONES

// Implementación del método Validar de la interfaz IValidable
public virtual void Validar()
{
    // Llamamos a los metodos privados de validación
    ValidarTexto();
    ValidarTitulo();
}

// Método privado para validar que el texto no esté vacío
private void ValidarTexto()
{
    //Validamos texto no vacío
    if (string.IsNullOrEmpty(Texto))
    {
        throw new Exception("El texto no debe estar vacio");
    }
}

private void ValidarTitulo()
{
    //Verificamos si el titulo no es vacio
    if (string.IsNullOrEmpty(Titulo)) //Validamos titulo no vacío
    {
        throw new Exception("El titulo no debe estar vacio");
    }
```

```
        //Validamos que el titulo tenga al menos 3 caracteres.
        if (Titulo.Length < 3)
        {
            throw new Exception("El titulo debe tener al menos 3 caracteres");
        }
    }


    #endregion


    #region ManejoDeString


    // Implementación del método ToString para generar un mensaje personalizado.
    public override string ToString()
    {
        return $"Tipo: {GetTipo()}; Id : {Id}, Fecha de Publicacion :  {FechaPublicacion},
Titulo : {Titulo}, Texto: {CortarTexto(Texto)}..."; // Redefinimos toString para
adaptarlo a nuestras necesidades
    }


    // Metodo que corta un texto para que no supere los 50 caracteres, si el texto dado
tiene un largo menor a 50 caracteres no necesita ser cortado
    public string CortarTexto(string txt)
    {
        if (txt.Length > 50)
        {
            string nuevoTexto = "";
            for (int i = 0; i <= txt.Length; i++)
            {
                if (i < 50)
                {
                    nuevoTexto += txt[i];
                }
                else
                {
                    break;
                }
            }
            return nuevoTexto;
        }
        else
        {
            return txt;
        }
```

```
        }

        #endregion
        // Métodos abstractos para ser implementados en las clases derivadas
        public abstract int CalcularValorAceptacion(); // Solo firma

        public abstract string GetTipo(); // Solo firma


        // Metodo de la interfaz IComparable<Post> para establecer un criterio de
    ordenamiento.
        public int CompareTo(Post other)
        {
            if (Titulo.CompareTo(other.Titulo) > 0)
            {
                return -1;
            }
            else if (Titulo.CompareTo(other.Titulo) < 0)
            {
                return 1;
            }
            else
            {
                return 0;
            }
        }
        public string TiempoTranscurrido() //Calcula el tiempo que pasó desde la publicacion a
    la hora actual
{
    TimeSpan diferencia = DateTime.Now - FechaPublicacion;
    double segundos = Math.Round(diferencia.TotalSeconds);
    string ret = segundos.ToString();
    if(segundos > 59)
    {
        double minutos = Math.Round(diferencia.TotalMinutes);
        if (minutos >= 1 && minutos <= 59)
        {
            ret = minutos.ToString();
            return $"Hace {ret} minutos";
        }
        else if (minutos > 59 && minutos < 1440)
        {
```

```
            double horas = Math.Round(diferencia.TotalHours);
            ret = horas.ToString();
            return $"Hace {ret} horas";
        }
        else if (minutos >= 1440)
        {
            double dias = Math.Round(diferencia.TotalDays);
            ret = dias.ToString();
            return $"Hace {ret} dias";
        }
    }
    return $"Hace {ret} segundos";
}
}
}
```

Post:

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

namespace Dominio
{

    // La clase Post hereda de Publicacion e implementa la interfaz IValidable
    public class Post : Publicacion, IValidable
    {
        public string Imagen { get; set; }
        private List<Comentario> _comentarios { get; } = new List<Comentario>();
        public bool Censurado { get; set; }

        //Creacion de constructores
        public Post()
        {

        }

        public Post(string imagen, List<Comentario> comentarios, bool censurado, Miembro
    autor, string titulo, string texto, bool privado, List<Reaccion> reacciones) :
    base(autor, titulo, texto, privado, reacciones)
```

```
    {
        Imagen = imagen;
        Censurado = censurado;
        _comentarios = comentarios;

    }

    //Generamos un constructor para poder cargar post con fechas anteriores y así poder darle uso a la 3ra funcionalidad del Menú usuario
    public Post(string imagen, List<Comentario> comentarios, bool censurado, Miembro autor, DateTime fecha, string titulo, string texto, bool privado, List<Reaccion> reacciones) : base(autor, titulo, texto, privado, reacciones)
    {
        Imagen = imagen;
        Censurado = censurado;
        _comentarios = comentarios;
        FechaPublicacion = fecha;

    }

    //Metodo para agregar comentarios a la lista de comentarios si es que se cumplen las validaciones impuestas, por lo contrario se lanzarán las excepciones correspondientes
    public void AltaComentario(Comentario c)
    {
        try
        {
            if (c != null)
            {
                c.Validar();
                _comentarios.Add(c);
            }
        }
        catch (Exception e)
        {
            throw new Exception(e.Message);
        }
    }

    // Metodo para obtener la lista de comentarios de manera segura, sin acceder directamente a la lista privada
    public List<Comentario> GetComentarios()
    {
```

```
        return _comentarios;
    }



    // Implementación del método abstracto GetTipo de la clase base
    public override string GetTipo()
    {
        return "Post";
    }

    #region VALIDACIONES

    // Implementación del método abstracto Validar de la interfaz IValidable
    public override void Validar()
    // Utilizamos validaciones del padre (Publicacion) y ademas validamos extension
    {
        base.Validar();
        ValidarExtension();
    }

    //Validamos que el texto de la imagen no sea vacío y la extencion de la imagen sea la
correcta
    private void ValidarExtension()

    {
        if (string.IsNullOrEmpty(Imagen))
        {
            throw new Exception("La imagen del post no puede estar vacia");
        }

        //Usamos la opción de hacerlo insensible a mayusculas y minusculas con
".OrdinalIgnoreCase"

        if (!Imagen.EndsWith(".jpg", StringComparison.OrdinalIgnoreCase) &&
!Imagen.EndsWith(".png", StringComparison.OrdinalIgnoreCase))

        {
            // Solo lanza la excepcion en caso que la extencion no termine en las dos
formas
            throw new Exception("La extension de la imagen debe ser (.jpg) o (.png)");
        }

    }
```

```
        #endregion

        #region UtilizableAFuturo

        // El metodo lo utilizaremos a futuro
        public override int CalcularValorAceptacion()
        {
            int likes = GetReaccionesTipo("like");
            int dislikes = GetReaccionesTipo("dislike");
            int retorno = (likes * 5) + (dislikes * -2);
            if (!Privado)
            {
                retorno += 10;
            }
            return retorno;
        }

        //public void MostrarPostPrivado(List<Miembro> listaAmigo)  //Lo guardamos para el
    futuro
        //{
        //    if (Privado && !listaAmigo.Contains(Autor))
        //    {
        //        throw new Exception("No tiene acceso al post privado");
        //    }
        //}

        #endregion


    }
}
```

Comentario:

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

namespace Dominio
{
```

```
//La clase Comentario hereda de Publicacion.
public class Comentario : Publicacion
{

    //Creacion de contructores
    public Comentario()
    {

    }

    public Comentario(Miembro autor, string titulo, string texto, bool privado,
List<Reaccion> reacciones) : base(autor, titulo, texto, privado, reacciones)
    {

    }

    // Implementación del método abstracto Validar de la clase base
    public override void Validar()

    {
        // Utilizamos las mismas validaciones del Padre (Publicacion)
        base.Validar();
    }

    // Implementación del método abstracto GetTipo de la clase base
    public override string GetTipo()
    {
        return "Comentario";
    }

    // Implementación del método abstracto CalcularValorAceptacion de la clase base
    public override int CalcularValorAceptacion()
    {
        int likes = GetReaccionesTipo("like");
        int dislikes = GetReaccionesTipo("dislike");
        int retorno = (likes * 5) + (dislikes * -2);

        return retorno;
    }

}
```

```
}
```

Reaccion:

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

namespace Dominio
{
    public class Reaccion
    {
        public int Id { get; set; }
        public static int UltimoId { get; set; } = 1;
        public Tipo Tipo { get; set; }
        public Miembro Autor { get; set; }

        //Creamos los constructores
        public Reaccion()
        {
            Id = UltimoId;
            UltimoId++;
        }

        public Reaccion(Tipo tipo, Miembro autor)
        {
            Id = UltimoId;
            UltimoId++;
            Tipo = tipo;
            Autor = autor;
        }

        //Sobreescribimos el metodo ToString para mostrar un mensaje personalizado
        public override string ToString()
        {
            return $"{Tipo}";
        }
    }
}
```

Tipo:

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

namespace Dominio
{
    //Enumeramos los tipo de reacciones posibles.
    public enum Tipo
    {
        like,
        dislike
    }
}
```

IValidable:

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

namespace Dominio
{
    public interface IValidable
    {
        //firma del metodo sin implementacion
        public void Validar();
    }
}
```

EstadoInvitacion:

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

```
namespace Dominio
{
    //Enumeramos los posibles estados de una invitacion
    public enum EstadoInvitacion
    {
        Pendiente_Aprobacion,
        Aprobada,
        Rechazada
    }
}
```

1.5.2 IU WebApp:

HomeController:

```
using Dominio;
using IUWebApp.Models;
using Microsoft.AspNetCore.Mvc;
using System.Diagnostics;

namespace IUWebApp.Controllers
{
    public class HomeController : Controller
    {
        Sistema s = Sistema.Instancia();
        public IActionResult Index()
        {
            int? idUsuarioLogueado = HttpContext.Session.GetInt32("idUsuarioLogueado");
            string? rolUsuarioLogueado = HttpContext.Session.GetString("tipoUsuarioLogueado");
            Usuario u = s.GetUsuarioLogueado(idUsuarioLogueado);

            if (u != null)
            {
                if (rolUsuarioLogueado == "Administrador")
                {
                    ViewBag.msg = "Administrador";
                }
                else
                {
                    Miembro m = (Miembro)u;
                    ViewBag.msg = "Miembro " + m.Nombre;
```

```
                if (m.Bloqueado)
                {
                    ViewBag.msgBloqueado = "Le informamos que usted ha sido bloqueado por un administrador de la red, por lo cual tendrá acciones restringidas";
                }
            }
        }
        else
        {
            ViewBag.msg = "Inicie Sesión para acceder a las funcionalidades de nuestra red, si aún no está registrado puede Registrarse de froma gratuita";
        }
        return View();
    }

    public IActionResult Privacy()
    {
        return View();
    }
    public IActionResult IniciarSesion()
    {
        return View();
    }

    [HttpPost]
    public IActionResult IniciarSesion(string Email, string Contrasenia)
    {
        try
        {
            Usuario usuarioBuscado = s.GetUsuarioPorEmail(Email);
            if (usuarioBuscado.Contrasenia.Equals(Contrasenia))
            {
                HttpContext.Session.SetInt32("idUsuarioLogueado", usuarioBuscado.Id);
                HttpContext.Session.SetString("tipoUsuarioLogueado", usuarioBuscado.GetTipo());
                return RedirectToAction("Index");
            }
            else
            {
                ViewBag.msgError = "El usuario y la contraseña no coinciden";
                return View();
            }
```

```
        }
        catch (Exception e)
        {
            ViewBag.msgError = e.Message;
            return View();
        }

    }

    public IActionResult Registrarse()
    {
        return View();
    }

    [HttpPost]
    public IActionResult Registrarse(Miembro m)
    {
        try
        {
            s.AltaUsuario(m);
            ViewBag.msgOk = "Se ha registrado correctamente";
            return View();
        }
        catch (Exception e)
        {
            ViewBag.msgError = e.Message;
            return View();
        }
    }

    public IActionResult Logout()
    {
        string? rolUsuarioLogueado = HttpContext.Session.GetString("tipoUsuarioLogueado");
        if (rolUsuarioLogueado != null)
        {
            HttpContext.Session.Clear();
            TempData.Clear(); //Intento de eliminar loa TempData al cerrar sesión
        }
        return RedirectToAction("Index");
    }

}
```

```
}
```

Vista Index:

```
<link href="@Url.Content("~/css/indexHome.css")" rel="stylesheet" />
@{
    ViewData["Title"] = "Inicio";
}

<div class="text-center">
    <h1 class="display-4">Bienvenido a Soccer Social Network </h1>
    <h4>@ViewBag.msg</h4>
    <h5>@ViewBag.msgBloqueado</h5>
</div>
<div class="image">
    <img src="img/social.jpg"/>
</div>
```

Vista Privacy:

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

<h2>Universidad ORT Uruguay</h2>
<h4>@ViewData["Title"]</h4>

<ul>
    <li>
        Angel Vaz - N° 320162
    </li>
    <li>
        Rodrigo Cristaldo - N° 310992
    </li>
</ul>

<h4>Tutor</h4>

<ul>
    <li>
        Andres Ureta
    </li>
</ul>
```

Vista IniciarSesion:

```
@using Dominio;

@model Dominio.Usuario;

@{
    ViewData["Title"] = "Iniciar Sesión";
}

<div class="container mt-5">
    <h1 class="text-center">Iniciar Sesión</h1>
    <hr class="my-4">
    <div class="row justify-content-center">

        <div class="col-md-6">

            <form method="post">
                <div class="form-group">
                    <input type="email" class="form-control mt-3" asp-for="Email"
    placeholder="Email" required />
                </div>
                <div class="form-group">
                    <input type="password" class="form-control mt-3" asp-for="Contrasenia"
    placeholder="Contraseña" required />
                </div>
                <input type="submit" value="Ingresar" class="btn btn-primary btn-block mt-3"
    />
            </form>
            @if (ViewBag.msgError != null)
            {
                <div class="alert alert-danger mt-3">
                    <p>@ViewBag.msgError</p>
                </div>
            }
            else
            {
                <div class="alert alert-success mt-3">
                    <p>@ViewBag.msgOk</p>
                </div>
            }
            <hr class="my-4">
            <span class="d-block text-center">
```

```
                ¿Aún no tienes una cuenta? @Html.ActionLink("Regístrate", "Registrarse",
    "Home", null, new { @class = "btn btn-outline-primary" })
            </span>
        </div>


    </div>
</div>


<div class="mt-3">
    @Html.ActionLink("Atrás", "Index", "Home", null, new { @class = "btn btn-outline-
    secondary"})
</div>
```

Vista Registrarse:

```
@using Dominio;


@model Dominio.Miembro;


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


<h1 class="text-center">Registrarse como Usuario</h1>
<hr />
<div class="row">
    <div class="col-md-4"></div>
    <div class="col-md-4">
        <form method="post">
            <div class="form-group">
                <input type="text" asp-for="Nombre" placeholder="Nombre" class="form-control
    mt-3" required />
            </div>
            <div class="form-group">
                <input type="text" asp-for="Apellido" placeholder="Apellido" class="form-
    control mt-3" required />
            </div>
            <div class="form-group mt-3">
                <label class="control-label">Fecha de Nacimiento</label>
                <input type="date" asp-for="FechaNacimiento" value="1990-01-01" class="form-
    control" required />
            </div>
            <div class="form-group">
```

```
                <input type="email" asp-for="Email" placeholder="Email" class="form-control
    mt-3" required />


            </div>
            <div class="form-group">
                <input type="password" asp-for="Contrasenia" placeholder="Contraseña"
    class="form-control mt-3" required />
            </div>
            <div class="form-group  mt-3">
                <input type="checkbox" class="form-check-input" required />
                <label class="control-label">Acepta nuestros Términos y Condiciones</label>
            </div>
            <div class="form-group mt-3">
                <input type="submit" value="Confirmar Registro" class="btn btn-primary" />
            </div>

        </form>
    </div>
</div>

@if (ViewBag.msgError != null)
{
    <div class="msg alert-danger">
        <p>@ViewBag.msgError</p>
    </div>
}
else
{
    <div class="msg alert-success">
        <p>@ViewBag.msgOk</p>
    </div>
}
<div class="mt-3">
    @Html.ActionLink("Atrás","Index","Home",null, new {@class="btn btn-outline-secondary"})
</div>
```

UsuarioController:

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

namespace IUWebApp.Controllers
{
```

```
public class UsuarioController : Controller
{
    Sistema s = Sistema.Instancia();

    public IActionResult Index() //Mostramos los Miembros habilitados para el usuario
logueado, (si es Admin mostramos todos)
    {
        int? idUsuarioLogueado = HttpContext.Session.GetInt32("idUsuarioLogueado");
        if (idUsuarioLogueado != null)
        {
            string? tipoUsuarioLogueado =
HttpContext.Session.GetString("tipoUsuarioLogueado");

            if (tipoUsuarioLogueado == "Administrador")
            {
                return View(s.GetMiembros());
            }
            else
            {
                return View(s.GetMiembrosHabilitados(idUsuarioLogueado));
            }
        }
        else
        {
            return RedirectToAction("Index", "Home");
        }

    }

    public IActionResult Edit(string emailMiembro)
    {
        string? tipoUsuarioLogueado =
HttpContext.Session.GetString("tipoUsuarioLogueado");
        if (tipoUsuarioLogueado == "Administrador")
        {
            return View(s.GetMiembroPorEmail(emailMiembro));
        }
        else
        {
            return RedirectToAction("Index", "Home");
        }
    }
```

```
    [HttpPost]
    public IActionResult Edit(bool Check, string emailMiembro)
    {
        string? tipoUsuarioLogueado =
HttpContext.Session.GetString("tipoUsuarioLogueado");
        if (tipoUsuarioLogueado == "Administrador")
        {
            if (Check)
            {
                int? idUsuarioLogueado =
HttpContext.Session.GetInt32("idUsuarioLogueado");
                Usuario a = s.GetUsuarioLogueado(idUsuarioLogueado);
                Miembro miembroBuscado = s.GetMiembroPorEmail(emailMiembro);

                Administrador admin = (Administrador)a;
                admin.BloquearUsuario(miembroBuscado);
            }
            else
            {
                ViewBag.msg = "Debe seleccionar la opción para bloquear";
            }
            return RedirectToAction("Index");
        }
        else
        {
            return RedirectToAction("Index", "Home");
        }

    }

    public IActionResult Solicitudes()
    {
        int? idUsuarioLogueado = HttpContext.Session.GetInt32("idUsuarioLogueado");
        string? tipoUsuarioLogueado =
HttpContext.Session.GetString("tipoUsuarioLogueado");

        if (tipoUsuarioLogueado == "Miembro")
        {
            return View(s.GetInvitacionesMiembro(idUsuarioLogueado));
        }
        else
        {
            return RedirectToAction("Index", "Home");
```

```
        }
    }

    public IActionResult ProcesarSolicitud(string respuesta, int idInvitacion)
    {
        string? tipoUsuarioLogueado =
HttpContext.Session.GetString("tipoUsuarioLogueado");
        if (tipoUsuarioLogueado == "Miembro")
        {
            int? idUsuarioLogueado = HttpContext.Session.GetInt32("idUsuarioLogueado");
            Usuario u = s.GetUsuarioLogueado(idUsuarioLogueado);
            Miembro m = (Miembro)u;
            Invitacion invitacion = s.GetInvitacionPorID(idInvitacion);
            try
            {
                if (respuesta == "aceptar")
                {
                    m.AceptarSolicitud(invitacion);
                    TempData["solicitud"] = $"Has aceptado la solicitud de
{invitacion.MiembroSolicitante.Nombre} {invitacion.MiembroSolicitante.Apellido}";
                }
                else
                {
                    m.RechazarSolicitud(invitacion);
                    TempData["solicitud"] = $"Has rechazado la solicitud de
{invitacion.MiembroSolicitante.Nombre} {invitacion.MiembroSolicitante.Apellido}";
                }
                return RedirectToAction("Solicitudes");
            }
            catch (Exception e)
            {
                TempData["solicitud"] = e.Message;
                return RedirectToAction("Solicitudes");
            }

        }
        else
        {
            return RedirectToAction("Index", "Home");
        }
    }

    public IActionResult Solicitar(string emailMiembro)
```

```
        {
            string? tipoUsuarioLogueado =
    HttpContext.Session.GetString("tipoUsuarioLogueado");
            if(tipoUsuarioLogueado == "Miembro")
            {
                int? idLogueado = HttpContext.Session.GetInt32("idUsuarioLogueado");
                Usuario u = s.GetUsuarioLogueado(idLogueado);
                Miembro miembroSolicitante = (Miembro)u;
                Miembro miembroSolicitado = s.GetMiembroPorEmail(emailMiembro);
                try
                {
                    if (!s.MiembroTieneInvitacionPendiente(miembroSolicitante,
    miembroSolicitado)) /* && !miembroSolicitante.Bloqueado Lo podemos agregar, pero no nos
    tira el mensaje personalizado del Exception*/
                    {
                        s.NuevaInvitacion(miembroSolicitante, miembroSolicitado);
                        TempData["solicitudOk"] = $"Solicitud Enviada a
    {miembroSolicitado.Nombre} {miembroSolicitado.Apellido}";
                    }
                    return View("Index", s.GetMiembrosHabilitados(idLogueado));
                }
                catch (Exception e)
                {
                    TempData["solicitudError"] = e.Message;
                    return View("Index", s.GetMiembrosHabilitados(idLogueado));
                }

            }
            else
            {
                return RedirectToAction("Index", "Home");
            }
        }

    }
}
```

Vista Index:

```
@using Dominio;

@model IEnumerable<Dominio.Miembro>;
@if (Context.Session.GetString("tipoUsuarioLogueado") == "Administrador")
{
    ViewData["Title"] = "Miembros";
```

```
    <h1 class="text-center">Lista de Miembros</h1>
}
else
{
    ViewData["Title"] = "Agregar Amigos";

    <h1 class="text-center">Personas que quizás conozcas</h1>
}
<hr />

@if (Model != null && Model.Count() != 0)
{
    <table class="table">

        <tr>
            @if (Context.Session.GetString("tipoUsuarioLogueado") == "Administrador")
            {
                <th>ID</th>
                <th>Nombre</th>
                <th>Fecha de Nacimiento</th>
                <th>Correo</th>
                <th>Estado</th>
            }
            else
            {
                <th>Nombre</th>
                <th>Correo</th>
                <th>Solicitar</th>
            }


        </tr>


        @foreach (Miembro m in Model)
        {
            @if (m != null)
            {
                <tr>
                    @if (Context.Session.GetString("tipoUsuarioLogueado") == "Administrador")
                    {
```

```
                        <td>@m.Id</td>
                        <td>@m.Apellido @m.Nombre</td>
                        <td>@m.FechaNacimiento</td>
                        <td>@m.Email</td>

                        @if (m.Bloqueado)
                        {
                            <td>Bloqueado</td>
                        }
                        else
                        {
                            <td>@Html.ActionLink("Bloquear","Edit","Usuario", new{emailMiembro
    = m.Email}, new{@class="btn btn-outline-danger btn-sm"})</td>
                        }
                    }
                    else
                    {
                        <td>@m.Nombre @m.Apellido</td>
                        <td>@m.Email</td>
                        <td>@Html.ActionLink("Enviar Solicitud","Solicitar","Usuario",
    new{emailMiembro = m.Email}, new{@class="btn btn-outline-primary btn-sm"})</td>
                    }
                </tr>
            }
        }
    </table>


}
else
{
    <span>No hay Usuarios para mostrar</span>
}
@if (TempData["solicitudOk"] != null)
{
    <div class="alert-success mt-3">
        <p>@TempData["solicitudOk"]</p>
    </div>
}
else
{
    <div class="alert-danger mt-3">
```

```
        <p>@TempData["solicitudError"]</p>
    </div>
}
<div class="mt-3">
    @Html.ActionLink("Atrás", "Index", "Home", null, new { @class = "btn btn-outline-secondary
    btn-sm"})
</div>
```

Vista Edit:

```
@model Dominio.Miembro
@{
    ViewData["Title"] = "Bloquear Usuario";
}

<h1>Bloquear Usuario</h1>

<form method="post">
    <span>Está seguro de bloquear al usuario <strong>@Model.Nombre
    @Model.Apellido</strong>?</span>
    <input type="checkbox" name="Check" value="true" required/>
    <br />
    <input class="btn btn-danger btn-sm mt-3" type="submit" value="Bloquear"/>
</form>

<p class="alert-warning"> @ViewBag.msg </p>

<div class="mt-3">
    @Html.ActionLink("Atrás", "Index", "Usuario", null, new { @class = "btn btn-outline-
    secondary btn-sm"})
</div>
```

Vista Solicitudes:

```
@using Dominio;
@model IEnumerable<Dominio.Invitacion>

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

<h1 class="text-center">Solicitudes de Amistad</h1>
<hr />
@if (Model != null && Model.Count() != 0)
```

```
{
    <table class="table">
        <tr>
            <th>Miembro</th>
            <th>Fecha</th>
            <th></th>
            <th></th>
            <th></th>
        </tr>

        @foreach (Invitacion i in Model)
        {
            <tr>
                <td>@i.MiembroSolicitante.Nombre @i.MiembroSolicitante.Apellido</td>
                <td>@i.FechaSolicitud</td>
                <td>@i.TiempoTranscurrido()</td>
                @if (!i.MiembroSolicitado.Bloqueado)
                {
                    <td>@Html.ActionLink("Aceptar", "ProcesarSolicitud", "Usuario",
    new{respuesta="aceptar", idInvitacion = i.Id}, new { @class = "btn btn-outline-success
    btn-sm"})</td>
                    <td>@Html.ActionLink("Rechazar", "ProcesarSolicitud", "Usuario",
    new{respuesta="rechazar", idInvitacion = i.Id}, new { @class = "btn btn-outline-danger
    btn-sm"})</td>
                }
                else
                {
                    <td><span class="alert-danger">Usted ha sido bloqueado, no puede aceptar
    ni rechazar solicitudes</span></td>
                }

            </tr>
        }
    </table>

}
else
{
    <span>No tienes Solicitudes de amistad pendientes!</span>
}
@if (TempData["solicitud"] != null)
{
    <div class="alert alert-info mt-3">
```

```
        <p>@TempData["solicitud"]</p>
    </div>
}
<div class="mt-3">
    @Html.ActionLink("Atrás", "Index", "Home", null, new { @class = "btn btn-outline-secondary
    btn-sm"})
</div>
```

PublicacionController:

```
using Dominio;
using Microsoft.AspNetCore.Mvc;
using Microsoft.Extensions.Hosting;

namespace IUWebApp.Controllers
{
    public class PublicacionController : Controller
    {
        Sistema s = Sistema.Instancia();

        public IActionResult Index() //Mostramos los Posts habilitados para el usuario
    logueado, (si es Admin mostramos todos)
        {
            string? rolLogueado = HttpContext.Session.GetString("tipoUsuarioLogueado");
            if (rolLogueado != null)
            {
                if (rolLogueado == "Administrador")
                {
                    return View(s.GetPosts());
                }
                else
                {
                    int? idLogueado = HttpContext.Session.GetInt32("idUsuarioLogueado");
                    Miembro m = (Miembro)s.GetUsuarioLogueado(idLogueado);
                    if (m.Bloqueado)
                    {
                        TempData["bloqueado"] = "Bloqueado";
                    }
                    return View(s.GetPostsHabilitados(idLogueado));
                }
            }
            return RedirectToAction("Index", "Home");
        }
```

```
    public IActionResult Buscar() //Buscador de Publicaciones
    {
        string? rolLogueado = HttpContext.Session.GetString("tipoUsuarioLogueado");
        if (rolLogueado == "Miembro")
        {
            return View();
        }
        return RedirectToAction("Index", "Home");
    }


    [HttpPost]
    public IActionResult Buscar(string buscar, int numero) //Filtramos las publicaciones
segun los datos que brinda el usuario
    {
        if (!String.IsNullOrWhiteSpace(buscar))
        {
            string? rolLogueado = HttpContext.Session.GetString("tipoUsuarioLogueado");
            if (rolLogueado == "Miembro")
            {
                int? idLogueado = HttpContext.Session.GetInt32("idUsuarioLogueado");
                Usuario u = s.GetUsuarioLogueado(idLogueado);
                Miembro m = (Miembro)u;
                if (m.Bloqueado)
                {
                    TempData["bloqueado"] = "Bloqueado";
                }
                return View(s.GetPublicacionesFiltradas(buscar, numero, idLogueado));
            }
            return RedirectToAction("Index", "Home");
        }
        else
        {
            TempData["error"] = "El texto no debe ser vacío";
            return View();
        }
    }


    public IActionResult Edit(int idPost) //Mostramos la vista para confirmacion de baneo
de Post
    {
        string? rolLogueado = HttpContext.Session.GetString("tipoUsuarioLogueado");
```

```
        if (rolLogueado == "Administrador")
        {
            return View(s.GetPostPorId(idPost));
        }
        return RedirectToAction("Index", "Home");
    }

    [HttpPost]
    public IActionResult Edit(bool Check, int idPost) //Realizamos el baneo del post
    {
        string? rolLogueado = HttpContext.Session.GetString("tipoUsuarioLogueado");
        if (rolLogueado == "Administrador")
        {
            if (Check)
            {
                int? idUsuarioLogueado =
HttpContext.Session.GetInt32("idUsuarioLogueado");
                Usuario a = s.GetUsuarioLogueado(idUsuarioLogueado);
                Post postBuscado = s.GetPostPorId(idPost);

                Administrador admin = (Administrador)a;
                admin.CensurarPost(postBuscado);
            }
            else
            {
                ViewBag.msg = "Debe seleccionar la opción para banear";
            }
        }
        return RedirectToAction("Index", "Home");
    }

    public IActionResult Details(int idPost) //Mostramos los comentarios habilitados para
el miembro/admin logueado segun el idPost
    {
        string? rolLogueado = HttpContext.Session.GetString("tipoUsuarioLogueado");
        if (rolLogueado != null)
        {
            Post p = s.GetPostPorId(idPost);
            ViewBag.Post = p;
            if (rolLogueado == "Administrador")
            {
                return View(p.GetComentarios());
```

```
                }
                else
                {
                    int? idLogueado = HttpContext.Session.GetInt32("idUsuarioLogueado");
                    Miembro m = (Miembro)s.GetUsuarioLogueado(idLogueado);
                    if (m.Bloqueado)
                    {
                        TempData["bloqueado"] = "Bloqueado";
                    }
                    return View(s.GetComentariosHabilitados(idPost, idLogueado));
                }
            }
            return RedirectToAction("Index", "Home");
        }

        [HttpPost]
public IActionResult Details(int idPost, string titulo, string texto, bool check) // Nuevo
    Comentario a un Post
{
    string? rolLogueado = HttpContext.Session.GetString("tipoUsuarioLogueado");
    if (rolLogueado == "Miembro")
    {
        try
        {
            int? idLogueado = HttpContext.Session.GetInt32("idUsuarioLogueado");
            Usuario u = s.GetUsuarioLogueado(idLogueado);
            Miembro m = s.GetMiembroPorEmail(u.Email);
            Post p = s.GetPostPorId(idPost);
            if (!string.IsNullOrWhiteSpace(titulo) || !string.IsNullOrWhiteSpace(titulo))
            {
                s.NuevoComentario(m, titulo, texto, check, new List<Reaccion>(), p);
                TempData["msgComentario"] = "Comentario publicado!";
            }
            else
            {
                TempData["msgComentarioError"] = "El titulo y el comentario son obligatorios,
    no pueden ser vacíos";
            }
            return RedirectToAction("Details", "Publicacion", new { idPost = p.Id });

        }
        catch (Exception e)
```

```
        {
            TempData["msgComentarioError"] = e.Message;
            return RedirectToAction("Details");
        }
    }
    return RedirectToAction("Index", "Home");
}
    public IActionResult Create(int idPublicacion, string reaccion, int idPosteo) // Nueva reaccion a Post o Comentario
    {
        string? rolLogueado = HttpContext.Session.GetString("tipoUsuarioLogueado");
        if (rolLogueado == "Miembro")
        {
            Publicacion p = s.GetPublicacionPorId(idPublicacion);
            int? idLogueado = HttpContext.Session.GetInt32("idUsuarioLogueado");
            Miembro m = (Miembro)s.GetUsuarioLogueado(idLogueado);
            try
            {
                s.NuevaReaccion(reaccion, m, p);
            }
            catch (Exception e)
            {
                TempData["errorReaccion"] = e.Message;
            }

            //Agregar link de reacciones?
            //if (p.GetTipo() == "Post" || idPosteo == 0) //El idPosteo será 0 cuando el tipo sea Post y Cuando sea Comentario en la vista Buscar, de lo contrario se va al else cuando es Comentario llamado desde la vista Details
            if (p.GetTipo() == "Post")
            {
                //return RedirectToAction("Buscar"); //Agregando Reacciones en el buscador deberíamos evaluar doble retorno para recargar la página dependiendo de la vista en que haya sido llamado el metodo
                return RedirectToAction("Index");
            }
            else
            {
                return RedirectToAction("Details", "Publicacion", new { idPost = idPosteo });
            }
        }
        return RedirectToAction("Index", "Home");
    }
```

```
        public IActionResult CreatePost()
        {
            string? rolLogueado = HttpContext.Session.GetString("tipoUsuarioLogueado");
            if (rolLogueado == "Miembro")
            {
                return View();
            }
            return RedirectToAction("Index", "Home");

        }
        [HttpPost]
        public IActionResult CreatePost(string Titulo, string Texto, string Imagen, bool
    Privado) //Nuevo Post
        {
            string? rolLogueado = HttpContext.Session.GetString("tipoUsuarioLogueado");
            if (rolLogueado == "Miembro")
            {
                try
                {
                    int? idLogueado = HttpContext.Session.GetInt32("idUsuarioLogueado");
                    Usuario u = s.GetUsuarioLogueado(idLogueado);
                    Miembro m = s.GetMiembroPorEmail(u.Email);
                    s.NuevoPost(m, Titulo, Texto, Imagen, Privado, false, new
    List<Reaccion>());
                    TempData["msgPublicacion"] = "Posteo Realizado!";
                    return RedirectToAction("CreatePost", "Publicacion");

                }
                catch (Exception e)
                {
                    TempData["msgPublicacionError"] = e.Message;
                    return RedirectToAction("CreatePost", "Publicacion");
                }
            }
            return RedirectToAction("Index", "Home");
        }

    }
}
```

Vista Index:

```
@using Dominio;
```

```
@model IEnumerable<Dominio.Post>
@{
    ViewData["Title"] = "Publicaciones";
}
@if (Model != null)
{
    <h2 class="text-center">Lista de Posteos</h2>
    @foreach (Post p in Model)
    {
        <section class="container mt-4 col-md-8">
            <hr />
            <article class="card">
                <div class="card-header">
                    <h4 class="card-title">@p.Titulo</h4>
                    <p>@p.TiempoTranscurrido()</p>
                    <div class="d-flex justify-content-between">
                        <p title="Valor de Aceptación">VA: @p.CalcularValorAceptacion()</p>

                        @if (p.Privado)
                        {
                            <p title="Solo los Amigos de @p.Autor.Nombre pueden ver este
    posteo">Privado</p>
                        }
                        else
                        {
                            <p title="Cualquier Usuario que tenga una cuenta puede ver este
    posteo">Publico</p>
                        }

                    </div>
                </div>
                <div class="card-body">
                    <h6 class="card-subtitle mb-2 text-muted">@p.Autor.Nombre
    @p.Autor.Apellido</h6>
                    <p class="card-text">@p.Texto</p>

                    <img src="@p.Imagen" class="card-img-top" alt="@p.Imagen">
                </div>
                <div class="card-footer">
                    <div class="row d-flex align-items-center justify-content-between">

                        @if (Context.Session.GetString("tipoUsuarioLogueado") == "Miembro")
```

```
                {
                    @if (TempData["bloqueado"] == "Bloqueado")
                    {
                        <div class="mt-3">
                            <h6 class="msg alert-danger text-center">Usted no puede
reaccionar a las publicaciones, está bloqueado!</h6>
                        </div>
                    }
                    else
                    {
                        <div class="col-md-2 text-center">
                            @Html.ActionLink("Like", "Create", "Publicacion",
new{idPublicacion=@p.Id, reaccion="like"}, new { @class = "btn btn-outline-success btn-sm
" })
                        </div>

                        <div class="col-md-2 text-center">
                            @Html.ActionLink("Dislike", "Create", "Publicacion",
new{idPublicacion=@p.Id, reaccion="dislike"}, new { @class = "btn btn-outline-danger btn-
sm " })
                        </div>
                    }
                }

                <div class="col-md-4 text-center">
                    @Html.ActionLink("Comentarios", "Details", "Publicacion",
new{idPost=@p.Id}, new { @class = "btn btn-outline-secondary btn-sm" })

                </div>

                <div class="col-md-3 text-center">
                    @if (Context.Session.GetString("tipoUsuarioLogueado") ==
"Administrador")
                    {
                        @if (p.Censurado)
                        {
                            <span>Baneado</span>
                        }
                        else
                        {
                            @Html.ActionLink("Banear", "Edit","Publicacion",
new{idPost = p.Id},new{@class="btn btn-outline-danger btn-sm"})
                        }
                    }
```

```
                    </div>
                    <span class="text-center text-black-50">@p.GetReaccionesTipo("like")
    like | @p.GetReaccionesTipo("dislike") dislike</span>
                </div>
            </div>
        </article>
    </section>
}
}
<hr />
<div class="mt-3">
    @Html.ActionLink("Atrás", "Index", "Home", null, new { @class = "btn btn-outline-secondary
    btn-sm"})
</div>
```

Vista Buscar:

```
@using Dominio;

@model IEnumerable<Dominio.Publicacion>

@{
    ViewData["Title"] = "Buscar";
}
<div class=" row justify-content-center">
    <h2 class="text-center">Buscador de Publicaciones</h2>

    <div class="col-md-6">
        <form method="post">
            <div class="form-group">
                <input type="text" class="form-control mt-3" name="buscar"
    placeholder="Ingrese un texto" required />
            </div>
            <div class="form-group">
                <input type="number" class="form-control mt-3" name="numero"
    placeholder="Valor Aceptación" required />
            </div>
            <input type="submit" value="Buscar" class="btn btn-primary btn-block mt-3" />
        </form>
    </div>

</div>

<hr />
```

```
@if (Model != null)
{
    <section class="container mt-4 col-md-8">
        <h5 class="text-center">Se muestran las Publicaciones con un Valor de Aceptación superior al ingresado y que contengan ese texto</h5>

        @foreach (Publicacion p in Model)
        {
            <hr />
            <article class="card">
                <div class="card-header">
                    <h4 class="card-title">@p.Titulo</h4>
                    <p title="@p.FechaPublicacion">@p.TiempoTranscurrido()</p>
                    <div class="d-flex justify-content-between">
                        <h6 title="Valor de Aceptación">VA: @p.CalcularValorAceptacion()</h6>

                        @if (p.Privado)
                        {
                            <p title="Solo los Amigos de @p.Autor.Nombre pueden ver este posteo">@p.GetTipo() Privado</p>
                        }
                        else
                        {
                            <p title="Cualquier Usuario que tenga una cuenta puede ver este posteo">@p.GetTipo() Público</p>
                        }

                    </div>
                </div>
                <div class="card-body">
                    <h6 class="card-subtitle mb-2 text-muted">@p.Autor.Nombre @p.Autor.Apellido</h6>
                    <p class="card-text">@p.Texto</p>
                    @if (p.GetTipo() == "Post" && p is Post po)
                    {
                        <img src="@po.Imagen" class="card-img-top" alt="@po.Imagen">
                    }
                </div>
                <div class="card-footer">

                    <div class="row d-flex align-items-center justify-content-between">
                        @* MOSTRAMOS REACCIONES Y COMENTARIOS? *@
                        @*
```

```
            @if (Context.Session.GetString("tipoUsuarioLogueado") == "Miembro")
            {
            @if (TempData["bloqueado"] == "Bloqueado")
            {
            <div class="mt-3">
            <h6 class="msg alert-danger text-center">Usted no puede reaccionar a las
publicaciones, está bloqueado!</h6>
            </div>
            }
            else
            {
            @if (p.GetTipo() == "Post" && p is Post post)
            {
            <div class="col-md-2 text-center">
            @Html.ActionLink("Like", "Create", "Publicacion", new{idPublicacion=@post.Id,
reaccion="like"}, new { @class = "btn btn-outline-success btn-sm " })
            </div>

            <div class="col-md-2 text-center">
            @Html.ActionLink("Dislike", "Create", "Publicacion",
new{idPublicacion=@post.Id, reaccion="dislike"}, new { @class = "btn btn-outline-danger
btn-sm " })
            </div>
            }
            else
            {
            <div class="col-md-3 text-center">
            @Html.ActionLink("Like", "Create", "Publicacion", new{idPublicacion=@p.Id,
reaccion="like"}, new { @class = "btn btn-outline-success btn-sm " })
            </div>

            <div class="col-md-3 text-center">
            @Html.ActionLink("Dislike", "Create", "Publicacion", new{idPublicacion=@p.Id,
reaccion="dislike"}, new { @class = "btn btn-outline-danger btn-sm " })
            </div>
            }
            }
            }
            @if (p.GetTipo() == "Post" && p is Post pos)
            {
            <div class="col-md-4 text-center">
            @Html.ActionLink("Comentarios", "Details", "Publicacion", new{idPost=@pos.Id},
new { @class = "btn btn-outline-secondary btn-sm" })
```

```
                </div>
                } *@
                        <span class="text-center text-black-50">@p.GetReaccionesTipo("like")
    like | @p.GetReaccionesTipo("dislike") dislike</span>
                    </div>
                </div>
            </article>
        }
    </section>


}


<div class="mt-3">
    @Html.ActionLink("Atrás", "Index", "Home", null, new { @class = "btn btn-outline-secondary
    btn-sm"})
</div>
```

Vista Edit:

```
@model Dominio.Post


<h1>Banear Posteo "@Model.Titulo"</h1>
@{
    ViewData["Title"] = "Banear Post";
}
<form method="post">
    <span>Está seguro de banear este post <cite><strong>"@Model.Titulo"</strong></cite> del
    miembro <strong>@Model.Autor.Nombre @Model.Autor.Apellido</strong>?</span>
    <input type="checkbox" name="Check" value="true" required />
    <br />
    <input class="btn btn-danger btn-sm mt-3" type="submit" value="Banear" />
</form>


<p class="alert-warning"> @ViewBag.msg </p>


<div class="mt-3">
    @Html.ActionLink("Atrás", "Index", "Publicacion", null, new { @class = "btn btn-outline-
    secondary btn-sm"})
</div>
```

Vista Details:

```
@using Dominio;
@model IEnumerable<Dominio.Comentario>
```

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

@if (ViewBag.Post != null && Model.Count() != 0)
{
    <h4 class="text-center">Comentarios del posteo @ViewBag.Post.Titulo</h4>
    <hr />
    @foreach (Comentario c in Model)
    {
        <section class="container mt-4 col-md-8">
            <article class="card">
                <div class="card-header">
                    <h4 class="card-title">@c.Autor.Nombre @c.Autor.Apellido</h4>
                    <p title="@c.FechaPublicacion">@c.TiempoTranscurrido()</p>
                    <div class="d-flex justify-content-between">
                        <p title="Valor de Aceptación">VA: @c.CalcularValorAceptacion()</p>

                        @if (c.Privado)
                        {
                            <p title="Solo los Amigos de @c.Autor.Nombre pueden ver este
    comentario">Privado</p>
                        }
                        else
                        {
                            <p title="Cualquier Usuario que tenga una cuenta puede ver este
    comentario">Publico</p>
                        }
                    </div>
                </div>
                <div class="card-body">
                    <h6 class="card-subtitle mb-2 text-muted">@c.Titulo</h6>
                    <p class="card-text">@c.Texto</p>
                </div>
                <div class="card-footer">
                    <div class="row d-flex align-items-center justify-content-center">
                        @if (Context.Session.GetString("tipoUsuarioLogueado") == "Miembro")
                        {
                            @if (TempData["bloqueado"] == "Bloqueado")
                            {
                                <div class="mt-3">
                                    <h6 class="msg alert-danger text-center">Usted no puede
    reaccionar a los comentarios, está bloqueado!</h6>
```

```
                            </div>
                        }
                        else
                        {
                            <div class="col-md-3 text-center">
                                @Html.ActionLink("Like", "Create", "Publicacion",
    new{idPublicacion=@c.Id, reaccion="like", idPosteo = ViewBag.Post.Id}, new { @class =
    "btn btn-outline-success btn-sm " })
                            </div>

                            <div class="col-md-3 text-center">
                                @Html.ActionLink("Dislike", "Create", "Publicacion",
    new{idPublicacion=@c.Id, reaccion="dislike", idPosteo = ViewBag.Post.Id}, new { @class =
    "btn btn-outline-danger btn-sm " })
                            </div>
                        }
                    }

                    <span class="text-center text-black-50">@c.GetReaccionesTipo("like")
    like | @c.GetReaccionesTipo("dislike") dislike</span>
                </div>
            </div>
        </article>
    </section>
}
<hr />
@if (TempData["errorReaccion"] != null)
{
    <div class="alert-danger">
        <p>@TempData["errorReaccion"]</p>
    </div>
}
}
else
{
    <span>Aún no se han realizado comentarios</span>
    <hr />
}
@if (Context.Session.GetString("tipoUsuarioLogueado") == "Miembro")
{
    <section class="container mt-4">
        <article>
            <div class="row justify-content-center">
```

```
            <div class="col-md-8">
                @if (TempData["bloqueado"] == "Bloqueado")
                {
                    <div class="mt-3">
                        <h6 class="msg alert-danger text-center">Usted no puede publicar
comentarios, está bloqueado!</h6>
                    </div>
                }
                else
                {
                    <h4 class="text-center">Escribe un comentario</h4>
                    <hr />
                    <form method="post">
                        @* <input name="idPost" value="@ViewBag.Post.Id" hidden> *@
                        <div class="form-group mt-3">
                            <input type="text" name="titulo" class="form-control mt-3"
placeholder="Titulo" required />
                        </div>

                        <div class="form-group mt-3">
                            <textarea name="texto" class="form-control mt-3"
placeholder="Comentario" maxlength="500" required></textarea>
                        </div>

                        <div class="form-group mt-3">
                            <label class="control-label">Seleccione para publicar en
privado</label>
                            <input type="checkbox" name="check" class="form-check-input"
value="true" />
                        </div>

                        <div class="form-group mt-3">
                            <input type="submit" value="Comentar" class="btn btn-primary"
/>
                        </div>

                    </form>
                    <div class="mt-3">
                        <span class="msg alert-success">@TempData["msgComentario"]</span>
                        <span class="msg alert-
danger">@TempData["msgComentarioError"]</span>
                    </div>
                }
            </div>
```

```
                <hr>
            </div>
        </article>
    </section>
}
<div class="mt-3">
    @Html.ActionLink("Atrás", "Index", "Publicacion", null, new { @class = "btn btn-outline-
    secondary btn-sm"})
</div>
```

Vista CreatePost:

```
@using Dominio;

@model Dominio.Post;

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

<div class="container mt-5">
    <h1 class="text-center">Crear Posteo</h1>
    <hr class="my-6">
    <div class="row justify-content-center">

        <div class="col-md-6">

            <form method="post">
                <div class="form-group">
                    <input type="text" class="form-control mt-3" asp-for="Titulo"
    placeholder="Titulo" required />
                </div>
                <div class="form-group">
                    <input type="text" class="form-control mt-3" asp-for="Texto"
    placeholder="Contenido" required />
                </div>
                <div class="form-group">
                    <input type="text" class="form-control mt-3" asp-for="Imagen"
    placeholder="Imagen" required />
                </div>
                <div class="form-group  mt-3">
                    <label class="control-label">Seleccione si desea publicar en
    Privado</label>
```

```
                    <input type="checkbox" asp-for="Privado" value="true" class="form-check-
    input"/>
                </div>
                <input type="submit" value="Postear" class="btn btn-primary btn-block mt-3" />
            </form>
            @if (TempData["msgPublicacionError"] != null)
            {
                <div class="alert alert-danger mt-3">
                    <p>@TempData["msgPublicacionError"]</p>
                </div>
            }
            else
            {
                <div class="alert alert-success mt-3">
                    <p>@TempData["msgPublicacion"]</p>
                </div>
            }
            <hr class="my-4">
        </div>

    </div>
</div>

<div class="mt-3">
    @Html.ActionLink("Atrás", "Index", "Home", null, new { @class = "btn btn-outline-
    secondary"})
</div>
```

Vista Shared

Layout

```
@using Microsoft.AspNetCore.Http

<!DOCTYPE html>
<html lang="es">
<head>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>@ViewData["Title"] - SSN</title>
    <link rel="icon" href="@Url.Content("~/img/soccer.ico")" />
    <link rel="stylesheet" href="~/lib/bootstrap/dist/css/bootstrap.min.css" />
    <link rel="stylesheet" href="~/css/site.css" asp-append-version="true" />
```

```
    <link rel="stylesheet" href="~/IUWebApp.styles.css" asp-append-version="true" />
</head>
<body>
    <header>
        <nav class="navbar navbar-expand-sm navbar-toggleable-sm navbar-light bg-white border-bottom box-shadow mb-3">
            <div class="container-fluid">
                <a class="navbar-brand" asp-area="" asp-controller="Home" asp-action="Index">Soccer Social Network</a>
                <button class="navbar-toggler" type="button" data-bs-toggle="collapse" data-bs-target=".navbar-collapse" aria-controls="navbarSupportedContent"
                        aria-expanded="false" aria-label="Toggle navigation">
                    <span class="navbar-toggler-icon"></span>
                </button>
                <div class="navbar-collapse collapse d-sm-inline-flex justify-content-between">
                    <ul class="navbar-nav flex-grow-1">

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

                        @if (Context.Session.GetInt32("idUsuarioLogueado") != null)
                        {
                            @if (Context.Session.GetString("tipoUsuarioLogueado") == "Miembro")
                            {
                                <li class="nav-item">
                                    <a class="nav-link text-dark" asp-area="" asp-controller="Usuario" asp-action="Solicitudes">Solicitudes</a>
                                </li>
                                <li class="nav-item">
                                    <a class="nav-link text-dark" asp-area="" asp-controller="Publicacion" asp-action="CreatePost">Publicar</a>
                                </li>
                                <li class="nav-item">
                                    <a class="nav-link text-dark" asp-area="" asp-controller="Publicacion" asp-action="Buscar">Buscar</a>
                                </li>
                                <li class="nav-item">
                                    <a class="nav-link text-dark" asp-area="" asp-controller="Usuario" asp-action="Index">Agregar amigos</a>
                                </li>
                            }
```

```
                    else
                    {
                        <li class="nav-item">
                            <a class="nav-link text-dark" asp-area="" asp-
controller="Usuario" asp-action="Index">Miembros</a>
                        </li>
                    }

                    <li class="nav-item">
                        <a class="nav-link text-dark" asp-area="" asp-
controller="Publicacion" asp-action="Index">Publicaciones</a>
                    </li>

                    <li class="nav-item">
                        <a class="nav-link text-dark" asp-area="" asp-
controller="Home" asp-action="Logout">Cerrar Sesión</a>
                    </li>

                }
                else
                {
                    <li class="nav-item">
                        <a class="nav-link text-dark" asp-area="" asp-
controller="Home" asp-action="IniciarSesion">Iniciar Sesión</a>
                    </li>

                    <li class="nav-item">
                        <a class="nav-link text-dark" asp-area="" asp-
controller="Home" asp-action="Registrarse">Registrarse</a>
                    </li>
                }




            </ul>
        </div>
    </div>
</nav>
</header>
<div class="container">
    <main role="main" class="pb-3">
```

```
            @RenderBody()
        </main>
    </div>

    <footer class="border-top footer text-muted">
        <div class="container">
            &copy; 2023 - SSN - Soccer Social Network - <a asp-area="" asp-controller="Home"
    asp-action="Privacy">Autores</a>
        </div>
    </footer>
    <script src="~/lib/jquery/dist/jquery.min.js"></script>
    <script src="~/lib/bootstrap/dist/js/bootstrap.bundle.min.js"></script>
    <script src="~/js/site.js" asp-append-version="true"></script>
    @await RenderSectionAsync("Scripts", required: false)
</body>
</html>
```

Program

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

            builder.Services.AddSession(); // Preparamos nuestro proyecto para trabajar con
    variables de session

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

            var app = builder.Build();

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

```
            // The default HSTS value is 30 days. You may want to change this for
    production scenarios, see https://aka.ms/aspnetcore-hsts.
            app.UseHsts();
        }

        app.UseHttpsRedirection();
        app.UseStaticFiles();

        app.UseRouting();

        app.UseAuthorization();

        app.UseSession(); // Antes de ejecutar la aplicación le decimos que use la session

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

        app.Run();
    }
}
}
```