UNIVERSIDAD ORT URUGUAY

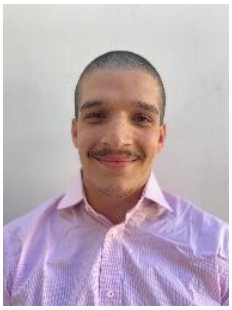
ANALISTA EN TECNOLOGÍAS DE LA INFORMACIÓN

OBLIGATORIO 1

Asignatura: Programación 2

Profesor/a: Nicolas Elizabelar

FELIPE CHAVARRIA - 260512



MONTEVIDEO

2023

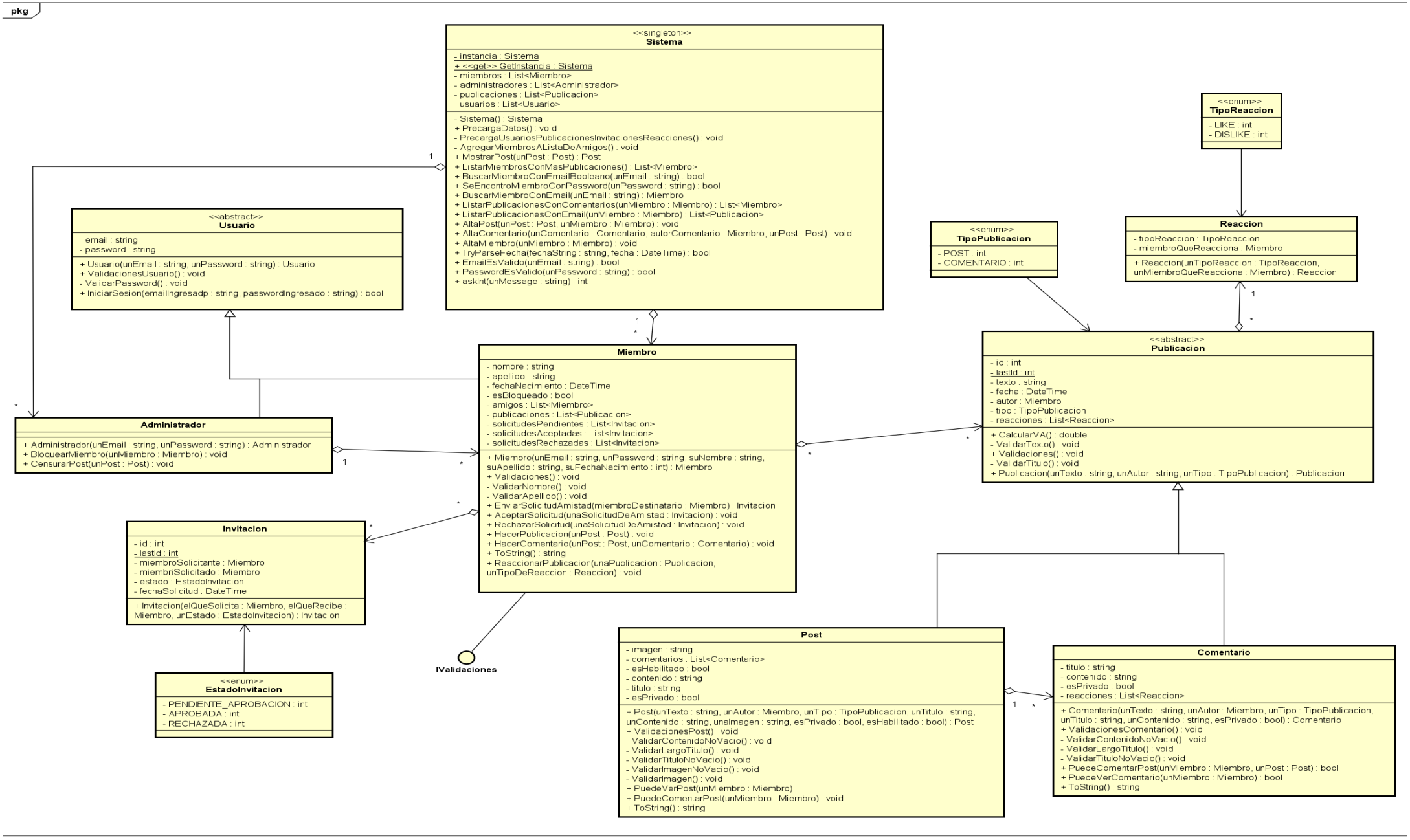
INDICE

1. INTRODUCCIÓN

El presente trabajo corresponde al obligatorio de la materia Programación 2, vinculada al 2do semestre de la carrera Analista en Tecnología de la Información de la Universidad ORT Uruguay.

Dada una realidad, planteada en la letra del obligatorio (Anexo 1, p.x) es que se realiza el presente documento.

2. DIAGRAMA DE CLASES



3. TESTING DE ALTA MIEMBRO

N°	Descripción	Resultado	Observaciones
1	Se le pide al usuario que ingrese todos los datos necesarios para dar de alta un miembro.	Aprobado	
2	Se verifica que todos los strings (e-mail, password, nombre y apellido) no estén vacíos.	Aprobado	Da la posibilidad a que el usuario ingrese un carácter especial o numérico. Simplemente con que no sea nulo es correcto. (*1*)
3	El email posee un “@”.	Aprobado	Simplemente se verifica que el string posea en alguna posición tal carácter, a excepción de la posición cero y la última del largo del string. (*2*)
4	Se verifica correctamente la fecha de nacimiento, y se guarda como dato “DateTime”.	Aprobado	Se parsea el string a formato DateTime correctamente.
5	Se verifica que ya exista un miembro con el mismo e-mail.	Aprobado	
6	Salir del alta de miembro	Fallido	Sólo se puede salir cuando se termina el proceso de alta, ya sea con todos los datos correctamente ingresados o con algún error en alguno de estos. (*3*)

1, *2*, *3*- La letra del trabajo no exigía verificaciones más exhaustivas.

4. TABLA DE DATOS PRECARGADOS

En el presente apartado se mostrarán tablas mostrando los datos precargados en el sistema. Cada una de estas tablas pertenece a una clase distinta.

4.1 Miembro

EMAIL	PASSWORD	NOMBRE	FECHA NACIMIENTO	APELLIDO	PUBLICACIONES	PUBLICACIONES ACEPTADAS	PUBLICACIONES PENDIENTES	PUBLICACIONES RECHAZADAS	AMIGOS	BLOQUEADO
miembro1@hotmail.com	miembro1	miembro1	01/01/2000	ApellidoMiembro1	6	9 (es amigo de todos)	0	0	9 (es amigo de todos)	false
miembro2@hotmail.com	miembro2	miembro2	02/02/2001	ApellidoMiembro2	3	9 (es amigo de todos)	0	0	9 (es amigo de todos)	false
miembro3@hotmail.com	miembro3	miembro3	03/03/2002	ApellidoMiembro3	3	2	1	0	2	false
miembro4@hotmail.com	miembro4	miembro4	04/04/2003	ApellidoMiembro4	4	2	1	0	2	false
miembro5@hotmail.com	miembro5	miembro5	05/05/2004	ApellidoMiembro5	1	2	1	0	2	false
miembro6@hotmail.com	miembro6	miembro6	06/06/2005	ApellidoMiembro6	0	2	1	0	2	false
miembro7@hotmail.com	miembro7	miembro7	07/07/2006	ApellidoMiembro7	0	2	0	1	2	false
miembro8@hotmail.com	miembro8	miembro8	08/08/2007	ApellidoMiembro8	0	2	0	1	2	false
miembro9@hotmail.com	miembro9	miembro9	09/09/2008	ApellidoMiembro9	0	2	0	1	2	false
miembro10@hotmail.com	miembro10	miembro10	10/10/2009	ApellidoMiembro10	0	2	0	1	2	false

Tabla de datos precargados de miembros.

4.2 Administrador

EMAIL	PASSWORD
administrador1@hotmail.com	administrador1

Tabla de datos precargados de administradores.

4.3 Post

TEXTO	AUTOR	TIPO	TITULO	CONTENIDO	IMAGEN	PRIVADO	HABILITADO
Texto 1	miembro1	POST	Título Post 1	Este es el contenido del post 1 público	imagen1.png	false	true
Texto 2	miembro1	POST	Título Post 2	Este es el contenido del post 2 público	imagen2.png	false	true
Texto 3	miembro3	POST	Título Post 3	Este es el contenido del post 3 público	imagen3.png	false	true
Texto 4	miembro4	POST	Título Post 4	Este es el contenido del post 4 público	imagen4.png	false	true
Texto 5	miembro4	POST	Título Post 5	Este es el contenido del post 5 privado y no habilitado	imagen5.png	true	false

4.4 Comentario

TEXTO	AUTOR	TIPO	TITULO	CONTENIDO	PRIVADO
Texto 6	miembro1	COMENTARIO	Título comentario 1	Este es el contenido del comentario 1	false
Texto 7	miembro1	COMENTARIO	Título comentario 2	Este es el contenido del comentario 2	false
Texto 8	miembro1	COMENTARIO	Título comentario 3	Este es el contenido del comentario 3	false
Texto 9	miembro9	COMENTARIO	Título comentario 4	Este es el contenido del comentario 4	false
Texto 10	miembro10	COMENTARIO	Título comentario 5	Este es el contenido del comentario 5	false
Texto 11	miembro10	COMENTARIO	Título comentario 6	Este es el contenido del comentario 6	false
Texto 12	miembro2	COMENTARIO	Título comentario 7	Este es el contenido del comentario 7	false
Texto 13	miembro2	COMENTARIO	Título comentario 8	Este es el contenido del comentario 8	false
Texto 14	miembro3	COMENTARIO	Título comentario 9	Este es el contenido del comentario 9	false
Texto 15	miembro4	COMENTARIO	Título comentario 10	Este es el contenido del comentario 10	false
Texto 16	miembro4	COMENTARIO	Título comentario 11	Este es el contenido del comentario 11	false
Texto 17	miembro4	COMENTARIO	Título comentario 12	Este es el contenido del comentario 12	false
Texto 18	miembro1	COMENTARIO	Título comentario 13	Este es el contenido del comentario 13	false
Texto 19	miembro2	COMENTARIO	Título comentario 14	Este es el contenido del comentario 14	false
Texto 20	miembro3	COMENTARIO	Título comentario 15	Este es el contenido del comentario 15	false

Tabla de datos precargados de comentarios.

4.5 Invitaciones

MIEMBROSOLICITANTE	MIEMBROSOLICITADO	ESTADO
miembro2	miembro1	APROBADA
miembro3	miembro1	APROBADA
miembro4	miembro1	APROBADA
miembro5	miembro1	APROBADA
miembro6	miembro1	APROBADA
miembro7	miembro1	APROBADA
miembro8	miembro1	APROBADA
miembro9	miembro1	APROBADA
miembro10	miembro1	APROBADA
miembro1	miembro2	APROBADA
miembro3	miembro2	APROBADA
miembro4	miembro2	APROBADA
miembro5	miembro2	APROBADA
miembro6	miembro2	APROBADA
miembro7	miembro2	APROBADA
miembro8	miembro2	APROBADA
miembro9	miembro2	APROBADA
miembro10	miembro2	APROBADA

Tabla de datos precargados de invitaciones aceptadas.

MIEMBROSOLICITANTE	MIEMBROSOLICITADO	ESTADO
miembro3	miembro4	PENDIENTE_APROBACION
miembro5	miembro6	PENDIENTE_APROBACION

Tabla de datos precargados de invitaciones pendientes de aprobación.

MIEMBROSOLICITANTE	MIEMBROSOLICITADO	ESTADO
miembro7	miembro8	RECHAZADA
miembro9	miembro10	RECHAZADA

Tabla de datos precargados de invitaciones rechazadas.

4.6 Reacciones

TIPO	QUIEN
LIKE	miembro1
LIKE	miembro5

TIPO	QUIEN
DISLIKE	miembro1
DISLIKE	miembro10

Tabla de datos precargados de likes y dislikes.

5. CÓDIGO FUENTE

5.1 Interface IValidaciones

```
namespace Dominio.Interfaces
{
    public interface IValidaciones
    {
        public void Validaciones();
    }
     }
```

5.2 Clase Administrador

```
public class Administrador : Usuario
{
    //Constructor

    public Administrador(string unEmail, string unPassword) : base(unEmail,
unPassword)
    {
    }

    //M É T O D O S

    public void BloquearMiembro(Miembro unMiembro)
    {
        if (!unMiembro.esBloqueado)
        {
            throw new Exception($"El miembro ya está bloqueado.");
        }
        else
        {
        unMiembro.esBloqueado = true;
        }
    }

    public void CensurarPost(Post unPost)
    {
        if (!unPost.esHabilitado)
        {
            throw new Exception($"La publicación ya está censurada.");
        }
        else
        {
            unPost.esHabilitado = false;
        }
         }
```

5.3 Clase Comentario

```
public class Comentario : Publicacion, IValidaciones
{
    //A T R I B U T O S
    public string Titulo { get; set; }

    public string Contenido { get; set; }

    public bool esPrivado { get; set; } = false;

    public List<Reaccion> reacciones { get; set; }

    //Constructor
    public Comentario(string unTexto, Miembro unAutor, TipoPublicacion unTipo,
string unTitulo, string unContenido, bool esPrivado) : base (unTexto, unAutor,
unTipo)
    {
        this.Titulo = unTitulo;
        this.Contenido = unContenido;
        this.esPrivado = esPrivado;
        this.reacciones = new List<Reaccion>();
    }

    //M É T O D O S

    public void Validaciones()
    {
        ValidarTituloNoVacio();
        ValidarLargoTitulo();
        ValidarContenidoNoVacio();
    }

    private void ValidarContenidoNoVacio()
    {
        if (this.Contenido == null || this.Contenido.Trim().Length == 0)
        {
            throw new Exception($"El campo del contenido no debe estar vacío.");
        }
    }

    private void ValidarLargoTitulo()
    {
        if (this.Titulo.Trim().Length <= 3)
        {
            throw new Exception($"El campo del título debe ser más largo.");
        }
    }

    private void ValidarTituloNoVacio()
    {
        if (this.Titulo == null || this.Titulo.Trim().Length == 0)
        {
            throw new Exception($"El campo del título no debe estar vacío.");
        }
    }

    public bool PuedeComentarPost(Miembro unMiembro, Post unPost)
    {
        if (unPost.esPrivado)
        {
            if (this.Autor.amigos.Contains(unMiembro))
```

```
            {
                this.esPrivado = true;
            }
            return this.Autor.amigos.Contains(unMiembro); //Si el post es privado
verifica que el miembro esté en la lista de amigos
        }
        return true; //Si es público cualquiera lo puede comentar
    }

    public bool PuedeVerComentario(Miembro unMiembro)
    {
        if (this.esPrivado)
        {
            return this.Autor.amigos.Contains(unMiembro); //Si el comentario es
privado verifica que el miembro esté en la lista de amigos
        }
        return true; //Si es público todos lo pueden ver
    }

    public override string ToString()
    {
        return $"{this.Id}\t{this.Tipo}\t'{this.Titulo}'";
    }
     }
```

5.4 Clase EstadoInvitacion

```
public enum EstadoInvitacion
{
    PENDIENTE_APROBACION = 1,
    APROBADA = 2,
    RECHAZADA = 3
     }
```

5.5 Clase Invitacion

```
public class Invitacion
 {
     //A T R I B U T O S
     public int Id {  get; set; }

     private static int lastId { get; set; } = 0;

     public Miembro miembroSolicitante { get; set; }

     public Miembro miembroSolicitado { get; set; }

     public EstadoInvitacion Estado { get; set; }

     public DateTime fechaSolicitud { get; set; }

     //Constructor
     public Invitacion(Miembro elQueSolicita, Miembro elQueRecibe,
EstadoInvitacion unEstado)
     {
         this.Id = lastId++;
         this.miembroSolicitante = elQueSolicita;
         this.miembroSolicitado = elQueRecibe;
```

```
            this.Estado = unEstado;
            fechaSolicitud = DateTime.Now;
        }
         }
```

5.6 Clase Miembro

```
public class Miembro : Usuario, IValidaciones
 {
     //A T R I B U T O S
     public string Nombre { get; set; }

     public string Apellido { get; set; }

     public DateTime fechaNacimiento { get; set; }

     public bool esBloqueado = false;

     public List<Miembro> amigos { get; set; }

     public List<Publicacion> publicaciones { get; set; }

     public List<Invitacion> solicitudesPendientes { get; set; }

     public List<Invitacion> solicitudesAprobadas { get; set; }

     public List<Invitacion> solicitudesRechazadas { get; set; }

     //Constructor
     public Miembro(string unEmail, string unPassword, string suNombre, string
suApellido, DateTime suFechaNacimiento) : base(unEmail, unPassword)
     {
         this.Nombre = suNombre;
         this.Apellido = suApellido;
         this.fechaNacimiento = suFechaNacimiento;
         this.esBloqueado = false;
         this.amigos = new List<Miembro>(); //Inicializa las listas
         this.publicaciones = new List<Publicacion>();
         this.solicitudesPendientes = new List<Invitacion>();
         this.solicitudesAprobadas = new List<Invitacion>();
         this.solicitudesRechazadas = new List<Invitacion>();
     }

     //M É T O D O S

     public void Validaciones()
     {
         ValidarNombre();
         ValidarApellido();
     }

     private void ValidarNombre()
     {
         if (this.Nombre == null || this.Nombre.Trim().Length == 0)
         {
             throw new Exception("El campo del nombre no debe estar vacío.");
         }
     }

     private void ValidarApellido()
     {
```

```
        if (this.Apellido == null || this.Apellido.Trim().Length == 0)
        {
            throw new Exception($"El campo del apellido no debe estar vacío.");
        }
    }

    public Invitacion EnviarSolicitudAmistad(Miembro miembroDestinatario)
    {

        if (this.esBloqueado)
        {
            throw new Exception($"No puedes enviar la solicitud porque estás
bloqueado por un administrador.");
        }
        else if (miembroDestinatario.esBloqueado)
        {
            throw new Exception($"No puedes enviar la solicitud porque el
destinatario está bloqueado por un administrador.");
        }
        else
        {
            Invitacion solicitud = new Invitacion(this, miembroDestinatario,
EstadoInvitacion.PENDIENTE_APROBACION);

            miembroDestinatario.solicitudesPendientes.Add(solicitud);

            return solicitud;
        }
    }

    public void AceptarSolicitudAmistad(Invitacion unaSolicitudDeAmistad)
    {
        if (this.esBloqueado)
        {
            throw new Exception($"No puedes aceptar la solicitud porque estás
bloqueado por un administrador.");
        }
        else
        {
            if (unaSolicitudDeAmistad.Estado ==
EstadoInvitacion.PENDIENTE_APROBACION)
            {
                unaSolicitudDeAmistad.Estado = EstadoInvitacion.APROBADA;
                solicitudesAprobadas.Add(unaSolicitudDeAmistad);

unaSolicitudDeAmistad.miembroSolicitante.solicitudesAprobadas.Add(unaSolicitudDeA
mistad); //Agrega la solicitud a la lista de aprobadas del miembro que envió la
solicitud
            }
            else
            {
                throw new Exception($"No puedes aceptar la solicitud porque no
está en estado pendiente.");
            }
        }
    }

    public void RechazarSolicitudDeAmistad(Invitacion unaSolicitudDeAmistad)
    {
        if (this.esBloqueado)
        {
            throw new Exception($"No puedes aceptar la solicitud porque estás
bloqueado por un administrador.");
```

```
        }
        else
        {
            if (unaSolicitudDeAmistad.Estado ==
EstadoInvitacion.PENDIENTE_APROBACION)
            {
                unaSolicitudDeAmistad.Estado = EstadoInvitacion.RECHAZADA;
                solicitudesRechazadas.Add(unaSolicitudDeAmistad);

unaSolicitudDeAmistad.miembroSolicitante.solicitudesRechazadas.Add(unaSolicitudDe
Amistad); //Agrega la solicitud a la lista de rechazadas del miembro que envió la
solicitud
            }
            else
            {
                throw new Exception($"No puedes rechazar la solicitud porque no
está en estado pendiente.");
            }
        }
    }

    public void HacerPublicacion(Post unPost)
    {
        if (!unPost.esHabilitado)
        {
            throw new Exception($"No se puede comentar este post porque está
deshabilitado por un administrador.");
        }
        else if (this.esBloqueado)
        {
            throw new Exception($"No puedes hacer una publicación porque estas
bloqueado por un administrador.");
        }
        else
        {
            publicaciones.Add(unPost);
        }
    }

    public void HacerComentario(Post unPost, Comentario unComentario)
    {
        if (!unPost.esHabilitado)
        {
            throw new Exception($"No se puede comentar este post porque está
deshabilitado por un administrador.");
        }
        else
        {
            publicaciones.Add(unComentario);
            unPost.comentarios.Add(unComentario);
        }
    }

    public void ReaccionarPublicacion(Publicacion unaPublicacion, TipoReaccion
unTipoDeReaccion)
    {
        // Verificar si el miembro ya dio una reacción a esta publicación o
comentario.
        if (unaPublicacion.reacciones.Any(reaccion =>
reaccion.miembroQueReacciona == this))
        {
            throw new Exception($"Ya has reaccionado a esta publicación!");
        }
```

```
        else
        {
            // Agregar la reacción a la lista de reacciones de la publicación o
comentario.
            Reaccion nuevaReaccion = new Reaccion(unTipoDeReaccion, this);
            unaPublicacion.reacciones.Add(nuevaReaccion);
        }
    }

    public override string ToString()
    {
        return
$"{this.Email}\t\t{this.Nombre}\t{this.Apellido}\t{this.fechaNacimiento.ToShortDa
teString()}";
         }
```

5.7 Clase Post

```
public class Post : Publicacion
{
    //A T R I B U T O S
    public string Imagen {  get; set; }

    public List<Comentario> comentarios { get; set; }

    public bool esHabilitado { get; set; } = true; //Comienzan todos habilitados

    public string Contenido { get; set; }

    public string Titulo { get; set; }

    public bool esPrivado { get; set; }

    //Constructor
    public Post(string unTexto, Miembro unAutor, TipoPublicacion unTipo,string
unTitulo, string unContenido, string unaImagen, bool esPrivado, bool
esHabilitado) : base(unTexto, unAutor, unTipo)
    {
        this.Titulo = unTitulo;
        this.Contenido = unContenido;
        this.Imagen = unaImagen;
        this.esPrivado = esPrivado;
        this.comentarios = new List<Comentario>();
        this.esHabilitado = esHabilitado;
    }

    //M É T O D O S

    public void ValidacionesPost()
    {
        ValidarImagenNoVacio();
        ValidarImagen();
        ValidarTituloNoVacio();
        ValidarLargoTitulo();
        ValidarContenidoNoVacio();
    }

    private void ValidarContenidoNoVacio()
    {
        if (this.Contenido == null || this.Contenido.Trim().Length == 0)
        {
```

```
            throw new Exception($"El campo del contenido no debe estar vacío.");
        }
    }

    private void ValidarLargoTitulo()
    {
        if (this.Titulo.Trim().Length <= 3)
        {
            throw new Exception($"El campo del título debe ser más largo.");
        }
    }

    private void ValidarTituloNoVacio()
    {
        if (this.Titulo == null || this.Titulo.Trim().Length == 0)
        {
            throw new Exception($"El campo del título no debe estar vacío.");
        }
    }

    private void ValidarImagenNoVacio()
    {
        if(this.Imagen == null || this.Imagen.Trim().Length == 0)
        {
            throw new Exception($"El campo de la imagen no debe estar vacío.");
        }
    }

    private void ValidarImagen()
    {
        if (!this.Imagen.EndsWith(".jpg") || !this.Imagen.EndsWith(".png"))
        {
            throw new Exception($"La imagen debe tener terminación '.jpg' o
'.png'.");
        }
    }

    public bool PuedeVerPost(Miembro unMiembro)
    {
        if (this.esPrivado)
        {
            return this.Autor.amigos.Contains(unMiembro); //Si el post es privado
verifica que el miembro esté en la lista de amigos
        }
        else if (!this.esHabilitado)
        {
            throw new Exception($"No puedes ver el post porque el mismo está
censurado por un administrador.");
        }
        return true; //Si es público todos lo pueden ver
    }

    public bool PuedeComentarPost(Miembro unMiembro)
    {
        if (this.esPrivado)
        {
            return this.Autor.amigos.Contains(unMiembro); //Si el post es privado
verifica que el miembro esté en la lista de amigos
        }
        else if (!this.esHabilitado)
        {
            throw new Exception($"No puedes ver el post porque el mismo está
censurado por un administrador.");
```

```
        }
        return true; //Si es público cualquiera lo puede comentar
    }

    public override string ToString()
    {
        return $"{this.Id}\t{this.Tipo}\t\t'{this.Titulo}'";
         }
```

5.8 Clase Publicacion

```
public abstract class Publicacion
{
    //A T R I B U T O S
    public int Id { get; set; }

    private static int lastId {  get; set; } = 1;

    public string Texto { get; set; }

    public  DateTime Fecha { get; set; }

    public Miembro Autor { get; set; }

    public TipoPublicacion Tipo { get; set; }

    public List<Reaccion> reacciones { get; set; }

    //Constructor
    public Publicacion(string unTexto, Miembro unAutor, TipoPublicacion unTipo)
    {
        this.Id = lastId++;
        this.Texto = unTexto;
        Fecha = DateTime.Now;
        this.Autor = unAutor;
        this.Tipo = unTipo;
        this.reacciones = new List<Reaccion>();
    }

    //M É T O D O S

    public void ValidacionesPublicacion()
    {
        ValidarTexto();
    }

    private void ValidarTexto()
    {
        if (this.Texto == null || this.Texto.Trim().Length == 0)
        {
            throw new Exception("El campo del texto no debe estar vacío.");
        }
    }

    public int CompareTo(object? obj) //this primero ascendente, this despues
descendente
    {
        Publicacion comparar = (Post)obj;
        return comparar.Fecha.CompareTo(this.Fecha);
    }
```

```
    public override string ToString()
    {
        return $"{this.Id}\t{this.Tipo}";
         }
```

5.9 Clase Reacción

```
public class Reaccion
 {
     public TipoReaccion TipoReaccion { get; set; }

     public Miembro miembroQueReacciona { get; set; }

     //Contructor
     public Reaccion(TipoReaccion unTipoReaccion, Miembro unMiembroQueReacciona)
     {
         this.TipoReaccion = unTipoReaccion;
         this.miembroQueReacciona = unMiembroQueReacciona;
     }
      }
```

5.10 Clase Sistema

```
public class Sistema
{
    //Patrón Singleton
    private static Sistema instancia;

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

    //A T R I B U T O S
    public List<Miembro> miembros { get; set; }

    public List<Administrador> administradores { get; set; }

    public List<Publicacion> publicaciones { get; set; }

    public List<Usuario> usuarios { get ; set; }

    //Constructor
    private Sistema()
    {
        this.miembros = new List<Miembro>(); //Inicializa las listas
        this.administradores = new List<Administrador>();
        this.publicaciones = new List<Publicacion>();
        this.usuarios = new List<Usuario>();
        PrecargaDatos(); //Se precargan los datos al iniciar el sistema
    }
```

```
//M É T O D O S

//Precarga de datos
private void PrecargaDatos()
{
    PrecargaUsuariosPublicacionesInvitacionesReacciones();
    AgregarMiembrosAListaDeAmigos();
}

private void PrecargaUsuariosPublicacionesInvitacionesReacciones()
{
    //Miembros
    Miembro miembro1 = new Miembro("miembro1@hotmail.com", "miembro1",
"miembro1", "apellidoMiembro1", new DateTime(2000, 01, 01));
    Miembro miembro2 = new Miembro("miembro2@hotmail.com", "miembro2",
"miembro2", "apellidoMiembro2", new DateTime(1986, 12, 22));
    Miembro miembro3 = new Miembro("miembro3@hotmail.com", "miembro3",
"miembro3", "apellidoMiembro3", new DateTime(2008, 10, 20));
    Miembro miembro4 = new Miembro("miembro4@hotmail.com", "miembro4",
"miembro4", "apellidoMiembro4", new DateTime(2005, 07, 17));
    Miembro miembro5 = new Miembro("miembro5@hotmail.com", "miembro5",
"miembro5", "apellidoMiembro5", new DateTime(2001, 08, 09));
    Miembro miembro6 = new Miembro("miembro6@hotmail.com", "miembro6",
"miembro6", "apellidoMiembro6", new DateTime(1995, 10, 03));
    Miembro miembro7 = new Miembro("miembro7@hotmail.com", "miembro7",
"miembro7", "apellidoMiembro7", new DateTime(1993, 03, 02));
    Miembro miembro8 = new Miembro("miembro8@hotmail.com", "miembro8",
"miembro8", "apellidoMiembro8", new DateTime(1990, 11, 08));
    Miembro miembro9 = new Miembro("miembro9@hotmail.com", "miembro9",
"miembro9", "apellidoMiembro9", new DateTime(1992, 12, 04));
    Miembro miembro10 = new Miembro("miembro10@hotmail.com", "miembro10",
"miembro10", "apellidoMiembro10", new DateTime(2000, 12, 20));

    miembros.Add(miembro1);
    miembros.Add(miembro2);
    miembros.Add(miembro3);
    miembros.Add(miembro4);
    miembros.Add(miembro5);
    miembros.Add(miembro6);
    miembros.Add(miembro7);
    miembros.Add(miembro8);
    miembros.Add(miembro9);
    miembros.Add(miembro10);
    usuarios.Add(miembro1);
    usuarios.Add(miembro2);
    usuarios.Add(miembro3);
    usuarios.Add(miembro4);
    usuarios.Add(miembro5);
    usuarios.Add(miembro6);
    usuarios.Add(miembro7);
    usuarios.Add(miembro8);
    usuarios.Add(miembro9);
    usuarios.Add(miembro10);

    //Administrador
    Administrador administrador1 = new
Administrador("administrador1@hotmail.com", "administrador1");

    administradores.Add(administrador1);
    usuarios.Add(administrador1);

    //Publicaciones
```

```
    Post post1 = new Post("Texto 1", miembro1, TipoPublicacion.POST, "Título Post
1", "Este es el contenido del post 1 público", "imagen1.png", false, true);
    Post post2 = new Post("Texto 2", miembro1, TipoPublicacion.POST, "Título Post
2", "Este es el contenido del post 2 público", "imagen2.png", false, true);
    Post post3 = new Post("Texto 3", miembro3, TipoPublicacion.POST, "Título Post
3", "Este es el contenido del post 3 público", "imagen3.png", false, true);
    Post post4 = new Post("Texto 4", miembro4, TipoPublicacion.POST, "Título Post
4", "Este es el contenido del post 4 público", "imagen4.png", false, true);
    Post post5 = new Post("Texto 5", miembro5, TipoPublicacion.POST, "Título Post
5", "Este es el contenido del post 5 privado", "imagen5.png", true, false);

    publicaciones.Add(post1);
    publicaciones.Add(post2);
    publicaciones.Add(post3);
    publicaciones.Add(post4);
    publicaciones.Add(post5);
    //Post dentroo de la lista de -publicaciones del miembro
    miembro1.publicaciones.Add(post1);
    miembro1.publicaciones.Add(post2);
    miembro3.publicaciones.Add(post3);
    miembro4.publicaciones.Add(post4);
    miembro5.publicaciones.Add(post5);

    //Comentarios
    //Del post1
    Comentario comentario1 = new Comentario("Texto 6", miembro1,
TipoPublicacion.COMENTARIO, "Título Comentario 1", "Este es el contenido del
comentario 1 público", false);
    Comentario comentario2 = new Comentario("Texto 7", miembro1,
TipoPublicacion.COMENTARIO, "Título Comentario 2", "Este es el contenido del
comentario 2 público", false);
    Comentario comentario3 = new Comentario("Texto 8", miembro1,
TipoPublicacion.COMENTARIO, "Título Comentario 3", "Este es el contenido del
comentario 3 público", false);

    //Del post2
    Comentario comentario4 = new Comentario("Texto 9", miembro9,
TipoPublicacion.COMENTARIO, "Título Comentario 4", "Este es el contenido del
comentario 4 público", false);
    Comentario comentario5 = new Comentario("Texto 10", miembro10,
TipoPublicacion.COMENTARIO, "Título Comentario 5", "Este es el contenido del
comentario 5 público", false);
    Comentario comentario6 = new Comentario("Texto 11", miembro10,
TipoPublicacion.COMENTARIO, "Título Comentario 6", "Este es el contenido del
comentario 6 público", false);

    //Del post3
    Comentario comentario7 = new Comentario("Texto 12", miembro2,
TipoPublicacion.COMENTARIO, "Título Comentario 7", "Este es el contenido del
comentario 7 público", false);
    Comentario comentario8 = new Comentario("Texto 13", miembro2,
TipoPublicacion.COMENTARIO, "Título Comentario 8", "Este es el contenido del
comentario 8 público", false);
    Comentario comentario9 = new Comentario("Texto 14", miembro3,
TipoPublicacion.COMENTARIO, "Título Comentario 9", "Este es el contenido del
comentario 9 público", false);

    //Del post4
    Comentario comentario10 = new Comentario("Texto 15", miembro4,
TipoPublicacion.COMENTARIO, "Título Comentario 10", "Este es el contenido del
comentario 10 público", false);
```

```
    Comentario comentario11 = new Comentario("Texto 16", miembro4,
TipoPublicacion.COMENTARIO, "Título Comentario 11", "Este es el contenido del
comentario 11 público", false);
    Comentario comentario12 = new Comentario("Texto 17", miembro4,
TipoPublicacion.COMENTARIO, "Título Comentario 12", "Este es el contenido del
comentario 12 público", false);

    //Del post5
    Comentario comentario13 = new Comentario("Texto 18", miembro1,
TipoPublicacion.COMENTARIO, "Título Comentario 13", "Este es el contenido del
comentario 13 privado", true);
    Comentario comentario14 = new Comentario("Texto 19", miembro2,
TipoPublicacion.COMENTARIO, "Título Comentario 14", "Este es el contenido del
comentario 14 privado", true);
    Comentario comentario15 = new Comentario("Texto 20", miembro3,
TipoPublicacion.COMENTARIO, "Título Comentario 15", "Este es el contenido del
comentario 15 privado", true);

    //Comentario dentro lista -publicaciones
    publicaciones.Add(comentario1);
    publicaciones.Add(comentario2);
    publicaciones.Add(comentario3);
    publicaciones.Add(comentario4);
    publicaciones.Add(comentario5);
    publicaciones.Add(comentario6);
    publicaciones.Add(comentario7);
    publicaciones.Add(comentario8);
    publicaciones.Add(comentario9);
    publicaciones.Add(comentario10);
    publicaciones.Add(comentario11);
    publicaciones.Add(comentario12);
    publicaciones.Add(comentario13);
    publicaciones.Add(comentario14);
    publicaciones.Add(comentario15);
    //Comentarios dentro de lis lista -comentarios de los posts
    post1.comentarios.Add(comentario1);
    post1.comentarios.Add(comentario2);
    post1.comentarios.Add(comentario3);
    post2.comentarios.Add(comentario4);
    post2.comentarios.Add(comentario5);
    post2.comentarios.Add(comentario6);
    post3.comentarios.Add(comentario7);
    post3.comentarios.Add(comentario8);
    post3.comentarios.Add(comentario9);
    post4.comentarios.Add(comentario10);
    post4.comentarios.Add(comentario11);
    post4.comentarios.Add(comentario12);
    post5.comentarios.Add(comentario13);
    post5.comentarios.Add(comentario14);
    post5.comentarios.Add(comentario15);
    //Comentarios dentro de la lista -publicaciones de los miembros
    miembro1.publicaciones.Add(comentario1);
    miembro1.publicaciones.Add(comentario2);
    miembro1.publicaciones.Add(comentario3);
    miembro9.publicaciones.Add(comentario4);
    miembro10.publicaciones.Add(comentario5);
    miembro10.publicaciones.Add(comentario6);
    miembro2.publicaciones.Add(comentario7);
    miembro2.publicaciones.Add(comentario8);
    miembro3.publicaciones.Add(comentario9);
    miembro4.publicaciones.Add(comentario10);
    miembro4.publicaciones.Add(comentario11);
    miembro4.publicaciones.Add(comentario12);
```

```
    miembro1.publicaciones.Add(comentario13);
    miembro2.publicaciones.Add(comentario14);
    miembro3.publicaciones.Add(comentario15);

    //Invitaciones
    //Miembro1 amigo de todos
    Invitacion solicitudAceptada1 = new Invitacion(miembro2, miembro1,
EstadoInvitacion.APROBADA);
    Invitacion solicitudAceptada2 = new Invitacion(miembro3, miembro1,
EstadoInvitacion.APROBADA);
    Invitacion solicitudAceptada3 = new Invitacion(miembro4, miembro1,
EstadoInvitacion.APROBADA);
    Invitacion solicitudAceptada4 = new Invitacion(miembro5, miembro1,
EstadoInvitacion.APROBADA);
    Invitacion solicitudAceptada5 = new Invitacion(miembro6, miembro1,
EstadoInvitacion.APROBADA);
    Invitacion solicitudAceptada6 = new Invitacion(miembro7, miembro1,
EstadoInvitacion.APROBADA);
    Invitacion solicitudAceptada7 = new Invitacion(miembro8, miembro1,
EstadoInvitacion.APROBADA);
    Invitacion solicitudAceptada8 = new Invitacion(miembro9, miembro1,
EstadoInvitacion.APROBADA);
    Invitacion solicitudAceptada9 = new Invitacion(miembro10, miembro1,
EstadoInvitacion.APROBADA);

    miembro1.solicitudesAprobadas.Add(solicitudAceptada1);
    miembro1.solicitudesAprobadas.Add(solicitudAceptada2);
    miembro1.solicitudesAprobadas.Add(solicitudAceptada3);
    miembro1.solicitudesAprobadas.Add(solicitudAceptada4);
    miembro1.solicitudesAprobadas.Add(solicitudAceptada5);
    miembro1.solicitudesAprobadas.Add(solicitudAceptada6);
    miembro1.solicitudesAprobadas.Add(solicitudAceptada7);
    miembro1.solicitudesAprobadas.Add(solicitudAceptada8);
    miembro1.solicitudesAprobadas.Add(solicitudAceptada9);

    //Miembro2 amigo de todos
    Invitacion solicitudAceptada01 = new Invitacion(miembro1, miembro2,
EstadoInvitacion.APROBADA);
    Invitacion solicitudAceptada02 = new Invitacion(miembro3, miembro2,
EstadoInvitacion.APROBADA);
    Invitacion solicitudAceptada03 = new Invitacion(miembro4, miembro2,
EstadoInvitacion.APROBADA);
    Invitacion solicitudAceptada04 = new Invitacion(miembro5, miembro2,
EstadoInvitacion.APROBADA);
    Invitacion solicitudAceptada05 = new Invitacion(miembro6, miembro2,
EstadoInvitacion.APROBADA);
    Invitacion solicitudAceptada06 = new Invitacion(miembro7, miembro2,
EstadoInvitacion.APROBADA);
    Invitacion solicitudAceptada07 = new Invitacion(miembro8, miembro2,
EstadoInvitacion.APROBADA);
    Invitacion solicitudAceptada08 = new Invitacion(miembro9, miembro2,
EstadoInvitacion.APROBADA);
    Invitacion solicitudAceptada09 = new Invitacion(miembro10, miembro2,
EstadoInvitacion.APROBADA);

    miembro2.solicitudesAprobadas.Add(solicitudAceptada01);
    miembro2.solicitudesAprobadas.Add(solicitudAceptada02);
    miembro2.solicitudesAprobadas.Add(solicitudAceptada03);
    miembro2.solicitudesAprobadas.Add(solicitudAceptada04);
    miembro2.solicitudesAprobadas.Add(solicitudAceptada05);
    miembro2.solicitudesAprobadas.Add(solicitudAceptada06);
    miembro2.solicitudesAprobadas.Add(solicitudAceptada07);
    miembro2.solicitudesAprobadas.Add(solicitudAceptada08);
```

```
    miembro2.solicitudesAprobadas.Add(solicitudAceptada09);

    //m3
    miembro3.solicitudesAprobadas.Add(solicitudAceptada2);
    miembro3.solicitudesAprobadas.Add(solicitudAceptada02);

    Invitacion solicitudPendiente1 = new Invitacion(miembro3, miembro4,
EstadoInvitacion.PENDIENTE_APROBACION);
    miembro3.solicitudesPendientes.Add(solicitudPendiente1); //Pendiente con el
m4
    //m4
    miembro4.solicitudesAprobadas.Add(solicitudAceptada3);
    miembro4.solicitudesAprobadas.Add(solicitudAceptada03);
    miembro4.solicitudesPendientes.Add(solicitudPendiente1); //Pendiente con el
m3
    //m5
    miembro5.solicitudesAprobadas.Add(solicitudAceptada4);
    miembro5.solicitudesAprobadas.Add(solicitudAceptada04);

    Invitacion solicitudPendiente2 = new Invitacion(miembro5, miembro6,
EstadoInvitacion.PENDIENTE_APROBACION);
    miembro5.solicitudesPendientes.Add(solicitudPendiente2); //Pendiente con el
m6
    //m6
    miembro6.solicitudesAprobadas.Add(solicitudAceptada5);
    miembro6.solicitudesAprobadas.Add(solicitudAceptada05);
    miembro6.solicitudesPendientes.Add(solicitudPendiente2); //Pendiente con el
m5
    //m7
    miembro7.solicitudesAprobadas.Add(solicitudAceptada6);
    miembro7.solicitudesAprobadas.Add(solicitudAceptada06);

    Invitacion solicitudRechazada1 = new Invitacion(miembro7, miembro8,
EstadoInvitacion.RECHAZADA);
    miembro7.solicitudesRechazadas.Add(solicitudRechazada1); //Rechazada con el
m8
    //m8
    miembro8.solicitudesAprobadas.Add(solicitudAceptada7);
    miembro8.solicitudesAprobadas.Add(solicitudAceptada07);
    miembro8.solicitudesRechazadas.Add(solicitudRechazada1); //Rechazada con el
m7
    //m9
    miembro9.solicitudesAprobadas.Add(solicitudAceptada8);
    miembro9.solicitudesAprobadas.Add(solicitudAceptada08);

    Invitacion solicitudRechazada2 = new Invitacion(miembro9, miembro10,
EstadoInvitacion.RECHAZADA);
    miembro9.solicitudesRechazadas.Add(solicitudRechazada2); //Rechazada con el
m10
    //m10
    miembro10.solicitudesAprobadas.Add(solicitudAceptada9);
    miembro10.solicitudesAprobadas.Add(solicitudAceptada09);
    miembro10.solicitudesRechazadas.Add(solicitudRechazada2); //Rechazada con el
m9

    //Reacciones
    Reaccion like = new Reaccion(TipoReaccion.LIKE, miembro1);
    Reaccion dislike = new Reaccion(TipoReaccion.DISLIKE, miembro1);

    //Reaccion dentro de posts
    post4.reacciones.Add(like);
    post5.reacciones.Add(dislike);
```

```
    Reaccion like3 = new Reaccion(TipoReaccion.LIKE, miembro5);
    Reaccion dislike3 = new Reaccion(TipoReaccion.DISLIKE, miembro10);

    //Reaccion dentro de comentarios
    comentario1.reacciones.Add(like3);
    comentario3.reacciones.Add(dislike3);
}
    private void AgregarMiembrosAListaDeAmigos()
    {
        foreach (Miembro miembro in miembros)
        {
            foreach (Invitacion invitacionAceptada in
miembro.solicitudesAprobadas)
            {
                if (invitacionAceptada.miembroSolicitante == miembro)
                {
                miembro.amigos.Add(invitacionAceptada.miembroSolicitado);
                }
            }
        }
         }


public Post MostrarPost(Post unPost)
{
    if (!unPost.esHabilitado)
    {
        throw new Exception($"No se puede mostrar este post porque está
deshabilitado por un administrador.");
    }
    else
    {
        return unPost;
    }
}

public List<Miembro> ListarMiembrosConMasPublicaciones()
{
    List<Miembro> elMasPublicador = new List<Miembro>();
    int maxPublicaciones = 0;

    foreach (Miembro miembro in miembros)
    {
        int totalPublicaciones = miembro.publicaciones.Count; //Calcula la
cantidad de publicaciones (Post + Comentarios)

        if (totalPublicaciones > maxPublicaciones) //Verifica si el miembro tiene
más publicaciones que el máximo
        {
            elMasPublicador.Clear(); //Limpia la lista
            maxPublicaciones = totalPublicaciones;
        }

        if (totalPublicaciones == maxPublicaciones)
        {
            elMasPublicador.Add(miembro);
        }
    }
    return elMasPublicador;
}

public bool BuscarMiembroConEmailBooleano(string unEmail)
{
```

```
    foreach (Miembro miembro in miembros)
    {
        if (miembro.Email.Equals(unEmail))
        {
            return true;
        }
    }
    return false;
}

public bool SeEncontroMiembroConPassword(string unPassword)
{
    foreach (Miembro miembro in miembros)
    {
        if (miembro.Password.Equals(unPassword))
        {
            return true;
        }
    }
    return false;
}

public Miembro BuscarMiembroConEmail(string unEmail)
{
    Miembro miembroEncontrado = null;

    foreach (Miembro miembro in miembros)
    {
        if (miembro.Email.Equals(unEmail))
        {
            miembroEncontrado = miembro;
            break; //Porque ya lo encontró
        }
    }
    return miembroEncontrado;
}

public List<Publicacion> ListarPublicacionesConComentarios(Miembro unMiembro)
{
    List<Publicacion> publicacionesConComentarios = new List<Publicacion>();

    foreach (Publicacion publicacion in publicaciones)
    {
        if (publicacion is Post post)
        {
            foreach (Comentario comentario in post.comentarios)
            {
                if (comentario.Autor == unMiembro)
                {
                    publicacionesConComentarios.Add(post);
                    break; //Hizo un comentario en ese post, no es necesario
volver a iterar.
                }
            }
        }
    }
    return publicacionesConComentarios;
}

public List<Publicacion> ListarPublicacionesConEmail(Miembro unMiembro)
{
    List<Publicacion> listaPublicacionesFiltradas = new List<Publicacion>();
```

```
    foreach (Miembro miembro in miembros)
    {
        if (miembro.Email == unMiembro.Email)
        {
            foreach (Publicacion publicacion in publicaciones)
            {
                if (publicacion.Autor == miembro)
                {
                    listaPublicacionesFiltradas.Add(publicacion);
                }
            }
        }
    }

    if (listaPublicacionesFiltradas.Count == 0)
    {
        throw new Exception("El miembro ingresado no tiene publicaciones
hechas.");
    }

    return listaPublicacionesFiltradas;
     }


public void AltaPost(Post unPost, Miembro autorPost)
 {
     unPost.ValidacionesPublicacion();
     unPost.ValidacionesPost();
     autorPost.HacerPublicacion(unPost);
     publicaciones.Add(unPost);
 }

 public void AltaComentario(Comentario unComentario, Miembro autorComentario,
Post unPost)
 {
     unComentario.ValidacionesPublicacion();
     unComentario.Validaciones();
     autorComentario.HacerComentario(unPost, unComentario);
     publicaciones.Add(unComentario);
 }

 public void AltaMiembro(Miembro unMiembro)
 {
     unMiembro.ValidacionesUsuario();
     unMiembro.Validaciones();
     usuarios.Add(unMiembro); //Se agrega a la lista
     miembros.Add(unMiembro);
 }

 public bool TryParseFecha(string fechaString, out DateTime fecha) //Para parsear
una fecha al formato deseado
 {
     return DateTime.TryParseExact(fechaString, "dd/MM/yyyy", null,
System.Globalization.DateTimeStyles.None, out fecha);
 }

 public bool EmailEsValido(string unEmail)
 {
     return unEmail.Trim().Length > 0 && unEmail.Contains("@") &&
!unEmail.StartsWith("@") && !unEmail.EndsWith("@");
 }

 public bool PasswordEsValido(string unPassword)
```

```
 {
     return unPassword != null && unPassword.Trim().Length > 0;
 }

 public int askInt(string unMessage) //Verifica que se introduzca un int
 {
     bool validSelection = false;
     int selected = -1;
     while (!validSelection)
     {
         try
         {
             Console.WriteLine(unMessage);
             selected = int.Parse(Console.ReadLine());
             validSelection = true;
         }
         catch (Exception e)
         {
             validSelection = false;
             Console.WriteLine("Solo se aceptan números. Inténtalo de nuevo...");
//Única línea de consola en el dominio!
         }
     }
     return selected;
      }
```

5.11 Clase TipoPublicacion

```
public enum TipoPublicacion
{
    POST = 1,
    COMENTARIO = 2
     }
```

5.12 Clase TipoReaccion

```
public enum TipoReaccion
 {
     LIKE = 1,
     DISLIKE = 2
      }
```

5.13 ClaseUsuario

```
public abstract class Usuario
 {
     public string Email { get; set; }

     public string Password { get; set; }

     public Usuario(string unEmail, string unPassword)
     {
         this.Email = unEmail;
```

```
        this.Password = unPassword;
    }

    public void ValidacionesUsuario()
    {
        ValidarPassword();
    }

    private void ValidarPassword()
    {
        if (this.Password.Trim() == null || this.Password.Trim().Length == 0)
        {
            throw new Exception($"El campo de la contraseña no debe estar
vacío.");
        }
    }

    protected bool IniciarSesion(string emailIngresado, string
passwordIngresado)
    {
        return this.Email == emailIngresado && this.Password ==
passwordIngresado;
    }
     }
```

6. ANEXOS

ANEXO 1: LETRA DEL OBLIGATORIO

EVALUACIÓN	Obligatorio 1	**GRUPO**	Todos	**FECHA**	Agosto 2023
MATERIA	Programación 2				
CARRERA	Analista Programador/Analista en Tecnologías de la Información				
CONDICIONES	**Puntos:** 15 **Fecha de entrega: 05/10/2023 (hasta las 21:00 horas en gestion.ort.edu.uy).** LA ENTREGA SE REALIZA EN FORMA ONLINE EN ARCHIVO NO MAYOR A 40MB EN FORMATO ZIP, RAR O PDF. **IMPORTANTE:** • Inscribirse • Formar grupos de hasta 2 integrantes, • Aquellos de ustedes que presenten alguna dificultad con su inscripción o tengan inconvenientes técnicos, por favor contactarse con el Coordinador o Coordinación adjunta **antes de las 20:00hs.** del día de la entrega, a través de los mails gervaz@ort.edu.uy, alamon@ort.edu.uy y fernandez_ma@ort.edu.uy, o telefónicamente al 29021505 - int 1156 o 1138 • Subir el trabajo a Gestión antes de la hora indicada, ver hoja al final del documento: "RECORDATORIO"				

Una empresa de medios nos ha encomendado la tarea de crear una aplicación de redes sociales, la nueva Social.NetWork.

Se trata de una red social de tema libre que constará de miembros, publicaciones y administradores. Tanto los miembros como administradores comparten los siguientes atributos: un email y una contraseña, ambos serán utilizados para verificar su identidad y acceder a la aplicación en el futuro. De los miembros se conoce además el nombre, apellido, fecha de nacimiento y la lista de amigos.

Un miembro le puede solicitar a otro establecer un vínculo de amistad, generando una invitación, de la cual se conoce un id, el miembro solicitante, el miembro solicitado, el estado de esa solicitud que puede tomar valores PENDIENTE_APROBACION, APROBADA o RECHAZADA y la fecha de solicitud. La fecha de solicitud será la fecha actual.

Cuando el miembro acepta la solicitud de amistad, se establece un vínculo recíproco, es decir que el vínculo de amistad se genera para los dos. Un miembro puede estar bloqueado por el administrador y en ese caso se verán restringidas algunas funcionalidades. Si el miembro está bloqueado no podrá ni enviar solicitudes ni aceptarlas/rechazarlas.

Los miembros son los únicos capaces de realizar publicaciones, las que contienen un identificador único secuencial, un texto no vacío, una fecha (la fecha actual) y el autor (es decir, el mismo miembro que la publicó).

Existen dos tipos de publicaciones: Los Posts y los comentarios a un Post. De cada uno se conoce un id, el título (no vacío, al menos de 3 caracteres), el autor, la fecha, el contenido que no puede ser vacío.

Los Posts tienen además una imagen obligatoriamente (en esta etapa solo se guardará el nombre de la imagen, que no puede ser vacío y debe terminar en ".jpg" o ".png"), pueden ser públicos o privados, tienen una lista de comentarios y pueden estar censurados por el administrador, y en ese caso están

deshabilitados, no se pueden mostrar ni comentar. Los posts privados solo podrán ser visualizados y comentados por los amigos de su autor.

Los comentarios no tendrán ningún tipo de publicación anidada. Los comentarios a un post privado serán privados, y los correspondientes a un post público serán públicos.

Las publicaciones, tanto Posts y comentarios, pueden tener reacciones de las que interesa saber el tipo de reacción (Like o Dislike) y el miembro que lo realizó. Un usuario puede poner un solo like/dislike por publicación.

De las publicaciones nos interesa conocer/calcular el valor de aceptación (VA) cuya fórmula varía dependiendo del tipo de publicación. Veamos:

- Para calcular el VA de un Post, se cuentan todas las reacciones del tipo "Like" y se multiplica por un Factor de VA (FVA) igual a 5. Este mismo cálculo se realiza para "Dislikes" pero multiplicado por un FVA igual a -2. Sumamos ambas partes y por último se suma 10 si el Post es público, de lo contrario 0.
- Para comentarios se realiza el mismo cálculo, pero sin incluir los 10 puntos extras. Lo que quedaría:

$$VA = (Likes \times 5) + (Dislikes \times -2)$$

IMPORTANTE: Para esta entrega no se solicita realizar el método que permite determinar el valor de aceptación de un POST, pero debe estar diseñado y presente en el diagrama de clases.

Se pide:

Punto 1: Diseño

- Diseño de la realidad planteada representada en el diagrama de clases completo del Dominio (Reglas del negocio) que modele la situación anterior. Se seguirá el estándar UML y debe ser presentado en formato Astah.

Punto 2: Implementación

Implementar al menos dos proyectos 1) biblioteca de clases y 2) aplicación de consola – que incluyan el código que corresponda - en Visual Studio 2022 usando .NET 7 y C# como lenguaje de programación, que incluya:

1. Codificación de las clases del dominio necesarias para cumplir con todos los requerimientos del sistema solicitados para este obligatorio (atributos, propiedades, constructor/es, ToString, validaciones y métodos necesarios para llevar adelante los requerimientos planteados).

2. Precarga de datos en el sistema para que permita hacer pruebas con distintos escenarios. Se deberá implementar como mínimo la precarga de:

- 10 miembros
- 1 administrador
- Invitaciones para todos los miembros: donde 2 de ellos tengan al menos un vínculo de amistad con cada uno de los demás miembros restantes y además deberá precargar invitaciones en todos los estados posibles. (aprobadas, rechazadas y en proceso).
- 5 posts y al menos 3 comentarios para cada uno.
- Reacciones para al menos 2 post, y para al menos 2 comentarios.

3. Desplegar un menú en consola que permita:

1) Alta de miembro. Recuerde verificar que se cumplan las reglas mencionadas en la realidad planteada.
2) Dado un email de miembro listar todas las publicaciones que ha realizado, diferenciando en la lista su tipo (si es post o comentario).
3) Dado un email de miembro, listar los posts en los que haya realizado comentarios. Se listarán solamente los posts, no se incluirán los comentarios.
4) Dadas dos fechas listar los posts realizados entre esas fechas inclusive. Se deberá mostrar id, fecha, título y el texto del post. Si el texto del post supera los 50 caracteres, solo se mostrarán los primeros 50. No se mostrarán los comentarios de dichos posts. El listado estará ordenado por título en forma descendente.
5) Obtener los miembros que hayan realizado más publicaciones de cualquier tipo. Si hay más de un miembro con la misma cantidad de publicaciones mostrarlos todos. Se mostrarán todos los datos de los miembros.

ATENCIÓN: LOS DATOS DE PRUEBA DEBEN PERMITIR TESTEAR TODOS LOS CASOS DE LOS REQUERIMIENTOS SOLICITADOS.