Informe Laboratorio 2

“Paradigma Lógico”

Asignatura Paradigmas de la Programación

Nombre: Agustín Henríquez Rojas

Rut: 20.382.184-0

Profesor: Roberto González

Índice

Introducción

Breve descripción del problema

Descripción del Paradigma y conceptos aplicados en el proyecto

Análisis del problema respecto de los requisitos que se deben cubrir

Diseño de la solución

Aspectos de la implementación

Instrucciones de uso

Resultados y autoevaluación

Conclusiones del proyecto

Referencias

Introducción

El siguiente informe muestra el desarrollo y el resultado del proyecto de simular un sistema de redes sociales, que es una implementación generalizada de una Social Network, pero se centra en la modelación de un paradigma Lógico.

En este informe se presentará una descripción del problema que se tuvo que abordar para lograr la implementación, también se incluyen los conceptos teóricos. De igual manera se realizó un análisis del problema junto con sus requisitos a cubrir, y una solución para este, describiendo sus aspectos de implementación e instrucciones de uso. Concluyendo el informe con los resultados obtenidos de la solución implementada junto a una autoevaluación, y finalmente las conclusiones obtenidas del proyecto.

Breve descripción del problema

El problema que se debe resolver desde el enfoque del Paradigma Lógico consiste en realizar la implementación como una simulación de una Social Network, es decir, una red social que permita registrarse, iniciar sesión, realizar un post, compartir las publicaciones con otros usuarios, además de poder seguir sus perfiles, etc.

Descripción del paradigma y conceptos aplicados en el proyecto

Se utilizó el concepto de Tipo de Dato Abstracto (TDA) para la realización del siguiente proyecto. Se puede utilizar TDA para especificar casi cualquier cosa que deba integrarse en una solución de software. los niveles de implementación pueden ser tanto una representación, constructores, pertenencia, selectores, modificadores, como otras funciones.

En este proyecto se aplicó el concepto de TDA para realizar una abstracción de los elementos de una Social Network, contando con una representación, constructores, pertenencia, selectores, etc.

Por otro lado, en el caso del Paradigma Lógico, este cuenta con una gran ventaja, pues al ser un paradigma declarativo, resulta más fácil pensar en la solución del problema en lugar de los detalles procedurales para llegar a esta solución. Además, en este paradigma se describe lo que es una Base de Conocimientos con Hechos y Reglas, y luego se pueden realizar consultas sobre esta.

Para este proyecto se utilizó el lenguaje de programación lógico “Prolog”, el cual se basa en 3 mecanismos lógicos: Unificación, Back tracking automático y Estructuras de datos basadas en árboles.

Además de todo lo descrito anteriormente, entre los conceptos más destacables del Paradigma Lógico, y que fueron utilizados para la implementación de los predicados necesarios para lograr los requerimientos obligatorios, se puede mencionar el uso de hechos, reglas, metas, los diferentes tipos de datos de Prolog (átomos, números, variables, estructuras), también se utilizó la unificación, la sustitución, el back tracking, la negación, listas, relaciones recursivas.

Análisis del problema respecto de los requisitos que se deben cubrir

Empezando el proyecto, el primer requisito que había que abordar era sobre una buena abstracción de elementos adecuada para el problema. Para esto se debía implementar una cantidad de TDAs, incluyendo constructores, selectores, modificadores, etc. Todos los TDAs siendo, además, estructuras basadas en listas.
Primero se realizó una estructura base, TDA SocialNetwork, la que corresponde al contenedor de usuarios, publicaciones y el usuario que esta con la sesión activa. También se hizo un TDA Usuario, el cual almacena sus datos de acceso, seguidores, una lista con IDs de las publicaciones que ha hecho, y una lista con IDs de las publicaciones que ha compartido.

El TDA Usuario estaría representados de la siguiente forma.

USUARIO => [Nombre, Contraseña, Fecha, Seguidores, Seguidos, Id publicaciones, post compartidos hacia otros, post compartidos para el usuario]

Desarrollado en Porlog.

USUARIO => [str x str x list x list x list x list x list]

Para el caso de una publicación, se hizo una representación mediante un TDA que contiene el Id de la publicación, el contenido, el autor y la fecha de la publicación. Un ejemplo en Prolog es el siguiente:

PUBLICACION => [Id post, Contenido post, Autor post, Fecha post]

Desarrollado en Porlog.

PUBLICACION => [int x str x str x list]

Finalmente, la SocialNetwork, estaría compuesta por una lista de usuarios, una lista de publicaciones, una lista con el usuario activo, el nombre de la red social, y la fecha de creación de la socialnetwork. Quedaría de la siguiente manera:

SOCIALNETWORK => [lista de usuarios, lista de publicaciones, Usuario activo, Nombre de la socialnetwork, fecha de creación]

SOCIALNETWORK => [[Usuario1, Usuario2, ...], [Post1, Post2, ...], [Usuario_activo], Name, Date]

Diseño de la solución

Para poder desarrollar una solución eficaz se implementarán varios TDAs; TDA socialnetwork, TDA usuario, TDA publicaciones, TDA fecha. La realización de estos TDAs ayuda a resolver de manera organizada y a la misma vez cumplir con los requisitos funcionales a medida que se avanza con la solución.

Comenzando con el predicado socialNetworkRegister, permite consultar el valor que torna socialnetwork al realizar un registro de un nuevo usuario. Esta requerirá de parámetros como el socialnetwork, un nombre, una contraseña, una fecha, y un SocialNetwork2. y comienza consultando si hay un usuario activo en la SocialNetwork, en caso de haber uno, entonces la consulta arroja falso de inmediato. En caso de no haber un usuario activo, entonces se verifica que SocialNetwork corresponda a uno, y que Nombre y Contraseña sean strings, si se cumple entonces la consulta pasa a verificar que Nombre se encuentre disponible para registrar, luego se actualiza la lista de Usuarios, retornando en la variable SocialNetwork2 el valor del nuevo SocialNetwork generado.

La siguiente consulta requerida, predicado socialNetworkLogin, permite autenticar a un usuario registrado, y pide como parámetros las variables SocialNetwork, Nombre, Contraseña y un SocialNetwork2, comienza consultando si hay un usuario activo en SocialNetwork, en caso de haber uno, entonces la consulta arroja falso de inmediato. En caso de no haber un usuario activo, entonces se verifica que SocialNetwork corresponda a uno, y que Nombre y Contraseña sean strings, si se cumple entonces la consulta pasa a verificar que el Usuario se encuentre registrado, si todo se cumple, entonces se mueve el usuario seleccionado a la lista del Usuario_activo, retornando en la variable SocialNetwork2 el valor del nuevo SocialNetwork generado.

Para el caso de la consulta socialNetworkPost, permite a un usuario con sesión iniciada en la SocialNetwork realizar una nueva publicación. Pide como parámetros las variables SocialNetwork, Fecha, Texto un SocialNetwork2 y comienza consultando si hay un usuario activo en SocialNetwork, en caso de no haber uno, entonces la consulta arroja falso de inmediato. En caso de sí haber un usuario activo, entonces se verificará que SocialNetwork corresponda a uno, que Fecha corresponda una lista de enteros y también que Texto sea un string, si se cumple todo entonces la consulta pasa a crear la nueva publicación, y actualizará la lista de IDs de las publicaciones que ha realizado el usuario que se encuentra activo, retornando en la variable SocialNetwork2 el valor del nuevo SocialNetwork generado.

Para la consulta socialNetworkFollow, esta permite consultar el valor que toma SocialNetwork al realizar un follow a otro usuario. Pide como parámetros las variables SocialNetwork, Nombre de un usuario y un SocialNetwork2, y comienza consultando si hay un usuario activo, en caso de no haber uno, entonces la consulta arroja falso de inmediato. En caso de sí haber un usuario activo, se verificará que SocialNetwork corresponda a uno y también que Nombre sea un string. Si se cumple todo, entonces la

consulta pasa a crear al nuevo seguidor, y actualiza la lista de seguidores del usuario junto con la lista de seguidos del usuario.

Para la consulta socialNetworkShare, permite consultar el valor que toma SocialNetwork al momento en que un usuario comparte una publicación a otro usuario. Pide como parámetros las variables SocialNetwork, Fecha, ID de la publicación, Nombre de usuario a compartir y un SocialNetwork2, y comienza consultando si hay un usuario activo en SocialNetwork, en caso de no haber uno, entonces la consulta arroja falso de inmediato. En caso de sí haber un usuario activo, entonces se verificará que SocialNetwork corresponda a uno, que Nombre sea un string, que Fecha corresponda a una lista de enteros y que el ID sea un entero. Si las condiciones se cumplen, la consulta pasa a crear la nueva publicación compartida al usuario destinatario.

Finalmente, para la consulta socialNetworkToString, permite consultar el valor que toma un string al intentar representar en éste un socialnetwork. Pide como parámetros las variables SocialNetwork y un SocialNetwork2, y comienza consultando si hay un usuario activo en SocialNetwork, ya que dependiendo de si hay o no un usuario activo es el valor que tomará SocialNetwork2.

Aspectos de la implementación

Para la realización de este proyecto se utilizó la implementación en código abierto de Prolog llamada SWI-Prolog, versión 8.2.2. y como editor de texto se utilizó Visual Studio Code.

Instrucciones de uso

Para registrarse en la red social se utiliza la consulta socialNetworkRegister, la que requiere la socialnetwork a trabajar, el nombre de usuario, la contraseña, la fecha y la variable (SocialNetwork2) que tomará el valor de la socialnetwork resultante. Si todo fue ingresado correctamente, entonces debería retornar el valor de la variable como socialnetwork actualizado con el nuevo usuario registrado. Si al llamar esta consulta de nuevo se le ingresara el valor de la variable, esta retornará true.

Para iniciar sesión en la red social sirve la consulta socialNetworkLogin, la que requiere el socialnetwork a trabajar, el nombre de usuario, la contraseña y la variable (SocialNetwork2) que tomará el valor del socialnetwork resultante. Si todo fue ingresado correctamente, entonces se debería retornar el valor de la variable como el socialnetwork actualizado con el usuario activo. Si al llamar esta consulta de nuevo se le ingresara el valor de la variable, esta retornará true.

Para realizar una publicación en la red social sirve la consulta socialNetworkPost, la que necesita el SocialNetwork a trabajar, la fecha de la publicación, el contenido de la publicación(texto) y la variable (SocialNetwork2) que tomará el valor del SocialNetwork resultante. Si todo fue ingresado correctamente, entonces se debería retornar el valor de la variable como el socialnetwork actualizado con la nueva publicación y la lista de ids de publicaciones actualizada del usuario activo. Si al llamar esta consulta de nuevo se le ingresara el valor de la variable, esta retornará true.

Para seguir a un usuario sirve la consulta socialNetworkFollow, la que requiere el SocialNetwork a trabajar, el Nombre del usuario a seguir y la variable (SocialNetwork2) que tomará el valor del SocialNetwork resultante. Si todo fue ingresado correctamente, entonces se debería retornar el valor de la variable como el SocialNetwork actualizado con el nuevo seguidor y la lista de seguidos actualizada del usuario activo, junto con la lista de seguidores del usuario seguido. Si al llamar esta consulta de nuevo se le ingresara el valor de la variable, esta retornará true.

Para compartir una publicación a otro usuario sirve la consulta socialNetworkShare, la que requiere el SocialNetwork a trabajar, fecha, el id de la publicación a compartir, el nombre de usuario destinatario y la variable (SocialNetwork2) que tomará el valor del SocialNetwork resultante. Si todo fue ingresado correctamente, entonces se debería retornar el valor de la variable como el SocialNetwork actualizado con la nueva publicación compartida y la lista de ids de publicaciones compartidas del usuario activo, junto con la lista de ids de publicaciones que le fueron compartidas al usuario destinatario. Si al llamar esta consulta de nuevo se le ingresara el valor de la variable, esta retornará true.

Finalmente, si se desea visualizar de forma comprensible y mucho más cómoda el socialnetwork, se puede hacer uso de la consulta socialNetworkToString, la que sirve como una “interfaz” para el socialnetwork, mostrándolo de una forma más visual. Esta requiere el SocialNetwork a trabajar y la variable (SocialNetwork2) que tomará el valor del SocialNetwork resultante. Dependiendo de si hay un usuario activo o no es si muestra

todo el SocialNetwork o solo los datos del usuario activo junto a sus publicaciones. Si al llamar esta consulta de nuevo se le ingresara el valor de la variable, esta retornará true.

Resultados y autoevaluación

Para este proyecto había varios requerimientos que debían cumplirse, habiendo requerimientos funcionales y no funcionales.

0	No realizado
0.25	Implementación con problemas mayores
0.5	Implementación con funcionamiento irregular
0.75	Implementación con problemas menores
1	Implementación completa sin problemas

Requerimientos no Funcionales	Evaluación
Autoevaluación	1
Lenguaje	1
Versión	1
Documentación	1
Historial	1
Ejemplos	1
Prerrequisitos	1

Requerimientos Funcionales	Evaluación
TDAs	1
Hechos	1
Register	1
Login	1
Post	0.75
Follow	1
Share	0.75
SocialnetworkToString	1
Comment	0
Like	0
Viral	0
Search	0

A partir de las tablas, se puede decir que en los requerimientos no funcionales no hubo problemas en la implementación ni en las pruebas, sin embargo, en el caso de los requerimientos funcionales, en las consultas post y share no se pudo hacer una implementación completa, ya que se pedía que tuvieran como uno de sus parámetros de entrada una lista con destinatarios, pero solo se pudo implementar con 1 solo destinatario a la vez.

Conclusiones del proyecto

Como conclusión podría partir mencionando que el proyecto fue un poco más sencillo de abordar con respecto al laboratorio en scheme, ya que ahora se comprende mejor el manejo de estructura basadas en listas.

Con respecto al objetivo del proyecto, se puede decir que en parte se logró crear una simulación de una red social con un paradigma lógico en el cual se pudo trabajar con abstracciones de datos, y comprender bien su utilidad.

Lo negativo que se podría mencionar de este laboratorio con paradigma lógico es que al igual que el paradigma funcional, la comprensión del funcionamiento del lenguaje es poco intuitivo para un usuario. Quizás el programa es poco práctico ya que para cada consulta en la entrada hay que estar copiando y pegando el socialnetwork completo.

Referencias

https://www.swi-prolog.org/pldoc/man?section=syntax

https://www.swi-prolog.org/pldoc/man?section=string-predicates