**Universidad de Santiago de chile**

**Facultad de ingeniería**

**Departamento de ingeniería informática**

**Paradigma de programación**

**Informe Laboratorio 2***Simón Sáez*

Profesor: Daniel Gacitua

**Índice**

*Introducción 3*

*Descripción del problema 3*

*Descripción del paradigma y sus conceptos 3*

*Análisis del problema 4*

*Diseño de la solución 5 - 6*

*Aspectos de implementación. 6*

*Instrucciones de uso 6 - 9*

*Resultado y autoevaluación 10*

*Conclusión 10*

*Referencias 11*

**Introducción**

En el siguiente informe se tratará de varios aspectos de la implementación del paradigma funcional en un proyecto de un parecido 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 lógico se basa en los conceptos matemáticos de las proposiciones lógicas mediante predicados. Cabe destacar que es un paradigma declarativo.

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

* Unificación = Es el concepto en el que permite “unir” las variables en un hecho para saber en cuales se cumple.
* Predicados = Concepto fundamental del paradigma y con el que está ligado toda la estructura.
* Recursividad = Es una función que se llama a si misma dentro de ella.
* Patter-matching = Permite la “Unificación” de predicados, por lo que es un concepto esencial en la recursividad.

**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 como 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 es crear:

* Socialnetwork -> La base de una socialnetwork
* Publicacion -> Es una estructura de una publicación
* Cuenta -> Es una estructura de una cuenta de un usuario
* Comentario -> Es una estructura de un comentario que se hace a una publicación o comentario

Y luego crear los predicados esenciales del programa:

* socialNetworkRegister -> Es el predicado que permite hacer un register
* socialNetworkLogin -> Es el predicado que permite hacer un login
* socialNetworkPost -> Es el predicado que permite hacer un post
* socialNetworkFollow -> Es el predicado que permite hacer un follow
* socialNetworkShare -> Es el predicado que permite hacer un share
* socialNetworkToString -> Es el predicado que permite mostrar el socialnetwork en string
* socialNetworkComment -> Es el predicado que permite hacer un comentario
* socialNetworkLike -> Es el predicado que permite hacer un like

Pero dentro de código se tendrá que usar obligadamente lo que es llamado recursión para poder recorrer listas, buscar cosas, hacer ciclos, entre otras razones. Y además el uso esencial de Patter-matching en la recursión.

**Diseño de la solución**

Se debe ver la representación a usar de los TDA’s que son los siguientes:

* Se tendrá una socialnetwork que contendrá:
* Nombre de la socialnetwork
* Fecha de creacion
* Lista de usuarios
* Lista de publicaciones
* Lista de comentarios
* Los usuarios contendrán:
* Un ID único
* Nombre
* Contraseña
* Actividad (Si se encuentra activo o inactivo)
* Lista de publicaciones mediante su ID
* Lista de publicaciones compartidas mediante su ID
* Fecha de creación
* Cantidad de Follows (Personas que sigue)
* Cantidad de Seguidores (Personas que lo siguen)
* Lista de Follows (A qué persona sigue)
* Lista de Seguidores (Que persona lo sigue)
* Las publicaciones contendrán:
* Un ID único
* Autor
* Fecha de la creación
* Tipo de la publicación (Sera “text” para ese caso)
* Contenido
* Lista de comentarios según su ID
* Cantidad de likes
* Cantidad de veces compartidas
* Los comentarios contendrán:
* Un ID único
* Autor
* Fecha de creacion
* Contenido
* Cantidad de likes
* Lista de comentarios según su ID

­Como se dijo anteriormente, se crearán funciones donde:

* register 🡪 Permite registrar un usuario
* login 🡪 Permite iniciar sesión
* post 🡪 Permite hacer una publicación en tu cuenta o en la de otros usuarios
* follow 🡪 Permite seguir a una persona
* share 🡪 Permite compartir una publicación a tu cuenta o a la de otros usuarios
* socialnetwork->string 🡪 Permite mostrar el contenido de toda la socialnetwork o de un usuario
* comment 🡪 Permite el comentar una publicación
* like 🡪 Permite dar like a una publicación o a un comentario

Aunque todos esos predicados son provenientes de login, exceptuando register. Por lo que primero habrá que hacer login y luego el predicado que se desea hacer.

Aunque en el 90% de esas funciones se edita la socialnetwork, por lo que se necesitará retornar una nueva socialnetwork con los nuevos datos y modificaciones necesarias a ella.

Y como también se trabaja con listas en la social, por ejemplo, las listas de publicaciones, se necesitará si o si la recursión para recorrer aquella lista

**Aspectos de implementación.**

Se usó la versión de SWI-Prolog 8.2.2 de 64 bits.

La estructura para el proyecto fue con unas subdivisiones del código en partes en las que iban los hechos y las reglas, dentro de las reglas hubo otras subdivisiones en las que fueron los modificadores, selectores, otras funciones, etc. Además de los predicados importantes como login, register, etc.

Debido a ello hizo complejo el trabajo en aquel trabajo por tener todo tan acoplado. Aunque fue la manera más ordenada que se encontró para trabajar en aquel.

**Instrucciones de uso:**

**Register**

socialNetwork("Feibo", [24,06,2021], SN), socialNetworkRegister(SN, [24,06,2020], "Viajador del tiempo", "xd", SN1).

socialNetwork("Feibo", [24,06,2021], SN), socialNetworkRegister(SN, [24,06,2020], "Viajador del tiempo", "xd", SN1), socialNetworkRegister(SN1, [25,06,2021], "Hacker", "Intento de hacker", SN2).

socialNetwork("Feibo", [24,06,2021], SN), socialNetworkRegister(SN, [24,06,2020], "Viajador del tiempo", "xd", SN1), socialNetworkRegister(SN1, [25,06,2021], "Hacker", "Intento de hacker", SN2), socialNetworkRegister(SN2, [25,06,2021], "User normal UwU", "Normalito UwU", SN3).

**Login**

socialNetwork("Feibo", [24,06,2021], SN), socialNetworkRegister(SN, [24,06,2020], "Viajador del tiempo", "xd", SN1), socialNetworkRegister(SN1, [25,06,2021], "Hacker", "Intento de hacker", SN2), socialNetworkRegister(SN2, [25,06,2021], "User normal UwU", "Normalito UwU", SN3), socialNetworkLogin(SN3, "Viajador del tiempo", "xd", SN4).

socialNetwork("Feibo", [24,06,2021], SN), socialNetworkRegister(SN, [24,06,2020], "Viajador del tiempo", "xd", SN1), socialNetworkRegister(SN1, [25,06,2021], "Hacker", "Intento de hacker", SN2), socialNetworkRegister(SN2, [25,06,2021], "User normal UwU", "Normalito UwU", SN3), socialNetworkLogin(SN3, "Hacker", "Intento de hacker", SN4).

socialNetwork("Feibo", [24,06,2021], SN), socialNetworkRegister(SN, [24,06,2020], "Viajador del tiempo", "xd", SN1), socialNetworkRegister(SN1, [25,06,2021], "Hacker", "Intento de hacker", SN2), socialNetworkRegister(SN2, [25,06,2021], "User normal UwU", "Normalito UwU", SN3), socialNetworkLogin(SN3, "User normal UwU", "Normalito UwU", SN4).

**Post**

socialNetwork("Feibo", [24,06,2021], SN), socialNetworkRegister(SN, [24,06,2020], "Viajador del tiempo", "xd", SN1), socialNetworkRegister(SN1, [25,06,2021], "Hacker", "Intento de hacker", SN2), socialNetworkRegister(SN2, [25,06,2021], "User normal UwU", "Normalito UwU", SN3), socialNetworkLogin(SN3, "Viajador del tiempo", "xd", SN4), socialNetworkPost(SN4, [29,06,2020], "Mensaje antes de que se cree todo", [], SN5).

socialNetwork("Feibo", [24,06,2021], SN), socialNetworkRegister(SN, [24,06,2020], "Viajador del tiempo", "xd", SN1), socialNetworkRegister(SN1, [25,06,2021], "Hacker", "Intento de hacker", SN2), socialNetworkRegister(SN2, [25,06,2021], "User normal UwU", "Normalito UwU", SN3), socialNetworkLogin(SN3, "Hacker", "Intento de hacker", SN4), socialNetworkPost(SN4, [32,06,2021], "Como que la fecha no coincide", ["User normal UwU", "Viajador del tiempo"], SN5).

socialNetwork("Feibo", [24,06,2021], SN), socialNetworkRegister(SN, [24,06,2020], "Viajador del tiempo", "xd", SN1), socialNetworkRegister(SN1, [25,06,2021], "Hacker", "Intento de hacker", SN2), socialNetworkRegister(SN2, [25,06,2021], "User normal UwU", "Normalito UwU", SN3), socialNetworkLogin(SN3, "Hacker", "Intento de hacker", SN4), socialNetworkPost(SN4, [32,06,2021], "Como que la fecha no coincide", [], SN5).

**Follow**

socialNetwork("Feibo", [24,06,2021], SN), socialNetworkRegister(SN, [24,06,2020], "Viajador del tiempo", "xd", SN1), socialNetworkRegister(SN1, [25,06,2021], "Hacker", "Intento de hacker", SN2), socialNetworkRegister(SN2, [25,06,2021], "User normal UwU", "Normalito UwU", SN3), socialNetworkLogin(SN3, "User normal UwU", "Normalito UwU", SN4), socialNetworkFollow(SN4, "Hacker", SN5).

socialNetwork("Feibo", [24,06,2021], SN), socialNetworkRegister(SN, [24,06,2020], "Viajador del tiempo", "xd", SN1), socialNetworkRegister(SN1, [25,06,2021], "Hacker", "Intento de hacker", SN2), socialNetworkRegister(SN2, [25,06,2021], "User normal UwU", "Normalito UwU", SN3), socialNetworkLogin(SN3, "User normal UwU", "Normalito UwU", SN4), socialNetworkFollow(SN4, "Viajador del tiempo", SN5).

socialNetwork("Feibo", [24,06,2021], SN), socialNetworkRegister(SN, [24,06,2020], "Viajador del tiempo", "xd", SN1), socialNetworkRegister(SN1, [25,06,2021], "Hacker", "Intento de hacker", SN2), socialNetworkRegister(SN2, [25,06,2021], "User normal UwU", "Normalito UwU", SN3), socialNetworkLogin(SN3, "Hacker", "Intento de hacker", SN4), socialNetworkFollow(SN4, "Viajador del tiempo", SN5).

**Share**

socialNetwork("Feibo", [24,06,2021], SN), socialNetworkRegister(SN, [24,06,2020], "Viajador del tiempo", "xd", SN1), socialNetworkRegister(SN1, [25,06,2021], "Hacker", "Intento de hacker", SN2), socialNetworkRegister(SN2, [25,06,2021], "User normal UwU", "Normalito UwU", SN3), socialNetworkLogin(SN3, "Viajador del tiempo", "xd", SN4), socialNetworkPost(SN4, [29,06,2020], "Mensaje antes de que se cree todo", [], SN5), socialNetworkLogin(SN5, "Viajador del tiempo", "xd", SN6), socialNetworkShare(SN6, [29,06,2020], 1, [], SN7).

socialNetwork("Feibo", [24,06,2021], SN), socialNetworkRegister(SN, [24,06,2020], "Viajador del tiempo", "xd", SN1), socialNetworkRegister(SN1, [25,06,2021], "Hacker", "Intento de hacker", SN2), socialNetworkRegister(SN2, [25,06,2021], "User normal UwU", "Normalito UwU", SN3), socialNetworkLogin(SN3, "Viajador del tiempo", "xd", SN4), socialNetworkPost(SN4, [29,06,2020], "Mensaje antes de que se cree todo", [], SN5), socialNetworkLogin(SN5, "Viajador del tiempo", "xd", SN6), socialNetworkShare(SN6, [29,06,2020], 1, ["Hacker"], SN7).

socialNetwork("Feibo", [24,06,2021], SN), socialNetworkRegister(SN, [24,06,2020], "Viajador del tiempo", "xd", SN1), socialNetworkRegister(SN1, [25,06,2021], "Hacker", "Intento de hacker", SN2), socialNetworkRegister(SN2, [25,06,2021], "User normal UwU", "Normalito UwU", SN3), socialNetworkLogin(SN3, "Viajador del tiempo", "xd", SN4), socialNetworkPost(SN4, [29,06,2020], "Mensaje antes de que se cree todo", [], SN5), socialNetworkLogin(SN5, "Viajador del tiempo", "xd", SN6), socialNetworkShare(SN6, [29,06,2020], 1, ["Hacker", "User normal UwU"], SN7).

**ToString**

socialNetwork("Feibo", [24,06,2021], SN), socialNetworkRegister(SN, [24,06,2020], "Viajador del tiempo", "xd", SN1), socialNetworkRegister(SN1, [25,06,2021], "Hacker", "Intento de hacker", SN2), socialNetworkRegister(SN2, [25,06,2021], "User normal UwU", "Normalito UwU", SN3), socialNetworkLogin(SN3, "Viajador del tiempo", "xd", SN4), socialNetworkPost(SN4, [29,06,2020], "Mensaje antes de que se cree todo", [], SN5), socialNetworkLogin(SN5, "Viajador del tiempo", "xd", SN6), socialNetworkShare(SN6, [29,06,2020], 1, ["Hacker", "User normal UwU"], SN7), socialNetworkToString(SN7,StringSalida).

socialNetwork("Feibo", [24,06,2021], SN), socialNetworkRegister(SN, [24,06,2020], "Viajador del tiempo", "xd", SN1), socialNetworkRegister(SN1, [25,06,2021], "Hacker", "Intento de hacker", SN2), socialNetworkRegister(SN2, [25,06,2021], "User normal UwU", "Normalito UwU", SN3), socialNetworkLogin(SN3, "Viajador del tiempo", "xd", SN4), socialNetworkPost(SN4, [29,06,2020], "Mensaje antes de que se cree todo", [], SN5), socialNetworkLogin(SN5, "Viajador del tiempo", "xd", SN6), socialNetworkShare(SN6, [29,06,2020], 1, ["Hacker", "User normal UwU"], SN7), socialNetworkLogin(SN7, "Viajador del tiempo", "xd", SN8), socialNetworkToString(SN8,StringSalida).

socialNetwork("Feibo", [24,06,2021], SN), socialNetworkRegister(SN, [24,06,2020], "Viajador del tiempo", "xd", SN1), socialNetworkRegister(SN1, [25,06,2021], "Hacker", "Intento de hacker", SN2), socialNetworkRegister(SN2, [25,06,2021], "User normal UwU", "Normalito UwU", SN3), socialNetworkLogin(SN3, "Viajador del tiempo", "xd", SN4), socialNetworkPost(SN4, [29,06,2020], "Mensaje antes de que se cree todo", [], SN5), socialNetworkLogin(SN5, "Viajador del tiempo", "xd", SN6), socialNetworkShare(SN6, [29,06,2020], 1, ["Hacker", "User normal UwU"], SN7), socialNetworkLogin(SN7, "Hacker", "Intento de hacker", SN8), socialNetworkToString(SN8,StringSalida).

**Comment**

socialNetwork("Feibo", [24,06,2021], SN), socialNetworkRegister(SN, [24,06,2020], "Viajador del tiempo", "xd", SN1), socialNetworkRegister(SN1, [25,06,2021], "Hacker", "Intento de hacker", SN2), socialNetworkRegister(SN2, [25,06,2021], "User normal UwU", "Normalito UwU", SN3), socialNetworkLogin(SN3, "Viajador del tiempo", "xd", SN4), socialNetworkPost(SN4, [29,06,2020], "Mensaje antes de que se cree todo", [], SN5), socialNetworkLogin(SN5, "Hacker", "Intento de hacker", SN6), socialNetworkComment(SN6, [33, 13, 2021], 1, 0, "Este es un comentario de un post", SN7).

socialNetwork("Feibo", [24,06,2021], SN), socialNetworkRegister(SN, [24,06,2020], "Viajador del tiempo", "xd", SN1), socialNetworkRegister(SN1, [25,06,2021], "Hacker", "Intento de hacker", SN2), socialNetworkRegister(SN2, [25,06,2021], "User normal UwU", "Normalito UwU", SN3), socialNetworkLogin(SN3, "Viajador del tiempo", "xd", SN4), socialNetworkPost(SN4, [29,06,2020], "Mensaje antes de que se cree todo", [], SN5), socialNetworkLogin(SN5, "Hacker", "Intento de hacker", SN6), socialNetworkComment(SN6, [33, 13, 2021], 1, 0, "Este es un comentario de un post", SN7), socialNetworkLogin(SN7, "Hacker", "Intento de hacker", SN8), socialNetworkComment(SN8, [33, 13, 2021], 1, 1, "Este es un comentario de un comentario", SN9).

socialNetwork("Feibo", [24,06,2021], SN), socialNetworkRegister(SN, [24,06,2020], "Viajador del tiempo", "xd", SN1), socialNetworkRegister(SN1, [25,06,2021], "Hacker", "Intento de hacker", SN2), socialNetworkRegister(SN2, [25,06,2021], "User normal UwU", "Normalito UwU", SN3), socialNetworkLogin(SN3, "Viajador del tiempo", "xd", SN4), socialNetworkPost(SN4, [29,06,2020], "Mensaje antes de que se cree todo", [], SN5), socialNetworkLogin(SN5, "User normal UwU", "Normalito UwU", SN6), socialNetworkComment(SN6, [20, 10, 2021], 1, 0, "Este es un comentario de un post", SN7).

**Like**

socialNetwork("Feibo", [24,06,2021], SN), socialNetworkRegister(SN, [24,06,2020], "Viajador del tiempo", "xd", SN1), socialNetworkRegister(SN1, [25,06,2021], "Hacker", "Intento de hacker", SN2), socialNetworkRegister(SN2, [25,06,2021], "User normal UwU", "Normalito UwU", SN3), socialNetworkLogin(SN3, "Hacker", "Intento de hacker", SN4), socialNetworkPost(SN4, [32,06,2021], "Como que la fecha no coincide", [], SN5), socialNetworkLogin(SN5, "User normal UwU", "Normalito UwU", SN6), socialNetworkLike(SN6, [29,10,2021], 1, 0, SN7).

socialNetwork("Feibo", [24,06,2021], SN), socialNetworkRegister(SN, [24,06,2020], "Viajador del tiempo", "xd", SN1), socialNetworkRegister(SN1, [25,06,2021], "Hacker", "Intento de hacker", SN2), socialNetworkRegister(SN2, [25,06,2021], "User normal UwU", "Normalito UwU", SN3), socialNetworkLogin(SN3, "Viajador del tiempo", "xd", SN4), socialNetworkPost(SN4, [29,06,2020], "Mensaje antes de que se cree todo", [], SN5), socialNetworkLogin(SN5, "Hacker", "Intento de hacker", SN6), socialNetworkComment(SN6, [33, 13, 2021], 1, 0, "Este es un comentario de un post", SN7), socialNetworkLogin(SN7, "Hacker", "Intento de hacker", SN8), socialNetworkComment(SN8, [33, 13, 2021], 1, 1, "Este es un comentario de un comentario", SN9), socialNetworkLogin(SN9, "User normal UwU", "Normalito UwU", SN10), socialNetworkLike(SN10, [29,10,2021], 1, 1, SN11).

socialNetwork("Feibo", [24,06,2021], SN), socialNetworkRegister(SN, [24,06,2020], "Viajador del tiempo", "xd", SN1), socialNetworkRegister(SN1, [25,06,2021], "Hacker", "Intento de hacker", SN2), socialNetworkRegister(SN2, [25,06,2021], "User normal UwU", "Normalito UwU", SN3), socialNetworkLogin(SN3, "Hacker", "Intento de hacker", SN4), socialNetworkPost(SN4, [32,06,2021], "Como que la fecha no coincide", [], SN5), socialNetworkLogin(SN5, "Hacker", "Intento de hacker", SN6), socialNetworkLike(SN6, [-1,-0,2021], 1, 0, SN7).

**Resultados**

Los predicados abordados fueron completamente terminados haciendo que logre su función principal y como llegar a ese resultado.

Aunque existen momentos de errores cuando no se respeta el formato de escritura en los parámetros de los predicados como por ejemplo en el register al usar átomos en lugar de string

Predicado mal escrita

socialNetworkRegister(SN, [10,10,2010], user1, pass1, SNSalida).

Predicado bien escrita

socialNetworkRegister(SN, [10,10,2010], “user1”, “pass1”, SNSalida).

Y para corroborar si los predicados fueron bien hechos, comprobarlo con los ejemplos puestos en “Instrucciones de uso” son más que suficiente.

Por lo que se terminó en un 100% los requerimientos y funcionamiento de los predicados obligatorios y opcionales.

**Conclusión**

El paradigma fue difícil de abordar por el hecho de que no se puede almacenar los datos y modificar, además el sistema muy cerrado que tiene prolog hace un poco más difícil trabajarlo, como también su interfaz de trabajo. Pero está claro que estos predicados son para tener un resultado booleano, por lo que no es necesaria el almacenar los datos, sino que solo basta con el resultado simulado que da y si puede llegar al resultado.

Comparado al paradigma anterior, el paradigma funcional, fue más sencillo de trabajar porque no era tan lineal y además no existe la currificación. Algo que fue muy difícil de entender y trabajar a su vez.

**Referencia**

Uqbar Foundation. (s. f.). *Paradigma logico*. uqbar-wiki. Recuperado 5 de diciembre de 2020, de https://wiki.uqbar.org/wiki/articles/paradigma-logico.html