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

Informe De Laboratorio: Creación De Un Sistema De Chatbots

Integrantes: Gonzalo Moncada
Profesor: Gonzalo Martinez
Asignatura: Paradigmas de programación

Tabla de contenido

1.- Introducción

En este informe se expondrá una solución al problema de laboratorio presentado utilizando el paradigma funcional, todo esto mediante el lenguaje de programación Racket.

1.1.- Descripción del problema

Se desea construir un programa que permita a un usuario realizar distintas operaciones (crear, vincular, interactuar, etc.) en un sistema centrado en chatbots, las estructuras base son:

Option: Almacenan los mensajes y vínculos dentro del programa, este acepta tanto el codigo como una keyword(palabra clave).

Flow: Almacenan las opciones dentro del programa.

Chatbot: Almacenan los flujos dentro del programa.

System: Almacenan los chatbots dentro del programa, debe gestionar todas las interacciones en el programa.

User: Genera un usuario que realice las interacciones.

ChatHistory: Almacenan las interacciones realizadas en el programa.

1.2.- Descripción del paradigma

El paradigma funcional, se enfoca en la resolución de problemas a través de funciones, respondiendo al ¿Qué? Y no al ¿Cómo?

Donde una función se puede describir como algoritmo que ante una entrada concreta, entrega una salida siendo estos su dominio y recorrido respectivamente, siempre teniendo en cuenta que la misma entrada siempre tendrá una misma salida. esta omite el uso de variables para una optimización en la funcion, las herramientas que proporciona este paradigma son:

Funciones anónimas: Se llama función anónima aquella que no tiene un nombre, por lo tanto, no es guardada en memoria.

Recursividad: Siendo una propiedad de las funciones, aunque no aplicable a las anónimas y se reconoce cuando una función se llama a sí misma, existen 3 tipos: de cola,natural y arbórea.

Funciones de orden superior: La principal característica es que reciben las funciones como parámetros.

Currificación: Una función currificada es aquella recibe los argumentos de manera secuencial, en racket dejando un procedimiento en espera de aquellos que faltan.

1.3.- Objetivos

Como objetivo se tiene el desarrollar entendimiento del paradigma funcional a través de racket con la idea de realizar por completo la solución al problema presentado.

2.- Desarrollo

2.1.- Análisis de problema

Se tiene como entidad principal el sistema, el cual va a almacenar todos los chatbots y crear sus interacciones, además de esto se presenta como requisito especifico la creación de funciones en las entidades de option, chatbot, flow y system para el correcto funcionamiento del programa, las funciones

option, flow, flow-add-option, chatbot, chatbot-add-flow, system, system-add-chatbot, system-add-user, system-login, system-logout, system-talk-rec, system-talk-norec, system-synthesis, system-simulate.

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

2.2.- Diseño de solución

Al diseñar la solución se ocuparon los tipos de datos nativos del lenguaje y se crean 7 TDA que serán utilizados para solucionar el problema, la base en la cual se enfoca la solución es en tener ciertas opciones que se dirijan a flujos y estos estar presentes en chatbots almacenados en un system, todo esto a cabo de listas anidadas, además de esto se presentaran los usuarios para poder especificar quien interactúa con el system, además de esto ser guardado en un chatHistory para tener registro de lo que se realizo, por ultimo cabe aclarar que frente a la función de interactuar se opto por tener 2 numeros que fueran cambiando a medida que el usuario ingresaba su mensaje, esto para tener constancia en que instancia del sistema esta el usuario y poder desde el mensaje poder verificar su elección al menú el cual el esta presente, así también se puede que si se deslogea el usuario se actualizan los valores a los iniciales, esto en base al vinculo del chatbot inicial, las representaciones de cada TDA son las siguientes:

TDA Option: Una lista que contiene de entrada un codigo identificador(int), un mensaje(string), un vinculo hacia un chatbot(int), un vínculo hacia un flow(int) y una lista de palabras clave(lista de string)
TDA Flow: Una lista que contiene de entrada un codigo identificador(int), el nombre y

su mensaje(string), y una lista de opciones(lista de option).
TDA Chatbot: Una lista que contiene de entrada un codigo identificador(int), el nombre(string), su mensaje de entrada(string), el vínculo al flow de inicio(int) y una lista de flujos (lista de flow),
TDA System: Una lista que contiene de entrada un nombre(string), el vinculo a chatbot inicial(int) y una lista de chatbots (lista de chatbot), además se agrega la fecha de creación(number), un vínculo al chatbot actual(number), un vínculo al flow actual(number), una lista de usuarios(list) y una lista de las interacciones realizadas(list).
TDA User: Una lista que contiene de entrada un nombre(string), además se agrea un booleano que entrega el estado del usuario(#t = logeado/#f = offline)..
TDA ChatHistory: Una lista que contiene de entrada un nombre de usuario(string), un vínculo de chatbot(int), un vínculo de flow(int) y un mensaje(string), además se agrega la fecha de creación(number).
TDA Common: Tda con funciones comunes utilizadas entre los TDA.

Además, se tiene desde el anexo 1 al 7 las funciones utilizadas en cada TDA. Luego la relación entre estos TDA se podrá ver en Anexo 8, además de mencionar que, dada la similitud de varias funciones dentro del TDA estas se descomponen en funciones más pequeñas para lograr avances en más de una. Se utilizará principalmente recursión del tipo natural por la facilidad que otorganal momento de recorrer una lista o crearla.

2.3.- Aspectos de implementación

El compilador utilizado es DrRacket en versiones 6.11 o superior, se trabaja principalmente con listas, pero se hacen funciones para acceder a estas.

El código se estructura por TDA, para facilitar la edición a la hora de agregar contenido a alguno de estos, utilizando require / provide para importar y exportar información de estos a otros archivos y asi hacer utilizacion de estos.

Para este laboratorio no se usan bibliotecas externas por lo que solo se define que se utiliza el lenguaje racket siendo escogido por sobre scheme por simplicidad a la hora de acceder a la documentación.

2.4.- Instrucciones de uso

Se tiene un archivo main que contiene varios ejemplos y se tiene en funcionamiento solo los TDA requeridos(option, flow, chatbot y system).

En cambio para el el uso fuera de los ejemplos propuestos en este archivo se presenta el anexo 9, el cual presenta la manera de generar cada uno de los requerimientos para realizar una interaccion en el sistema, en cuanto a esta interacción será posible realizarse siempre y cuando los vínculos tengan una dirección correcta, ya que si llamo a un Flow inexistente se terminara el programa.

En cuanto a posibles errores, mientras se utilizen correctamente las entradas

no deberia existir errores.

2.5.- Resultados y autoevaluación

Los resultados son los esperados, logrando hacer un programa completamente funcional, donde se intentó colocando entradas erróneasy repetidas dentro de una misma entidad, como también el interactuar con opciones erróneas.

Se logran completar las 14 funciones propuestas para el proyecto

La autoevaluación radica entre valores de 0 (no implementado) hasta 1 (implementado y sin errores) aumentando en una escala de 0.25 y se puede observar en el anexo 10. Se les asigna a todas las funciones 1 dado que al momento de realizar diversas pruebas estas trabajan correctamente y sin errores.

3.- Conclusión

Una vez terminado el trabajo se logra cumplir los objetivos, mejorando el entendimiento del paradigma funcional, así como también lograr una aplicación en racket para interactuar con chatbots dentro de un sistema.

La mayor limitación de este paradigma y lo que causo más dificultad al empezar este proyecto fue la no utilización de variables, ya que la falta de estas complicaba el pensamiento respecto a cómo recorrer listas y modificar algún elemento de esta sin modificar lo demás, luego de encontrar una solución había problemas sobre cómo usar la recursión y como generar la misma salida sin la utilización de esta.

4.- Bibliografía

1.- Gonzales, R. (2023). “Proyecto semestral de laboratorio”. https://docs.google.com/document/d/1lErqBgtZSLxtedB4dsQjmg6ujSi5GwfJ-NKjsRvlXKY/edit

2.- Flatt, M. y Bruce, R. (2021). “The Racket Guide” https://docs.racket-lang.org/guide/

5.- Anexos

1.- Funciones del TDA Option:

Tipo de función	Nombre	Descripción
constructor	option	Crea opcion para un flujo de un chatbot
selector	get-option-code	Selecciona un code de una opcion
selector	get-option-message	Selecciona un mensaje de una opcion
selector	get-option-ChatbotCodelink	Selecciona un vinculo de chatbot de una opcion
selector	get-option-InitialFlowCodeLink	Selecciona un vinculo de flow de una opcion
selector	get-option-from-message-rec	Selecciona una opcion base a un mensaje de manera recursiva
selector	get-option-from-message	Selecciona una opcion base a un mensaje

2.- Funciones del TDA Flow:

Tipo de función	Nombre	Descripción
constructor	flow	Crea un flujo de un chatbot
selector	get-flow-id	Selecciona un id de un flow
selector	get-flow-name	Selecciona un nombre de un flow
selector	get-flow-option	Selecciona las opciones de un flow
selector	get-flow-option-from-message-rec	Selecciona una opcion base a un mensaje de manera recursiva
selector	get-flow-option-from-message	Obtiene una opcion base a un mensaje
modificador	set-flow-new-option	Define un nuevo flow con una lista nueva de opciones
modificador	flow-add-option	Añade una opcion a un flujo
otras funciones	flow-options-format	Obtiene las opciones que contiene el flow en un formato de texto

3.- Funciones del TDA Chatbot:

Tipo de función	Nombre	Descripción
constructor	chatbot	Crea un chatbot
selector	get-chatbot-id	Selecciona un id de un chatbot
selector	get-chatbot-name	Selecciona un nombre de un chatbot
selector	get-chatbot-welcomeMessage	Selecciona un mensaje de entrada de un flow
selector	get-chatbot-Initialflow	Selecciona un vinculo con el flow inicial de un chatbot
selector	get-chatbot-flows	Obtiene una lista de flows de un chatbot
selector	get-chatbot-flow-by-id	Obtiene un flow de un chatbot en base a una id

modificador	set-chatbot-new-flow	Define un nuevo chatbot con una lista nueva de flows
modificador	chatbot-add-flow	Añade un flow a un chatbot

4.- Funciones del TDA System:

Tipo de función	Nombre	Descripción
constructor	system	Crea un sistema con chatbots
constructor	make-system-message-notalk	Crea un mensaje de chatHistory sin exito de interaccion
constructor	make-system-message-notalk	Crea un mensaje de chatHistory con exito de interaccion
selector	get-system-date	Selecciona una fecha de creacion de un system
selector	get-system-name	Selecciona el nombre de un chatbot
selector	get-system-user	Selecciona una lista de usuarios de un system
selector	get-system-chatHistory	Selecciona el historial de mensajes de un system
selector	get-system-initialchatbotcodelink	Selecciona el vinculo del chatbot inicial de un system
selector	get-system-Activechatbotcodelink	Selecciona el vinculo al chatbot actual de un system
selector	get-system-Activeflowcodelink	Selecciona el vinculo al flow actual de un system
selector	get-system-chatbots	Selecciona los chatbots de un system
selector	get-system-chatbot-by-id	Selecciona un chatbot base a una id
selector	get-system-option-from-message-rec	Selecciona una opcion desde el system base a un mensaje de manera recursiva
selector	get-system-option-from-message	Selecciona una opcion desde el system base a un mensaje
modificador	set-system-new-chatbot	Define un nuevo system con una nueva lista de chatbots
modificador	set-system-new-user	Define un nuevo system con una nueva lista de users
modificador	set-system-login	Define un nuevo system con una nueva lista de users despues de login
modificador	set-system-logout	Define un nuevo system con una nueva lista de users despues de logout
modificador	set-system-talk	Define un nuevo system tras un system-talk con exito
modificador	set-system-notalk	Define un nuevo system tras un system-talk sin exito
modificador	set-system-chatHistory-new-message	Define un nuevo chatHistory tras un system-talk con exito
modificador	set-system-chatHistory-new-message-notalk	Define un nuevo chatHistory tras un system-talk sin exito
modificador	set-system-add-chatbot	Añade chatbots al sistema

modificador	set-system-add-user	Añade un usuario al sistema
otras funciones	system-login	Permite iniciar una sesion en el sistema
otras funciones	system-logout	Permite cerrar una sesion en el sistema
otras funciones	system-talk-rec	Permite interactuar con un chatbot mediante recursion
otras funciones	system-talk-norec	Permite interactuar con un chatbot
otras funciones	system-message-display-format	Selecciona un mensaje del chatHistory y le genera un formato para display desde el system
otras funciones	system-synthesis	Permite obtener sintesis de lo realizado por un usuario
otras funciones	system-simulate	Permite simular un dialogo entre dos chatbots del sistema

5.- Funciones del TDA User:

Tipo de función	Nombre	Descripción
constructor	make-user	Crea un usuario
constructor	make-user-name-by-seed	Crea un nombre de usuario en base a un numero de semilla
selector	get-user-name	Selecciona un nombre de un usuario
selector	get-user-state	Selecciona un estado de un usuario
selector	get-login-user	Selecciona un usuario logeado
modificador	set-login-user-list	Obtiene una nueva lista de usuarios con un user logeado
modificador	set-logout-user-list	Obtiene una nueva lista de usuarios con todos los usuarios offline
pertenencia	is-login-user	Verifica si un usuario esta logeado

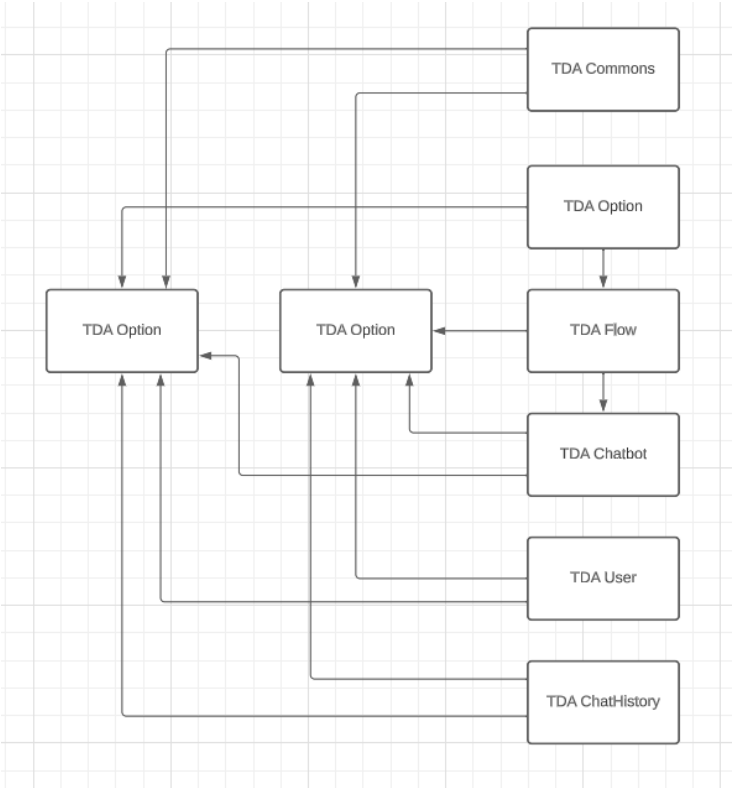
6.- Funciones del TDA ChatHistory:

Tipo de función	Nombre	Descripción
constructor	make-message	Crea un mensaje de chatHistory
selector	get-message-date	Selecciona una fecha de creacion de un message
selector	get-message-user	Selecciona un nombre de usuario de un message
selector	get-message-chatbot	Selecciona un chatbot id de un message
selector	get-message-flow	Selecciona un flow id de un message
selector	get-message-user-message	Selecciona un mensaje de usuario de un chatHistory

7.- Funciones del TDA Commons:

Tipo de función	Nombre	Descripción
otras funciones	eliminar-duplicados	Crea un mensaje de chatHistory
otras funciones	rec-list-add	Añade un elemento a una lista de manera recursiva
otras funciones	insertar-final-lista	Añade un elemento a una lista
otras funciones	myRandom	Genera un numero pseudocodigo

8.- Relación entre TDA y main:



9.- ejemplos de uso:

```
> (define option (option  1 "1) Viajar" 1 1 "viajar" "turistear" "conocer"))
> option
'(1 "1) Viajar" 1 1 ("viajar" "turistear" "conocer"))
> (define flow (flow 1 "Flujo Principal Chatbot 1\nBienvenido\n¿Qué te gustaría hacer?" op1))
> flow
'(1 "Flujo Principal Chatbot 1\nBienvenido\n¿Qué te gustaría hacer?" ((1 "1) Viajar" 1 1 ("viajar" "turistear" "conocer"))))
> (define chatbot (chatbot 0 "Inicial" "Bienvenido\n¿Qué te gustaría hacer?" 1 fl0))
> chatbot
'(0
  "Inicial"
  "Bienvenido\n¿Qué te gustaría hacer?"
  1
  ((1 "Flujo Principal Chatbot 1\nBienvenido\n¿Qué te gustaría hacer?" ((1 "1) Viajar" 1 1 ("viajar" "turistear" "conocer")) (2 "2) Estudiar" 2 1 ("estudiar" "aprender" "perfeccionarme"))))))
> (define system (system "Chatbots Paradigmas" 0 cb0))
> system
'(1696884745
  ("Chatbots Paradigmas"
   ()
   ()
   0
   0
   1
   ((0
     "Inicial"
     "Bienvenido\n¿Qué te gustaría hacer?"
     1
     ((1 "Flujo Principal Chatbot 1\nBienvenido\n¿Qué te gustaría hacer?" ((1 "1) Viajar" 1 1 ("viajar" "turistear" "conocer")) (2 "2) Estudiar" 2 1 ("estudiar" "aprender" "perfeccionarme"))))))))
```

10.- Autoevaluación:

Autoevaluacion	Puntaje
option	1
flow	1
flow-add-option	1
chatbot	1
chatbot-add-flow	1

system	1
system-add-chatbot	1
system-add-user	1
system-login	1
system-logout	1
system-talk-rec	1
system-talk	1
system-synthesis	1
system-simulate	1