UNIVERSIDAD
DE SANTIAGO
DE CHILE

Laboratorio 1

Paradigmas de Programación

Ka Zeng Zhong

Sección B-2

Tabla de contenidos

Introducción

En el mundo digital actual, los chatbots se han convertido en herramientas esenciales, facilitando desde respuestas rápidas hasta transacciones en línea. Sin embargo, detrás de esa aparente simplicidad, la creación y gestión de estos chatbots es un desafío. El propósito de este laboratorio es explorar cómo Scheme puede ser utilizado para desarrollar un sistema que facilite el diseño, lanzamiento y administración de chatbots de manera eficiente y sencilla.

Descripción breve del problema

Se necesita un sistema para crear, desplegar y administrar chatbots estructurados que interactúan mediante opciones predefinidas o palabras clave. Estos chatbots deben poder enlazarse entre sí, guiando al usuario de uno a otro según sus elecciones. Aunque los chatbots son herramientas esenciales en la era digital, desarrollarlos presenta desafíos. Usando la programación funcional y el lenguaje Scheme, se busca simplificar este proceso. El sistema permitirá a los usuarios configurar un entorno para múltiples chatbots, crear e interactuar con ellos y, al final, recibir un resumen de la conversación. La interacción puede ser a través de comandos o, opcionalmente, una interfaz gráfica similar a las plataformas de ecommerce.

Descripción breve del paradigma

El paradigma funcional de programación se basa en la construcción de programas mediante la evaluación de funciones, evitando el estado mutable y los datos cambiantes. En esencia, es como el cálculo en matemáticas: dada una entrada, siempre obtendrás la misma salida. Este enfoque trae consigo beneficios como la facilidad para razonar sobre el código y un comportamiento predecible.

En el proyecto de chatbots, se aplican varios conceptos del paradigma funcional: inmutabilidad, funciones como ciudadanos de primera clase, composición de funciones, recursividad y transparencia referencial.

El uso del lenguaje Scheme, con su naturaleza funcional, permite una implementación efectiva de estos conceptos, simplificando el diseño y administración de chatbots estructurados.

Análisis del problema.

El problema planteado busca construir un sistema para la creación, despliegue y administración de chatbots estructurados. Estos chatbots son singulares porque se basan en interacciones predefinidas, no en análisis avanzados de lenguaje natural. Los requisitos principales son:

1. **Ambiente centralizado:** un espacio unificado para todos los chatbots.
2. **Personalización de chatbots:** crear y modificar chatbots según la necesidad.
3. **Enlazar chatbots:** posibilidad de conectar diferentes chatbots.
4. **Interacción flexible:** a través de opciones o palabras clave.
5. **Resumen de conversaciones:** presentar una recapitulación.
6. **Interfaces variadas:** línea de comandos esencial.

Teniendo en cuenta estos requisitos, es evidente que el desafío no sólo está centrado en cómo se aborde el problema, sino que también requiere una comprensión profunda de las interacciones del usuario y una implementación que sea fácil de adaptar y expandir.

Diseño de la solución

Para gestionar las interacciones del chatbot, se abordará el diseño centrado en la manipulación de listas. El problema principal se descompondrá en varias tareas esenciales, tales como la búsqueda de chatbots por código, la actualización de la información de un chatbot y la obtención de respuestas según las palabras clave del usuario. Este nivel de descomposición permite abordar cada desafío de forma individual, simplificando la lógica y facilitando la implementación.

Utilizando las estructuras y funciones de Scheme, se implementarán procedimientos que iteren sobre listas para encontrar chatbots, flujos y opciones específicas. También se hará uso de estructuras condicionales para manejar la lógica de la conversación y asegurar que el sistema pueda responder adecuadamente a las interacciones del usuario. Por último, para mantener un registro ordenado de las conversaciones, se implementará una función que concatene y formatee el historial de chat entre el usuario y el sistema.

En resumen, se propondrá un diseño estructurado y modular que hace uso intensivo de listas y operaciones de lista para representar y manipular los flujos y opciones del chatbot, garantizando así una interacción fluida y eficiente.

Aspectos de la implementación

El código está estructurado en Scheme, favoreciendo las capacidades del lenguaje, en particular para el manejo de listas y recursión. El diseño se basa en una lógica modular, dividiendo la solución en funciones específicas con objetivos claros. Acompañando estas funciones, existen auxiliares diseñadas para abordar subproblemas y mejorar la claridad general del código. El código se apoya en las funcionalidades básicas de Scheme, sin bibliotecas externas. Se usa el intérprete de DrRacket. Todo se documenta de manera coherente para garantizar una legibilidad óptima y prevenir posibles confusiones en el desarrollo.

Instrucciones de uso

El primer paso para utilizar correctamente el programa es instalar DrRacket. Después de esto, se abre el archivo llamado “pruebas”, se ejecuta el código y se puede probar una función tras otra con los parámetros especificados en la sección de requerimientos funcionales. Para hacer que la comprobación de cada función sea más agradable, se incluyen varios ejemplos en el mismo archivo de pruebas (más instrucciones en el anexo del informe).

Resultados y autoevaluación

Al finalizar cada función se realizaron distintos tipos de pruebas para comprobar el funcionamiento de dicha función, esto también es evaluado en la tabla siguiente, ya que, si una prueba daba un resultado inesperado o directamente malo, se hace un descuento de puntaje, los ejemplos utilizados son listados en el archivo de script de pruebas. De manera desafortunada, la última parte del TDA system quedó incompleta, esto se debe a dos grandes razones, la primera fue la complejidad del TDA system, y la segunda razón es la imposibilidad de mi parte de simular un diálogo entre dos chatbots del sistema.

Requerimientos Funcionales	Grado de Alcance
Option	1
Flow	1
Flow-add-option	1
Chatbot	1
Chatbot-add-flow	1
System	1
System-add-chatbot	0.75
System-add-user	1
System-login	1
System-logout	1
System-talk-rec	0,5
System-talk-norec	0,5
System-synthesis	1
System-simultate	0

Conclusiones

Finalmente, luego de haber desarrollado parte del problema, se puede decir que fue una experiencia enriquecedora con respecto al paradigma funcional y a Scheme, ya que se trabajó constantemente con este lenguaje y se logró comprenderlo de una manera mucho mejor que si se hubiera visto solo de una manera teórica. Por lo tanto, el paradigma presentó numerosos desafíos, ya que la creación de funciones que cumplan objetivos específicos se vuelve extremadamente complicada sin la utilización de variables. Esto no se debe a que las funciones sean grandes o largas, sino a que es difícil pensar en cómo realizar dichas funciones sin la utilización de variables.

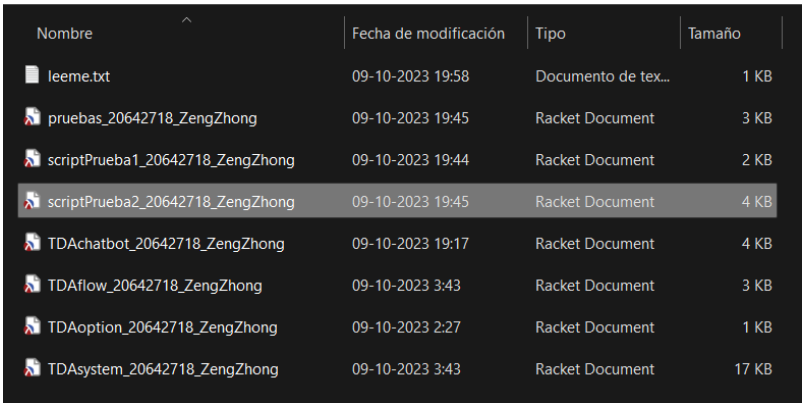
Referencias

Racket documentation. (s. f.). https://docs.racket-lang.org/

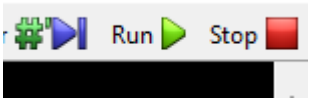
Anexos

Ejemplo ejecutando script de pruebas 2:

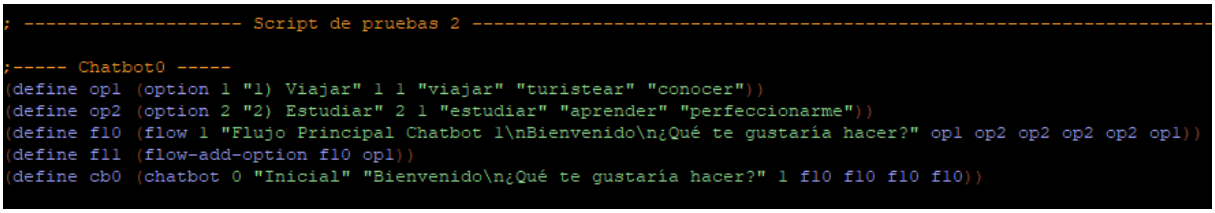
1. Abrir archivo



2. Presionar Run en la esquina superior derecha



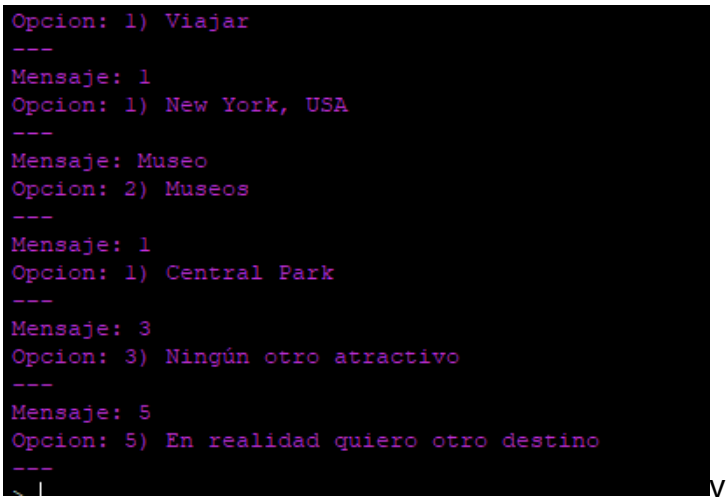
3. Revisar funciones



```
; -------------------- Script de pruebas 2 ---------------------------------------------------------------------

;----- Chatbot0 -----
(define op1 (option 1 "1) Viajar" 1 1 "viajar" "turistear" "conocer"))
(define op2 (option 2 "2) Estudiar" 2 1 "estudiar" "aprender" "perfeccionarme"))
(define f10 (flow 1 "Flujo Principal Chatbot 1\nBienvenido\n¿Qué te gustaría hacer?" op1 op2 op2 op2 op2 op1))
(define f11 (flow-add-option f10 op1))
(define cb0 (chatbot 0 "Inicial" "Bienvenido\n¿Qué te gustaría hacer?" 1 f10 f10 f10 f10))
```

4. Observar el retorno del programa



```
Opcion: 1) Viajar
---
Mensaje: 1
Opcion: 1) New York, USA
---
Mensaje: Museo
Opcion: 2) Museos
---
Mensaje: 1
Opcion: 1) Central Park
---
Mensaje: 3
Opcion: 3) Ningún otro atractivo
---
Mensaje: 5
Opcion: 5) En realidad quiero otro destino
---
```

V