**Universidad de Santiago de chile**

**Facultad de ingeniería**

**Departamento de ingeniería informática**

**Paradigma de programación**

**Informe Laboratorio 3***Simón Sáez*

Profesor: Daniel Gacitua

**Índice**

*Introducción 3*

*Descripción del problema 3*

*Descripción del paradigma y sus conceptos 3 - 4*

*Análisis del problema 4 - 5*

*Diseño de la solución 5 - 6*

*Aspectos de implementación. 6*

*Instrucciones de uso 7*

*Resultado y autoevaluación 8*

*Conclusión 8*

**Introducción**

En el siguiente informe se tratará de varios aspectos de la implementación del paradigma orientado a objetos en un proyecto parecido a una red social.

Para ello se explicará mediante varias secciones del informe:

**1. Descripción del problema:** Se describe el problema

**2. Descripción del paradigma y sus conceptos:** Se describe el paradigma

**3. Análisis del problema:** Se hace un análisis al problema planteado

**4. Diseño de solución:** Se describe como se resolverá el problema a detalle

**5. Aspectos de implementación:** Como se implementó la solución y en que

**6. Instrucciones de uso:** Cuales son las instrucciones para usar el programa

**7. Resultado y Autoevaluación:** Resultados obtenidos, alcance obtenido y problemas tenidos

**8. Conclusión**

**Descripción del problema**

Se creará un programa que simule un parecido a una red social como Facebook, Instagram, Twitter, entre otras redes sociales.

En donde esa red social podría hacer lo siguiente:

* Crear una cuenta
* Iniciar sesión de la cuenta
* Hacer una publicación
* Seguir a una persona
* Compartir una publicación
* Comentar una publicación
* Dar likes a publicaciones y/o comentarios

En donde cumplen funciones muy importantes dentro de una red social

**Descripción del paradigma y sus conceptos.**

El paradigma orientado a objetos define todo como un objeto (Seres vivos, cosas, etc), en donde tienen sus propios atributos y métodos. Además, los objetos pueden interactuar entre sí.

En los conceptos usados en este proyecto son nada más ni nada menos que:

* Atributos = Es un concepto que se refiere a las características de un objeto.
* Métodos = Es un concepto que se refiere a las acciones que puede hacer un objeto.
* Composición = Es un concepto que define en que un objeto A contiene un objeto B, pero si A deja de existir, B también. Es decir que el tiempo de vida de B está ligado al de A.
* Agregación = Es un concepto que define en un objeto A contiene un objeto B, pero si A deja de existir, B seguirá existiendo, Es decir que el tiempo de vida de B no está ligado al de A.
* Asociación = Es un concepto que define en un objeto A colabora con un objeto B, por lo que sus tiempos de vida son independientes
* Dependencia = Es un concepto que define que un objeto A depende de un objeto B y la conoce, pero B no conoce a A.
* Miembro de instancia = Es un tributo o método que solo pertenece a ese objeto.
* Miembro de clase = Es un atributo o método que se comparte con todos los objetos.
* Herencia = Es un concepto que dice que un objeto A heredo los atributos y métodos de otro objeto B. Es decir que A es hijo de B.
* Polimorfismo = Es un concepto que permite hacer la misma acción de formas distintas.
* Sobrecarga = Es un concepto proveniente del polimorfismo en el que su nombre es el mismo, pero sus argumentos son distintos. Y pertenecen a la misma clase.
* Interfaces = Es un concepto en el que se declaran solo métodos y no atributos.
* Encapsulamiento = Es un concepto en el que sus atributos están ocultos para los demás y la única forma de acceder es por medio de métodos.

**Análisis del problema**

**Requerimientos No Funcionales Obligatorios**

* Autoevaluacion -> Es una autoevaluación que permite saber el nivel que he cumplido en los requerimientos funcionales y no funcionales obligatorias.
* Lenguaje -> Si la implementación en código está hecha en prolog.
* Versión -> Si la versión de prolog es en la versión 8.X.X.
* Documentación -> Si el código esta correctamente documentado.
* Historial -> Para el historial de commits en Github donde deben ser mas de 10 en un lapso de 2 semana.
* Ejemplos -> Son ejemplos de cómo se usan los predicados.
* Prerrequisitos -> Son para los prerrequisitos que necesitan los predicados y deben ser cumplidos.

**Requerimientos Funcionales Obligatorios**

Para el problema se planteará hacer las siguientes cosas:

Los TDA’s a crear:

* Red Social -> Clase que representa la red social a hacer.
* Publicación -> Clase que representa las publicaciones dentro de la aplicación.
* Usuario -> Clase que representa al usuario en la red social.
* Reacción -> Clase que representa las respuestas a publicaciones dentro de la aplicación.

Y luego crear los métodos esenciales del programa:

* register -> Es el método que permite registrar un nuevo usuario a la red social.
* login -> Es el método que permite hacer conectar a un usuario en la red social.
* logout -> Es un método que permite desconectar un usuario de la red social.
* post -> Es el método que permite hacer una publicación en la red social.
* follow -> Es el método que permite hacer follows entre usuarios en la red social.
* share -> Es el método que permite compartir una publicación a otro usuario en la red social.
* visiualize -> Es el método que permite mostrar el socialnetwork en string o a al usuario conectado en usuario.
* comment -> Es el método que permite hacer un comentario a una publicación o a un comentario.
* like -> Es el método que permite dar un like a una publicación o un comentario.

**Diseño de la solución**

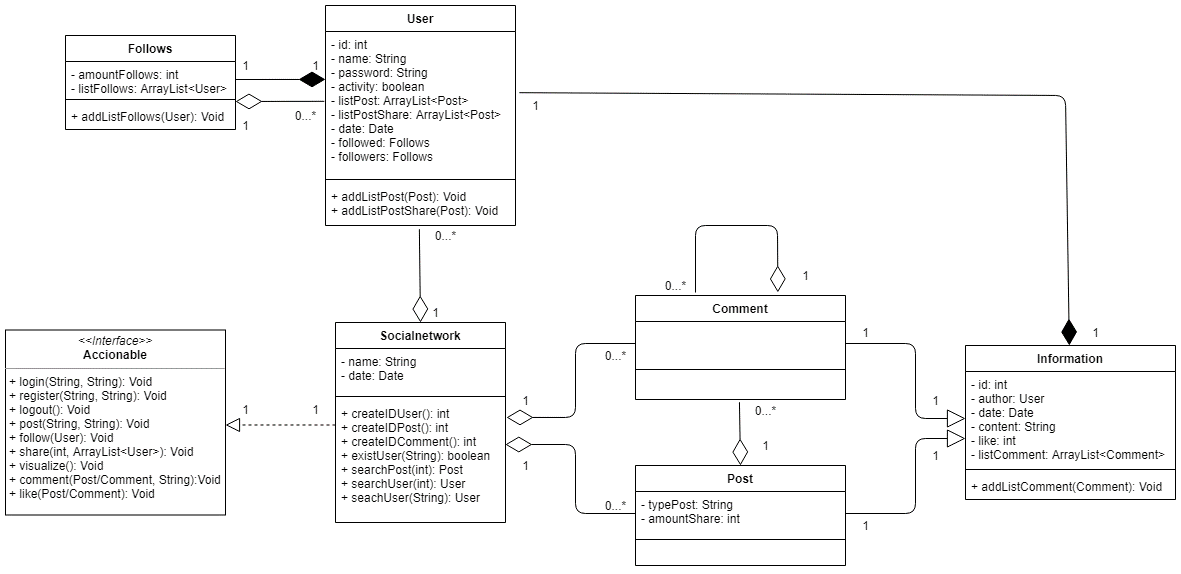
Para la solución se hizo un análisis inicial en donde el Socialnetwork contiene:

* Muchos usuarios (De 0 a muchos) porque comienza sin usuarios inicialmente porque es nueva.
* Muchas publicaciones (De 0 a muchos) porque comienza sin publicaciones inicialmente porque es nueva.
* Muchos comentarios (De 0 a muchos) porque comienza sin comentarios porque es nueva.
* También implementa una interfaz llamada Accionable.

El usuario contiene Follows y el Follow contiene a muchos usuarios (De 0 a muchos), pero los usuarios contenidos son distintos al usuario que se encuentra en el follow.

El Post contiene muchos comentarios (De 0 a muchos) y es hereditario de Information, lo mismo para Comment, contiene muchos comentarios (De 0 a muchos) y es hereditario de information.

Mientras tanto information tiene un autor el cual es un usuario, por lo que si o si contendrá un autor.

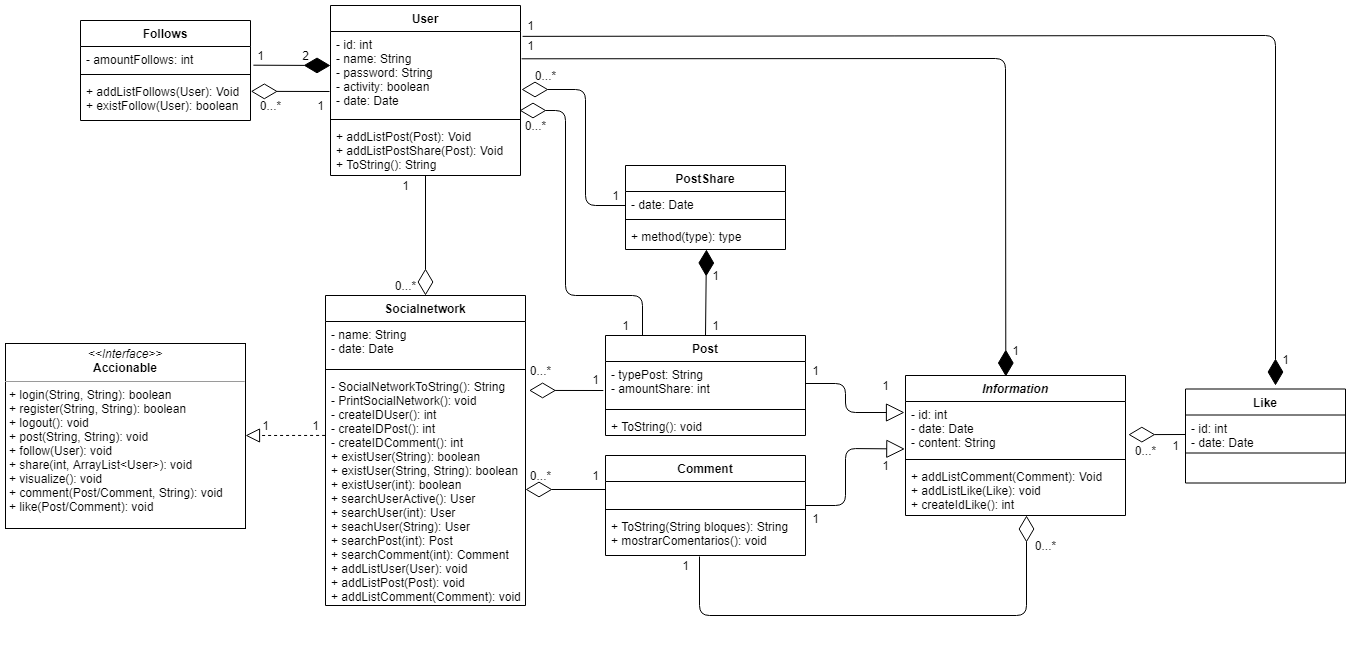


Mientras en los métodos a crear, el login y register harán inicio de sesión al usuario donde uno solo conecta a un usuario ya existente, mientras que el otro crea a un usuario y lo conecta respectivamente. El logout es para desconectar, por lo que solo buscará a uno que esté conectado y lo desconectará.

Mientras que el post y el share tendrán que pasar por un proceso donde si aquella publicación a enviar o a compartir va para otros, ambos se deben seguir. En caso contrario no se puede enviar. Y en el caso dar un like y comentar una publicación se debe ver el post, para ello se debe seguir a la persona.

Entonces el follow mutuo es indispensable para la acción de post, share, like, comment si va para otro.

Aunque ya para la solución final tuvo unos cambios en lo que se tuvo que añadir unas clases (PostShare y Like) y se añadieron también nuevos métodos para mayor facilidad como se logra apreciar en el diagrama de diseños, además de que se cambió la orientación de algunas flechas y su paridad. En consecuencia, también cambiaron unas clases al añadir otras clases.

****

**Aspectos de implementación.**

Se uso el lenguaje de Java utilizando OpenJDK 11 y el IDE Apache NetBeans 12.

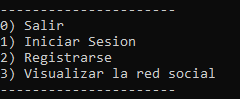
La estructura para el proyecto fue con unas subdivisiones del código en clases donde cada clase representa a un objeto de la red social. Donde se encuentras las clases Socialnetwork, User, Post, Comment, PostShare, Like, Follows; además una clase abstracta llamada Information y una interfaz llamada Accionable.

También hay una Main donde junta todas las clases y las hace funcionar mediante un menú interactivo.

**Instrucciones de uso.**

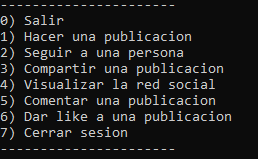
Para este programa se inicia con un menú donde se puede elegir 4 opciones, las cuales son:

* Salir: Sale del programa.
* Iniciar Sesion: Inicia sesión a la red social.
* Registrarse: Registra a un nuevo usuario.
* Visualizar la red social: Visualiza todo el contenido de la red social mediante los usuarios, por lo que no mostrará todas las publicaciones ordenadas, sino que los mostrará en los usuarios todas las publicaciones.



Luego de iniciar sesión o registrarse mostrara un menú donde se pueden elegir 8 opciones, en las cuales son:

* Salir: Sale del programa.
* Hacer una publicación: Hace una publicación a la red social donde puede ser a si mismo o a otros.
* Seguir a una persona: Permite hacer follow a otro usuario.
* Compartir una publicación: Permite compartir la publicación de alguien o de otra persona a un usuario.
* Visualizar la red social: Mostrará el contenido del usuario conectado de la red social.
* Comentar una publicación: Permite comentar una publicación, pero también comentar un comentario de la publicación.
* Cerrar sesión: Permite desconectarse de la red social y volver a ingresar con otro usuario o crear otro usuario.



Luego al acceder a alguna opción, solo debes seguir lo que te dicen y listo. Se muestran los usuarios, las publicaciones, los comentarios disponibles para que sea más intuitivo.

**Resultados**

En el proyecto se alcanzó un grado satisfactorio y completo del programa haciendo que sea agradable, intuitivo y sencillo de comprender. Sin errores, con posibilidades de regresar en caso de apretar mal en algunas opciones.

Se probaron muchas pruebas al programa, donde la mas importante correspondía hacer un comentario de un comentario de un comentario de un comentario de un comentario y luego mostrarlo bien identado y todos los comentarios.

**Conclusión**

En conclusión, se pudo ver que el paradigma fue mas sencillo que los anteriores, además por el lenguaje que se usó que fue Java, por ser muy cercano al paradigma imperativo. Aunque en comparación a los demás se tuvo que hacer un poco más de código de lo normal, pero en parte seguía siendo mas sencillo y tener un mejor orden en el código en comparación al paradigma lógico y mejor análisis y lectura respecto al paradigma funcional.

Se alcanzaron a completar todos los requerimientos obligatorios del proyecto a un buen grado, por lo que es muy satisfactorio.

El lenguaje Java, nombrado anteriormente, hizo que fuese muy cómodo trabajarlo por la razón de que es mas intuitivo la escritura y la lógica, además del IDE que se usó en el proyecto. El autocompletar código, las sugerencias, entre varias más.