Universidad ORT Uruguay

Facultad de Ingeniería

Lucas Giusiano – 305227

V2A

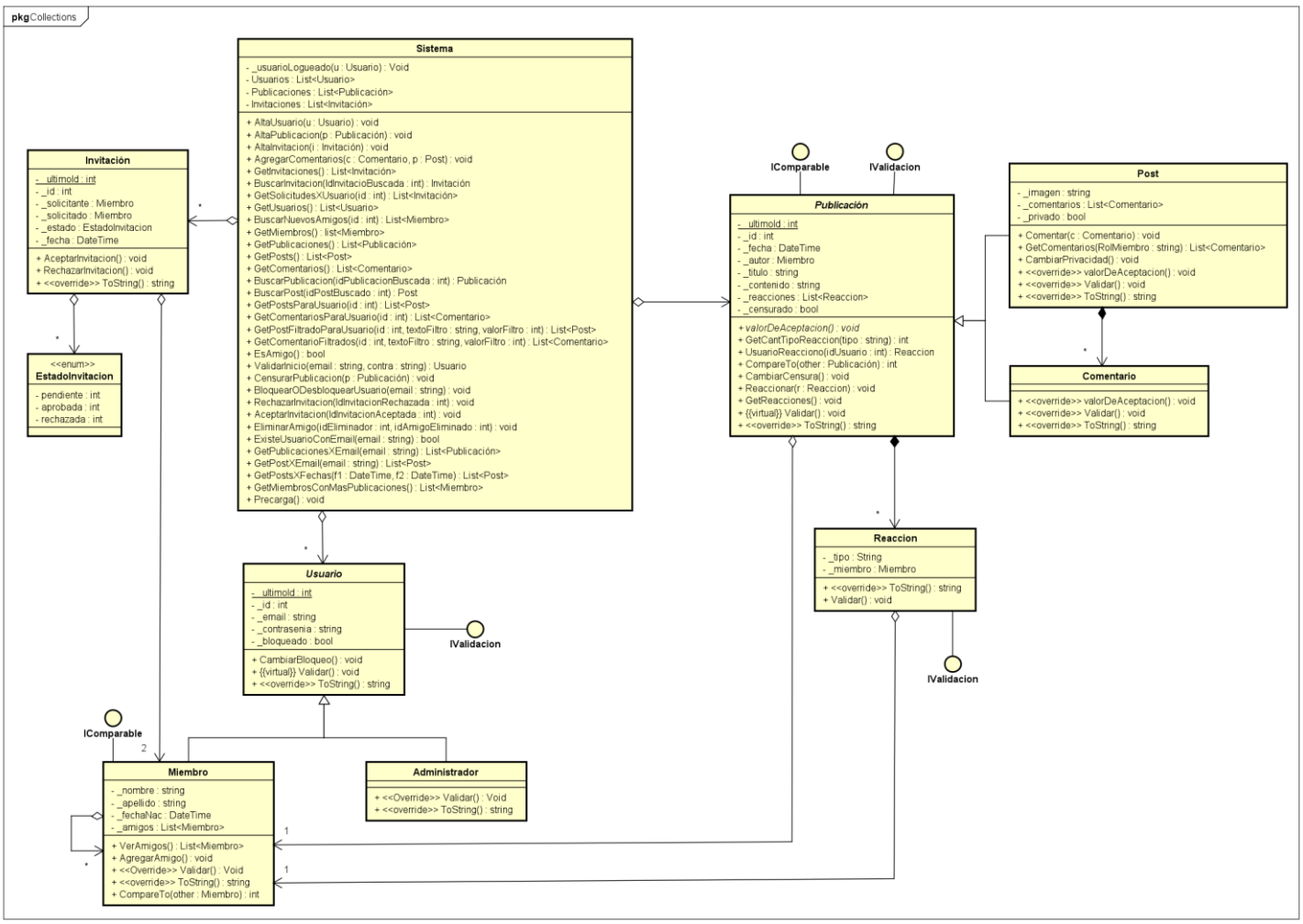
Joaquín Rodríguez

02/10/2023

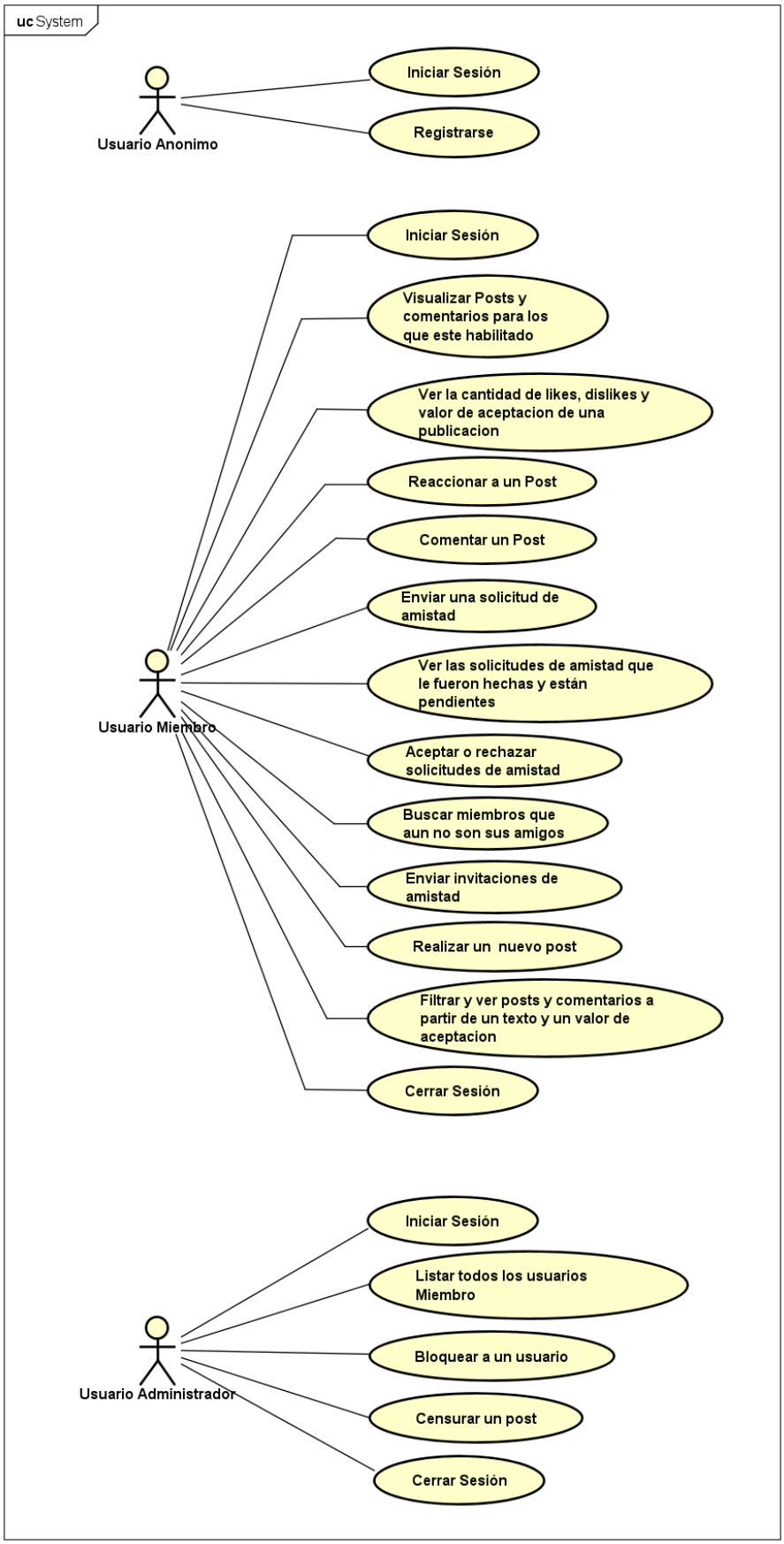
Índice

1. Diagramas

1.2 Diagrama de clases



1.2 Diagrama de casos de uso



2. Tabla de información

2.1 Miembros

ID	Correo	Contraseña	Nombre	Apellido	Fecha de Nacimiento
0	luucass26@gmail.com	contrasenia123	Lucas	Giusiano	2000/09/26
1	usuario2@hotmail.com	qwerty1	Alice	Smith	1988/10/20
2	ejemplo3@yahoo.com	password	Michael	Johnson	1993/08/07
3	nuevocorreo4@gmail.com	myp@ss1	Sophia	Brown	1985/03/25
4	usuario5@hotmail.com	letmein	Olivia	Taylor	1991/12/10
5	correo6@gmail.com	securepass	Matthew	Robinson	1987/07/14
6	otrocorreo7@yahoo.com	p@ssw0rd	Emma	Hernandez	1989/06/18
7	micorreo8@hotmail.com	ilovecoding	Daniel	Garcia	1994/04/05
8	nuevo_miembro9@gmail.com	mypassword	Sophie	Anderson	1996/09/22
9	correo10@yahoo.com	password123	Ryan	Baker	1986/02/12

2.2 Administradores

ID	Correo	Contraseña
10	admin@gmail.com	admin123

2.3 Invitaciones

ID	ID de Solicitante	ID de Solicitado	Estado
0	0	1	Aprobada
1	0	2	Aprobada
2	0	3	Aprobada
3	0	4	Aprobada
4	0	5	Aprobada
5	0	6	Aprobada
6	0	7	Aprobada
7	0	8	Aprobada
8	0	9	Aprobada
9	1	2	Aprobada
10	1	3	Aprobada
11	1	4	Aprobada
12	1	5	Aprobada
13	1	6	Aprobada
14	1	7	Aprobada
15	1	8	Aprobada
16	1	9	Aprobada
17	2	4	Aprobada
18	2	6	Rechazada
19	2	5	Pendiente
20	3	7	Aprobada
21	3	4	Aprobada

22	3	8	Aprobada
23	4	9	Rechazada
24	4	6	Rechazada
25	4	7	Aprobada
26	5	6	Aprobada
27	5	9	Rechazada
28	5	3	Aprobada
29	6	7	Pendiente
30	6	3	Pendiente
31	6	4	Pendiente
32	7	2	Rechazada
33	7	8	Rechazada
34	7	5	Rechazada
35	8	5	Aprobada
36	8	9	Aprobada
37	8	4	Pendiente
38	9	2	Rechazada
39	9	3	Aprobada
40	9	6	Pendiente

2.4 Posts

ID	Titulo	Contenido	Id Del Autor	Imagen	Fecha (aaaa/MM/dd)	Privacidad
0	Nuevo Zelda	'The Legend of Zelda: Breath of the Wild' es una obra maestra de la libertad y la aventura. Desde explorar el vasto reino de Hyrule hasta los emocionantes combates y desafiantes rompecabezas en santuarios, el juego cautiva. Con gráficos impresionantes y una banda sonora envolvente, ofrece una experiencia inolvidable. Es un hito en la historia de los videojuegos y esencial para los amantes de la fantasía y la acción.	0	zelda.png	2010/07/05	Publico
1	Épica batalla en Fortnite	Anoche tuve la batalla más intensa en Fortnite. ¡Conseguí esa tan ansiada victoria magistral! #Fortnite #VictoriaRoyale	1	fortnite_batalla.png	2012/04/15	Privado
2	Descubriendo secretos en Skyrim	Explorando Skyrim y encontré un lugar oculto lleno de tesoros. ¡La emoción de descubrir secretos en este juego es increíble! #Skyrim #Exploración	2	skyrim_secreto.png	2013/09/20	Publico
3	Speedrun récord en Super Mario	Hoy logré mi mejor tiempo en un speedrun de Super Mario. ¡Rompiendo récords y mejorando cada día! #SuperMario #Speedrun	3	mario_speedrun.png	2015/02/10	Publico
4	Logros desbloqueados en Dark Souls	Conseguí vencer a todos los jefes en Dark Souls. ¡Un desafío épico superado! #DarkSouls #GitGud	4	dark_souls_logros.png	2015/11/08	Publico
5	Emocionante partida en League of Legends	¡Increíble partida en League of Legends! Logramos una remontada épica y ganamos la partida en el último segundo. #LeagueOfLegends #GamerLife	5	lol_partida.png	2018/06/25	Privado

2.4 Comentarios

ID	Titulo	Contenido	ID del Autor	Id de Post Comentado
0	Amo la saga Zelda	The Legend of Zelda siempre ha sido mi saga de videojuegos favorita. Cada entrega es única y especial.	3	0
1	¿Alguien más emocionado por esto?	Estoy contando los días para el lanzamiento. ¡No puedo esperar para explorar Hyrule en Breath of the Wild!	6	0
2	Los gráficos se ven alucinantes	Vi algunos avances y los gráficos son impresionantes. La atención al detalle es fenomenal.	1	0
3	¡Quiero jugarlo ya!	Soy un gran fan de Zelda y esta nueva entrega se ve prometedora. ¡No puedo esperar a sumergirme en esta aventura!	5	0
4	¡Me encanta Fortnite!	Es un juego tan divertido, especialmente cuando llegas a la última fase. ¡Buena jugada!	6	1
5	¿Qué estrategia usaste?	Estoy tratando de mejorar en Fortnite, ¿tienes algún consejo sobre las mejores estrategias?	7	1
6	Genial, felicidades por la victoria	Sé lo difícil que es lograr una Victoria Royale en Fortnite, ¡buen trabajo!	8	1
7	¿Alguien quiere unirse para jugar?	Estoy buscando compañeros de escuadrón en Fortnite. ¡Si estás interesado, házmelo saber!	9	1
8	Skyrim es mi favorito	He pasado horas explorando Skyrim. Siempre hay algo nuevo por descubrir.	4	2
9	¡Qué emocionante!	Skyrim tiene algunos de los paisajes más impresionantes. Me encanta perderme en este mundo.	5	2
10	Esa sensación de encontrar secretos...	En Skyrim, descubrir esos lugares ocultos es tan satisfactorio. ¡Bien hecho!	6	2
11	¿Cuál es tu misión favorita?	Estoy interesado en saber cuál es tu misión favorita en Skyrim. Para mí, es 'La caída de Alduin'.	7	2
12	Adictivo juego	Super Mario es un clásico atemporal. ¡Nunca me canso de jugarlo!	2	3
13	Recuerdos de la infancia	Jugaba a Super Mario cuando era niño, ¡y aún lo juego ahora! La nostalgia es increíble.	3	3

14	¿En qué plataforma juegas?	Estoy pensando en empezar a jugar Super Mario. ¿En qué plataforma juegas? ¿Switch o emulador?	4	3
15	¿Tienes alguna estrategia?	¿Tienes alguna estrategia o consejo para avanzar rápidamente en Super Mario? ¡Estoy atascado en una fase difícil!	5	3
16	Dark Souls es un desafío	Este juego es todo un desafío, pero la satisfacción de vencer a un jefe es inigualable. ¡Buena suerte en tu travesía!	1	4
17	Combates intensos	Los combates en Dark Souls son tan intensos y estratégicos. Cada encuentro es una prueba de habilidad.	3	4
18	Mis momentos favoritos	Mis momentos favoritos en Dark Souls son cuando derroto a un jefe en el primer intento. Es una sensación increíble.	6	4
19	Dark Souls: un clásico moderno	Dark Souls redefinió los juegos de acción y RPG. Una obra maestra que todo jugador debería experimentar.	8	4
20	¿Cuál es tu campeón favorito?	Estoy interesado en saber cuál es tu campeón favorito en League of Legends y por qué. ¡Comparte tu opinión!	0	5
21	Amo jugar LoL	League of Legends es mi juego favorito. ¡Las estrategias y el trabajo en equipo hacen que cada partida sea única!	2	5
22	¡Esa remontada fue épica!	Me encanta cuando las partidas de League of Legends se vuelven intensas y logras una remontada asombrosa. ¡Muy bien hecho!	4	5

2.5 Reacciones a Posts

Id Post Reaccionado	Tipo	Id de Autor
0	Like	0
0	Dislike	1
0	Like	6
0	Dislike	7
1	Like	2
1	Dislike	3
1	Like	1
1	Dislike	2
2	Like	4
2	Dislike	5
3	Like	6
3	Dislike	7
4	Like	8
4	Dislike	0
5	Like	1
5	Dislike	2

2.6 Reacciones a comentarios

Id Comentario Reaccionado	Tipo	Id Autor
0	Like	3
0	Dislike	4
0	Like	5
1	Like	6
1	Dislike	7
1	Like	8
2	Like	0
2	Dislike	1
2	Dislike	2
3	Like	2
3	Dislike	3
3	Like	4
4	Like	5
4	Dislike	6
4	Like	7
5	Like	8
5	Dislike	0
5	Like	1
6	Like	3
6	Dislike	4
6	Like	5
7	Like	6
7	Dislike	7

7	Like	8
8	Like	0
8	Dislike	1
8	Like	2
9	Like	3
9	Dislike	4
9	Like	5
10	Like	6
10	Dislike	7
10	Like	8
11	Like	0
11	Dislike	1
11	Like	2
12	Like	3
12	Dislike	4
12	Like	5
13	Like	6
13	Dislike	7
13	Like	8
14	Like	0
14	Dislike	1
14	Like	2
15	Like	3
15	Dislike	4
15	Like	5
16	Like	6
16	Dislike	7
16	Like	8
17	Like	0
17	Dislike	1
17	Like	2

3. Evidencia de Testing

Funcionalidad	Datos Usados	Resultado esperado	Estado
Registro	Nombre: Mario Apellido: Giusiano Fecha nacimiento: 10/12/1971 Correo: correoMario@gmail.com Contraseña: contrasenia123	El usuario es registrado en el sistema y permite el posterior inicio de sesión en el mismo	Pasado
Registro	Nombre: m Apellido: g Fecha nacimiento: --/--/---- Correo: correogemail Contraseña: contra	El mail y la contraseña son reportados como inválidos y no se realiza el registro del nuevo usuario en el sistema (Se realiza solo el reporte de estos dos ya que son los primeros en comprobarse por el validador de la clase Usuario), además el validador HTML solicita completar el campo de la fecha	Pasado
Registro	Nombre: m Apellido: g Fecha nacimiento: 04/07/2023 Correo: uncorreo@gmail.com Contraseña: contra123	Al ser validos los campos de correo y contraseña se pasa a validar los atributos propios del miembro por tanto se notifica al usuario que el nombre y el apellido deben cumplir con un mínimo de caracteres	Pasado
Inicio de Sesión - Miembro	Correo: luucass26@gmail.com Contraseña: contrasenia123	Al ser un usuario que ya esta en el sistema se le permite ingresar y se le muestra la pagina principal de la red	Pasado
Inicio de Sesión - Miembro	Correo: correoRandom@gmail.com Contraseña: unacontrasenia123	Se le informa al usuario que los datos ingresados son incorrectos puesto que no es un usuario que exista en el sistema, sucede lo mismo en caso de poner un correo correcto pero una contraseña mal o viceversa	Pasado

Inicio de Sesión – Administrador – con credenciales válidas	Correo: admin@gmail.com Contraseña: admin123	El usuario es validado correctamente y se le permite entrar en la pagina principal y ver las opciones correspondientes a su tipo	Pasado
Inicio de Sesión – Administrador – con credenciales inválidas	Correo: unadminrandom@gmail.com Contraseña: admin123	Se le informa al usuario anónimos que los datos que ingresó no son correctos	Pasado
Realizar un post nuevo - Sin completar los campos	Titulo: --------- Contenido: ---------	El usuario es informado sobre los requisitos necesarios para poder realizar un post, que son al menos completar el titulo y el contenido que desea tenga su post	Pasado
Realizar un post nuevo – Con datos	Título: Red Dead Redeption 2 Contenido: Increíble historia y gráficos. Imprescindible para los amantes de los videojuegos. Foto: Imagen png seleccionada desde el pc (Asume el nombre del título de la publicación y el id del usuario) Privado: Si	El post es realizado con éxito y en caso de ser privado solo es visible por el usuario que lo creó, de lo contrario, todos pueden verlo	Pasado
Reaccionar a un post o comentario		Se suma la reacción al tipo que corresponde y se modifica el valor de aceptación en respuesta a ello, en caso de querer cambiar la reacción, se elimina la anterior y se reemplaza por la que haya sido seleccionada por ultima vez y es posible quitar la reacción pulsando nuevamente en la reacción ya hecha	Pasado
Realizar comentario a un post	Titulo: --------- Contenido: ---------	El usuario es informado sobre los requisitos necesarios para poder realizar un comentario, que son al menos completar	Pasado

		el titulo y el contenido que desea tenga su comentario	
Realizar comentario a un post – Con datos	Título: Un gran juego Contenido: Espero poder jugarlo pronto	El comentario es ingresado correctamente y se puede visualizar en el post correspondiente al que fue hecho	Pasado
Buscar y filtrar un post por valor de aceptación y un texto dado – Sin datos	Valor de aceptación: ----- Texto a buscar: --------	Se muestran todos los posts que estén a disposición del usuario logueado en ese momento	Pasado
Buscar y filtrar un post por valor de aceptación y un texto dado – Sin datos	Valor de aceptación: 9 Texto a buscar: Zelda	Se muestran todos los post y comentarios que contengan la frase a filtrar y estén por arriba del valor de aceptación dado. En caso de proporcionar solo uno de estos solo se aplica el filtro con el dato proporcionado	Pasado
Visualizar amigos		Se muestra una tabla con los amigos del usuario que este logueado en ese momento, dando opción a eliminar a alguno de ellos	Pasado
Eliminar un amigo		Se elimina un al amigo de la lista y aparecen en la lista de personas para enviar solicitud de amistad	Pasado
Visualizar solicitudes		Se muestran los usuarios que han enviado una solicitud al usuario logueado y se le permite a este ultimo las opciones de aceptar o rechazar la solicitud (Los usuarios que hayan enviado una solicitud de amistad al usuario logueado, no aparecerán en la búsqueda de nuevos amigos a no ser que se rechace la solicitud)	Pasado
Rechazar una solicitud		Se cambia el estado de la invitación a “Rechazada” y el usuario rechazado es	Pasado

		eliminado de la lista y ahora pasa a estar disponible en la lista de usuarios a los cuales se le puede enviar una solicitud de amistad	
Aceptar una solicitud		Se cambia el estado de la invitación a “Aprobada” y los usuarios son agregados a sus respectivas listas de amigos y ahora pueden visualizarse en la sección de la pagina provista para ello	Pasado
Visualizar personas que aún no son amigos del usuario (Buscar amigos nuevos)		Se muestran las personas que no están en la lista de amigos del usuario logueado ni tiene una solicitud pendiente con él	Pasado
Enviar una solicitud de amistad		Se crea una invitación en estado “Pendiente” y se elimina de la lista de personas que aún no son amigos del usuario logueado	Pasado
Censurar un post o un comentario		Al presionar el botón de censurar en una respectiva publicación, esta cambia si estado de censura y no es visible a los usuarios de tipo miembro. A su vez la misma puede ser des censurada.	Pasado
Listar todos los miembros ordenados de forma ascendente por nombre y apellido		Se muestra una lista con todos los usuarios de tipo miembro en el sistema y se habilita un botón “Bloquear” para cada uno de ellos	Pasado
Bloquear un usuario miembro		Al presionar el botón de bloqueo el usuario que fue bloqueado podrá acceder al sistema y ver los posts y comentarios pero no podrá interactuar con ellos, lo mismo sucederá con sus amigos y los miembros del sistema, además de no poder	

		ver, aceptar o rechazar solicitudes	
Cerrar Sesión		Al dar clic sobre el botón de “Cerrar sesión” se borran todos los datos de sesión del usuario y se le envía a la pestaña de inicio de sesión	

4. Código Fuente

4.1 Clase Sistema

```
    public class Sistema
    {
        #region Singleton
        private Sistema()
        {
            Precarga();
        }

        private static Sistema instancia = null;

        public static Sistema GetInstancia()
        {
            if (instancia == null)
            {
                instancia = new Sistema();
            }
            return instancia;
        }
        #endregion

        private List<Usuario> Usuarios = new List<Usuario>();
        private List<Publicacion> Publicaciones = new List<Publicacion>();
        private List<Invitacion> Invitaciones = new List<Invitacion>();

        #region Metodos de Alta

        public void AltaUsuario(Usuario u) //Comprueba que no haya un usuario con el mismo email
que el que esta intentando registrar y de ser asi hace el alta, de lo contrario arroja una
excepción
        {
            if (!ExisteUsuarioConEmail(u.Email))
            {
                u.Validar();
                Usuarios.Add(u);
            }
            else
            {
                throw new Exception("Ya existe un usuario el email " + u.Email);
            }
        }

        public void AltaPublicacion(Publicacion p)//Hace el alta de una nueva publicación al
sistema
        {
            p.Validar();
            Publicaciones.Add(p);
```

```
        }

        public void AltaInvitacion(Invitacion i)//Comprueba si el miembro solicitante ya se encuentra en la lista de amigos del miembro solicitado y de ser asi arroja una excepción, sino hace el alta de una nueva invitación
        {
            foreach (Miembro m in i.Solicitado.VerAmigos())
            {
                if (i.Solicitante.Id == m.Id)
                {
                    throw new Exception("Error: Ya es amigo de esa persona");
                }
            }
            Invitaciones.Add(i);
        }

        public void AgregarComentarios(Post p, Comentario c) //Agrega el comentario la lista del post correspondiente y luego a la lista de publicaciones del sistema
        {
            c.Validar();
            AltaPublicacion(c);
            p.Comentar(c);
        }

        #endregion

        #region Metodos Get
        public List<Invitacion> GetInvitaciones()
        {
            return Invitaciones;
        }

        public Invitacion BuscarInvitacion(int idInvitacionBuscada) //Busca una invitacion
        {
            foreach (var invitacion in Invitaciones)
            {
                if (invitacion.Id == idInvitacionBuscada)
                {
                    return invitacion;
                }
            }
            return null;
        }

        public List<Invitacion> GetSolicitudesXUsuario(int id) //Devuelve las invitaciones que han sido hechas al usuario logueado
        {
            List<Invitacion> ret = new List<Invitacion>();
```

```
            foreach (var solicitud in Invitaciones)
            {
                if (solicitud.Solicitado.Id == id && solicitud.Estado ==
EstadoInvitacion.Pendiente)
                {
                    ret.Add(solicitud);
                }
            }
            return ret;
        }

        public List<Usuario> GetUsuarios() //Devuelve la lista de todos los usuarios ingresados
en el sistema
        {
            return Usuarios;
        }

        public Miembro BuscarUsuario(int? IdUsuarioBuscado) //Busca un usuario por su id
        {
            foreach (var usuario in Usuarios)
            {
                if (usuario.Id == IdUsuarioBuscado && usuario is Miembro)
                {
                    return (Miembro)usuario;
                }
            }
            return null;
        }

        public List<Miembro> BuscarNuevosAmigos(int id) //Devuelve una lista de miembros con los
miembros que aun no sean amigos o tengan una invitación pendiente con el usuario logueado
        {
            List<Miembro> ret = new List<Miembro>();
            Miembro usuarioLog = BuscarUsuario(id);
            foreach (var miembro in Usuarios)
            {
                if (miembro is Miembro)
                {
                    if (!EsAmigo(usuarioLog, (Miembro)miembro) && miembro.Id != usuarioLog.Id &&
!Invitaciones.Any(i => ((i.Solicitante.Id == usuarioLog.Id && i.Solicitado.Id == miembro.Id) ||
(i.Solicitante.Id == miembro.Id && i.Solicitado.Id == usuarioLog.Id)) && i.Estado ==
EstadoInvitacion.Pendiente))
                    {
                        ret.Add((Miembro)miembro);
                    }
                }
            }
            return ret;
        }
```

```
        //!Invitaciones.Any(i => (i.Solicitante.Id == usuarioLog.Id || i.Solicitado.Id ==
miembro.Id) && i.Estado != EstadoInvitacion.Aprobada)

        public List<Miembro> GetMiembros() //Devuelve la lista de todos los usuarios miembros
ingresados en el sistema
        {
            List<Miembro> ret = new List<Miembro>();
            foreach (var m in Usuarios)
            {
                if (m is Miembro)
                {
                    ret.Add((Miembro)m);
                }
            }

            ret.Sort();
            return ret;
        }


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

        public List<Post> GetPosts() //Devuelve la lista de todas las publicaciones ingresadas
en el sistema
        {
            List<Post> ret = new List<Post>();

            foreach (var p in Publicaciones)
            {
                if (p is Post)
                {
                    ret.Add((Post)p);
                }
            }

            return ret;
        }

        public List<Comentario> GetComentarios() //Devuelve la lista de todas las publicaciones
ingresadas en el sistema
        {
            List<Comentario> ret = new List<Comentario>();

            foreach (var c in Publicaciones)
            {
```

```
                if (c is Comentario)
                {
                    ret.Add((Comentario)c);
                }
            }

            return ret;
        }

        public Publicacion BuscarPublicacion(int idPublicacionBuscada) //Busca una publicacion por su id y la devuelve, de no encontrarla, devuelve null
        {
            foreach (var publicacion in Publicaciones)
            {
                if (publicacion.Id == idPublicacionBuscada)
                {
                    return publicacion;
                }
            }
            return null;
        }

        public Post BuscarPost(int idPostBuscado) //Devuelve el post buscado a partir del id proporcionado
        {
            foreach (var post in Publicaciones)
            {
                if ( post.Id == idPostBuscado)
                {
                    return (Post)post;
                }
            }
            return null;
        }

        public List<Post> GetPostParaUsuario(int id) //Devuelve solo los post que hayan sido hechas por los amigos del usuario logeado y que no esten censuradas por un administrador
        {
            Miembro usuarioLogueado = BuscarUsuario(id);
            List<Post> ret = new List<Post>();

            foreach (Publicacion p in Publicaciones)
            {
                if (p is Post)
                {
                    Post post = (Post)p;
                    if (!post.Censurado && EsAmigo(usuarioLogueado, post.Autor) && !post.Privado || post.Autor.Id == usuarioLogueado.Id)
                    {
```

```
                        ret.Add(post);
                    }
                }
            }

            return ret;
        }

        public List<Comentario> GetComentariosParaUsuario(int id) //Devuelve solo los post que
hayan sido hechas por los amigos del usuario logeado y que no esten censuradas por un
administrador
        {
            Miembro usuarioLogueado = BuscarUsuario(id);
            List<Comentario> ret = new List<Comentario>();

            foreach (Publicacion c in Publicaciones)
            {
                if (c is Comentario)
                {
                    if (EsAmigo(usuarioLogueado, c.Autor) || c.Autor.Id == usuarioLogueado.Id &&
!c.Censurado)
                    {
                        ret.Add((Comentario)c);
                    }
                }
            }

            return ret;
        }

        //Devuelve solo los posts que hayan sido hechas por los amigos del usuario logeado y que
no esten censuradas por un administrador, que ademas contengan el texto por el cual estan siendo
filtrados
        //y cumplan con tener un valor de aceptacion mayor al recibido por parámetro
        public List<Post> GetPostFiltradoParaUsuario(int id,string textoFiltro, int valorFiltro)
        {
            List<Post> ret = new List<Post>();

            List<Post> postsAFiltrar = new List<Post>();

            if (BuscarUsuario(id) is Miembro)
            {
                postsAFiltrar = GetPostParaUsuario(id);
            }
            else
            {
                postsAFiltrar = GetPosts();
            }
```

```
            if (String.IsNullOrEmpty(textoFiltro) && valorFiltro != 0)
            {

                foreach (Post p in postsAFiltrar)
                {
                    if (p.ValorDeAceptacion() >= valorFiltro)
                    {
                        ret.Add(p);
                    }
                }
            }
            else if (!String.IsNullOrEmpty(textoFiltro) && valorFiltro == 0)
            {
                foreach (Post p in postsAFiltrar)
                {
                    if ((p.Contenido.ToLower().Contains(textoFiltro.ToLower()) ||
p.Titulo.ToLower().Contains(textoFiltro.ToLower())) && p.ValorDeAceptacion() >= valorFiltro)
                    {
                        ret.Add(p);
                    }
                }
            }
            else
            {
                foreach (Post p in postsAFiltrar)
                {
                    if ((p.Contenido.ToLower().Contains(textoFiltro.ToLower()) ||
p.Titulo.ToLower().Contains(textoFiltro.ToLower())) && p.ValorDeAceptacion() >= valorFiltro)
                    {
                        ret.Add(p);
                    }
                }
            }
            return ret;
        }


        //Devuelve solo los comentarios que hayan sido hechas por los amigos del usuario logeado
y que no esten censuradas por un administrador, que ademas contengan el texto por el cual estan
siendo filtrados
        //y cumplan con tener un valor de aceptacion mayor al recibido por parámetro
        public List<Comentario> GetComentarioFiltrados(int id, string textoFiltro, int
valorFiltro)
        {
            Miembro usuarioLogueado = BuscarUsuario(id);

            List<Comentario> ret = new List<Comentario>();

            List<Comentario> comentariosAFiltrar = new List<Comentario>();
```

```
            if (BuscarUsuario(id) is Miembro)
            {
                comentariosAFiltrar = GetComentariosParaUsuario(id);
            }
            else
            {
                comentariosAFiltrar = GetComentarios();
            }

            if (String.IsNullOrEmpty(textoFiltro) && valorFiltro != 0)
            {
                foreach (var c in comentariosAFiltrar)
                {
                    double valor1 = c.ValorDeAceptacion();
                    bool nose = valor1 > valorFiltro;
                    if (c.ValorDeAceptacion() >= valorFiltro)
                    {
                        ret.Add(c);
                    }
                }
            }
            else if (!String.IsNullOrEmpty(textoFiltro) && valorFiltro == 0)
            {
                foreach (var c in comentariosAFiltrar)
                {
                    double valor1 = c.ValorDeAceptacion();
                    bool nose = valor1 > valorFiltro;
                    if (c.Contenido.ToLower().Contains(textoFiltro.ToLower()) ||
c.Titulo.ToLower().Contains(textoFiltro.ToLower()))
                    {
                        ret.Add(c);
                    }
                }
            }
            else
            {
                foreach (var c in comentariosAFiltrar)
                {
                    double valor1 = c.ValorDeAceptacion();
                    bool nose = valor1 > valorFiltro;
                    if ((c.Contenido.ToLower().Contains(textoFiltro.ToLower()) ||
c.Titulo.ToLower().Contains(textoFiltro.ToLower())) && c.ValorDeAceptacion() >= valorFiltro)
                    {
                        ret.Add(c);
                    }
                }
            }
```

```
            return ret;
        }

        public bool EsAmigo(Miembro m,Miembro miembroAmigo) //Busca si un usuario es amigo del
usuario logueado actualmente en caso de que este sea un miembro
        {
            if (m is Miembro)
            {
                foreach (Miembro amigo in m.VerAmigos())
                {
                    if (amigo.Id == miembroAmigo.Id)
                    {
                        return true;
                    }
                }
            }
            else
            {
                throw new Exception("Error: El usuario actualmente logueado no tiene una lista
de amigos porque es una administrador");
            }

            return false;
        }

        #endregion

        #region Metodos de acción

        public Usuario ValidarInicio(string email, string contra) // Valida el usuario que esta
intentando iniciar sesión contra los datos del sistema
        {
            foreach (var usuario in Usuarios)
            {
                if (usuario.Email == email && usuario.Contrasenia == contra)
                {
                    return usuario;
                }
            }
            return null;
        }

        public void CensurarPublicacion(int IdPublicacionCensurada) //Cambia el estado de
censura de una publicacion
        {
            Publicacion p = BuscarPublicacion(IdPublicacionCensurada);
            p.CambiarCensura();
        }
```

```
        public void BloquearODesbloquearUsuario(string email) //Cambia el estado de bloqueo de
un usuario
        {
            if (ExisteUsuarioConEmail(email))
            {
                foreach (Usuario u in Usuarios)
                {
                    if (u.Email == email)
                    {
                        u.CambiarBloqueo();
                    }
                }
            }
            else
            {
                throw new Exception("Error: No existe ningun usuario con el email que desea
bloquear");
            }
        }

        public void RechazarInvitacion(int IdInvitacionRechazada) //Cambia el estado de una
invitacion a "rechazada"
        {
            Invitacion InvitacionARechazar = BuscarInvitacion(IdInvitacionRechazada);
            if (InvitacionARechazar.Id == IdInvitacionRechazada)
            {
                InvitacionARechazar.RechazarInvitacion();
            }
        }

        public void AceptarInvitacion(int IdInvitacionAceptada) //Cambia el estado de una
invitacion a "aprobada"
        {
            Invitacion InvitacionAAceptar = BuscarInvitacion(IdInvitacionAceptada);
            if (InvitacionAAceptar.Id == IdInvitacionAceptada)
            {
                InvitacionAAceptar.AceptarInvitacion();

            }
        }

        public void EliminarAmigo(int idEliminador, int idAmigoEliminado) //Elimina la relación
de amistad entre dos usuarios.
        {
            Miembro eliminador = BuscarUsuario(idEliminador);
            Miembro eliminado = BuscarUsuario(idAmigoEliminado);

            eliminador.VerAmigos().Remove(eliminado);
            eliminado.VerAmigos().Remove(eliminador);
```

```
        }


        #endregion

        #region Metodos para Requerimientos de Consola

        public bool ExisteUsuarioConEmail(string email)//Recibe un email por parametros y
verifica si existe un usuario en el sistema que ya tenga ese email devolviendo un bool en true
en caso de ser asi
        {
            bool ret = false;

            foreach (Usuario u in Usuarios)
            {
                if (u.Email == email)
                {
                    ret = true;
                    break;
                }
            }

            return ret;
        }

        //En caso de existir un usuario con el email del cual se desea filtrar devuelve la lista
de publicaciones hechas por ese usuario
        //El tipo de publicacion es mostrado mediante polimorfismo con el ToString de cada tipo
de publicación
        public List<Publicacion> GetPublicacionesXEmail(string email)
        {
            List<Publicacion> ret = new List<Publicacion>();

            if (ExisteUsuarioConEmail(email))
            {
                foreach (Publicacion p in Publicaciones)
                {
                    if (p.Autor.Email == email)
                    {
                        ret.Add(p);
                    }
                }
            }
            else
            {
                throw new Exception($"Error: No existe ningun usuario con el email '{email}'");
            }

            return ret;
```

```
        }

        public List<Post> GetPostXMail(string email)//En caso de existir un usuario con el email del cual se desea filtrar devuelve una lista con los post hechos por ese usuario
        {
            List<Post> ret = new List<Post>();

            if (ExisteUsuarioConEmail(email))
            {
                foreach (Publicacion p in Publicaciones)
                {
                    if (p is Post)
                    {
                        Post post = (Post)p;
                        foreach (Comentario c in post.GetComentarios("Administrador"))
                        {
                            if (c.Autor.Email == email)
                            {
                                ret.Add(post);
                                break;
                            }
                        }
                    }

                }
            }
            else
            {
                throw new Exception($"Error: No existe ningun usuario con el email '{email}'");
            }

            return ret;
        }


        //Recorre la lista de publicaciones y en caso de que sean un Post y esten en el rango de fechas especificados, las agrega a una lista para retornar
        //Luego ordena la lista por el titulo de cada post en orden alfabetico y la retorna
        public List<Post> GetPostsXFechas(DateTime f1, DateTime f2)
        {
            List<Post> ret = new List<Post>();

            if (f1 > f2)
            {
                DateTime aux = f1;
                f1 = f2;
                f2 = aux;
            }
```

```
            foreach (Publicacion p in Publicaciones)
            {
                if (p is Post && p.Fecha >= f1 && p.Fecha <= f2)
                {
                    ret.Add((Post)p);
                }
            }

            if (ret.Count == 0)
            {
                throw new Exception($"Error: No se encontro ningun post entre {f1.ToString("dd-MM-yyyy")} y {f2.ToString("dd-MM-yyyy")}");
            }

            ret.Sort();

            return ret;
        }


        //Recorre la lista de usuarios y en caso de ser un miembro comprueba la cantidad de publiaciones que ha realizado usando la funcion utilizada para el otro requerimiento
        //Guarda en una variable de control de maximos la cantidad de publicaciones realizada en caso de ser mayor a la anteriormente guardada en la variable
        //Y por ultimo en caso de que haya un usuario con mas publicaciones, limpia la lista y agrega a ese usuario, o en caso de que el usuario tenga la misma cantidad de publiaciones
        //Solo lo agrega y al finalizar retorna la lista
        public List<Miembro> GetMiembrosConMasPublicaciones()
        {
            List<Miembro> ret = new List<Miembro>();
            int CantidadMayorPublicador = 0;

            foreach (Usuario u in Usuarios)
            {
                if (u is Miembro && GetPublicacionesXEmail(u.Email).Count > CantidadMayorPublicador)
                {
                    CantidadMayorPublicador = GetPublicacionesXEmail(u.Email).Count;
                    ret.RemoveRange(0, ret.Count);
                    ret.Add((Miembro)u);
                }
                else if (GetPublicacionesXEmail(u.Email).Count == CantidadMayorPublicador)
                {
                    ret.Add((Miembro)u);
                }
            }

            return ret;
        }
```

```
        #endregion

        public void Precarga() //Precarga de datos
        {

            #region Alta de miembros

            Miembro m1 = new Miembro("luucass26@gmail.com", "contrasenia123", "Lucas",
"Giusiano", new DateTime(2000, 9, 26));
            Miembro m2 = new Miembro("usuario2@hotmail.com", "qwerty1", "Alice", "Smith", new
DateTime(1988, 10, 20));
            Miembro m3 = new Miembro("ejemplo3@yahoo.com", "password", "Michael", "Johnson", new
DateTime(1993, 8, 7));
            Miembro m4 = new Miembro("nuevocorreo4@gmail.com", "myp@ss1", "Sophia", "Brown", new
DateTime(1985, 3, 25));
            Miembro m5 = new Miembro("usuario5@hotmail.com", "letmein", "Olivia", "Taylor", new
DateTime(1991, 12, 10));
            Miembro m6 = new Miembro("correo6@gmail.com", "securepass", "Matthew", "Robinson",
new DateTime(1987, 7, 14));
            Miembro m7 = new Miembro("otrocorreo7@yahoo.com", "p@ssw0rd", "Emma", "Hernandez",
new DateTime(1989, 6, 18));
            Miembro m8 = new Miembro("micorreo8@hotmail.com", "ilovecoding", "Daniel", "Garcia",
new DateTime(1994, 4, 5));
            Miembro m9 = new Miembro("nuevo_miembro9@gmail.com", "mypassword", "Sophie",
"Anderson", new DateTime(1996, 9, 22));
            Miembro m10 = new Miembro("correo10@yahoo.com", "password123", "Ryan", "Baker", new
DateTime(1986, 2, 12));

            Administrador a1 = new Administrador("admin@gmail.com", "admin123");

            AltaUsuario(m1);
            AltaUsuario(m2);
            AltaUsuario(m3);
            AltaUsuario(m4);
            AltaUsuario(m5);
            AltaUsuario(m6);
            AltaUsuario(m7);
            AltaUsuario(m8);
            AltaUsuario(m9);
            AltaUsuario(m10);
            AltaUsuario(a1);

            #endregion

            #region Alta de invitaciones

            Invitacion i1 = new Invitacion(m1, m2);
            Invitacion i2 = new Invitacion(m1, m3);
```

```
        Invitacion i3 = new Invitacion(m1, m4);
        Invitacion i4 = new Invitacion(m1, m5);
        Invitacion i5 = new Invitacion(m1, m6);
        Invitacion i6 = new Invitacion(m1, m7);
        Invitacion i7 = new Invitacion(m1, m8);
        Invitacion i8 = new Invitacion(m1, m9);
        Invitacion i9 = new Invitacion(m1, m10);

        Invitacion in2 = new Invitacion(m3,m2);
        Invitacion in3 = new Invitacion(m4,m2);
        Invitacion in4 = new Invitacion(m5,m2);
        Invitacion in5 = new Invitacion(m6,m2);
        Invitacion in6 = new Invitacion(m7,m2);
        Invitacion in7 = new Invitacion(m8,m2);
        Invitacion in8 = new Invitacion(m9,m2);
        Invitacion in9 = new Invitacion(m10,m2);

        Invitacion in10 = new Invitacion(m3, m5);
        Invitacion in11 = new Invitacion(m3, m7);
        Invitacion in12 = new Invitacion(m3, m6);
        Invitacion in13 = new Invitacion(m4, m8);
        Invitacion in14 = new Invitacion(m4, m5);
        Invitacion in15 = new Invitacion(m4, m9);
        Invitacion in16 = new Invitacion(m5, m10);
        Invitacion in17 = new Invitacion(m5, m7);
        Invitacion in18 = new Invitacion(m5, m8);
        Invitacion in19 = new Invitacion(m6, m7);
        Invitacion in20 = new Invitacion(m6, m10);
        Invitacion in21 = new Invitacion(m6, m4);
        Invitacion in22 = new Invitacion(m7, m8);
        Invitacion in23 = new Invitacion(m7, m4);
        Invitacion in24 = new Invitacion(m7, m5);
        Invitacion in25 = new Invitacion(m8, m3);
        Invitacion in26 = new Invitacion(m8, m9);
        Invitacion in27 = new Invitacion(m8, m6);
        Invitacion in28 = new Invitacion(m9, m6);
        Invitacion in29 = new Invitacion(m9, m10);
        Invitacion in30 = new Invitacion(m9, m5);
        Invitacion in31 = new Invitacion(m10, m3);
        Invitacion in32 = new Invitacion(m10, m4);
        Invitacion in33 = new Invitacion(m10, m7);


        AltaInvitacion(i1);
        AltaInvitacion(i2);
        AltaInvitacion(i3);
        AltaInvitacion(i4);
        AltaInvitacion(i5);
        AltaInvitacion(i6);
```

```
        AltaInvitacion(i7);
        AltaInvitacion(i8);
        AltaInvitacion(i9);
        AltaInvitacion(in2);
        AltaInvitacion(in3);
        AltaInvitacion(in4);
        AltaInvitacion(in5);
        AltaInvitacion(in6);
        AltaInvitacion(in7);
        AltaInvitacion(in8);
        AltaInvitacion(in9);
        AltaInvitacion(in10);
        AltaInvitacion(in11);
        AltaInvitacion(in12);
        AltaInvitacion(in13);
        AltaInvitacion(in14);
        AltaInvitacion(in15);
        AltaInvitacion(in16);
        AltaInvitacion(in17);
        AltaInvitacion(in18);
        AltaInvitacion(in19);
        AltaInvitacion(in20);
        AltaInvitacion(in21);
        AltaInvitacion(in22);
        AltaInvitacion(in23);
        AltaInvitacion(in24);
        AltaInvitacion(in25);
        AltaInvitacion(in26);
        AltaInvitacion(in27);
        AltaInvitacion(in28);
        AltaInvitacion(in29);
        AltaInvitacion(in30);
        AltaInvitacion(in31);
        AltaInvitacion(in32);
        AltaInvitacion(in33);

        i1.AceptarInvitacion();
        i2.AceptarInvitacion();
        i3.AceptarInvitacion();
        i4.AceptarInvitacion();
        i5.AceptarInvitacion();
        i6.AceptarInvitacion();
        i7.AceptarInvitacion();
        i8.AceptarInvitacion();
        i9.AceptarInvitacion();
        in2.AceptarInvitacion();
        in3.AceptarInvitacion();
        in4.AceptarInvitacion();
        in5.AceptarInvitacion();
```

```
            in6.AceptarInvitacion();
            in7.AceptarInvitacion();
            in8.AceptarInvitacion();
            in9.AceptarInvitacion();
            in10.AceptarInvitacion();
            in11.RechazarInvitacion();
            in13.AceptarInvitacion();
            in14.AceptarInvitacion();
            in15.AceptarInvitacion();
            in16.RechazarInvitacion();
            in17.RechazarInvitacion();
            in18.AceptarInvitacion();
            in19.AceptarInvitacion();
            in20.RechazarInvitacion();
            in21.AceptarInvitacion();
            in25.RechazarInvitacion();
            in26.RechazarInvitacion();
            in27.RechazarInvitacion();
            in28.AceptarInvitacion();
            in29.AceptarInvitacion();
            in31.RechazarInvitacion();
            in32.AceptarInvitacion();

            #endregion

            #region Alta de Posts y Comentarios con Reacciones

            //Posts
            Post p1 = new Post("Nuevo Zelda", "'The Legend of Zelda: Breath of the Wild' es una obra maestra de la libertad y la aventura. Desde explorar el vasto reino de Hyrule hasta los emocionantes combates y desafiantes rompecabezas en santuarios, el juego cautiva. Con gráficos impresionantes y una banda sonora envolvente, ofrece una experiencia inolvidable. Es un hito en la historia de los videojuegos y esencial para los amantes de la fantasía y la acción.", m1, "zelda.png", false);
            Post p2 = new Post("Épica batalla en Fortnite", "Anoche tuve la batalla más intensa en Fortnite. ¡Conseguí esa tan ansiada victoria magistral! #Fortnite #VictoriaRoyale", m2, "fortnite_batalla.png", true);
            Post p3 = new Post("Descubriendo secretos en Skyrim", "Explorando Skyrim y encontré un lugar oculto lleno de tesoros. ¡La emoción de descubrir secretos en este juego es increíble! #Skyrim #Exploración", m3, "skyrim_secreto.png", false);
            Post p4 = new Post("Speedrun récord en Super Mario", "Hoy logré mi mejor tiempo en un speedrun de Super Mario. ¡Rompiendo récords y mejorando cada día! #SuperMario #Speedrun", m4, "mario_speedrun.png", false);
            Post p5 = new Post("Logros desbloqueados en Dark Souls", "Conseguí vencer a todos los jefes en Dark Souls. ¡Un desafío épico superado! #DarkSouls #GitGud", m5, "dark_souls_logros.png", false);
            Post p6 = new Post("Emocionante partida en League of Legends", "¡Increíble partida en League of Legends! Logramos una remontada épica y ganamos la partida en el último segundo. #LeagueOfLegends #GamerLife", m6, "lol_partida.png", true);
```

```
            p1.Fecha = new DateTime(2010, 7, 5);
            p2.Fecha = new DateTime(2012, 4, 15);
            p3.Fecha = new DateTime(2013, 9, 20);
            p4.Fecha = new DateTime(2015, 2, 10);
            p5.Fecha = new DateTime(2015, 11, 8);
            p6.Fecha = new DateTime(2018, 6, 25);

            // Comentarios para el post p1 (Nuevo Zelda)
            Comentario c1_p1 = new Comentario("Amo la saga Zelda", "The Legend of Zelda siempre ha sido mi saga de videojuegos favorita. Cada entrega es única y especial.", m4);
            Comentario c2_p1 = new Comentario("¿Alguien más emocionado por esto?", "Estoy contando los días para el lanzamiento. ¡No puedo esperar para explorar Hyrule en Breath of the Wild!", m7);
            Comentario c3_p1 = new Comentario("Los gráficos se ven alucinantes", "Vi algunos avances y los gráficos son impresionantes. La atención al detalle es fenomenal.", m2);
            Comentario c4_p1 = new Comentario("¡Quiero jugarlo ya!", "Soy un gran fan de Zelda y esta nueva entrega se ve prometedora. ¡No puedo esperar a sumergirme en esta aventura!", m6);

            // Comentarios para el post p2 (Fortnite)
            Comentario c1_p2 = new Comentario("¡Me encanta Fortnite!", "Es un juego tan divertido, especialmente cuando llegas a la última fase. ¡Buena jugada!", m7);
            Comentario c2_p2 = new Comentario("¿Qué estrategia usaste?", "Estoy tratando de mejorar en Fortnite, ¿tienes algún consejo sobre las mejores estrategias?", m8);
            Comentario c3_p2 = new Comentario("Genial, felicidades por la victoria", "Sé lo difícil que es lograr una Victoria Royale en Fortnite, ¡buen trabajo!", m9);
            Comentario c4_p2 = new Comentario("¿Alguien quiere unirse para jugar?", "Estoy buscando compañeros de escuadrón en Fortnite. ¡Si estás interesado, házmelo saber!", m10);

            // Comentarios para el post p3 (Skyrim)
            Comentario c1_p3 = new Comentario("Skyrim es mi favorito", "He pasado horas explorando Skyrim. Siempre hay algo nuevo por descubrir.", m5);
            Comentario c2_p3 = new Comentario("¡Qué emocionante!", "Skyrim tiene algunos de los paisajes más impresionantes. Me encanta perderme en este mundo.", m6);
            Comentario c3_p3 = new Comentario("Esa sensación de encontrar secretos...", "En Skyrim, descubrir esos lugares ocultos es tan satisfactorio. ¡Bien hecho!", m7);
            Comentario c4_p3 = new Comentario("¿Cuál es tu misión favorita?", "Estoy interesado en saber cuál es tu misión favorita en Skyrim. Para mí, es 'La caída de Alduin'.", m8);

            // Comentarios para el post p4 (Super Mario)
            Comentario c1_p4 = new Comentario("Adictivo juego", "Super Mario es un clásico atemporal. ¡Nunca me canso de jugarlo!", m3);
            Comentario c2_p4 = new Comentario("Recuerdos de la infancia", "Jugaba a Super Mario cuando era niño, ¡y aún lo juego ahora! La nostalgia es increíble.", m4);
            Comentario c3_p4 = new Comentario("¿En qué plataforma juegas?", "Estoy pensando en empezar a jugar Super Mario. ¿En qué plataforma juegas? ¿Switch o emulador?", m5);
            Comentario c4_p4 = new Comentario("¿Tienes alguna estrategia?", "¿Tienes alguna estrategia o consejo para avanzar rápidamente en Super Mario? ¡Estoy atascado en una fase difícil!", m6);
```

```
            // Comentarios para el post p5 (Dark Souls)
            Comentario c1_p5 = new Comentario("Dark Souls es un desafío", "Este juego es todo un desafío, pero la satisfacción de vencer a un jefe es inigualable. ¡Buena suerte en tu travesía!", m2);
            Comentario c2_p5 = new Comentario("Combates intensos", "Los combates en Dark Souls son tan intensos y estratégicos. Cada encuentro es una prueba de habilidad.", m4);
            Comentario c3_p5 = new Comentario("Mis momentos favoritos", "Mis momentos favoritos en Dark Souls son cuando derroto a un jefe en el primer intento. Es una sensación increíble.", m7);
            Comentario c4_p5 = new Comentario("Dark Souls: un clásico moderno", "Dark Souls redefinió los juegos de acción y RPG. Una obra maestra que todo jugador debería experimentar.", m8);

            // Comentarios para el post p6 (League of Legends)
            Comentario c1_p6 = new Comentario("¿Cuál es tu campeón favorito?", "Estoy interesado en saber cuál es tu campeón favorito en League of Legends y por qué. ¡Comparte tu opinión!", m1);
            Comentario c2_p6 = new Comentario("Amo jugar LoL", "League of Legends es mi juego favorito. ¡Las estrategias y el trabajo en equipo hacen que cada partida sea única!", m3);
            Comentario c3_p6 = new Comentario("¡Esa remontada fue épica!", "Me encanta cuando las partidas de League of Legends se vuelven intensas y logras una remontada asombrosa. ¡Muy bien hecho!", m5);

            // Asignación de comentarios a sus respectivos posts

            // Para el post p1 (Nuevo Zelda)
            p1.Comentar(c1_p1);
            p1.Comentar(c2_p1);
            p1.Comentar(c3_p1);
            p1.Comentar(c4_p1);

            // Para el post p2 (Fortnite)
            p2.Comentar(c1_p2);
            p2.Comentar(c2_p2);
            p2.Comentar(c3_p2);
            p2.Comentar(c4_p2);

            // Para el post p3 (Skyrim)
            p3.Comentar(c1_p3);
            p3.Comentar(c2_p3);
            p3.Comentar(c3_p3);
            p3.Comentar(c4_p3);

            // Para el post p4 (Super Mario)
            p4.Comentar(c1_p4);
            p4.Comentar(c2_p4);
            p4.Comentar(c3_p4);
            p4.Comentar(c4_p4);
```

```
// Para el post p5 (Dark Souls)
p5.Comentar(c1_p5);
p5.Comentar(c2_p5);
p5.Comentar(c3_p5);
p5.Comentar(c4_p5);

// Para el post p6 (League of Legends)
p6.Comentar(c1_p6);
p6.Comentar(c2_p6);
p6.Comentar(c3_p6);

//Reacciones para los post
// Reacciones para el post p1 (Nuevo Zelda)
p1.Reaccionar(new Reaccion("like", m1));
p1.Reaccionar(new Reaccion("dislike", m2));
p1.Reaccionar(new Reaccion("like", m7));
p1.Reaccionar(new Reaccion("dislike", m8));

// Reacciones para el post p2 (Fortnite)
p2.Reaccionar(new Reaccion("like", m3));
p2.Reaccionar(new Reaccion("dislike", m4));
p2.Reaccionar(new Reaccion("like", m2));
p2.Reaccionar(new Reaccion("dislike", m9));

// Reacciones para el post p3 (Skyrim)
p3.Reaccionar(new Reaccion("like", m5));
p3.Reaccionar(new Reaccion("dislike", m6));

// Reacciones para el post p4 (Super Mario)
p4.Reaccionar(new Reaccion("like", m7));
p4.Reaccionar(new Reaccion("dislike", m8));

// Reacciones para el post p5 (Dark Souls)
p5.Reaccionar(new Reaccion("like", m9));
p5.Reaccionar(new Reaccion("dislike", m1));

// Reacciones para el post p6 (League of Legends)
p6.Reaccionar(new Reaccion("like", m2));
p6.Reaccionar(new Reaccion("dislike", m3));


//Reacciones para comentarios
// Reacciones para el comentario c1_p1 (Amo la saga Zelda)

c1_p1.Reaccionar(new Reaccion("like", m4));
c1_p1.Reaccionar(new Reaccion("dislike", m5));
c1_p1.Reaccionar(new Reaccion("like", m6));
```

```
// Reacciones para el comentario c2_p1 (¿Alguien más emocionado por esto?)
c2_p1.Reaccionar(new Reaccion("like", m7));
c2_p1.Reaccionar(new Reaccion("dislike", m8));
c2_p1.Reaccionar(new Reaccion("like", m9));

// Reacciones para el comentario c3_p1 (Los gráficos se ven alucinantes)
c3_p1.Reaccionar(new Reaccion("like", m1));
c3_p1.Reaccionar(new Reaccion("dislike", m2));
c3_p1.Reaccionar(new Reaccion("dislike", m3));

// Reacciones para el comentario c4_p1 (¡Quiero jugarlo ya!)
c4_p1.Reaccionar(new Reaccion("like", m1));
c4_p1.Reaccionar(new Reaccion("like", m2));
c4_p1.Reaccionar(new Reaccion("like", m3));

// Reacciones para el comentario c1_p2 (¡Me encanta Fortnite!)
c1_p2.Reaccionar(new Reaccion("like", m3));
c1_p2.Reaccionar(new Reaccion("dislike", m4));
c1_p2.Reaccionar(new Reaccion("like", m5));

// Reacciones para el comentario c2_p2 (¿Qué estrategia usaste?)
c2_p2.Reaccionar(new Reaccion("like", m6));
c2_p2.Reaccionar(new Reaccion("dislike", m7));
c2_p2.Reaccionar(new Reaccion("like", m8));

// Reacciones para el comentario c3_p2 (Genial, felicidades por la victoria)
c3_p2.Reaccionar(new Reaccion("like", m9));
c3_p2.Reaccionar(new Reaccion("dislike", m1));
c3_p2.Reaccionar(new Reaccion("like", m2));

// Reacciones para el comentario c4_p2 (¿Alguien quiere unirse para jugar?)
c4_p2.Reaccionar(new Reaccion("like", m9));
c4_p2.Reaccionar(new Reaccion("dislike", m1));
c4_p2.Reaccionar(new Reaccion("like", m2));

// Reacciones para el comentario c1_p3 (Skyrim es mi favorito)
c1_p3.Reaccionar(new Reaccion("like", m4));
c1_p3.Reaccionar(new Reaccion("dislike", m5));
c1_p3.Reaccionar(new Reaccion("like", m6));

// Reacciones para el comentario c2_p3 (¡Qué emocionante!)
c2_p3.Reaccionar(new Reaccion("like", m7));
c2_p3.Reaccionar(new Reaccion("dislike", m8));
c2_p3.Reaccionar(new Reaccion("like", m9));

// Reacciones para el comentario c3_p3 (Esa sensación de encontrar secretos...)
c3_p3.Reaccionar(new Reaccion("like", m1));
c3_p3.Reaccionar(new Reaccion("dislike", m2));
c3_p3.Reaccionar(new Reaccion("like", m3));
```

```
// Reacciones para el comentario c3_p3 (Esa sensación de encontrar secretos...)
c4_p3.Reaccionar(new Reaccion("like", m1));
c4_p3.Reaccionar(new Reaccion("dislike", m2));
c4_p3.Reaccionar(new Reaccion("like", m3));

// Reacciones para el comentario c1_p4 (Adictivo juego)
c1_p4.Reaccionar(new Reaccion("like", m4));
c1_p4.Reaccionar(new Reaccion("dislike", m5));
c1_p4.Reaccionar(new Reaccion("like", m6));

// Reacciones para el comentario c2_p4 (Recuerdos de la infancia)
c2_p4.Reaccionar(new Reaccion("like", m7));
c2_p4.Reaccionar(new Reaccion("dislike", m8));
c2_p4.Reaccionar(new Reaccion("like", m9));

// Reacciones para el comentario c3_p4 (¿En qué plataforma juegas?)
c3_p4.Reaccionar(new Reaccion("like", m1));
c3_p4.Reaccionar(new Reaccion("dislike", m2));
c3_p4.Reaccionar(new Reaccion("like", m3));

// Reacciones para el comentario c4_p4 (¿En qué plataforma juegas?)
c4_p4.Reaccionar(new Reaccion("like", m1));
c4_p4.Reaccionar(new Reaccion("dislike", m2));
c4_p4.Reaccionar(new Reaccion("like", m3));

// Reacciones para el comentario c1_p5 (Dark Souls es un desafío)
c1_p5.Reaccionar(new Reaccion("like", m4));
c1_p5.Reaccionar(new Reaccion("dislike", m5));
c1_p5.Reaccionar(new Reaccion("like", m6));

// Reacciones para el comentario c2_p5 (Combates intensos)
c2_p5.Reaccionar(new Reaccion("like", m7));
c2_p5.Reaccionar(new Reaccion("dislike", m8));
c2_p5.Reaccionar(new Reaccion("like", m9));

// Reacciones para el comentario c3_p5 (Mis momentos favoritos)
c3_p5.Reaccionar(new Reaccion("like", m1));
c3_p5.Reaccionar(new Reaccion("dislike", m2));
c3_p5.Reaccionar(new Reaccion("like", m3));

// Reacciones para el comentario c4_p5 (Dark Souls: un clásico moderno)
c4_p5.Reaccionar(new Reaccion("like", m1));
c4_p5.Reaccionar(new Reaccion("dislike", m2));
c4_p5.Reaccionar(new Reaccion("like", m3));

// Reacciones para el comentario c1_p6 (¿Cuál es tu campeón favorito?)
c1_p6.Reaccionar(new Reaccion("like", m4));
c1_p6.Reaccionar(new Reaccion("dislike", m5));
```

```
c1_p6.Reaccionar(new Reaccion("like", m6));

// Reacciones para el comentario c2_p6 (Amo jugar LoL)
c2_p6.Reaccionar(new Reaccion("like", m7));
c2_p6.Reaccionar(new Reaccion("dislike", m8));
c2_p6.Reaccionar(new Reaccion("like", m9));

// Reacciones para el comentario c3_p6 (¡Esa remontada fue épica!)
c3_p6.Reaccionar(new Reaccion("like", m1));
c3_p6.Reaccionar(new Reaccion("dislike", m2));
c3_p6.Reaccionar(new Reaccion("like", m3));

//Alta de post
AltaPublicacion(p1);
AltaPublicacion(p2);
AltaPublicacion(p3);
AltaPublicacion(p4);
AltaPublicacion(p5);
AltaPublicacion(p6);

//Alta de comentarios
AltaPublicacion(c1_p1);
AltaPublicacion(c2_p1);
AltaPublicacion(c3_p1);
AltaPublicacion(c4_p1);

AltaPublicacion(c1_p2);
AltaPublicacion(c2_p2);
AltaPublicacion(c3_p2);
AltaPublicacion(c3_p2);

AltaPublicacion(c1_p3);
AltaPublicacion(c2_p3);
AltaPublicacion(c3_p3);
AltaPublicacion(c4_p3);

AltaPublicacion(c1_p4);
AltaPublicacion(c2_p4);
AltaPublicacion(c3_p4);
AltaPublicacion(c4_p4);

AltaPublicacion(c1_p5);
AltaPublicacion(c2_p5);
AltaPublicacion(c3_p5);
AltaPublicacion(c4_p5);

AltaPublicacion(c1_p6);
AltaPublicacion(c2_p6);
AltaPublicacion(c3_p6);
```

```
        }

        #endregion
    }
```

4.2 Clase Usuario

```
    public abstract class Usuario : IValidacion
    {
        private static int UltimoId { get; set; }
        public int Id { get; }
        public string Email { get; set; }
        public string Contrasenia { get; set; }
        public bool Bloqueado { get; set; }

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

        public Usuario()
        {
            Id = UltimoId++;
            Bloqueado = false;
        }

        public override string ToString()
        {
            return $"Email: {Email} || Contraseña: {Contrasenia}\n";
        }

        //Valida que el email y la contraseña estén correctos. En caso de que alguna de las dos
no cumpla con los requerimiento, se guarda el mensaje para el usuario en una variable
        // y por ultimo si esa variable no esta vacía, se arroja la excepción expecificando
todos los campos que estan mal y porque
        public virtual void Validar()
        {
            string mensajeUsuario = String.Empty;

            if (Email.IndexOf("@") != Email.LastIndexOf("@") || Email.IndexOf("@") == -1)
            {
                mensajeUsuario += "Email inválido: El email no contiene arroba";
            }
            if (Email.IndexOf(".") != Email.LastIndexOf(".") || Email.IndexOf(".") == -1)
            {
                mensajeUsuario += "\nEmail inválido: El email no contiene punto";
            }
            if (String.IsNullOrEmpty(Contrasenia) || Contrasenia.Length <= 5)
            {
```

```
                mensajeUsuario += "\nContraseña inválida: La contraseña debe tener más de cinco
caracteres";
            }
            if (!String.IsNullOrEmpty(mensajeUsuario))
            {
                throw new Exception(mensajeUsuario);
            }
        }

        public void CambiarBloqueo() // Cambia el estado del atributo bloqueado dependiendo del
estado actual del mismo
        {
            if (!Bloqueado)
            {
                Bloqueado = true;
            }
            else
            {
                Bloqueado = false;
            };
        }
    }
```

4.3 Clase Miembro

```
public class Miembro : Usuario, IComparable<Miembro>
    {
        public string Nombre { get; set; }
        public string Apellido { get; set; }
        public DateTime FechaNac { get; set; }
        private List<Miembro> Amigos = new List<Miembro>();

        public Miembro(string email, string contrasenia, string nombre, string apellido,
DateTime fechaNac) : base(email, contrasenia)
        {
            Nombre = nombre;
            Apellido = apellido;
            FechaNac = fechaNac;
        }

        public Miembro() : base()
        {

        }

        public List<Miembro> VerAmigos()//Devuelve la lista de amigos del ususario
        {
            return Amigos;
        }

        public bool BuscarAmigo(int idAmigoBuscado)
        {
            foreach (var amigo in Amigos)
            {
                if (amigo.Id == idAmigoBuscado)
                {
                    return true;
                }
            }
            return false;
        }

        public void AgregarAmigo(Miembro nuevoAmigo)//Agrega un nuevo miembro a la lista de
amigos
        {
            Amigos.Add(nuevoAmigo);
        }

        public override string ToString()
        {
            string estaBloqueado = "No";
            if (Bloqueado)
            {
```

```
                estaBloqueado = "Si";
            }

            return base.ToString() + $"Nombre y Apellido: {Nombre} {Apellido} || Fecha de
nacimiento: {FechaNac} || Está bloqueado: {estaBloqueado}";
        }

        //Valida los atributos del nuevo miembro y en caso de que no cumpla con alguno, guarda
el mensaje de error en una variable y al final se la variable de mensajes no se encuentra vacía
        //Arroja la excepción con todos los mensajes de los campos que hayan estado incorrectos
        public override void Validar()
        {
            base.Validar();
            string mensajeMiembro = string.Empty;

            if (String.IsNullOrEmpty(Nombre) || Nombre.Length < 3)
            {
                mensajeMiembro += "\nNombre inválido: El nombre debe tener mas de tres
caracteres";
            }
            if (String.IsNullOrEmpty(Apellido) || Apellido.Length < 3)
            {
                mensajeMiembro += "\nApellido inválido: El apellido debe tener mas de tres
caracteres";
            }
            if (FechaNac == DateTime.MinValue)
            {
                mensajeMiembro += "\nFecha inválida: No ha ingresado una fecha";
            }
            if (!String.IsNullOrEmpty(mensajeMiembro))
            {
                throw new Exception(mensajeMiembro);
            }
        }

        public int CompareTo(Miembro? other)
        {
            if (Nombre.CompareTo(other.Nombre) > 0)
            {
                return 1;
            }
            else if (Nombre.CompareTo(other.Nombre) < 0)
            {
                return -1;
            }
            else
            {
                if (Apellido.CompareTo(other.Apellido) > 0)
                {
```

```
                return 1;
            }
            else if (Apellido.CompareTo(other.Apellido) < 0)
            {
                return -1;
            }
            else
            {
                return 0;
            }
        }
    }
}
```

4.4 Clase Administrador

```
namespace Dominio
{
    public class Administrador : Usuario
    {
        public Administrador(string email, string contrasenia) : base(email,
contrasenia)
        {

        }

        public Administrador() : base()
        {

        }

        public override string ToString()
        {
            return base.ToString();
        }

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

4.5 Clase Publicación

```
public abstract class Publicacion : IValidacion, IComparable<Publicacion>
   {
        private static int UltimoId { get; set; }
        public int Id { get; }
        public string Titulo { get; set; }
        public string Contenido { get; set; }
        public bool Censurado { get; set; }
        public DateTime Fecha { get; set; }
        public Miembro Autor { get; set; }
        protected List<Reaccion> Reacciones = new List<Reaccion>();

        public Publicacion(string titulo, string contenido, Miembro autor)
        {
            Id = UltimoId++;
            Titulo = titulo;
            Contenido = contenido;
            Censurado = false;
            Fecha = DateTime.Now;
            Autor = autor;
        }

        public Publicacion()
        {
            Id = UltimoId++;
            Censurado = false;
            Fecha = DateTime.Now;
        }

        public override string ToString()
        {
            string txtCensurado = "No";
            if (Censurado)
            {
                txtCensurado = "Si";
            }
            return $"Autor: {Autor.Nombre} {Autor.Apellido} || Titulo: {Titulo} || Fecha:
{Fecha.ToString("dd/MM/yyyy")} || Censurado: {txtCensurado} \nContenido: {Contenido}";
        }

        public int GetCantTipoReaccion(string tipo) //Devuelve la cantidad de reacciones de un
tipo que tiene una publicacion
        {
            int ret = 0;

            foreach (var reaccion in Reacciones)
            {
                if (reaccion.Tipo == tipo)
```

```
                {
                    ret++;
                }
            }
            return ret;
        }

        public virtual void Validar()//Valida que el titulo y el contenido de la publicación no
esten vacios
        {
            if (String.IsNullOrEmpty(Titulo) || Titulo.Length < 5)
            {
                throw new Exception("El titulo debe tener al menos cinco caracteres");
            }
            if (String.IsNullOrEmpty(Contenido))
            {
                throw new Exception("El contenido no puede estar vacío");
            }
        }

        //Verifica si el usuario que está reaccionando a la publicación ya lo hizo anteriormente
y en caso de que asi sea pero la reacción actual sea otra, la cambia, de lo contrario arroja una
excepción
        //Si el usuario que reacciona es uno nuevo, agrega la reaccion a la lista de reacciones
de la publicación
        public void Reaccionar(Reaccion r)
        {
            bool agregar = true;

            foreach (Reaccion reaccionDePubli in Reacciones)
            {
                if (reaccionDePubli.Autor.Id == r.Autor.Id && reaccionDePubli.Tipo == r.Tipo)
                {
                    agregar = false;
                    Reacciones.Remove(reaccionDePubli);
                    break;
                }
                else if (reaccionDePubli.Autor.Id == r.Autor.Id && reaccionDePubli.Tipo !=
r.Tipo)
                {
                    agregar = false;
                    reaccionDePubli.Tipo = r.Tipo;
                    break;
                }
            }

            if (agregar)
            {
                Reacciones.Add(r);
```

```
            }

        }

        public Reaccion UsuarioReacciono(int idUsuario) //Busca si el usuario logueado ya
reaccionó a esa publicacion
        {
            foreach (var reaccion in Reacciones)
            {
                if (reaccion.Autor.Id == Id)
                {
                    return reaccion;
                }
            }
            return null;
        }

        public List<Reaccion> GetReacciones()//Devuelve la lista de reacciones que tiene la
publicación
        {
            return Reacciones;
        }

        public abstract double ValorDeAceptacion();//Funcion abastracta para acceder a las
funciones de calculo del valor de aceptación de posts y comentarios.

        public void CambiarCensura()//Cambia el estado de censura dependiendo el estado actual
del mismo
        {
            if (Censurado)
            {
                Censurado = false;
            }
            else
            {
                Censurado = true;
            }
        }

        public int CompareTo(Publicacion? other)
        {
            if (Titulo.CompareTo(other.Titulo) > 0)
            {
                return 1;
            }
            else if (Titulo.CompareTo(other.Titulo) < 0)
            {
                return -1;
            }
```

```
            else
            {
                return 0;
            }
        }
    }
```

4.6 Clase Post

```
public class Post : Publicacion
    {
        public string Imagen { get; set; }
        public bool Privado { get; set; }
        private List<Comentario> Comentarios = new List<Comentario>();

        public Post(string titulo, string contenido, Miembro autor, string imagen, bool privado)
: base(titulo, contenido, autor)
        {
            Imagen = imagen;
            Privado = privado;
        }

        public Post() : base()
        {

        }

        public override string ToString()
        {
            string txtPrivacidad = "Público";
            if (Privado)
            {
                txtPrivacidad = "Privado";
            }
            return base.ToString() + $"\nImagen: {Imagen} || Privacidad: {txtPrivacidad} || Tipo
de Publicación: Post";
        }

        public override void Validar() //Valida que el texto de la imagen no este vacia y tenga
las terminaciones .png o .jpg
        {
            base.Validar();

            bool imagenTieneFormato = false;

            if (String.IsNullOrEmpty(Imagen))
            {
                throw new Exception("Imagen inválida: Debe ingresar el nombre de una imagen");
            }
            if (Imagen.EndsWith(".jpg"))
            {
                imagenTieneFormato = true;
            }
            if (Imagen.EndsWith(".png") && !imagenTieneFormato)
            {
                imagenTieneFormato = true;
```

```
            }
            if (!imagenTieneFormato)
            {
                throw new Exception("Imagen inválida: La imagen debe tener una extensión de
formato '.png' o '.jpg'");
            }
        }

        public void Comentar(Comentario c)//Añade un comentario a la lista de comentarios
        {
            c.Validar();
            Comentarios.Add(c);
        }

        public List<Comentario> GetComentarios(string RolMiembro)//Devuelve la lista de
comentarios que tiene el post dependiendo el rol del usuario que los solicite
        {
            if (RolMiembro == "Miembro")
            {
                List<Comentario> ret = new List<Comentario>();
                foreach (Comentario c in Comentarios)
                {
                    if (!c.Censurado)
                    {
                        ret.Add(c);
                    }
                }
                return ret;
            }
            return Comentarios;
        }

        //Recorre la lista de reacciones de la publicación contabilizando la cantidad de likes y
dislikes, realiza el calculo acorde a los post
        //en caso de ser un post publico, suma 10 puntos al resultado
        public override double ValorDeAceptacion()
        {
            int ret;
            int cantLikes = 0;
            int cantDislikes = 0;

            foreach (Reaccion r in Reacciones)
            {
                if (r.Tipo == "like")
                {
                    cantLikes++;
                }
                if (r.Tipo == "dislike")
                {
```

```
                    cantDislikes++;
                }
            }

            ret = (cantLikes * 5) + (cantDislikes * -2);

            if (!Privado)
            {
                ret += 10;
            }

            return ret;
        }

        public void CambiarPrivacidad()//Cambia el estado de la privacidad dependiendo del
estado actual de la misma
        {
            if (Privado)
            {
                Privado = false;
            }
            else
            {
                Privado = true;
            }
        }
    }
```

4.7 Clase Comentario

```
 public class Comentario : Publicacion
    {
        public Comentario(string titulo, string contenido, Miembro autor) : base(titulo,
contenido, autor)
        {

        }

        public Comentario() : base()
        {

        }

        public override string ToString()
        {
            return base.ToString() + "\n Tipo de Publicación: Comentario";
        }

        public override void Validar()//Valida que el titulo y el contenido del comentario no
esten vacíos
        {
            base.Validar();
            if (String.IsNullOrEmpty(Titulo) || Titulo.Length < 3)
            {
                throw new Exception("Error: El titulo debe contener al menos tres caracteres");
            }
            if (String.IsNullOrEmpty(Contenido))
            {
                throw new Exception("Error: El contenido del comentario no puede ser vacío");
            }
        }

        //Recorre la lista de reacciones del comentario contabilizando la cantidad de likes y
dislikes para hacer el calculo respectivo al comentario
        public override double ValorDeAceptacion()
        {
            int cantLikes = 0;
            int cantDislikes = 0;


            foreach (Reaccion r in Reacciones)
            {
                if (r.Tipo.ToLower() == "like")
                {
                    cantLikes++;
                }
                if (r.Tipo.ToLower() == "dislike")
                {
```

```
                    cantDislikes++;
                }
            }

            return (cantLikes * 5) + (cantDislikes * -2);
        }
    }
```

4.8 Clase Reacción

```
namespace Dominio
{
    public class Reaccion : IValidacion
    {
        public string Tipo { get; set; }
        public Miembro Autor { get; set; }

        public Reaccion(string tipo, Miembro autor)
        {
            Tipo = tipo.ToLower(); //El tipo se transforma a minusculas para facilitar el
trabajo de validacion del tipo
            Autor = autor;
        }

        public Reaccion()
        {

        }

        public void Validar() //Valida el tipo de la reaccion para que solo sea un tipo acorde a
los utilizados y permitidos para la aplicación
        {
            if (Tipo != "like" || Tipo != "dislike")
            {
                throw new Exception("Error: La reacción solo puede ser del tipo 'Like' o
'Dislike'");
            }
        }

        public override string ToString()
        {
            return $"Tipo: {Tipo} || Autor: {Autor.Email}";
        }
    }
}
```

4.9 Clase Invitación

```
 public class Invitacion
    {
        private static int UltimoId { get; set; }
        public int Id { get; }
        public Miembro Solicitante { get; set; }
        public Miembro Solicitado { get; set; }
        public EstadoInvitacion Estado { get; set; }
        public DateTime Fecha { get; set; }

        public Invitacion(Miembro solicitante, Miembro solicitado)
        {
            Id = UltimoId++;
            Solicitante = solicitante;
            Solicitado = solicitado;
            Estado = EstadoInvitacion.Pendiente; //La invitación se crea por defecto con el
estado PENDIENTE
            Fecha = DateTime.Now; //La fecha de la invitacion se fija en el momento de su
creación
        }

        public Invitacion()
        {
            Id = UltimoId++;
            Estado = EstadoInvitacion.Pendiente;
            Fecha = DateTime.Now;
        }

        public void AceptarInvitacion()//Cambia el estado de la invitaciona a APROBADA y agrega
a cada miembro a la lista de amigos del otro
        {
            Estado = EstadoInvitacion.Aprobada;

            Solicitante.AgregarAmigo(Solicitado);
            Solicitado.AgregarAmigo(Solicitante);
        }

        public void RechazarInvitacion() //Cambia el estado de la invitacion a RECHAZADA
        {
            Estado = EstadoInvitacion.Rechazada;
        }

        public override string ToString()
        {
            return $"Solicitante: {Solicitante.Nombre} {Solicitante.Apellido} || Solicitado:
{Solicitado.Nombre} {Solicitado.Apellido} || Estado de la initación: {Estado} || Fecha:
{Fecha.ToString("dd/MM/yyyy")}";
        }
    }
```

4.10 Enumerado Estado de Invitación

```
namespace Dominio
{
    public enum EstadoInvitacion
    {
        Pendiente = 0,
        Aprobada = 1,
        Rechazada = 2
    }
}
```

5. Links de Deploy Somee

http://Lucas-Giusiano-Obligatorio-2.somee.com
http://www.Lucas-Giusiano-Obligatorio-2.somee.com