UNIVERSIDAD DE SANTIAGO DE CHILE
FACULTAD DE INGENIERÍA
DEPARTAMENTO DE INGENIERÍA INFORMÁTICA

Paradigmas de Programación

Laboratorio N°2

Alumno: Cristhofer Parada Salazar
sección: B-2.
Profesor: Víctor Flores Sánchez

Santiago - Chile
1-2021

TABLA DE CONTENIDOS

TABLA DE CONTENIDOS

CAPITULO 1. Introducción

El objetivo del presente informe corresponde a mostrar al lector detalles relacionados al Laboratorio N°2 del curso de Paradigmas de Programación. Estos detalles pueden ser resumidos como descripción del problema, donde se conocerá el contexto que hizo que el laboratorio se llevase a cabo, cómo se resolvió el problema y dar paso a explicar las distintas estructuras de datos o TDAs que fueron utilizados en la implementación de la solución, además de dar paso a la explicación de cómo se debe utilizar este programa y en que consiste Prolog, que es una de las herramientas utilizadas para la solución.

1.1.1 Descripción del problema

A los alumnos del curso de Paradigmas de Programación, se les requirió la creación de una red social simple, la cual debía contener funciones básicas dentro de cualquier red social, como crearse una cuenta de usuario, crear publicaciones, dar reacciones a publicaciones, tener followers, etc. Para poder realizar esto, se les solicito a los alumnos que el programa debiese estar escrito en el lenguaje de Prolog y hacer usos de TDAs.

1.1.2 Descripción del paradigma utilizado

En este laboratorio específico, fue solicitado que se utilizase el Paradigma Lógico para la solución del problema. El paradigma lógico posee características bastante representativas, entre las cuales podemos encontrar una base de conocimiento, a la cual se le hace consultas. Esta base de conocimientos está conformada por hechos, que es la información del sistema relacionada entre datos y reglas lógicas. No es necesario que la base de conocimientos contenga absolutamente toda la información, sino que solamente aquella que es necesaria para que el programa pueda deducir lo que necesite.

CAPÍTULO 2. Solución del problema

2.1.1 Análisis del problema con respecto a sus requisitos específicos

Para la creación de la solución, fueron solicitados los siguientes requisitos:

- Creación de TDAs apropiados que sirvan para moldear de la mejor manera el problema, además, el TDA SocialNetwork será en lo que estará enfocado completamente el programa.
- socialNetworkRegister: Función que permite registrar a un usuario dentro de una red social ya existente, el retorno de esta función es true si el usuario que se quiere registrar no existe previamente en la red social y dicha red social es actualizada en la salida RSout.
- socialNetworkLogin: Función que posibilita a un usuario preexistente en la red social conectarse. Si un usuario quiere realizar algún cambio dentro de la red social, debe primero haber pasado por esta función, siempre.
- socialNetworkPost: Función que permite crear una publicación con autoría del usuario que se encuentra en línea en ese momento, retorna si hay un usuario en línea y la red social actualizada queda guardada en RSout.
- socialNetworkFollow: Función que permite que el usuario que se encuentre online pueda darle “follow” a otro usuario que exista en la red social.
- socialNetworkShare: Función que permite al usuario conectado en la red social compartir una publicación.
- socialNetworkToString: Función que genera un string con la información relacionada a la red social o referente al usuario que se encuentre online en ese momento.

Además de las funciones anteriores a los alumnos del curso se les da la libertad de implementar otras funciones adicionales, de dichas funciones adicionales, se han seleccionado las siguientes:

- socialNetworkComment: Función que logra que un usuario con la sesión iniciada en la red social pueda comentar una publicación o un comentario de una publicación.
- socialNetworkLike: Función que posibilita a un usuario con la sesión iniciada reaccionar con un like a una publicación.

2.1.2 Diseño de la solución.

Para la implementación de este laboratorio, se utilizaron distintos tipos de TDAs que sirvieron para modelar el problema y hacer las cosas más sencillas y metódicas. Dichos TDAs serán explicados a continuación:

TDA Fecha
Corresponde a la representación de una fecha como una lista de 3 enteros siguiendo el formato de DD/MM/YYYY, se espera que, para su correcto uso, el usuario siga este formato, de otro modo el programa puede arrojar errores.

TDA User
Representación que corresponde a un usuario dentro de la red social, dentro de esta lista podemos encontrar su ID, su nombre de usuario, su contraseña, la fecha en la cual se creó la cuenta (representado por el TDA Fecha), una lista con las IDs de los demás usuarios que siguen a la cuenta y una lista con las publicaciones/comentarios hechos por el usuario o que han sido compartidas por él.

TDA Publicación
TDA perteneciente a una publicación, la cual comparte el formato con los comentarios, es decir, dentro de esta red social tanto publicaciones como comentarios son exactamente lo mismo ¿Cómo diferenciarlos? Un comentario siempre estará asociado a una publicación original, por lo que de esa forma se pueden hacer búsquedas de comentarios. Este TDA esta compuesto por una ID única para cada publicación, el contenido de la publicación como un texto (string), una lista con IDs de usuarios correspondiente a las personas que han reaccionado a dicha publicación, una lista con IDs de publicaciones que pertenecen a comentarios hechos en dicha publicación, la fecha de creación de la publicación, una lista con los nombres de usuario los cuales han sido etiquetados por el autor y finalmente una lista de IDs con los usuarios que han compartido dicha publicación.

TDA Social Network
TDA esencial en el programa, en el se realizan todas las modificaciones que el User quiera (y pueda). Este TDA este compuesto por un string correspondiente al nombre de la red social, una fecha perteneciente a la fecha de creación de la red social, una lista con todos los usuarios que se han registrado en la red social, una lista con todas las publicaciones hechas en la red social y finalmente un numero correspondiente al usuario conectado en la red social, si dicho numero es equivalente a 0, quiere decir que no hay ningún usuario online en dicha red social en el momento. Para demostración, se creará una red social vacía:

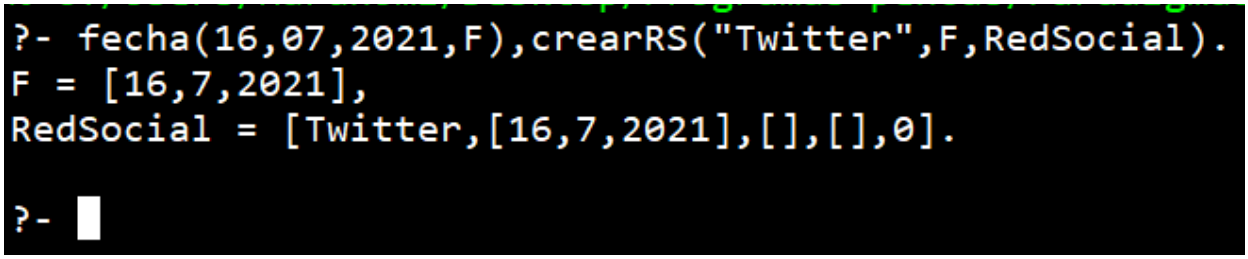

Figura 1: Creación de una red social con 0 usuarios registrados, 0 publicaciones y ningún usuario en línea (red social vacía).

De esta forma se puede crear una fecha, la cual fue guardada en la variable F, y podemos ocupar dicha variable como parámetro en la función crearRS, la cual tiene como entrada un string que pertenece al nombre de la social network, una fecha que fue provista con anterioridad y una variable que será entregada de vuelta como una red social. Con la red social creada, se pueden comenzar a registrar usuarios. La funcion socialNetworkRegister es la encargada de crear usuarios y posee el uso de una función auxiliar, la cual permite la búsqueda de usuarios dentro del TDA Red Social, esto, con el fin de que no existan usuarios duplicados (con el mismo nombre). Dentro del lenguaje de Prolog se utiliza la técnica de funciones recursivas, las cuales hacen el llamado de si mismas para generar ciclos. A continuación, se mostrara un ejemplo de una función recursiva en Prolog:

```
% Dominio: lista x number x var
% Descripcion: permite saber si existe una publicacion dada su id en una lista de publicaciones
id_to_publicacion([H|_],ID,Retorno):-
    getPublicacionID(H,ID2),
    (ID2 = ID),
    H = Retorno.

id_to_publicacion([_|T],ID,Retorno):-
    id_to_publicacion(T,ID,Retorno).
```

Figura 2: Función recursiva que dada una ID retorna una publicación de una lista de publicaciones.

Como bien se puede ver, la función anterior se divide en 2 partes. Dado que este lenguaje utiliza la lógica matemática como base, siempre busca que los predicados de las funciones puedan llegar a ser verdaderos, por lo que si hay 2 predicados que son aparentemente iguales, revisa ambos, hasta que pruebe que sea verdadero o no pueda probar la veracidad y retorne false. En la primera parte de la función, podemos ver que la lista inicial que corresponde a la lista de publicaciones es separada en Cola y Cabeza, para esta primera parte, solo necesitaremos la Cabeza de la lista o “la primera publicación de la lista”, se revisa si el ID de dicha publicación corresponde a la buscada y de ser así, se retorna la publicación. Si el predicado anterior llegase a no ser verdadero, prolog consultará la segunda parte, en importa la cola de las publicaciones, es decir, todas excepto la primera, y de esta forma se logra realizar una recursión y consultar, en este caso, si existe el ID entregado dentro de la lista de publicaciones.

2.1.3 Aspectos de la implementación

El programa fue escrito en el lenguaje de Prolog, el cual fue provisto por SWI-Prolog en su versión 8.2.4. No se hicieron uso de librerías y solamente se utilizaron predicados nativos del lenguaje de programación de Prolog.

Al abrir el código fuente "main_203877544_CristhoferParada.pl" se puede ver que se encuentra organizado de la siguiente forma:

- TDA Fecha
- TDA User
- TDA Publicacion
- TDA Social Network
- Funciones Auxiliares
- Predicados requeridos.

2.1.4 Instrucciones de uso

A continuación, se presentarán las instrucciones básicas para poder usar el programa correctamente:

- Abrir el programa SWI-Prolog y antes de consultar por el archivo "main_20387754_CristhoferParada.pl" se recomienda hacer uso del siguiente comando:
  set_prolog_flag(answer_write_options,[max_depth(0)]),use_module(library(theme/dark)),write('\e[2J').
  El cual tiene como finalidad que las listas se puedan ver en su totalidad, además de cambiar la paleta de colores de la consola (puesto que el blanco a ciertos usuarios daña la vista).
- Dentro del archivo consultado, se pueden ver ejemplos de uso, los cuales, además, se encuentran en el inicio de cada función. Se recomienda revisar estos predicados de ejemplo con cautela, puesto que un mal uso de ellos puede generar errores que no son culpa del programa en sí, sino de la forma en que se está utilizando.
- Un pequeño ejemplo de uso del programa puede ser el siguiente:
  `fecha(15,7,2021,Fecha),crearRS("Twitter",Fecha,RedSocial),socialNetworkRegister(RedSocial1,Fecha, "Usuario", "SecretPassword", RedSocial2).`
  Este comando puede ser copiado y pegado en la consola de SWI-Prolog (Siempre asegurándose que se este consultando el archivo principal.

2.2 Resultados y autoevaluación.

En la siguiente sección, se revisarán los resultados obtenidos, para ellos fueron revisados los requisitos funcionales y no funcionales:

Lenguaje de programación Prolog	Logrado: el lenguaje usado en el programa es scheme.
versión: Usar SWI-Prolog 8.x.x	Logrado: La versión usada es la 8.2.4.
documentación de funciones.	Logrado: Todas las funciones cuentan con su encabezado que corresponde a su dominio y descripción de la funcion, además de ejemplos.
Dom->Rec	Logrado: Se respecta las entradas de las funciones, sin efectos colaterales.
Organización	Logrado: cada TDA tiene su pequeña sección dentro del código, permitiendo una buena lectura.
Historial: al menos 10 commits en un mínimo de 2 semanas, con un mínimo de 10 commits	Logrado: el primer commit fue el día 4 de junio de 2021, el ultimo commit fue hecho el 16 de julio de 2021, con un total de 12 commits.
Ejemplos:	Logrado: Cada función principal cuenta con mínimo de 3 ejemplos.
Prerequisitos de funciones	Logrado: cada función tiene su prerrequisito implementado como se solicitó.

Requerimientos Funcionales:

Todas las funciones cumplen con los requerimientos funcionales exigidos por la coordinación, por lo que este punto fue respetado en su totalidad.

2.3 Conclusión

Analizando los datos obtenidos en el punto 2.2, puedo decir que estoy bastante conforme con el resultado de la implementación del programa. Los TDAs fueron lo más sencillo de hacer, puesto que con la sobriedad que resulta el programar con prolog, las funciones no requerían tanto código, puesto que más allá de los selectores no se podía hacer mucho dentro de los TDA. Algo que ayudó bastante fue el hecho de aprender a manejar las listas, tanto como recorrerlas, modificarlas y comparar, puesto que en base a esas funciones se comenzó a armar el programa y los predicados funcionales requeridos.

Cosas para mejorar a futuro puede ser un mejor manejo de los tiempos en los cuales se hizo este programa, puesto que creo que no se le dedico el tiempo adecuado. El lenguaje en un principio es algo complejo de entender, pero una vez que llegas a crear funciones de búsqueda de listas, se te soluciona bastante las cosas, por lo que en ese sentido me hace sentir mal el hecho de no haberle prestado atención al lenguaje cuando debí hacerlo y más bien comenzar a prestarle atención cuando ya estábamos “presionados” por la entrega.