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

INFORME LABORATORIO2

Paradigma Lógico
Paradigmas de programación

Dyllan Salgado

Profesor: Víctor Flores.
Fecha de entrega:06-12-2020

Santiago - Chile
2020

Tabla de contenidos

Tabla de Figuras

1.INTRODUCCIÓN

A través de la historia se han presentado una gran cantidad de problemáticas, la cual gracias a la ciencia computacional se han podido resolver a través de distintas abstracciones, generando diferentes soluciones para un mismo problema. A estas soluciones las denominaremos algoritmos, ya que son un método preciso para poder resolver un determinado problema, estos algoritmos serán creados en distintos lenguajes de programación con el paso del tiempo. En el informe se presenta una de las posibles soluciones a la problemática que se nos ha entregado, el cual será resuelto en el paradigma lógico y en el lenguaje de programación Prolog. La problemática consiste en la simulación de Stack overflow, pero ¿Qué es Stack Overflow? [1].

Stack overflow es un foro que construye una base de conocimientos sobre programación. Nos permite realizar preguntas y respuestas sobre problemáticas que tengamos al momento de programar o en lenguajes de programación en específico. Este foro está dirigido para programadores aficionados y profesionales. Fue creado en el año 2008 por Jeff Atwood y Joel Spolsky.

El objetivo principal de este laboratorio consiste en simular el foro stack overflow con requisitos que nos plantea la problemática en el paradigma Lógico.

Para lograr el objetivo principal debemos tener en consideración los objetivos específicos, los cuales son:

- Leer y comprender el enunciado.
- Creación de un TDA correcto que será utilizado para realizar el algoritmo.
- Realizar hechos y predicados que trabajen con el tda.
- Explicar entradas y salidas de las funciones.
- Ordenar y organizar el código para que sea más sencillo de comprender.

2. DESCRIPCIÓN BREVE DEL PROBLEMA

En la asignatura de Paradigmas de Programación a los estudiantes se les ha presentado la tarea de realizar el Laboratorio Número dos, el cual consistirá en una simulación del foro Stack Overflow con el paradigma Lógico, el cual será implementado a través del lenguaje Prolog.

El objetivo principal del foro Stack Overflow, es permitir plantear preguntas que luego pueden ser respondidas por otros usuarios, consta de cinco elementos claves para su funcionamiento, los cuales son:

- **Etiquetas:** Son palabras claves que sirven para categorizar una pregunta, nos permite agrupar preguntas que tengan relación entre sí. Además, cada etiqueta posee una descripción que el usuario puede visualizar al situar el puntero sobre ella, permitiendo ver preguntas y respuestas que tengan relación con ella.
- **Preguntas:** Las preguntas son publicadas por el usuario, las cuales vienen siendo las dudas que este tenga. Las preguntas tienen ciertas características: etiquetas, fecha de publicación, autor, estado, respuestas, votos, visualizaciones, entre otras.
- **Repuestas:** Las respuestas son realizadas por otros usuarios, y permite resolver la duda de la pregunta que ha sido ingresada por un usuario. Las características de las respuestas son: tienen ID propio, que nos permite identificar las respuestas, fecha, ID de la pregunta que ha sido respondida, la respuesta, etiquetas, autor.

- **Usuarios y reputación:** Los usuarios son quienes plantean y responden a preguntas, pero para asegurar la calidad de dichas respuestas el foro nos entrega un sistema de calificación o reputación para usuarios. De esta manera si un usuario tiene alta reputación es porque constantemente participa respondiendo preguntas a usuarios los cuales se sienten satisfechos por la respuesta entregada, generando una mayor fiabilidad a su respuesta.

El problema que se presenta en el laboratorio es simular parte de las características que tiene Stack Overflow y poder ser llevado a cabo por medio de la programación lógica en el lenguaje Prolog.

3. DESCRIPCIÓN BREVE DEL PARADIGMA

¿Qué es el Paradigma lógico?
El paradigma lógico está basado en la lógica de primer orden, estudia el uso de la lógica para el planteamiento de problemas. Este paradigma al igual que el paradigma funcional forman parte de los paradigmas declarativos, el cual se enfoca más en el cómo resolver el problema, este problema se trata de resolver mediante predicados y la base de dato para realizar las consultas.[2].

¿Qué es Prolog?
La palabra Prolog proviene del francés programmation en logique, es un lenguaje de programación lógico e interpretado usado habitualmente en el campo de la Inteligencia artificial, está basado en el paradigma lógico. Este lenguaje de programación se enfoca en resolver los problemas mediante la relación entre objetos, y emplea las principales características del paradigma lógico como backtracking, el cual nos permite realizar una búsqueda sistemática a través de todas las configuraciones posibles dentro de una base de conocimientos y unificaciones, la cual nos permite realizar asignaciones de variables para que haga idénticas a variables que se desean unificar. [2]

Características del paradigma lógico
Algunas características del paradigma lógico son los siguientes:

- Los programas para los lenguajes de programación lógicos son conjuntos de hechos y reglas.
- Unificación de términos.
- Recursión como estructura de control básica.
- La aplicación de las reglas de la lógica para inferir conclusiones a partir de datos.
- No tiene un algoritmo que indique los pasos que detallen la manera de llegar a un resultado.
- Visión lógica de la computación.

4. ANÁLISIS DEL PROBLEMA RESPECTO DE LOS REQUISITOS ESPECÍFICOS QUE DEBEN CUBRIR

Para poder llevar a cabo un informe y algoritmo óptimo, se debió seguir una serie de requisitos, ya sean de tipo funcionales y no funcionales.

Requisitos funcionales

La problemática planteada en este laboratorio consiste en crear una simulación del foro Stack Overflow, en donde un usuario se puede registrar, iniciar sesión, formular preguntas, formular respuestas, aceptar respuestas a preguntas que haya realizado, etc. Estas acciones el usuario las podrá realizar a través de consultas dentro del intérprete de Prolog.

El primer requerimiento funcional que se presenta, es la creación de un Tipo de dato abstracto que represente a un Stack Overflow, el cual contendrá a lista de registrados, lista de preguntas, listas de respuestas y una lista donde se encuentra el usuario logeado. Además, debemos crear predicados de tipo constructor, selectores, modificadores y predicados adicionales para un funcionamiento correcto.

Al tener definido un Tda, el usuario podrá ser capaz de manipular e ingresar elementos al stack overflow. En el tda se efectuarán todas las acciones que el usuario necesite, con esto me refiero a que las funciones no servirán si el tda está mal definido. De igual manera para que el usuario manipule el algoritmo o código necesitara predicados específicos que más adelante se presentaran.

Algunas de las funcionalidades que se han creado con los predicados y que el usuario puede ocupar son los siguientes:

- StackRegister: Consiste en que el usuario ingresa un nombre y una clave, si el nombre ingresado ya se encuentra en la lista de registros, le entregara un false, por lo tanto, debe ingresar un nombre nuevo para registrarse.
- StackLogin: Al momento de tener un registro exitoso, el usuario debe ingresar con los datos de registro correctos, ya que, si ingresa nombre o clave incorrecta, no se podrá completar el ingreso y se mostrará en pantalla un false.
- Ask: Al momento de realizar un login exitoso, el usuario podrá publicar preguntas que más adelante serán respondidas por otros usuarios.
- Answer: Consiste en que un usuario al estar logeado puede responder a la pregunta de otro usuario, si la persona que responde a la pregunta no se encuentra logeado, esta no podrá ser publicada y se mostrará un false.
- Accept: Función que permite al usuario que realizo una pregunta, aceptar una respuesta. Cabe destacar que esta función solo podrá ser utilizada por el usuario que ha formulado la pregunta, ya que si otro usuario trata de aceptar una respuesta que no es de su pregunta no se le permitirá.
- StackToString: La siguiente función permite pasar todo el stack overflow a un string para que sea visualizado de una manera más comprensible para el usuario.

Al tener dichas funciones bien planteadas e implementadas se podrá realizar una breve simulación de un stack overflow.

Requisitos No funcionales

Estos requerimientos son requisitos para la implementación de buenas prácticas y además de generar una estructura al momento de formar el algoritmo. Algunos de estos requerimientos son obligatorios y con puntajes, si no se cumplen los obligatorios el proyecto será evaluado con nota mínima.

Los requerimientos no funcionales son:

- Autoevaluación: Consiste en realizar una autoevaluación a los requerimientos funcionales mencionados anteriormente.
- Lenguaje y versión: Cada laboratorio se debe implementar con un lenguaje distinto y una versión actualizada de este mismo, en este caso el lenguaje utilizado es Prolog y su versión es la 8.2.3.
- Documentación: Todos los predicados deben estar debidamente comentados. Indicando descripción de la relación, términos de entrada y de salida.
- Realizar tres ejemplos para cada predicado funcional obligatorio y extra.
- Historial de trabajo en Github tomando en consideración la evolución en el desarrollo de su proyecto en distintas etapas. Se requieren **al menos 10 commits** distribuidos en un periodo de tiempo **mayor o igual a 1 semanas**
- Para cada predicado se establecen prerrequisitos. Estos deben ser cumplidos para que se proceda con la evaluación de la funcionalidad implementada.

5. DISEÑO DE LA SOLUCIÓN

Para llegar a una solución y creación del algoritmo, primero se realizó una lluvia de ideas, el cual nos permitió tener una mayor visión del cómo se compone el stack overflow, permitiendo tener una idea más clara y detallada de lo que se debe implementar en dicho stack. Cabe destacar que al momento de implementar el algoritmo no se han utilizado todos los datos de la lluvia de ideas, ya sea por la complejidad que esto conllevaba o más bien porque el enunciado no lo pedía.

Para nuestra implementación el stack contendrá a la lista de registros o usuarios, la lista de preguntas, la lista de respuestas y una lista que mostrará quien esta logeado. Esto se puede ver representado como una lista de cuatro listas en su interior. Además, se han implementado predicados para constructor del tda, pertenencia, selectores, modificadores y algunos predicados extras los cuales serán utilizados en los predicados obligatorios.

Para la lista de registros, se utilizan tres parámetros, los cuales son: Username, Password y reputación. El username nos permitirá identificar a cada usuario que se registre, para que de esta manera no existan usuarios con mismo nombre, la password nos permite asegurarnos de que el usuario debe ingresar dicha contraseña para poder logearse, ya que si no calzan no podrá ingresar al stackoverflow y la reputación nos permite saber que tan activos son los usuarios, en donde si realizan preguntas y aceptan respuestas se obtendrán ganancias para aumentar dicha reputación.

Para la lista de preguntas, se han utilizado los siguientes elementos:

- Id: Consiste en un numero entero que nos permitirá identificar a la pregunta. Cada vez que realicemos una pregunta nueva este id se irá incrementando,
- Pregunta: Se añade el string con la pregunta.
- Usuario: Nombre del usuario que formulo la pregunta.
- Lista de etiquetas: Consiste en una lista con nombres representativos en la pregunta.
- Fecha: Se añade la fecha de formulación de la pregunta.
- Estado: Si esta respondida o no.

Para la lista de respuestas, se han utilizado los siguientes elementos:

- Id pregunta: el id de la pregunta nos sirve para saber a qué pregunta corresponde la respuesta que se ha realizado.
- Id respuesta: el id de la respuesta nos sirve para saber a qué respuesta de la pregunta corresponde.
- Usuario: nombre de usuario que realizo la respuesta.
- Fecha: fecha en que ha sido respondida la pregunta.
- Respuesta: es el string con la respuesta de la pregunta.
- Etiqueta: Consiste en una lista con nombres representativos en la respuesta.

Para la lista de login, se utiliza el usuario y su contraseña, que se obtiene al momento de realizar el predicado stackLogin.
Al momento de tener planteado el tda, y las listas que he nombrado anteriormente se puede empezar a realizar los predicados que son especificados en el laboratorio2. Es por este motivo que el paso de abstracción fue muy esencial para plantear una solución a la problemática, ya que nos permitió tener una idea más general de lo que se debe realizar.

6. ASPECTOS DE IMPLEMENTACIÓN

El código realizado en Prolog está estructurado solo en un archivo de código fuente llamado “lab2_StackOverflow_20227250_Salgado.pl”. El desarrollo del algoritmo del programa fue implementado por el IDE SWI-Prolog en su versión 8.2.3. Para esta implementación no se han utilizado librerías externas.

El archivo anteriormente descrito, se encuentra junto a dos archivos, los cuales son la autoevaluacion.txt y el repositorio.txt, los cuales serán subidos vía consola a la nube de datos de GitHub. Esto nos permite ir subiendo avances que quedaran registrados y guardados en dicha nube. Esta nube es muy importante, ya que, si se nos llega a borrar algo por X motivo nuestro código o avances que estemos realizando, estarán resguardados, ya que se guardan todos los cambios que se han realizado.

La simulación al foro de Stack Overflow que se ha realizado es muy básica, ya que al ingresar al foro real se pueden apreciar una gran cantidad de funcionalidades que este nos permite realizar. Cabe destacar que de igual manera hay funcionalidades que deben ser muy complejas a nivel de código, provocando que sea más elevado de realizar su simulación.

7. INSTRUCCIONES DE USO

A continuación, se va a presentar como se debe compilar el archivo.pl en el IDE de SWI-Prolog y cuáles son las consultas iniciales que se podrán realizar:

Paso 1: Se abre el programa SWI-Prolog.

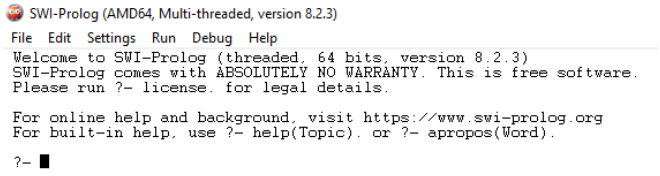

Figura 1: Ingresando a SWI-Prolog

Paso2: Debe seleccionar **File** y luego **Consult.**

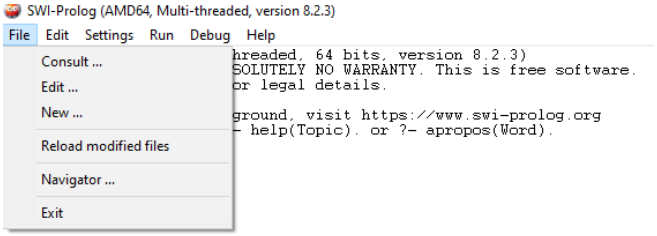

Figura 2: Seleccionar File y Consult

Paso3: Se debe ingresar a la carpeta donde se tenga el archivo y lo seleccionas.

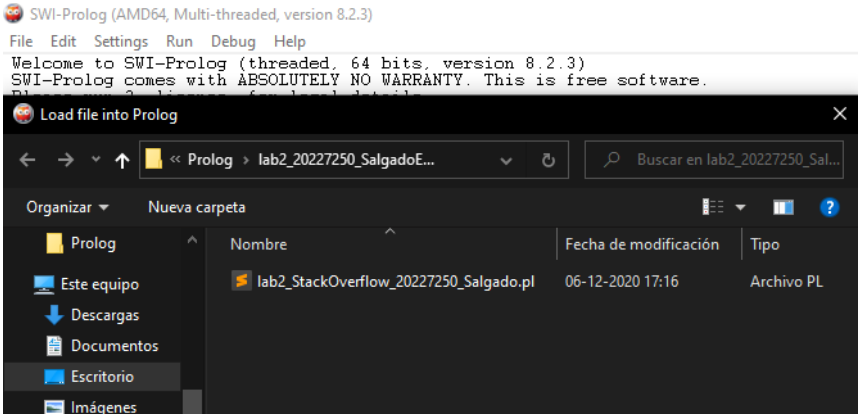

Figura 3: Abrir archivo.pl

Paso4: Luego de seleccionar puedes ingresar las consultas necesarias para empezar a trabajar.

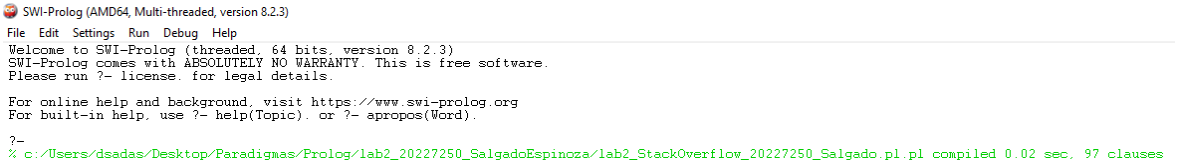

Figura 4: Programa compilado.

En el archivo llamado “lab2_StackOverflow_20227250_Salgado.pl”, se podrán encontrar una mayor cantidad de ejemplos del funcionamiento de los predicados de nuestro programa. A continuación, se presentarán ejemplos de uso de predicados:

crearStack: Este predicado nos permite crear la base de nuestro stack el cual consta de una lista de listas, las cuales contienen a lista de registros, lista de preguntas, lista de respuestas y usuario activo. Para utilizarlo en el IDE se debe compilar el programa y se debe ingresar lo siguiente:

- crearStack(Stack).

stackRegister: Este predicado nos permite crear un usuario que se añadirá a nuestro stack inicial, si el nombre de usuario ya se encuentra registrado entregara por pantalla un false, es por este motivo que debes registrar a usuarios nuevos. Para utilizarlo en el IDE se debe compilar el programa y se debe ingresar lo siguiente:

- crearStack(Stack),stackRegister(Stack,"Dyllan","clave123",Stack1),stackRegister(Stack1,"Salgado","clave456",Stack2).

stackLogin: Este predicado nos permite logear a un usuario que ya se encuentra registrado, si te equivocas en el nombre de usuario o contraseña, te entregara un false. Para utilizarlo en el IDE se debe compilar el programa y se debe ingresar lo siguiente:

- crearStack(Stack),stackRegister(Stack,"Dyllan","clave123",Stack1),stackRegister(Stack1,"Salgado","clave456",Stack2),stackLogin(Stack2,"Salgado","clave456",Stack3).

ask: Este predicado nos permite generar una pregunta con un usuario ya logeado, si el usuario no se encuentra logeado no se podrá realizar preguntas y entregará un false. Para utilizarlo en el IDE se debe compilar el programa y se debe ingresar lo siguiente:

- crearStack(Stack),stackRegister(Stack,"Dyllan","clave123",Stack1),stackRegister(Stack1,"Salgado","clave456",Stack2),stackLogin(Stack2,"Salgado","clave456",Stack3),ask(Stack3,"04-12-2020","preguntita",["prolog","C"],Stack4).

answer: Este predicado nos permite responder una pregunta que se ha realizado, cabe destacar que se debe logear nuevamente el usuario y podrá responder a las preguntas de otro usuario. Para utilizarlo en el IDE se debe compilar el programa y se debe ingresar lo siguiente:

- crearStack(Stack),stackRegister(Stack,"Dyllan","clave123",Stack1),stackRegister(Stack1,"Salgado","clave456",Stack2),stackLogin(Stack2,"Salgado","clave456",Stack3),ask(Stack3,"04-12-2020","preguntita",["prolog","C"],Stack4),stackLogin(Stack4,"Dyllan","clave123",Stack5), answer(Stack5,"05-12-2020",1,"mi respuesta",["lol","dota"],Stack6).

accept: Este predicado nos permite aceptar una respuesta a una pregunta, cabe destacar que se debe logear nuevamente un usuario, pero además solo el usuario que realizo la pregunta puede aceptar respuestas a esta misma, si otro usuario trata de aceptar una respuesta que no es de su pregunta entregara un false. Para utilizarlo en el IDE se debe compilar el programa y se debe ingresar lo siguiente:

- crearStack(Stack),stackRegister(Stack,"Dyllan","clave123",Stack1),stackRegister(Stack1,"Salgado","clave456",Stack2),stackLogin(Stack2,"Salgado","clave456",Stack

3),ask(Stack3,"04-12-2020","preguntita",["prolog","C"],Stack4),stackLogin(Stack4,"Dyllan","clave123",Stack5), answer(Stack5,"05-12-2020",1,"mi respuesta",["lol","dota"],Stack6),stackLogin(Stack6,"Salgado","clave456",Stack7),accept(Stack7,1,1,Stack8).

stackToString: Este predicado nos permite transformar todo el stack en un string, tiene dos formas de utilizarse, la primera es cuando un usuario no esta logeado se mostrará por pantalla todo el stack, pero si el usuario se encuentra logeado se mostrará por pantalla los datos del usuario y las preguntas que ha realizado.

- Caso sin usuario activo: crearStack(Stack),stackRegister(Stack,"Dyllan","clave123",Stack1),stackRegister(Stack1,"Salgado","clave456",Stack2),stackLogin(Stack2,"Salgado","clave456",Stack3),ask(Stack3,"04-12-2020","preguntita",["prolog","C"],Stack4),stackLogin(Stack4,"Dyllan","clave123",Stack5), answer(Stack5,"05-12-2020",1,"mi respuesta",["lol","dota"],Stack6),stackLogin(Stack6,"Salgado","clave456",Stack7),accept(Stack7,1,1,Stack8), stackToString(Stack8, StackStr).
- Caso con usuario Activo:

  crearStack(Stack),stackRegister(Stack,"Dyllan","clave123",Stack1),stackRegister(Stack1,"Salgado","clave456",Stack2),stackLogin(Stack2,"Salgado","clave456",Stack3),ask(Stack3,"04-12-2020","preguntita",["prolog","C"],Stack4),stackLogin(Stack4,"Dyllan","clave123",Stack5),answer(Stack5,"05-12-2020",1,"mi respuesta",["lol","dota"],Stack6),stackLogin(Stack6,"Salgado","clave456",Stack7),accept(Stack7,1,1,Stack8), stackLogin(Stack8,"Salgado","clave456",Stack9), stackToString(Stack9, StackStr).

Como se puede observar en la función stackToString se utilizan todos los predicados funcionales que nos solicitan en el enunciado, es por este motivo que no se entrega nuevamente el ejemplo de como funcionan todos los predicados juntos.

Posibles errores: Hasta el momento no se han encontrados errores, he tratado de desbugear todos los errores posibles que se puedan encontrar, pero aun así mi código puede tener fallas.

8. RESULTADOS Y AUTOEVALUACIÓN

Se logra simular un foro tipo Stack Overflow. El programa permite registrar usuarios, generar preguntas, responder preguntas, aceptar respuestas de una pregunta, dar recompensa a quien realiza la pregunta y al usuario que se le acepto la respuesta, entre otras cosas. Puede que tenga algunas fallas que no he podido visualizar, pero si es que las tiene no debe ser muy difícil su corrección. No logre realizar predicados opcionales por el motivo de que tenia fallas en los obligatorios, se ha preferido realizar los predicados obligatorios y que funcionen bien, en vez de realizar los opcionales teniendo fallas en los obligatorios. Se realizaron bastantes pruebas en los predicados obligatorios, encontrando muchas fallas los cuales se fueron arreglando mientras avanzaba el tiempo.

Requerimientos no Funcionales:

Incluir autoevaluación los requerimientos funcionales solicitados.	Se adjunta un archivo de texto plano con la autoevaluación de los requerimientos.
La implementación debe ser en el lenguaje de programación Prolog	Se utiliza el lenguaje señalado
Usar SWI-Prolog versión 8.x.x	Se utiliza el IDE señalado en versión 8.2.3
Todos los predicados deben estar debidamente comentados. Indicando descripción de la relación, términos de entrada y de salida.	Se comenta cada predicado implementado, con dominio, recorrido y una breve descripción de lo que hace.
Historial de trabajo en Github tomando en consideración la evolución en el desarrollo de su proyecto en distintas etapas. Se requieren **al menos 10 commits** distribuidos en un periodo de tiempo **mayor o igual a 1 semanas**	Se cumple con el historial.
Al final de su código incluir al menos 3 ejemplos de uso para cada uno de los predicados correspondientes a requerimientos funcionales y extras.	Se muestran diferentes ejemplos para los predicados funcionales obligatorios, pero para los extras no, ya que no se ha implementado ninguno.
Para cada predicado se establecen prerrequisitos. Estos deben ser cumplidos para que se proceda con la evaluación de la funcionalidad implementada	Se han seguido todos los prerrequisitos para una buena implementación del programa.

Requerimientos funcionales:

Tdas	Fue alcanzado de manera óptima, se logró la implementación de las funciones de constructor, pertenencia, selectores y modificadores.
Hechos	Se ha cumplido, producto de que se han implementado los dos stack. Donde uno tiene los cuatro usuarios registrados, las cinco preguntas y las diez respuestas, mientras que el segundo stack tiene dos usuarios registrados, tres preguntas y una respuesta.
Predicado StackRegister	Se logra crear el predicado stackRegister, donde si se encuentra un nombre de usuario ya registrado se señala que no puede ingresar dicho nombre de usuario, y si no se encuentra se ingresa a la lista de usuarios
Predicado stackLogin	Se logra crear el predicado stackLogin, donde si se ingresa nombre de usuario y clave correcta deja logearse, pero si se ingresa uno de los dos de forma incorrecta se entrega un false.
Predicado ask	Se logra crear predicado ask, donde si se ingresa pregunta y no esta logeado entregara un false, se debe estar logeado para publicar pregunta.
Predicado answer	Se logra crear predicado answer, donde se debe estar logeado para responder la pregunta señalada con el id.
Predicado accept	Se logra crear predicado accept, donde solo el usuario que genera pregunta puede aceptar la respuesta.
Predicado stackToString	Se logra crear predicado stackToString, donde se muestra todo el stack en forma de string para que sea más entendible por usuario.
Predicado Vote	**No se logra implementar**
Predicado ReportOffense	**No se logra implementar**
Predicado Ranking	**No se logra implementar**
Predicado Search	**No se logra implementar**

9. CONCLUSIÓN

A través de los datos adquiridos por el siguiente informe, se puede concluir que:

Stack Overflow es un foro muy integro en comparación con otros foros de programación, por el motivo de que esta hecho para usuarios profesionales o expertos en el ámbito de programación y para personas que están recién aprendiendo. Al ser un foro que lo utilizan personas expertas, estas mismas pueden ir dando ideas para ir mejorando la página web, prestando servicios de mantención, entre otros. Esto genera que el Stack Overflow aparte de ser integro como lo dije al inicio, también será muy complejo a nivel de estructuración y código. Esto lo experimente al momento de simularlo, ya que, si bien era una simulación parcial, tenía que ir pensando en muchas formas en que podría hacer fallar el programa o faltar al momento de generar un predicado.

En este laboratorio tuve dificultades para llevarlo a cabo, al igual que el primero que se realizo con el paradigma funcional, producto de que no estoy familiarizado con la programación de otros paradigmas.

El código creado se puede mejorar considerablemente para que sea mucho más sencillo y fácil de entender, pero para esto necesitare más practica y tiempo, ya que como dije anteriormente se me es complicado trabajar en un paradigma distinto, puesto que pierdo mucho tiempo pensando en cómo desarrollar los predicados e implementarlo.

En el ámbito de resultados, se pudo haber llegado a uno mucho mejor, pero por temas de tiempo y mal organización no fue de esta manera. Si bien se llega a una implementación parcial, este puede presentar fallas.

En el ámbito de implementación del paradigma lógico, se trató de crear todos los predicados posibles para no utilizar librerías externas del mismo IDE. Si bien, el laboratorio2 que fue creado en prolog me resulto difícil al inicio, a diferencia del paradigma anterior producto de que me confundían demasiado realizar funciones para todo el código, en cambio, en el actual laboratorio los predicados eran muy similares. Por lo tanto, se trató de seguir al pie de la letra todas las condiciones que solicitaban los predicados.

10. REFERENCIAS

1) German Escobar, (Sep 21, 2015), ¿Qué es un stack overflow (no el sitio)? (30-11-2020), https://blog.makeitreal.camp/que-es-un-stack-overflow-desbordamiento-de-pila/
2) MARCOS MERINO,(09-08 2020), El lenguaje Prolog: un ejemplo del paradigma de programación lógica(30-11-2020)