Universidad ORT Uruguay
Facultad de Ingeniería
Escuela de tecnología

OBLIGATORIO PROGRAMACIÓN 2

[Gastón Varela - 316838]

Docente: [Joaquín Rodríguez Ferraro]
Carrera: [Analista en tecnologías de la información]
Grupo: [M2A (remoto)]
Fecha de entrega del documento: [XX-10-2023]

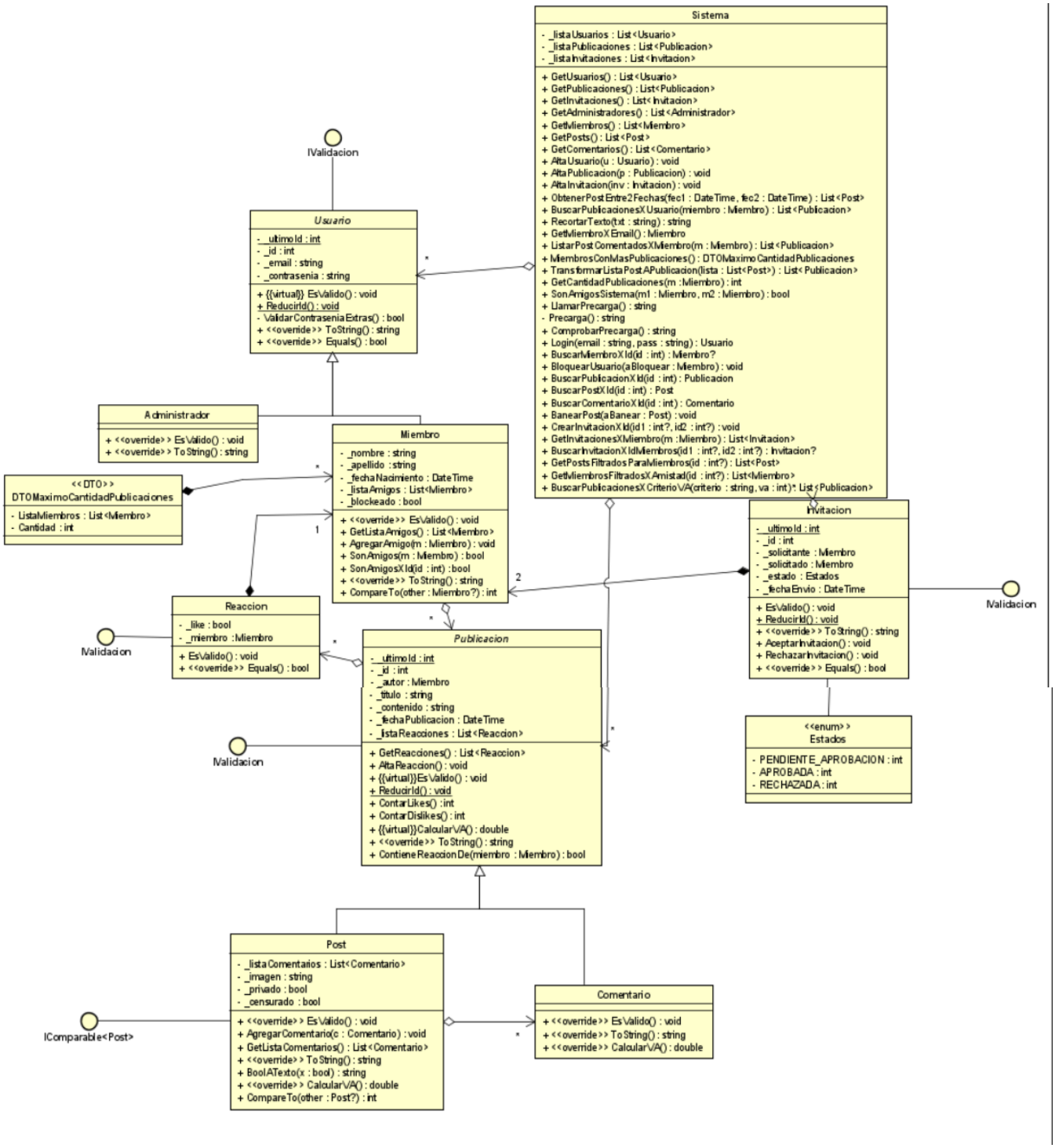
El obligatorio 1 fue realizado con un compañero, para el obligatorio 2, el grupo se disolvió. Esto ya fue aclarado con el profesor.

Indice

URL De Somee:

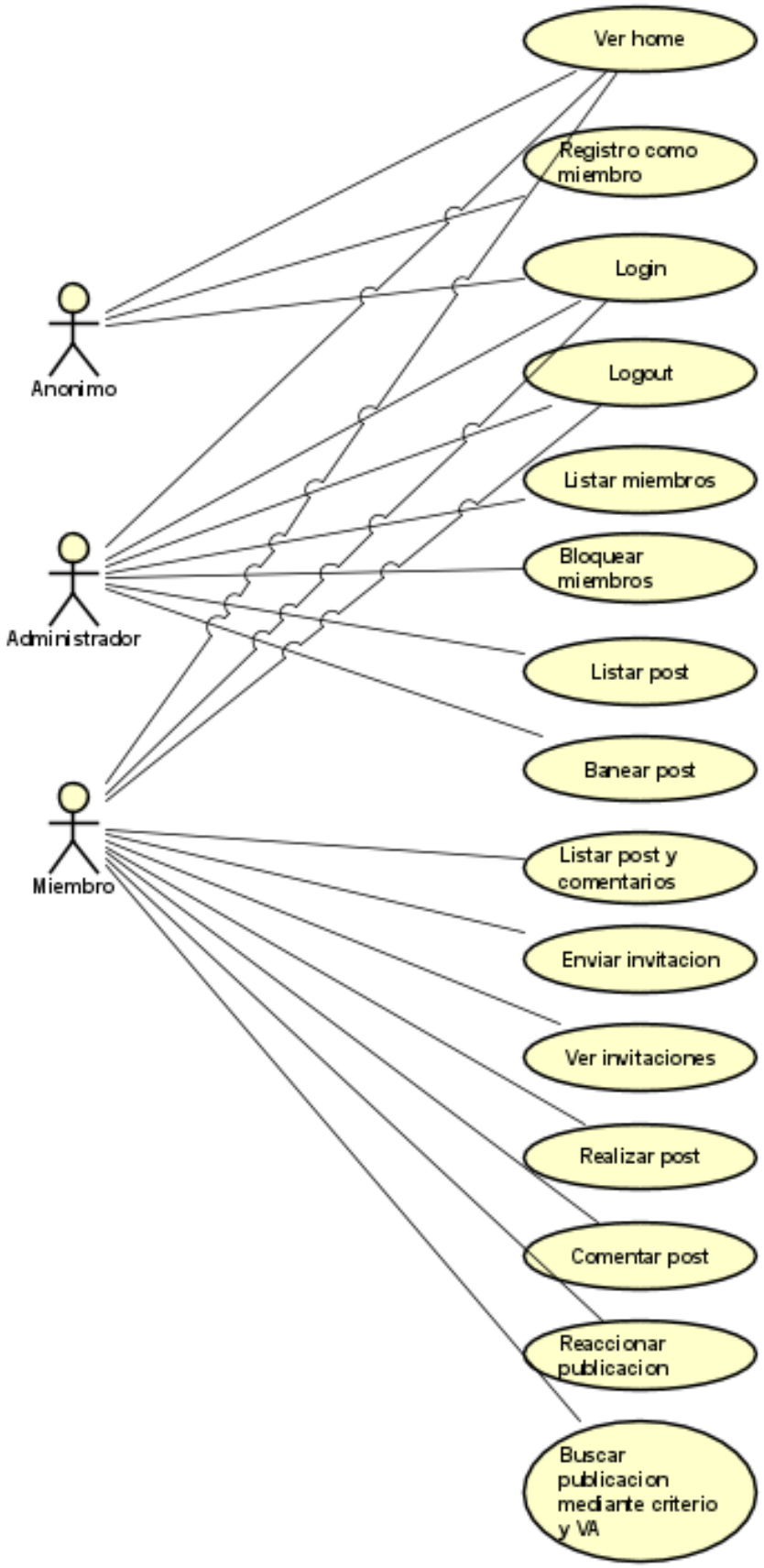
https://gv316838someeobl.somee.com/

Diagrama de clases



La navegabilidad de Sistema a Invitación es de *, esto está presente en el diagrama pero se movió en la foto y el archivo.

Diagrama de casos de uso



Precargas

Usuarios:

Nombre	Apellido	Fecha de Nacimiento	Correo	Contraseña
ADMIN	-	-	admin@gmail.com	Admin1234
Gastón	Varela	28/03/2004	gas@gmail.com	Gas1234
Joaquín	Rodriguez	01/07/1986	joaq@gmail.com	Joaq1234
Ana	Fernandéz	12/07/1990	ana@gmail.com	Ana1234
Juan	Perez	23/04/1999	juan@gmail.com	Juan1234
Juanchi	Juanza	12/08/2003	juanchi@gmail.com	Juanchi1234
Diego	Sabatel	29/12/1986	diego@gmail.com	Diego1234
Alejandra	Almeida	23/03/1988	alejandra@gmail.com	Alejandra1234
Jenny	González	18/05/1994	jenny@gmail.com	Jenny1234
Julieta	Vadetto	02/11/2000	julieta@gmail.com	Julieta1234
Pepe	Pérez	04/12/2003	pepe@gmail.com	Pepe1234
Gaspar	Mondadol	04/12/2003	gaspar@gmail.com	Gaspar1234
Carlos	Rodríguez	15/07/1990	carlos@gmail.com	Carlos1234
Laura	Martínez	08/09/1985	laura@gmail.com	Laura1234
Fernando	López	10/02/1998	fernando@gmail.com	Fernando1234
Ana	Sánchez	25/06/1992	ana2@gmail.com	Ana1234
Martín	Gómez	14/11/1987	martin@gmail.com	Martin1234
Lucía	Hernández	03/04/1995	lucia@gmail.com	Lucia1234
Pablo	Díaz	20/08/2001	pablo@gmail.com	Pablo1234
Isabel	Torres	07/10/1993	isabel@gmail.com	Isabel1234
Ricardo	Ramírez	30/01/2002	ricardo@gmail.com	Ricardo1234
María	Fernández	12/12/1989	maria@gmail.com	Maria1234
Jorge	Gutiérrez	05/03/1996	jorge@gmail.com	Jorge1234
Natalia	Luna	18/05/2000	natalia@gmail.com	Natalia1234
Andrés	Ortega	22/09/1986	andres@gmail.com	Andres1234
Valentina	Castro	08/04/1994	valentina@gmail.com	Valentina1234
Héctor	Mendoza	03/11/1999	hector@gmail.com	Hector1234
Sofía	Peralta	15/02/2004	sofia@gmail.com	Sofia1234
Eduardo	Guerrero	09/07/1984	eduardo@gmail.com	Eduardo1234
Camila	Rojas	28/12/1991	camila@gmail.com	Camila1234
Alejandro	Navarro	07/06/1997	alejandro@gmail.com	Alejandro1234

Publicaciones:
Posts:

Miembro	Titulo	Contenido	Imagen	Privacidad
Gastón	Agua salada	Las aguas saladas del océano en Uruguay son conocidas por sus tonos azules y verdes, ofreciendo hermosas playas y oportunidades para actividades marinas. Tomo agua salada todas las mañanas y tengo los riñones como piña.	oceano.png	Pública
Joaquín	Karate	El karate es un antiguo arte marcial japonés que se ha convertido en una disciplina globalmente reconocida y respetada.	karate.png	Pública
Ana	Caballos	Los caballos en Uruguay son parte esencial de su cultura, utilizados en el trabajo rural y en emocionantes deportes como el polo. Su presencia en la Semana Criolla y en estancias de renombre demuestra su importancia en la vida uruguaya.	caballo.png	Pública
Juan	Vacaciones	Las vacaciones son un bálsamo para el alma, un merecido descanso en medio de la rutina diaria.	playa.jpg	Privada
Juanchi	Dinosaurios	Los dinosaurios, esos gigantes antiguos, siguen fascinándonos. Su legado fósil nos conecta con un pasado misterioso y asombroso. ¿Cuál es tu dinosaurio favorito? 🦕 🦕 #Dinosaurios	dinosaurio.jpg	Privada

Comentarios:

Nombre del Miembro	Título	Contenido
Gastón	ME ENCANTA EL AGUA SALADA	AMO EL AGUA
Joaquín	DEJA DE SER TAN NABO	Como te va a gustar el agua salada, salame
Ana	Joaco, porque lo tratas mal?	Te parece bien? Todo bien en casa?
Juan	gente violenta!!	A ver si se calman un poco, así no se puede!!
Juanchi	El mejor arte	De todos los que existen es el mejor!!, escucho comentarios
Diego	mi abuela empezó Karate!!!	Tiene 104 años y es media rayada
Alejandra	Como hago para darle de comer a los caballos?	Hace dos meses les estoy intentando dar milanesas y ni las tocan
Jenny	Para al de arriba	Como le vas a dar milanesas al caballo?
Julieta	Que lindos!	Amo la mortadela
Pepe	Estoy en Rocha!!	Rocha es el mejor lugar para vacacionar....
Gastón	Para que sirven las vacaciones??	Es ese ansiado tiempo en el que dejamos atrás las preocupaciones laborales y nos sumergimos en un mundo de posibilidades y aventuras. Ya sea explorando destinos exóticos, relajándonos en la playa con el sonido del mar como banda sonora o simplemente disfrutando de la tranquilidad en casa, las vacaciones nos ofrecen la oportunidad de recargar energías, reconectar con seres queridos y crear recuerdos que perdurarán en nuestro corazón.
Joaquín	Que lindas son las licencias	AL FIN VACACIONESSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSS!!!!!!!, ya no podía mas con mi laburo!!
Ana	Los dinosaurios no existen	Nunca vi uno
Juan	Para el de arriba	Tu cerebro no existe
Juanchi	TAS LOCO	TODOS LOS POSTS SON ASI? DONDE ESTAN LOS MODS

Reacciones:

Tipo	Miembro	Reacción
Post	Gastón	Me gusta
Post	Joaquín	Me gusta
Post	Ana	Me gusta
Post	Juan	No me gusta
Post	Juanchi	No me gusta
Post	Diego	No me gusta
Post	Alejandra	No me gusta
Post	Jenny	No me gusta
Post	Julieta	Me gusta
Post	Pepe	Me gusta
Comentario	Gastón	Me gusta
Comentario	Joaquín	Me gusta
Comentario	Ana	Me gusta
Comentario	Juan	Me gusta
Comentario	Juanchi	Me gusta
Comentario	Diego	No me gusta
Comentario	Alejandra	No me gusta
Comentario	Jenny	No me gusta
Comentario	Julieta	No me gusta
Comentario	Pepe	No me gusta

Las primeras 5 reacciones de post corresponden a un post, las últimas 5 corresponden a otro.

Las primeras 5 reacciones de comentario corresponden a un comentario, las últimas 5 corresponden a otro.

Invitaciones

Invitación	Miembro Remitente	Miembro Receptor	Estado
i1	Gastón	Joaquín	Aceptada
i2	Gastón	Ana	Aceptada
i3	Gastón	Juan	Aceptada
i4	Gastón	Juanchi	Aceptada
i5	Gastón	Diego	Aceptada
i6	Gastón	Alejandra	Aceptada
i7	Gastón	Jenny	Aceptada
i8	Gastón	Julieta	Aceptada
i9	Gastón	Pepe	Aceptada
i10	Joaquín	Ana	Aceptada
i11	Joaquín	Juan	Aceptada
i12	Joaquín	Juanchi	Aceptada
i13	Joaquín	Diego	Aceptada
i14	Joaquín	Alejandra	Aceptada
i15	Joaquín	Jenny	Aceptada
i16	Joaquín	Julieta	Aceptada
i17	Joaquín	Pepe	Aceptada
i18	Joaquín	Gaspar	Aceptada
i19	Gastón	Gaspar	Aceptada
i20	Juan	Pepe	Aceptada
i21	Juanchi	Pepe	Aceptada
i22	Juan	Jenny	Aceptada
i23	Juanchi	Ana	Aceptada
i24	Juan	Diego	Aceptada
i25	Juanchi	Gaspar	Aceptada
i26	Juan	Gaspar	Aceptada
i27	Juanchi	Alejandra	Aceptada
i29	Juan	Juanchi	Aceptada
r1	Diego	Alejandra	Rechazada
r2	Alejandra	Jenny	Rechazada
r3	Jenny	Julieta	Rechazada
p1	Ana	Jenny	Pendiente
p2	Juan	Alejandra	Pendiente
p3	Juanchi	Diego	Pendiente

Código fuente

Sistema

```
using System;
using System.Collections.Generic;
using System.Dynamic;
using System.Linq;
using System.Net.Http.Headers;
using System.Security.Cryptography;
using System.Text;
using System.Threading.Tasks;


namespace ClasesObligatorioP2GVDS
{
    public class Sistema
    {
        #region Singleton
        private static Sistema _instancia = null;
        private Sistema()
        {
            string s = Precarga();
        }

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

        #region Listas
        //Declaracion de variables globales
        private List<Usuario> _listaUsuarios = new List<Usuario>();
        private List<Publicacion> _listaPublicaciones = new List<Publicacion>();
        private List<Invitacion> _listaInvitaciones = new List<Invitacion>();
        #endregion

        #region Getters
        //Metodo para acceder a las listas
        public List<Usuario> GetUsuarios() { return _listaUsuarios; }
        public List<Publicacion> GetPublicaciones() { return _listaPublicaciones; }
```

```
public List<Invitacion> GetInvitaciones() { return _listaInvitaciones; }

public List<Administrador> GetAdministradores()
{
    List<Administrador> administradores = new List<Administrador>();
    foreach (Usuario u in _listaUsuarios)
    {
        if (u is Administrador)
        {
            administradores.Add((Administrador)u);
        }
    }
    return administradores;
}
public List<Miembro> GetMiembros()
{
    List<Miembro> miembros = new List<Miembro>();
    foreach (Usuario u in _listaUsuarios)
    {
        if (u is Miembro)
        {
            miembros.Add((Miembro)u);
        }
    }
    return miembros;
}

public List<Post> GetPosts()
{
    List<Post> posts = new List<Post>();
    foreach (Publicacion p in _listaPublicaciones)
    {
        if (p is Post)
        {
            posts.Add((Post)p);
        }
    }
    return posts;
}

public List<Comentario> GetComentarios()
{
    List<Comentario> comentarios = new List<Comentario>();
    foreach (Publicacion p in _listaPublicaciones)
    {
        if (p is Comentario)
        {
            comentarios.Add((Comentario)p);
```

```
        }
    }
    return comentarios;
}

#endregion

#region Altas
//Metodos de alta, validan al objeto recibido por parametro.

//Recibe un usuario, lo valida y si no existe en el sistema lo agrega
public void AltaUsuario(Usuario u)
{
    try
    {
        u.EsValido();
        if (!_listaUsuarios.Contains(u))
        {
            _listaUsuarios.Add(u);
        }
        else
        {
            throw new Exception("Error, usuario ya registrado");
        }
    }
    catch (Exception e)
    {
        Usuario.ReducirId();
        throw e;
    }

}

//Valida la publicacion recibida como parametro y la agrega a la lista
public void AltaPublicacion(Publicacion p)
{
    try
    {
        p.EsValido();
        _listaPublicaciones.Add(p);
    }
    catch (Exception e)
    {
        Publicacion.ReducirId();
        throw e;
    }
```

```
    }

    //Valida la invitacion recibida como parametro y si aun no existe en el sistema entonces
la agrega.
    public void AltaInvitacion(Invitacion i)
    {
        try
        {
            i.EsValido();
            if (!_listaInvitaciones.Contains(i))
            {
                _listaInvitaciones.Add(i);
            }
            else
            {
                throw new Exception("Invitacion ya existente");
            }
        }
        catch (Exception e)
        {
            Invitacion.ReducirId();
            throw e;
        }


    }
    #endregion


    //Recibe dos fechas como parametro, las valida y retorna una lista con los post
realizados entre esas dos fechas
    //Si el contenido del post es mayor a 50 caracteres
    public List<Post> ObtenerPostEntre2Fechas(DateTime fec1, DateTime fec2)
    {
        if (fec1 < fec2)
        {
            List<Post> listaP = new List<Post>();
            foreach (Publicacion p in _listaPublicaciones)
            {
                if (p.FechaPublicacion >= fec1 && p.FechaPublicacion <= fec2)
                {
                    if (p is Post)
                    {
                        Post pAux1 = (Post)p;
                        Post pAux2 = new Post(pAux1.Autor, pAux1.Titulo,
RecortarTexto(pAux1.Contenido), pAux1.Imagen, pAux1.Privado, pAux1.Id);
                        listaP.Add(pAux2);
                    }
```

```
                }
            }

            if (listaP.Count > 0)
            {
                listaP.Sort();
                return listaP;
            }
            else
            {
                throw new Exception($"Error, ninguna publicación encontrada entre {fec1} y
{fec2}");
            }
        }
        else
        {
            throw new Exception("Error, fecha 1 es posterior a fecha 2");
        }


    }

    //Obtiene un miembro y busca las publicaciones de ese miembro en la lista de
publicaciones, las agrega a una lista auxiliar y la retorna
    public List<Publicacion> BuscarPublicacionesXUsuario(Miembro miembro)
    {

        List<Publicacion> listaDePublicacionesXUsuario = new List<Publicacion>();
        foreach (Publicacion p in _listaPublicaciones)
        {
            if (p.Autor.Equals(miembro))
            {
                listaDePublicacionesXUsuario.Add(p);
            }
        }
        if (listaDePublicacionesXUsuario.Count > 0)
        {
            return listaDePublicacionesXUsuario;
        }
        else
        {
            throw new Exception("Error, no se han encontrado publicaciones asociadas a este
usuario.");
        }



    }
```

```
    //Si el texto recibido es mayor a 50 caracteres entonces lo recorta hasta 50, de otra manera lo devuelve.
    public string RecortarTexto(string txt)
    {
        if (txt.Length > 50)
        {
            string nuevotxt = "";
            for (int i = 0; i < 50; i++)
            {
                nuevotxt += txt[i];
            }
            return nuevotxt;
        }
        else
        {
            return txt;
        }

    }

    //Obtiene un email y busca en el sistema un miembro con ese email asociado, si no lo encuentra lanza excepcion
    public Miembro GetMiembroXEmail(string email)
    {
        Miembro miembro = null;
        bool encontrado = false;
        for (int i = 0; i < _listaUsuarios.Count && !encontrado; i++)
        {
            Usuario u = _listaUsuarios[i];
            if (u.Email.ToLower().Equals(email.ToLower()))
            {
                miembro = (Miembro)u;
                encontrado = true;
            }
        }
        if (encontrado)
        {
            return miembro;
        }
        else
        {
            throw new Exception("Error, miembro no presente en el sistema.");
        }

    }

    //Lista los posts segun un miembro recibido como parametro.
```

```
        public List<Publicacion> ListarPostComentadosXMiembro(Miembro m)
        {
            List<Publicacion> retorno = new List<Publicacion>();
            //Recorremos la lista de posts.
            foreach (Post p in GetPosts())
            {
                //Recorremos los comentarios del post.
                foreach (Comentario c in p.GetListaComentarios())
                {
                    //Si el autor es el mismo que el recibido por parametro, guardamos el post en la
lista.
                    if (c.Autor.Equals(m) && !retorno.Contains(p))
                    {
                        retorno.Add(p);
                    }
                }
            }
            if (retorno.Count > 0)
            {
                return retorno;
            }
            else
            {
                throw new Exception("Error, no se han encontrado posts asociados a los
comentarios de este usuario.");
            }

        }

        //Devuelve una lista de miembros con la mayor cantidad de publicaciones
        public DTOMaximoCantidadPublicaciones MiembrosConMasPublicaciones()
        {
            int max = 0;
            List<Miembro> retorno = new List<Miembro>();
            foreach (Usuario u in _listaUsuarios)
            {
                if (u is Miembro)
                {
                    Miembro mAux = (Miembro)u;
                    if (GetCantidadPublicaciones(mAux) > max)
                    {
                        retorno.Clear();
                        retorno.Add(mAux);
                        max = GetCantidadPublicaciones(mAux);
                    }
                    else if (GetCantidadPublicaciones(mAux) == max)
                    {
                        retorno.Add(mAux);
```

```
                }
            }
        }
        if (retorno.Count > 0)
        {
            DTOMaximoCantidadPublicaciones DTO = new
DTOMaximoCantidadPublicaciones();
            DTO.ListaMiembros = retorno;
            DTO.Cantidad = max;
            return DTO;
        }
        else
        {
            throw new Exception("No existen publicaciones de miembros.");
        }
    }

    //Transforma una lista de posts en una lista de publicaciones
    public List<Publicacion> TransformarListaPostAPublicacion(List<Post> lista)
    {
        List<Publicacion> ret = new List<Publicacion>();
        foreach (Post p in lista)
        {
            ret.Add(p);
        }
        return ret;
    }

    //Obtiene la cantidad de publicaciones de un miembro buscandolas en el sistema
    public int GetCantidadPublicaciones(Miembro m)
    {
        int ret = 0;
        foreach (Publicacion p in _listaPublicaciones)
        {
            if (p.Autor.Equals(m))
            {
                ret++;
            }
        }
        return ret;
    }

    //Recibe dos usuarios por parametros y pregunta si son amigos (metodo no utilizado en
esta primera entrega.)
    public bool SonAmigosSistema(Miembro m1, Miembro m2)
    {
        return m1.GetListaAmigos().Contains(m2) && m2.GetListaAmigos().Contains(m1);
    }
```

```
        //Método que permite llamar y ejecutar la precarga desde el Program
        public string LlamarPrecarga()
        {
            return Precarga();
        }

        private string Precarga()
        {
            //Mensaje que se recibirá en el Program, si no hay errores será vacío. Si hay errores devolverá los mensajes de las Altas y Los Agregar.
            string mensajesDeError = "";

            //Usuarios y Administrador

            #region PrecargaUsuarios (Miembros y admin)
            Usuario ADMIN = new Administrador("admin@gmail.com", "Admin1234");
            Usuario m1 = new Miembro("Gastón", "Varela", new DateTime(2004, 3, 28), "gas@gmail.com", "Gas1234");
            Usuario m2 = new Miembro("Joaquín", "Rodriguez", new DateTime(1986, 7, 1), "joaq@gmail.com", "Joaq1234");
            Usuario m3 = new Miembro("Ana", "Fernandéz", new DateTime(1990, 7, 12), "ana@gmail.com", "Ana1234");
            Usuario m4 = new Miembro("Juan", "Perez", new DateTime(1999, 4, 23), "juan@gmail.com", "Juan1234");
            Usuario m5 = new Miembro("Juanchi", "Juanza", new DateTime(2003, 8, 12), "juanchi@gmail.com", "Juanchi1234");
            Usuario m6 = new Miembro("Diego", "Sabatel", new DateTime(1986, 12, 29), "diego@gmail.com", "Diego1234");
            Usuario m7 = new Miembro("Alejandra", "Almeida", new DateTime(1988, 3, 23), "alejandra@gmail.com", "Alejandra1234");
            Usuario m8 = new Miembro("Jenny", "González", new DateTime(1994, 5, 18), "jenny@gmail.com", "Jenny1234");
            Usuario m9 = new Miembro("Julieta", "Vadetto", new DateTime(2000, 11, 2), "julieta@gmail.com", "Julieta1234");
            Usuario m10 = new Miembro("Pepe", "Pérez", new DateTime(2003, 12, 4), "pepe@gmail.com", "Pepe1234");
            Usuario m11 = new Miembro("Gaspar", "Mondadol", new DateTime(2003, 12, 4), "gaspar@gmail.com", "Gaspar1234");
```

```
        Usuario m12 = new Miembro("Carlos", "Rodríguez", new DateTime(1990, 7, 15),
"carlos@gmail.com", "Carlos1234");
        Usuario m13 = new Miembro("Laura", "Martínez", new DateTime(1985, 9, 8),
"laura@gmail.com", "Laura1234");
        Usuario m14 = new Miembro("Fernando", "López", new DateTime(1998, 2, 10),
"fernando@gmail.com", "Fernando1234");
        Usuario m15 = new Miembro("Ana", "Sánchez", new DateTime(1992, 6, 25),
"ana2@gmail.com", "Ana1234");
        Usuario m16 = new Miembro("Martín", "Gómez", new DateTime(1987, 11, 14),
"martin@gmail.com", "Martin1234");
        Usuario m17 = new Miembro("Lucía", "Hernández", new DateTime(1995, 4, 3),
"lucia@gmail.com", "Lucia1234");
        Usuario m18 = new Miembro("Pablo", "Díaz", new DateTime(2001, 8, 20),
"pablo@gmail.com", "Pablo1234");
        Usuario m19 = new Miembro("Isabel", "Torres", new DateTime(1993, 10, 7),
"isabel@gmail.com", "Isabel1234");
        Usuario m20 = new Miembro("Ricardo", "Ramírez", new DateTime(2002, 1, 30),
"ricardo@gmail.com", "Ricardo1234");
        Usuario m21 = new Miembro("María", "Fernández", new DateTime(1989, 12, 12),
"maria@gmail.com", "Maria1234");
        Usuario m22 = new Miembro("Jorge", "Gutiérrez", new DateTime(1996, 3, 5),
"jorge@gmail.com", "Jorge1234");
        Usuario m23 = new Miembro("Natalia", "Luna", new DateTime(2000, 5, 18),
"natalia@gmail.com", "Natalia1234");
        Usuario m24 = new Miembro("Andrés", "Ortega", new DateTime(1986, 9, 22),
"andres@gmail.com", "Andres1234");
        Usuario m25 = new Miembro("Valentina", "Castro", new DateTime(1994, 4, 8),
"valentina@gmail.com", "Valentina1234");
        Usuario m26 = new Miembro("Héctor", "Mendoza", new DateTime(1999, 11, 3),
"hector@gmail.com", "Hector1234");
        Usuario m27 = new Miembro("Sofía", "Peralta", new DateTime(2004, 2, 15),
"sofia@gmail.com", "Sofia1234");
        Usuario m28 = new Miembro("Eduardo", "Guerrero", new DateTime(1984, 7, 9),
"eduardo@gmail.com", "Eduardo1234");
        Usuario m29 = new Miembro("Camila", "Rojas", new DateTime(1991, 12, 28),
"camila@gmail.com", "Camila1234");
        Usuario m30 = new Miembro("Alejandro", "Navarro", new DateTime(1997, 6, 7),
"alejandro@gmail.com", "Alejandro1234");
        List<Usuario> listaUsuariosPrecarga = new List<Usuario> { ADMIN, m1, m2, m3,
m4, m5, m6, m7, m8, m9, m10, m11, m12, m13, m14, m15, m16, m17, m18, m19, m20,
m21, m22, m23, m24, m25, m26, m27, m28, m29, m30 };

        foreach (Usuario m in listaUsuariosPrecarga)
        {
            try
            {
                AltaUsuario(m);
            }
```

```
    catch (Exception e)
    {
        mensajesDeError += "\nError en precarga miembro";
    }
}

#endregion

//Invitaciones

#region PrecargaInvitaciones
Invitacion i1 = new Invitacion((Miembro)m1, (Miembro)m2);
Invitacion i2 = new Invitacion((Miembro)m1, (Miembro)m3);
Invitacion i3 = new Invitacion((Miembro)m1, (Miembro)m4);
Invitacion i4 = new Invitacion((Miembro)m1, (Miembro)m5);
Invitacion i5 = new Invitacion((Miembro)m1, (Miembro)m6);
Invitacion i6 = new Invitacion((Miembro)m1, (Miembro)m7);
Invitacion i7 = new Invitacion((Miembro)m1, (Miembro)m8);
Invitacion i8 = new Invitacion((Miembro)m1, (Miembro)m9);
Invitacion i9 = new Invitacion((Miembro)m1, (Miembro)m10);
Invitacion i10 = new Invitacion((Miembro)m2, (Miembro)m3);
Invitacion i11 = new Invitacion((Miembro)m2, (Miembro)m4);
Invitacion i12 = new Invitacion((Miembro)m2, (Miembro)m5);
Invitacion i13 = new Invitacion((Miembro)m2, (Miembro)m6);
Invitacion i14 = new Invitacion((Miembro)m2, (Miembro)m7);
Invitacion i15 = new Invitacion((Miembro)m2, (Miembro)m8);
Invitacion i16 = new Invitacion((Miembro)m2, (Miembro)m9);
Invitacion i17 = new Invitacion((Miembro)m2, (Miembro)m10);
Invitacion i18 = new Invitacion((Miembro)m2, (Miembro)m11);
Invitacion i19 = new Invitacion((Miembro)m1, (Miembro)m11);
Invitacion i20 = new Invitacion((Miembro)m4, (Miembro)m10);
Invitacion i21 = new Invitacion((Miembro)m5, (Miembro)m10);
Invitacion i22 = new Invitacion((Miembro)m4, (Miembro)m8);
Invitacion i23 = new Invitacion((Miembro)m5, (Miembro)m3);
Invitacion i24 = new Invitacion((Miembro)m4, (Miembro)m6);
Invitacion i25 = new Invitacion((Miembro)m5, (Miembro)m11);

Invitacion i26 = new Invitacion((Miembro)m4, (Miembro)m11);
Invitacion i27 = new Invitacion((Miembro)m5, (Miembro)m7);
Invitacion i29 = new Invitacion((Miembro)m4, (Miembro)m5);


Invitacion r1 = new Invitacion((Miembro)m6, (Miembro)m7);
Invitacion r2 = new Invitacion((Miembro)m7, (Miembro)m8);
Invitacion r3 = new Invitacion((Miembro)m8, (Miembro)m9);

Invitacion p1 = new Invitacion((Miembro)m3, (Miembro)m8);
Invitacion p2 = new Invitacion((Miembro)m4, (Miembro)m7);
```

```
        Invitacion p3 = new Invitacion((Miembro)m5, (Miembro)m6);
        List<Invitacion> listaInvitacionAceptadaPrecarga = new List<Invitacion> { i1, i2, i3, i4,
i5, i6, i7, i8, i9, i10, i11, i12, i13, i14, i15, i16, i17, i18, i19, i20, i21, i22, i23, i24, i25, i26, i27,
i29 };
        List<Invitacion> listaInvitacionRechazadaPrecarga = new List<Invitacion> { r1, r2, r3
};
        List<Invitacion> listaInvitacionPendientePrecarga = new List<Invitacion> { p1, p2, p3
};
        //Aceptadas
        foreach (Invitacion i in listaInvitacionAceptadaPrecarga)
        {
           try
           {
              AltaInvitacion(i);
              i.AceptarInvitacion();
           }
           catch (Exception e)
           {

              mensajesDeError += $"\nPrecarga: {e.Message} Id invitacion: {i.Id}";
           }
        }
        //Rechazadas
        foreach (Invitacion i in listaInvitacionRechazadaPrecarga)
        {
           try
           {
              AltaInvitacion(i);
              i.RechazarInvitacion();
           }
           catch (Exception e)
           {

              mensajesDeError += $"\nPrecarga: {e.Message} Id invitacion: {i.Id}";
           }
        }
        //Pendientes
        foreach (Invitacion i in listaInvitacionPendientePrecarga)
        {
           try
           {
              AltaInvitacion(i);
           }
           catch (Exception e)
           {

              mensajesDeError += $"\nPrecarga: {e.Message} Id invitacion:{i.Id}";
           }
```

```
        }

        #endregion

        //Posts

        #region Posts
        Publicacion pos1 = new Post((Miembro)m1, "Agua salada", "Las aguas saladas del océano en Uruguay son conocidas por sus tonos azules y verdes, ofreciendo hermosas playas y oportunidades para actividades marinas. Tomo agua salada todas las mañanas y tengo los riñones como piña.", "oceano.png", false);
        Publicacion pos2 = new Post((Miembro)m2, "Karate", "El karate es un antiguo arte marcial japonés que se ha convertido en una disciplina globalmente reconocida y respetada.", "karate.png", false);
        Publicacion pos3 = new Post((Miembro)m3, "Caballos", "Los caballos en Uruguay son parte esencial de su cultura, utilizados en el trabajo rural y en emocionantes deportes como el polo. Su presencia en la Semana Criolla y en estancias de renombre demuestra su importancia en la vida uruguaya. ", "caballo.png", false);
        Publicacion pos4 = new Post((Miembro)m4, "Vacaciones", "Las vacaciones son un bálsamo para el alma, un merecido descanso en medio de la rutina diaria.", "playa.jpg", true);
        Publicacion pos5 = new Post((Miembro)m5, "Dinosaurios", "Los dinosaurios, esos gigantes antiguos, siguen fascinándonos. Su legado fósil nos conecta con un pasado misterioso y asombroso. ¿Cuál es tu dinosaurio favorito? 🦕🦖 #Dinosaurios ", "dinosaurio.jpg", true);
        List<Publicacion> listaPostsPrecarga = new List<Publicacion> { pos1, pos2, pos3, pos4, pos5 };
        foreach (Publicacion p in listaPostsPrecarga)
        {
            try
            {
                AltaPublicacion(p);
            }
            catch (Exception e)
            {

                mensajesDeError += $"\nPrecarga: {e.Message} Id publicacion: {p.Id}";
            }
        }
        #endregion

        //Comentarios y Reacciones

        #region Comentarios
        Publicacion com1 = new Comentario((Miembro)m1, "ME ENCANTA EL AGUAS SALADA", "AMO EL AGUA");
        Publicacion com2 = new Comentario((Miembro)m2, "DEJA DE SER TAN NABO", "Como te va a gustar el agua salada, salame");
```

```
        Publicacion com3 = new Comentario((Miembro)m3, "Joaco, porque lo tratas mal?",
"Te parece bien? Todo bien en casa?");

        Publicacion com4 = new Comentario((Miembro)m4, "gente violenta!!", "A ver si se
calman un poco, así no se puede!!");
        Publicacion com5 = new Comentario((Miembro)m5, "El mejor arte", "De todos los
que existen es el mejor!!, escucho comentarios");
        Publicacion com6 = new Comentario((Miembro)m6, "mi abuela empezó Karate!!!",
"Tiene 104 años y es media rayada");

        Publicacion com7 = new Comentario((Miembro)m7, "Como hago para darle de
comer a los caballos?", "Hace dos meses les estoy intentando dar milanesas y ni las
tocan");
        Publicacion com8 = new Comentario((Miembro)m8, "Para al de arriba", "Como le vas
a dar milanesas al caballo?");
        Publicacion com9 = new Comentario((Miembro)m9, "Que lindos!", "Amo la
mortadela");

        Publicacion com10 = new Comentario((Miembro)m10, "Estoy en Rocha!!", "Rocha es
el mejor lugar para vacacionar....");
        Publicacion com11 = new Comentario((Miembro)m1, "Para que sirven las
vacaciones??", "Es ese ansiado tiempo en el que dejamos atrás las preocupaciones
laborales y nos sumergimos en un mundo de posibilidades y aventuras. Ya sea explorando
destinos exóticos, relajándonos en la playa con el sonido del mar como banda sonora o
simplemente disfrutando de la tranquilidad en casa, las vacaciones nos ofrecen la
oportunidad de recargar energías, reconectar con seres queridos y crear recuerdos que
perdurarán en nuestro corazón.");
        Publicacion com12 = new Comentario((Miembro)m2, "Que lindas son las licencias",
"AL FIN VACACIONESSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSS!!!!!!!, ya no
podía mas con mi laburo!!");

        Publicacion com13 = new Comentario((Miembro)m3, "Los dinosaurios no existen",
"Nunca vi uno");
        Publicacion com14 = new Comentario((Miembro)m4, "Para el de arriba", "Tu cerebro
no existe");
        Publicacion com15 = new Comentario((Miembro)m5, "TAS LOCO", "TODOS LOS
POSTS SON ASI? DONDE ESTAN LOS MODS");
        List<Publicacion> listaComentariosPrecarga = new List<Publicacion> { com1, com2,
com3, com4, com5, com6, com7, com8, com9, com10, com11, com12, com13, com14,
com15 };



        foreach (Publicacion p in listaComentariosPrecarga)
        {
            try
            {
                AltaPublicacion(p);
```

```
            }

            catch (Exception e)
            {

                mensajesDeError += $"\nPrecarga: {e.Message} Id publicacion: {p.Id}";
            }
        }

        //Para cargar los comentarios en la lista de comentarios de cada post
        //Creamos auxiliares para que los post y los comentarios dejen de ser publicaciones
        //Y así poder usar el método Agregar comentario
        Post pAux1 = (Post)pos1;
        List<Publicacion> listaComentariospAux1 = new List<Publicacion> { com1, com2,
com3 };
        foreach (Publicacion p in listaComentariospAux1)
        {
            try
            {
                pAux1.AgregarComentario((Comentario)p);
            }
            catch (Exception e)
            {
                mensajesDeError += $"\nPrecarga: {e.Message} Id publicacion: {p.Id}";
            }
        }

        Post pAux2 = (Post)pos2;
        List<Publicacion> listaComentariospAux2 = new List<Publicacion> { com4, com5,
com6 };
        foreach (Publicacion p in listaComentariospAux2)
        {
            try
            {
                pAux2.AgregarComentario((Comentario)p);
            }
            catch (Exception e)
            {
                mensajesDeError += $"\nPrecarga: {e.Message} Id publicacion: {p.Id}";
            }
        }

        Post pAux3 = (Post)pos3;
        List<Publicacion> listaComentariospAux3 = new List<Publicacion> { com7, com8,
com9 };
        foreach (Publicacion p in listaComentariospAux3)
        {
            try
```

```
            {
                pAux3.AgregarComentario((Comentario)p);
            }
            catch (Exception e)
            {
                mensajesDeError += $"\nPrecarga: {e.Message} Id publicacion: {p.Id}";
            }
        }


        Post pAux4 = (Post)pos4;
        List<Publicacion> listaComentariospAux4 = new List<Publicacion> { com10, com11,
com12 };
        foreach (Publicacion p in listaComentariospAux4)
        {
            try
            {
                pAux4.AgregarComentario((Comentario)p);
            }
            catch (Exception e)
            {
                mensajesDeError += $"\nPrecarga: {e.Message} Id publicacion: {p.Id}";
            }
        }



        Post pAux5 = (Post)pos5;
        List<Publicacion> listaComentariospAux5 = new List<Publicacion> { com13, com14,
com15 };
        foreach (Publicacion p in listaComentariospAux5)
        {
            try
            {
                pAux5.AgregarComentario((Comentario)p);
            }
            catch (Exception e)
            {
                mensajesDeError += $"\nPrecarga: {e.Message} Id publicacion: {p.Id}";
            }
        }
        #endregion




        #region Reacciones
        //Reacciones post
```

```
        Reaccion rea1 = new Reaccion(true, (Miembro)m1);
        Reaccion rea2 = new Reaccion(true, (Miembro)m2);
        Reaccion rea3 = new Reaccion(true, (Miembro)m3);
        Reaccion rea4 = new Reaccion(false, (Miembro)m4);
        Reaccion rea5 = new Reaccion(false, (Miembro)m5);

        Reaccion rea6 = new Reaccion(false, (Miembro)m6);
        Reaccion rea7 = new Reaccion(false, (Miembro)m7);
        Reaccion rea8 = new Reaccion(false, (Miembro)m8);
        Reaccion rea9 = new Reaccion(true, (Miembro)m9);
        Reaccion rea10 = new Reaccion(true, (Miembro)m10);

        //Reacciones comentario
        Reaccion rea11 = new Reaccion(true, (Miembro)m1);
        Reaccion rea12 = new Reaccion(true, (Miembro)m2);
        Reaccion rea13 = new Reaccion(true, (Miembro)m3);
        Reaccion rea14 = new Reaccion(true, (Miembro)m4);
        Reaccion rea15 = new Reaccion(true, (Miembro)m5);

        Reaccion rea16 = new Reaccion(false, (Miembro)m6);
        Reaccion rea17 = new Reaccion(false, (Miembro)m7);
        Reaccion rea18 = new Reaccion(false, (Miembro)m8);
        Reaccion rea19 = new Reaccion(false, (Miembro)m9);
        Reaccion rea20 = new Reaccion(false, (Miembro)m10);
        List<Reaccion> reaccionesPost1 = new List<Reaccion> { rea1, rea2, rea3, rea4,
rea5 };
        List<Reaccion> reaccionesPost2 = new List<Reaccion> { rea6, rea7, rea8, rea9,
rea10 };

        foreach (Reaccion r in reaccionesPost1)
        {
            try
            {
                pos1.AltaReaccion(r);
            }
            catch (Exception e)
            {
                mensajesDeError += $"\nPrecarga: {e.Message} Id publicacion: {pos1.Id}";

            }
        }
        foreach (Reaccion r in reaccionesPost2)
        {
            try
            {
                pos2.AltaReaccion(r);
            }
            catch (Exception e)
```

```
            {
                mensajesDeError += $"\nPrecarga: {e.Message} Id publicacion: {pos2.Id}";

            }
        }

        List<Reaccion> reaccionesComentario1 = new List<Reaccion> { rea11, rea12,
rea13, rea14, rea15 };
        List<Reaccion> reaccionesComentario2 = new List<Reaccion> { rea16, rea17,
rea18, rea19, rea20 };
        foreach (Reaccion r in reaccionesComentario1)
        {
            try
            {
                com1.AltaReaccion(r);
            }
            catch (Exception e)
            {
                mensajesDeError += $"\nPrecarga: {e.Message} Id publicacion: {com1.Id}";
            }
        }
        foreach (Reaccion r in reaccionesComentario2)
        {
            try
            {
                com2.AltaReaccion(r);
            }
            catch (Exception e)
            {
                mensajesDeError += $"\nPrecarga: {e.Message} Id publicacion: {com2.Id}";
            }
        }


        #endregion

        return mensajesDeError;

    }
    //Método que comprueba que la precarga de las listas fue exitosa.
    public string ComprobarPrecarga()
    {
        if (_listaInvitaciones.Count < 34 || _listaPublicaciones.Count < 20 ||
_listaUsuarios.Count < 12)
        {
            return "HA HABIDO UN ERROR INESPERADO EN LA PRECARGA";
        }
        else
```

```
    {
        return "La precarga ha resultado exitosa.";
    }
}

//Busca en los usuarios alguno en el que correspondan email y contraseña
public Usuario Login(string email, string pass)
{
    foreach (Usuario u in _listaUsuarios)
    {
        if (u.Email.Equals(email) && u.Contrasenia.Equals(pass))
        {
            return u;
        }
    }
    return null;
}
//Busca un miembro por id
public Miembro BuscarMiembroXId(int? id)
{
    foreach (Miembro u in GetMiembros())
    {
        if (u.Id.Equals(id))
        {
            return u;
        }
    }
    return null;
}
//Bloquea un Miembro
public void BloquearMiembro(Miembro aBloquear)
{
    if (aBloquear.Bloqueado)
    {
        aBloquear.Bloqueado = false;
    }
    else
    {
        aBloquear.Bloqueado = true;
    }

}

public Publicacion BuscarPublicacionXId(int id)
{
    foreach (Publicacion p in _listaPublicaciones)
    {
        if (p.Id.Equals(id))
```

```
        {
            return p;
        }
    }
    return null;
}
//Busca un post por su id
public Post BuscarPostXId(int id)
{
    foreach (Post p in GetPosts())
    {
        if (p.Id.Equals(id))
        {
            return p;
        }
    }
    return null;
}
//Busca un comentario por su id
public Comentario BuscarComentarioXId(int id)
{
    foreach(Comentario c in GetComentarios())
    {
        if (c.Id.Equals(id))
        {
            return c;
        }
    }
    return null;
}
//Banea un post
public void BanearPost(Post aBanear)
{
    if (aBanear.Censurado)
    {
        aBanear.Censurado = false;
    }
    else
    {
        aBanear.Censurado = true;
    }

}
//Crea una invitacion mediante la id de los usuarios
public void CrearInvitacionXId(int? id1, int? id2)
{
    if (id1 != null && id2 != null)
    {
```

```
            Miembro u1 = BuscarMiembroXId(id1);
            Miembro u2 = BuscarMiembroXId(id2);

            if(u1!=null && u2 != null)
            {
                Invitacion i = new Invitacion(u1, u2);
                AltaInvitacion(i);
            }
            else
            {
                throw new Exception("ERROR UNO DE LOS MIEMBROS ES NULO");
            }

        }
        else
        {
            throw new Exception("ERROR UNO DE LOS MIEMBROS ES NULO");
        }

    }
    //Busca las invitaciones del miembro donde este es solicitado
    public List<Invitacion> GetInvitacionesXMiembro(Miembro m)
    {
        List<Invitacion> ret = new List<Invitacion>();
        foreach (Invitacion i in _listaInvitaciones)
        {
            if (i.Solicitado.Equals(m) && !ret.Contains(i))
            {
                ret.Add(i);
            }
        }
        return ret;
    }
    //Busca una invitacion donde los id recibidos por parametros esten presentes
    public Invitacion? BuscarInvitacionXIdMiembros(int? id1, int? id2)
    {
        foreach (Invitacion i in _listaInvitaciones)
        {

            if (i.Solicitado.Id == id1 && i.Solicitante.Id == id2 || i.Solicitado.Id == id2 &&
i.Solicitante.Id == id1)
            {
                return i;
            }


        }
        throw new Exception("Invitacion inexistente");
```

```
    }
    //Filtra los posts segun los requerimentos (censurado, amistad, autor, privacidad)
    public List<Post>? GetPostsFiltradosParaMiembros(int? id)
    {
        if (id != null)
        {
            int idaux = (int)id;

            List<Post> posts = new List<Post>();
            foreach (Publicacion p in _listaPublicaciones)
            {
                if (p is Post post)
                {
                    if (!post.Censurado) {
                        if (post.Privado && post.Autor.SonAmigosXId(id) || !post.Censurado &&
!post.Privado)
                        {
                            posts.Add((Post)p);
                        }
                    }

                }
            }
            return posts;
        }
        return null;
    }
    //Devuelve una lista de miembros que no son amigos del miembro con id recibida en
parametro
    public List<Miembro> GetMiembrosFiltradosXAmistad(int? id)
    {
        List<Miembro> ret = new List<Miembro>();
        foreach(Miembro m in GetMiembros())
        {
            if (!m.SonAmigosXId(id) && m.Id!=id)
            {
                ret.Add(m);
            }
        }
        return ret;
    }
    //Busca publicaciones que contengan el criterio y tengan un VA superior al recibido
    public List<Publicacion> BuscarPublicacionesXCriterioVA(string criterio, int va)
    {
        List<Publicacion> ret = new List<Publicacion>();
        if(criterio == null)
        {
            criterio = "";
```

```
            }
            foreach(Publicacion p in _listaPublicaciones)
            {
                if(p.Titulo.ToLower().Contains(criterio.ToLower()) && p.CalcularVA() > va ||
p.Contenido.ToLower().Contains(criterio.ToLower()) && p.CalcularVA() > va)
                {
                    ret.Add(p);
                }
            }
            return ret;
        }


    }
}
```

Usuario

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


namespace ClasesObligatorioP2GVDS
{
    public abstract class Usuario : IValidacion
    {
        private static int _ultimoId = 0;
        public int Id { get; set; }
        public string Email { get; set; }
        public string Contrasenia { get; set; }


        protected Usuario(string email, string contrasenia)
        {
            Id = _ultimoId++;
```

```
        Email = email;
        Contrasenia = contrasenia;

    }

    protected Usuario()
    {
        Id = _ultimoId++;
    }

    //Validaciones
    public virtual void EsValido()
    {

            if (this == null)
            {
                throw new Exception($"Usuario nulo");
            }
            if (String.IsNullOrEmpty(Email))
            {
                throw new Exception($"Email invalido");
            }
            if (!Email.Contains('@'))
            {
                throw new Exception($"Email invalido, debe contener @");
            }
            if (Email[0] == '@')
            {
                throw new Exception($"Email invalido, el @ no debe ir al inicio");
            }
            if (Email[Email.Length - 1] == '@')
            {
                throw new Exception($"Email invalido, el @ no debe ir al final");
            }
            if (String.IsNullOrEmpty(Contrasenia) || !ValidarContraseniaExtras()||
Contrasenia.Length<5)
            {
                throw new Exception($"Contraseña invalida, colocar al menos un número y una
mayúscula");
            }
            if(Email.Contains(' ')||Contrasenia.Contains(' '))
            {
            throw new Exception($"Email o contraseña invalidos, no deben presentar
espacios");
            }

    }
    //Es llamada en caso de error de validacion
```

```
        public static void ReducirId()
        {
            _ultimoId--;
        }

        //Método que valida la contraseña
        private bool ValidarContraseniaExtras()
        {

            bool mayus = false;
            bool num = false;
            foreach (char c in Contrasenia)
            {
                if (char.IsDigit(c))
                {
                    num = true;
                }
                if(char.IsUpper(c))
                {
                    mayus = true;
                }
            }
            return mayus && num;
        }


        public override string ToString()
        {
            return $"Usuario [ID: {Id}]\n[Email: {Email}]";
        }

        public override bool Equals(object? obj)
        {
            return obj is Usuario usuario &&
                   Email.ToLower() == usuario.Email.ToLower();
        }
    }
}
```

Administrador

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

```
using System.Threading.Tasks;

namespace ClasesObligatorioP2GVDS
{
    public class Administrador : Usuario
    {
        public Administrador(string email, string contrasenia) : base(email, contrasenia)
        {
            //Usuario se lleva a su constructor todos los parametros, pues Administrador no
difiere en atributos con su clase base.
        }
        public Administrador() : base()
        {

        }
        //Aplica la validacion de Usuario
        public override void EsValido()
        {
            base.EsValido();
        }
        //Aplica el ToString de Usuario
        public override string ToString()
        {
            return base.ToString();
        }
    }
}
```

Miembro

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

namespace ClasesObligatorioP2GVDS
{
    public class Miembro : Usuario, IComparable<Miembro>
    {
        public string Nombre { get; set; }
        public string Apellido { get; set; }
        public DateTime FechaNacimiento { get; set; }
        private List<Miembro> _listaAmigos = new List<Miembro>();
```

```
        public bool Bloqueado { get; set; }


        public Miembro(string nombre, string apellido, DateTime fechaNacimiento, string email,
string contrasenia) : base(email, contrasenia)
        {
            Nombre = nombre;
            Apellido = apellido;
            FechaNacimiento = fechaNacimiento;
            Bloqueado = false;
        }

        public Miembro() : base()
        {
            Bloqueado = false;
        }
        //Validaciones
        public override void EsValido()
        {
            base.EsValido();
                if (String.IsNullOrEmpty(Nombre))
                {
                    throw new Exception($"Nombre invalido");
                }
                if (String.IsNullOrEmpty(Apellido))
                {
                    throw new Exception($"Apellido invalido");
                }
                if(Nombre.Contains(' ')||Apellido.Contains(' '))
                {
                    throw new Exception($"Nombre o Apellido con espacio");
                }
                if (!(char.IsUpper(Nombre[0]) && char.IsUpper(Apellido[0])))
                {
                    throw new Exception($"Nombre o Apellido sin mayuscula");
                }
                if (FechaNacimiento < new DateTime(1900, 12, 31))
                {
                    throw new Exception($"Fecha no valida");
                }
        }
        //Devuelve la lista de amigos del miembro
        public List<Miembro> GetListaAmigos()
        {
            return _listaAmigos;
        }
        //Agrega un amigo a un miembro
        public void AgregarAmigo(Miembro m)
```

```
        {
            _listaAmigos.Add(m);
        }

        //Le pregunta a el miembro en que se invoca la funcion si el miembro recibido por parametro está incluido en su lista de amigos,
        //Tambien comprueba si el parametro recibido es uno mismo, se sobreentiende que un miembro es amigo de si mismo.
        public bool SonAmigos(Miembro m1)
        {
            return _listaAmigos.Contains(m1) || m1.Equals(this);
        }
        public bool SonAmigosXId(int? id)
        {
            //Busca a un miembro con la id recibida por parametro dentro de la lista de amigos
            foreach(Miembro m in _listaAmigos)
            {
                if(m.Id == id)
                {
                    //Si lo encuentra devuelve true
                    return true;
                }
            }
            if(this.Id == id)
            {
                //Si no lo encuentra se pregunta si es su misma id, si este es el caso entonces devuelve true
                return true;
            }
            return false;
        }

        //En la porcion final del ToString en la parte
        public override string ToString()
        {
            return base.ToString() + $"Miembro: [Nombre: {Nombre}]\n[Apellido: {Apellido}]\n[Fecha de nacimiento: {FechaNacimiento}]";
        }


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

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

Publicacion

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


namespace ClasesObligatorioP2GVDS
{
    public abstract class Publicacion : IValidacion
    {
        private static int _ultimoId { get; set; }
        public int Id { get; set; }
        public Miembro Autor { get; set; }
        public string Titulo { get; set; }
        public string Contenido { get; set; }
        public DateTime FechaPublicacion { get; set; }
        private List<Reaccion> _listaReacciones = new List<Reaccion>();

        protected Publicacion()
        {
            Id = _ultimoId++;
            FechaPublicacion = DateTime.Now;
        }
```

```
        protected Publicacion(Miembro autor, string titulo, string contenido)
        {
            Id = _ultimoId++;
            Autor = autor;
            Titulo = titulo;
            Contenido = contenido;
            FechaPublicacion = DateTime.Now;
        }

        //Constructor para crear copias:
        protected Publicacion(Miembro autor, string titulo, string contenido, int id)
        {
            Id = id;
            Autor = autor;
            Titulo = titulo;
            Contenido = contenido;
            FechaPublicacion = DateTime.Now;
        }
        //Devuelve la lista de reacciones de la publicacion
        public List<Reaccion> GetReacciones() { return _listaReacciones; }

        //Valida una reaccion recibida por parametro y la agrega a la lista de reacciones de la
publicacion
        public void AltaReaccion(Reaccion r)
        {
            try
            {
                r.EsValido();

                if (!ContieneReaccionDe(r.Miembro))
                {
                    _listaReacciones.Add(r);
                }
                else
                {
                    throw new Exception();
                }
            }
            catch (Exception)
            {

                throw new Exception("Usted ya reaccionó");
            }

        }
        //Busca si entre las reacciones existe alguna del miembro recibido en parametro
        private bool ContieneReaccionDe(Miembro miembro)
```

```
{
    foreach(Reaccion r in _listaReacciones)
    {
        if (r.Miembro == miembro)
        {
            return true;
        }
    }
    return false;
}

//Validaciones
public virtual void EsValido()
{

        if (this == null)
        {
            throw new Exception($"Publicación nula");
        }
        Autor.EsValido();
        if (String.IsNullOrEmpty(Titulo))
        {
            throw new Exception($"Titulo no valido");
        }
        if (String.IsNullOrEmpty(Contenido))
        {
            throw new Exception($"Titulo no valido");
        }
}


//Es llamada en caso de error de validacion
public static void ReducirId()
{
    _ultimoId--;
}
//Cuenta los likes de la publicacion
public int ContarLikes()
{
    int ret = 0;
    foreach (Reaccion r in _listaReacciones)
    {
        if (r.Like)
        {
            ret++;
        }
    }
    return ret;
```

```
        }
        //Cuenta los dislikes de la publicacion
        public int ContarDislikes()
        {
            int ret = 0;
            foreach (Reaccion r in _listaReacciones)
            {
                if (!r.Like)
                {
                    ret++;
                }
            }
            return ret;
        }

        //Metodo virtual para calcular el valor de aceptacion de la publicacion, es virtual pues Comentario lo utiliza como esta declarado abajo, Post necesita un paso extra.
        public virtual int CalcularVA()
        {
            return ContarLikes() * 5 + ContarDislikes() * -2;
        }

        public override string ToString()
        {
            return $"[ID: {Id}][Fecha: {FechaPublicacion}][Autor: {Autor.Nombre}]";
        }
    }
}
```

Post

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


namespace ClasesObligatorioP2GVDS
{
    public class Post : Publicacion, IComparable<Post>
    {
        private List<Comentario> _listaComentarios = new List<Comentario>();
        public string Imagen { get; set; }
        public bool Privado { get; set; }
        public bool Censurado { get; set; }
```

```
        public Post() : base()
        {

        }

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

        //Constructor para crear copias
        public Post(Miembro autor, string titulo, string contenido, string imagen, bool privado, int
id) : base(autor, titulo, contenido, id)
        {
            Imagen = imagen;
            Privado = privado;
            Censurado = false;
        }

        //Validaciones.
        public override void EsValido()
        {
            base.EsValido();
                if (string.IsNullOrEmpty(Imagen))
                {
                    throw new Exception($"Imagen no valida");
                }
                if (!(Imagen[Imagen.Length - 1] == 'g' && Imagen[Imagen.Length - 2] == 'n' &&
Imagen[Imagen.Length - 3] == 'p' || Imagen[Imagen.Length - 2] == 'p' &&
Imagen[Imagen.Length - 3] == 'j' && Imagen[Imagen.Length - 4] == '.'))
                {
                    throw new Exception($"Formato imagen no valida");
                }
        }

        //Valida un comentario recibido por parametro y lo agrega a la lista de comentarios del
post.
        //Prevé si el comentario es privado o no. Si no lo es, chequea que los miembros sean
amigos.
        public void AgregarComentario(Comentario c)
        {
            if (!Privado)
            {
                c.EsValido();
```

```
            _listaComentarios.Add(c);
        }
        else
        {
            if (Autor.SonAmigos(c.Autor))
            {
                c.EsValido();
                _listaComentarios.Add(c);
            }
            else
            {
                throw new Exception("Error, usted no es amigo del dueño del post");
            }
        }

    }

    //Método que devuelve la lista de comentarios del post
    public List<Comentario> GetListaComentarios() { return _listaComentarios; }

    public override string ToString()
    {
        return base.ToString() + $"\nPOST:\n[Titulo: {Titulo}]\n[Imagen: {Imagen}][Contenido:
{Contenido}]\n[Valor de aceptacion: {CalcularVA()}]\n[Privado:
{BoolATexto(Privado)}][Censurado: {BoolATexto(Censurado)}]\n\n";
    }

    //Método para no mostrar un bool (True o False) en ingles en el ToString.
    public string BoolATexto(bool x)
    {
        if (x)
        {
            return "SI";
        }
        else
        {
            return "NO";
        }
    }
    //Agrega la parte adicional a CalcularVA no presente en Publicacion y Comentario
    public override int CalcularVA()
    {
        int ret = base.CalcularVA();
        if (!Privado)
        {
            ret = ret + 10;
        }
        return ret;
```

```
        }



        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;
            }
        }
    }
}
```

Comentario

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

namespace ClasesObligatorioP2GVDS
{
    public class Comentario : Publicacion
    {
        public Comentario() : base()
        {

        }

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

```
        //Aplica la validacion de Publicacion
        public override void EsValido()
        {
            base.EsValido();
        }

        public override string ToString()
        {
            return base.ToString() + $"\nCOMENTARIO:\n[Contenido: {Contenido}]\n[Fecha de
publicacion: {FechaPublicacion}]\n[Valor de aceptacion: {CalcularVA()}]\n\n";
        }

        //Devuelve el metodo CalcularVA base, heredado de Publicacion
        public override int CalcularVA()
        {
            return base.CalcularVA();
        }

    }
}
```

DTOMaximoCantidadPublicaciones

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

namespace ClasesObligatorioP2GVDS
{
    public class DTOMaximoCantidadPublicaciones
    {
        public List<Miembro>ListaMiembros = new List<Miembro>();
        public int Cantidad { get; set; }
    }
}
```

Estado

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

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

namespace ClasesObligatorioP2GVDS
{
    public enum Estado
    {
        PENDIENTE_APROBACION = 1,
        APROBADA = 2,
        RECHAZADA = 3
    }
}
```

Invitacion

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


namespace ClasesObligatorioP2GVDS
{
    public class Invitacion : IValidacion
    {
        public static int _ultimoId { get; set; }
        public int Id { get; set; }
        public Miembro Solicitante { get; set; }
        public Miembro Solicitado { get; set; }
        public Estado Estado { get; set; }
        public DateTime FechaEnvio { get; set; }

        public Invitacion()
        {
            Id = _ultimoId++;
            FechaEnvio = DateTime.Now;
            Estado = Estado.PENDIENTE_APROBACION;
        }

        public Invitacion(Miembro solicitante, Miembro solicitado)
        {
            Id = _ultimoId++;
            Solicitante = solicitante;
```

```
            Solicitado = solicitado;
            Estado = Estado.PENDIENTE_APROBACION;
            FechaEnvio = DateTime.Now;
        }

        //Validaciones
        public void EsValido()
        {
            if (this == null)
            {
                throw new Exception($"Invitación ID {Id}: Invitación nula");
            }
            try
            {
                Solicitante.EsValido();
                Solicitado.EsValido();
            }
            catch (Exception e)
            {

                throw e;
            }

            if (Solicitante == null || Solicitado == null)
            {
                throw new Exception($"Solicitante o solicitado no validos");
            }
            if (Estado != Estado.PENDIENTE_APROBACION && Estado != Estado.APROBADA
&& Estado != Estado.RECHAZADA)
            {
                throw new Exception($"Estado invalido");
            }
            if (Solicitante.Equals(Solicitado))
            {
                throw new Exception($"Solicitante no puede ser igual a solicitado");
            }
        }
        //Es llamada en caso de error de validacion
        public static void ReducirId()
        {
            _ultimoId--;
        }

        public override string ToString()
        {
            return $"Invitacion: [Id: {Id}][Solicitante: {Solicitante.ToString()}][Estado:
{Estado}][Fecha de envio: {FechaEnvio}]";
        }
```

```
        //Método que chequea que los miembros no se tengan en sus listas de amigos y que la
invitacion este pendiente de aprobación.
        //De ser así se agregan a sus listas de amigos y se establece como aprobada.
        public void AceptarInvitacion()
        {
            EsValido();
            if
(!Solicitado.GetListaAmigos().Contains(Solicitante)&&!Solicitante.GetListaAmigos().Contains
(Solicitado)&&Estado==Estado.PENDIENTE_APROBACION) {
                Estado = Estado.APROBADA;
                Solicitado.AgregarAmigo(Solicitante);
                Solicitante.AgregarAmigo(Solicitado);
            }
            else
            {
                throw new Exception("Hubo un error inesperado");
            }

        }

        //Método para rechazar una invitación, cambia el estado a rechazada.
        public void RechazarInvitacion()
        {
            EsValido();
            Estado = Estado.RECHAZADA;

        }

        public override bool Equals(object? obj)
        {
            return obj is Invitacion invitacion &&
                   Solicitante.Equals(invitacion.Solicitante) &&
                   Solicitado.Equals(invitacion.Solicitado) || obj is Invitacion invitacionb &&
Solicitante.Equals(invitacionb.Solicitado) &&
                   Solicitado.Equals(invitacionb.Solicitante);
        }
    }
}
```

Reaccion

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

```
using System.Text;
using System.Threading.Tasks;

namespace ClasesObligatorioP2GVDS
{
    public class Reaccion : IValidacion
    {
        public bool Like { get; set; }
        public Miembro Miembro { get; set; }

        public Reaccion()
        {

        }

        public Reaccion(bool like, Miembro miembro)
        {
            Like = like;
            Miembro = miembro;
        }

        //Validaciones
        public void EsValido()
        {
            if (Miembro == null)
            {
                throw new Exception("Error miebro nulo");
            }
        }

        public override bool Equals(object? obj)
        {
            return obj is Reaccion reaccion &&
                   reaccion.Miembro.Equals(Miembro);
        }
    }}
```

Testing

Testing (T)	Escenario de prueba	Pasos	Datos utilizados	Resultados esperados	Resultados obtenidos	Estado F = falla P = pasa
T01	Registro como miembro	1.Click en el boton “Registrarse” 2.Rellenar todos los	1.Email 2.Contrase ña 3.Nombre 4.Apellido	El usuario se registra y hace login automáticamente	El usuario se registra y hace login automaticamente	P

		campos 3.Click en el botón azúl “Registrarse”	5.Fecha de nacimiento			
T02	Registro como miembro dejando cualquiera de los campos vacíos	1.Click en el boton “Registrarse” 2.Rellenar algunos de los campos del form 3.Click en el boton azúl “Registrarse”	1.Email 2.Contrase ña 3.Nombre 4.Apellido 5.Fecha de nacimiento Dejando al menos uno vacío	El usuario no puede registrarse	El usuario no se registra	P
T03	Registro como miembro sin cumplir los requerimientos de los campos (email con arriba al principio o al final, contraseña sin mayúscula y/o numero y/o longitud inferior a 5, nombre y apellido sin mayúscula)	1.Click en el boton “Registrarse” 2.Rellenar alguno de los campos sin cumplir los requisitos 3.Click en el boton azúl “Registrarse”	1.Email 2.Contrase ña 3.Nombre 4.Apellido 5.Fecha de nacimiento No cumpliendo los requisitos en al menos uno	El usuario no puede registrarse	El usuario no se registra	P
T04	Login como admin	1.Click en el botón login 2.Rellenar los campos 3.Click en el botón login	1.Email admin@gmail.com 2.Contrase ña Admin1234	El usuario admin se loguea	El usuario admin se logueo	P
T05	Login como miembro	1.Click en el botón login 2.Rellenar los campos 3.Click en el botón login	1.Email gas@gmail.com 2.Contrase ña Gas1234	El usuario gas se loguea como usuario	El usuario gas se logueo	P
T06	Login con credenciales invalidas	1.Click en el botón login 2.Rellenar los campos 3.Click en el botón login	1.Email sin registrar en el sistema 2.Contrase ña que no este asociada a ninguna cuenta	El usuario no se loguea	El usuario no se logueo	P

T07	Listar usuarios de tipo miembro ordenados por apellido y nombre ascendentemente logueado como admin	1.Loguearse como admin 2.Click en el botón “Ver miembros”	1.Email de admin 2.Contrase ña de admin	Se muestran los miembros ordenados por apellido y nombre ascendentement e	Se muestran los miembros ordenados por apellido y nombre ascendentemente	P
T08	Bloquear a un usuario como admin	1.Loguearse como admin 2.Click en el botón “Ver miembros” 3.Click en el botón “Bloquear / Desbloquear”	1.Email de admin 2.Contrase ña de admin	Se bloquea el miembro seleccionado	El miembro seleccionado fue bloqueado	P
T09	Login como miembro bloqueado	1.Loguearse como el miembro que fue bloqueado	1.Email del miembro bloqueado 2.Contrase ña del miembro bloqueado	Se puede loguear	Se pudo logueado	P
T10	Postear como miembro bloqueado	1.Loguearse como el miembro que fue bloqueado 2.Click en el botón “Postear”	1.Email del miembro bloqueado 2.Contrase ña del miembro bloqueado	No puede postear, el boton lo lleva al index home con un mensaje de error	No postea, va al index home con un mensaje de error	P
T11	Comentar como miembro bloqueado	1.Loguearse como el miembro que fue bloqueado 2.Click en el botón “Ver posts” 3.Rellenar el form para comentar 4.Click en el boton “Comentar”	1.Email del miembro bloqueado 2.Contrase ña del miembro bloqueado	No puede comentar, el boton lo lleva al index home con un mensaje de error	No comenta, va al index home con un mensaje de error	P
T12	Banear un post como admin	1.Loguearse como admin 2.Click en “Ver posts”	1.Email de admin 2.Contrase ña de	El post se censura	El post es censurado	P

		3.Click en “Banear / Desbanear”	admin			
T13	Ver post censurado como miembro	1.Loguearse como miembro 2.Click en “Ver posts”	1.Email de miembro 2.Contrase ña de miembro	El post es invisible	El post no se ve	P
T14	Ver post censurado como admin	1.Loguearse como admin 2.Click en “Ver posts”	1.Email de admin 2.Contrase ña de admin	El post es visible	El post se ve	P
T15	Logout	1.Loguearse 2.Click en “Cerrar Sesion”	1.Email 2.Contrase ña	Se desloguea	Se desloguea	P
T16	Ver posts como miembro	1.Loguearse como miembro 2.Click en “Ver posts”	1.Email 2.Contrase ña	Se ven los posts habilitados	Se ven los posts habilitados	P
T17	Ver post censurado como miembro	1.Loguearse como miembro 2.Click en “Ver posts”	1.Email 2.Contrase ña	No se ven los posts censurados	No se ven los posts censurados	P
T18	Ver post privado sin ser amigo del autor	1.Loguearse como miembro 2.Click en “Ver posts”	1.Email 2.Contrase ña	No se ven los posts privados	No se ven los posts privados	P
T19	Ver post privado siendo amigo del autor	1.Loguearse como miembro 2.Click en “Ver posts”	1.Email 2.Contrase ña	Se ve el post del amigo	Se ve el post del amigo	P

Testing (T)	Escenario de prueba	Pasos	Datos utilizados	Resultados esperados	Resultados obtenidos	Estado F = falla P = pasa
T20	Enviar invitación como miembro	1.Loguearse como miembro 2.Click en el botón “Enviar	1.Email de miembro 2.Contras eña de	La invitación se envía	La invitación se envía	P

		invitación” 3.Elegir un miembro y hacer click en “enviar invitacion”	miembro			
T21	Enviar invitación como miembro a alguien a quien ya tenemos relacion / invitación	1.Loguearse como miembro 2.Click en el botón “Enviar invitación” 3.Elegir un miembro con quien ya tenemos relacion / invitacion y hacer click en “enviar invitacion”	1.Email de miembro 2.Contraseña de miembro	La invitación no se envia	La invitación no se envia	P
T22	Ver solicitudes de amistad como miembro	1.Loguearse como miembro 2.Click en el boton “Invitaciones recibidas”	1.Email de miembro 2.Contraseña de miembro	Se ven las invitaciones	Se ven las invitaciones	P
T23	Aceptar solicitud de amistad	1.Loguearse como miembro 2.Click en el boton “Invitaciones recibidas” 2.Click en aceptar	1.Email de miembro 2.Contraseña de miembro	La invitación de aprueba	La invitación se aprueba	P
T24	Rechazar solicitud de amistad	1.Loguearse como miembro 2.Click en el boton “Invitaciones recibidas” 2.Click en rechazar	1.Email de miembro 2.Contraseña de miembro	La invitación de rechaza	La invitación de rechaza	
T25	Realizar post como miembro	1.Loguearse como miembro	1.Email de miembro	El post se publica en el muro	El post fue publicado	P

		2.Click en el boton “Postear” 3.Rellenar los campos y subir una imagen 4.Click en “Postear”	2.Contraseña de miembro			
T26	Comentar un post	1.Loguearse como miembro 2.Click en el boton “Ver posts” 3.Rellenar el campo de comentario 4.Click en “Comentar”	1.Email de miembro 2.Contraseña de miembro	El comentario se publica en el post	El comentario fue publicado en el post	P
T27	Reaccionar con un like o dislike a un post o comentario	1.Loguearse como miembro 2.Click en el boton “Ver posts” 3.Click en “Like” o “Dislike” debajo de un post o comentario	1.Email de miembro 2.Contraseña de miembro	La reacción se agrega y aplica al cálculo del VA	La reacción fue agregada y aplicada al cálculo del VA	P
T28	Utilizar buscador	1.Loguearse como miembro 2.Click en el boton “Buscador de publicaciones” 3.Ingresar criterios de busqueda 4.Click en “buscar”	1.Email de miembro 2.Contraseña de miembro	Se muestra una lista filtrada según los criterios ingresados	Se muestra la lista filtrada según los criterios ingresados	P