2023

Informe De Laboratorio: Creación De Un Sistema de Chatbots en Scheme

Integrantes: Aylin Castillo

Profesor: Víctor Flores

Asignatura: Paradigmas de programación

Tabla de contenido

Descripción del problema

En este informe se explicará una solución al problema de laboratorio utilizando el paradigma funcional, donde se utilizará el lenguaje Scheme con el intérprete DrRacket para la implementación de este.

Se desea construir un programa que permita crear Chatbots que estarán enlazados por flujos y conviviendo entre sí dentro de un sistema donde interactuarán con ellos los usuarios registrados.

Descripción del paradigma

El paradigma funcional se basa más en el "qué" que en el "cómo" para resolver problemas, donde solo se tendrá la misma salida para una misma entrada, junto a esto, en este paradigma no existe el concepto de variable, pero si de estados, lo cual es una ventaja al tener intenciones de volver a cierto punto del código.

Las características utilizadas para el proceso del laboratorio son:

- **Función anónima**: Se le llama función anónima a aquella que no tiene un nombre, se puede crear usando lambda. Es efectivo usarla cuando se quiere hacer un procedimiento en específico para una única función.
- **Función de Orden Superior**: Conocidas por ser capaces de recibir una función como parámetro, como por ejemplo: map, filter, apply.
- **Recursión**: Debido a que en este lenguaje no existen los ciclos (bucles), hay que usar si o si la recursividad, de las cuales hay dos tipos: natural y de cola, las cuales tienen como principal característica que se llaman a sí mismas.

Análisis del problema

Se reconocen cinco elementos principales, los cuales se organizan de la siguiente forma:

- **Sistema**: Es donde interactuarán los chatbots y los usuarios.
- **Chatbot**: Contiene los flujos de las distintas opciones a elegir.
- **Flujo**: Contiene las opciones elegidas por el usuario.
- **Fecha**: Entrega la fecha actual de cuando fue utilizado el programa.

- **Usuario**: Encargado de interactuar con los chatbots, tiene que registrarse antes de hacerlo.

Los elementos sistema, chatbot y flujo tienen funcionalidades muy parecidas, como el agregar, filtrar duplicados, contar con ID's para identificarlos, etc.

Por otro lado, el usuario debe poder registrarse y después lograr ingresar al sistema, pudiendo también cerrar sesión sin perder su registro.

Además dentro del sistema se debería lograr que los chatbots hablen entre sí.

También se solicita que cada TDA utilizado este en un archivo propio.

Diseño de solución

Para el problema expuesto anteriormente, se decidió hacer uso del concepto de TDA (tipo de dato abstracto) para poder hacer de una mejor forma la abstracción y construir una solución más apropiada al paradigma funcional. En el anexo 1 se encontrará un diagrama con los cuatro TDA's usados en la implementación, estos están presentados desde el elemento más pequeño del sistema (el cual sería el flujo) al sistema en sí que contiene todo lo que se usará.

Para llevar a cabo los TDA se usará el paradigma funcional en el lenguaje DrRacket donde se construirán en el mismo orden descrito en el diagrama junto a cada componente necesario para su funcionalidad.

Cabe mencionar que entre ellos hay funcionalidades muy similares, como por ejemplo el filtrar duplicados por su ID y agregar cierto elemento a una lista, por lo que al realizar las respectivas funciones para uno, se pueden adaptar a los otros fácilmente teniendo cambios mínimos.

También se hará uso de las funciones nativas de DrRacket que cumplan con el paradigma funcional, tales como remove-duplicates, flatten, member, entre otras. Lo cual simplificará aún más el proyecto.

Aspectos de implementación

El compilador utilizado es DrRacket en versiones 6.11 o superior, donde se hace el uso de listas principalmente creadas a través de funciones.

El proyecto es estructurado en base a la definición de TDA para poder tener un mejor orden en la implementación y en caso de haber un error no tener que cambiar todo el código. Debido a lo anterior se tiene que hacer uso de las funciones nativas de DrRacket llamadas "Require" y "Provide" las cuales sirven para importar y exportar funciones entre distintos archivos y así poder conectarlos.

Por otro lado, se hizo uso de la librería nativa de DrRacket en todos los TDA exceptuando el TDA Fecha, ya que para este fue necesario usar racket/date para obtener la fecha del sistema.

Instrucciones de uso

En el archivo .zip se encontrarán todos los archivos .rkt que servirán para hacer funcionar el programa. Para probarlo se debe usar el que se llama "pruebas_21060190_CastilloPerez.rkt", el cual contiene todas las pruebas realizadas al código. Para hacerlo funcionar, se deben llamar las funciones de dos maneras que se muestran en el anexo 2 (no se pueden llamar a todas con solo una llamada, tiene que ser si o si una llamada para cada una).

Después de llamar a las funciones deseadas, en caso de que estas funcionen, se mostrará una lista de listas en el terminal con sus respectivos componentes (anexo 3 con función s5), en caso contrario no se mostrará nada, ya que en pruebas previas realizadas, se dejaron como comentario las funciones que daban error, debido a que hay funciones dentro de los TDA que funcionan parcialmente o no fueron terminadas.

Por otro lado, es posible que el programa tenga errores cuando se ingresan listas vacías a ciertas funciones y se asume que se entregan correctamente las entradas para evitar otros posibles errores.

Resultados y autoevaluación

En general, el proyecto funciona en la mayoría de lo que se alcanzó a hacer, teniendo algunos problemas con ciertas pruebas, como por ejemplo el intentar hacer log in cuando otro usuario ya está conectado, que algunas opciones se agreguen al revés debido al filtro que se lleva a cabo al usar la función flow o flow-add-option, etc.

Se logró implementar once de las quince funciones exigidas debido al límite de tiempo y la falta de ingenio bajo presión para lograr hacer funcionar las faltantes.

Teniendo en cuenta lo anterior, la autoevaluación radica entre valores de 0 (no implementado) hasta 1 (implementado a la perfección) aumentando en una escala de 0.25. Esta se encontrará en el anexo 4.

Conclusiones

En conclusión, los alcances que tiene el proyecto son parcialmente satisfactorios debido a la falta de cuatro funciones y el rendimiento completo o parcial de las que si se construyeron, pero a pesar de esto se logró implementar por completo lo que es el paradigma funcional en el lenguaje exigido, sin la necesidad de usar funciones que

pertenecieran a otros paradigmas, como por ejemplo la función set!, que a pesar de venir en la librería de DrRacket, no cumple con el paradigma exigido.

La mayor limitación que se presentó al usar el paradigma funcional y lo que causo más dificultad al empezar este proyecto fue la no utilización de variables, ya que la falta de estas restringía el uso de ciclos para poder recorrer listas y modificar algún elemento de esta, pero esto tuvo solución con el uso de la recursión natural y el uso de car, cdr y list-ref (funciones nativas de DrRacket) para poder acceder a datos en posiciones específicas de las listas creadas.

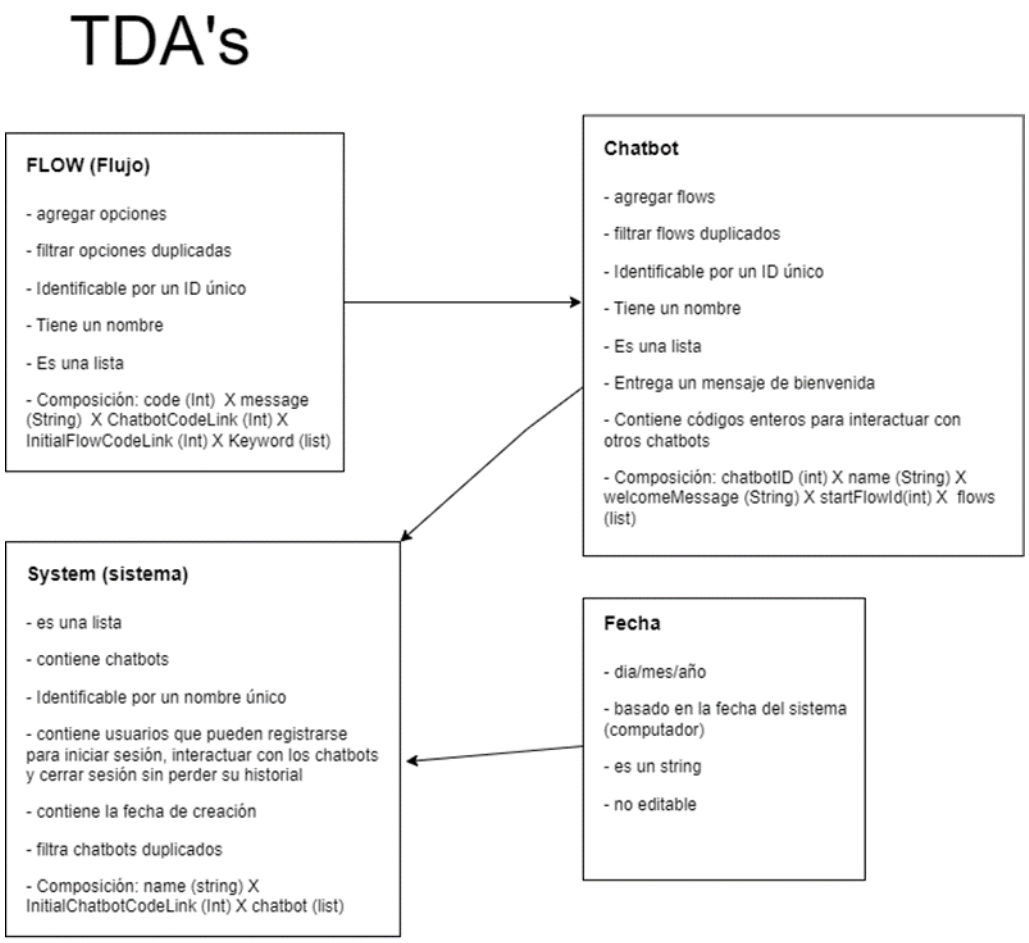
Referencias

1.- Flatt, M. y Bruce, R. (2021). “The Racket Guide”

https://docs.racket-lang.org/guide/

Anexos

Anexo 1:



Anexo 2:

```
(define op1 (option  1 "1) Viajar" 2 1 "viajar" "turistear" "conocer"))
(display op1)
op1
```

Anexo 3:

```
Welcome to DrRacket, version 8.10 [cs].
Language: racket, with debugging; memory limit: 128 MB.
'("9.10.2023"
  "Chatbots Paradigmas"
  0
  (0
   "Inicial"
   "Bienvenido\n¿Qué te gustaría hacer?"
   1
   (1
    "flujo1"
    (1 "1) Viajar" 2 1 ("viajar" "turistear" "conocer"))
    (2
     "2) Estudiar"
     3
     1
     ("estudiar" "aprender" "perfeccionarme")))
   0
   "Inicial"
   "Bienvenido\n¿Qué te gustaría hacer?"
   1
   (1
    "flujo1"
    (1 "1) Viajar" 2 1 ("viajar" "turistear" "conocer"))
    (2
     "2) Estudiar"
     3
     1
     ("estudiar" "aprender" "perfeccionarme"))))
  ("user1" "user2" "user3"))
```

Anexo 4 (autoevaluación):

Requerimiento no funcional / funcional	Puntaje
Autoevaluación	Obligatorio
Lenguaje	Obligatorio
Versión	Obligatorio
Standard	Obligatorio
Documentación	1
Dom->Rec	Obligatorio
Organización	1
Historial	1
Script de pruebas	Obligatorio
Prerrequisitos	Obligatorio
TDA	0.75
Option	1
Flow	1

Flow-add-option	0.75
Chatbot	1
Chatbot-add-flow	1
System	1
System-add-chatbot	1
System-add-user	1
System-login	0.5
System-logout	0.5
system-talk-rec	0
system-talk-norec	0
system-synthesis	0
system-simulate	0