# Informe Laboratorio 1
# Paradigma Funcional

Estudiante: Benjamín Zúñiga J.

Asignatura: Paradigmas de Programación

Profesor: Roberto González I.

Octubre 2023

# Tabla de Contenido

## Introducción

En la actualidad el masivo aumento del uso de las tecnologías digitales en casi cualquier ámbito de la vida cotidiana ha llevado a que el uso de un chatbot puede ser algo muy útil para cualquiera que quiera automatizar sus procesos, debido a que este una vez creado e implementado este en una web, aplicación, etc. Permite interactuar con usuarios a tiempo completo y entregar respuestas a estos de manera casi instantánea, a parte tampoco significa un costo extra a quien esté buscando implementarlo para el uso particular que le quiera dar, es por estoque en los últimos tiempos la "fama" e interés sobre los chatbots a aumentado de gran manera.

## Descripción del problema

Para efectos de este trabajo precisamente el problema que se busca solucionar es la creación de un sistema de chatbots que permita a un usuario interactuar con este y luego ser derivado a un chatbot especialista en el tema que quiera el usuario. Siendo más preciso lo que se busca es lograr diseñar, crear este sistema de chatbots mediante el lenguaje de programación Racket, el cual es un dialecto de Scheme, lenguajes que si bien no son puros en cuanto al paradigma de programación funcional que estos implementan, son bastante apegados a este, por sobre otros lenguajes, por lo tanto lo que se desea lograr es la creación del sistema de chatbots intentando aprovechar al máximo todas las "bondades" del lenguaje de programación Racket, sin despegarse del paradigma de programación funcional.

## Descripción del paradigma

Tal como fue mencionado anteriormente el paradigma sobre el cual se va a basar la solución propuesta próximamente es en el paradigma funcional, el cual se basa principalmente en pensar, considerar todo como funciones ya que estas son la unidad de abstracción básica del paradigma. Las funciones como tal las podríamos considerar como especies de "cajas negras" a las cuales se les entrega elementos y esta devuelve otro elemento, respetando el principio de dominio y recorrido, el cual establece el tipo de elementos que puede "entrar" y el tipo que "sale", también respetando que cada elemento tiene una salida única que pertenece siempre al tipo de elemento que "sale". Algunas consideraciones extra a la hora de hablar del paradigma funcional es que este pertenece a los paradigmas declarativos, los cuales se centran en el que hay que solucionar más que en el cómo solucionarlo, aparte de usar el cálculo lambda el cual establece la definición de función, indicando tanto los parámetros de la función, como el procedimiento mediante notación prefija, es decir, anteponiendo el operador a los operandos, un ejemplo puede ser $(\lambda\, n\, .+\, n\, 1)$, donde "n" es el parámetro, "+" el operador, "1" y "n" los operandos, siendo así una función donde "entra" un número "n" y "sale" su sucesor.

## Análisis del problema

Al profundizar en el problema, cabe mencionar que la creación de este sistema de chatbots tiene algunos requisitos específicos, relacionados a las implementaciones de los tipos de datos abstractos (TDAs) y las funciones requeridas. Respecto a los constructores de estos TDAs se ha establecido un dominio en particular para cada uno, limitando así la posible especificación de cada uno de estos TDAs, también al tratar de resolver el problema en el lenguaje Racket el cual basa casi todo en listas, las cuales son estructuras recursivas este mismo aspecto es algo presente en los requisitos específicos de cada función, teniendo que estar basada en recursión o ser mas bien declarativa, es decir, aprovechando directamente las funciones ya implementadas en el lenguaje. Otro punto importante es que este trabajo se basa en poner en práctica los conceptos del paradigma de programación funcional, es por esto que a pesar de que el lenguaje Racket tiene funciones que implementan otros paradigmas de programación distintos al funcional, no se ara uso de estas, funciones tales como "set!" la cual simula una variable, algo que no es propio del paradigma de programación funcional, ya permite la mutabilidad de un elemento, es por esto lo que si se puede hacer es definir como constante una función evaluada con sus parámetros correspondientes.

## Diseño de la solución

Respecto al diseño de solución, es importante comenzar mencionando que el principal enfoque fue la funcionalidad de tanto los TDAs como las propias funciones, es decir, lo principal fue lograr que cada función y TDA cumpliera su objetivo "como fuera" por sobre otros aspectos, respetando los requisitos específicos de cada función y TDA, lo cual se puede considerar como enumeración explicita o más comúnmente conocido como "fuerza bruta".

En relación a la descomposición del problema lo primero realizado fue intentar graficar un modelo del sistema para comprenderlo visualmente y entender en que lugar del modelo va cada estructura y cuales su función, cada estructura fue representada mediante un TDA distinto, en este caso los diseñados y empleados fueron el TDA Option, Flow, Chatbot, System, User, Chathistory, siendo tanto Option como User y Chathistory los TDAs más "simples" se podría decir, debido a que están compuestos netamente de estructuras y tipos de datos nativos del lenguaje, mientras que Flow, chatbot y system están compuestos de otros TDAs, como se puede ver en la imagen 1 en la sección "Anexos".

Respecto a algoritmos o técnicas usados para alunas solucionar problemas particulares, no se usaron ningunas en particular más que recursión en caso de estar indicado por los requerimientos específicos, y a la hora de diseñar los constructores de cada TDA, debido a la necesidad de verificar que tanto en el TDA Flow, Chatbot y System era necesario verificar que cada "sub-tda" tuviera un id único en cada "contenedor de estos, por lo tanto la opción en la cual fue más fácil lograr esta tarea fue diseñar funciones recursivas.

Por último lo que fue de más ayuda durante el diseño de la solución a la hora de implementarla fue revisar la mayoría de las funciones que ofrece el lenguaje, en especial las

que trabajan con listas, ya que esta fue la estructura básica para la implementación de cada TDA.

## Aspectos de implementación

Respecto a la implementación esta fue desarrollada en el ide DrRacket en su versión 8.10, el cual es para programar en el lenguaje Racket, en el desarrollo de este no fue usada ninguna biblioteca externa, solo funciones nativas del lenguaje y la estructura de la implementación del proyecto es 6 archivos con las funciones extra no obligatorias correspondientes a cada TDA, 1 archivo principal llamado "main" el cual contiene la implementación de las funciones obligatorias, cada uno de estos archivos contiene la respectiva documentación de cada función dentro de estos, y por último un archivo de pruebas el cual contiene un script para probar la implementación contextualizada, con comentarios de como debería funcionar cada cosa.

## Instrucciones de uso

Relacionado a las instrucciones de uso es importante dejar en claro que se asumió que el usuario va a ingresar cosas coherentes y respetar los dominios de cada función, es decir, por ejemplo en flows de un chatbot se van a ingresar efectivamente "n" Flow y no options, o que a la hora de buscar mensajes cuando se conversa con el sistema se ingresan mensajes relacionados o los números de las opciones son correlativos y siempre comienzan en 1, al igual que los ids de los TDAs. Luego de mencionar lo anterior se espera como resultado que el usuario pueda usar el cien por ciento de las funcionalidades obligatorias, sin errores siempre que siga lo mencionado anteriormente. (Ejemplo de uso en imagen 2 en la sección "Anexos")

## Resultados y Autoevaluación

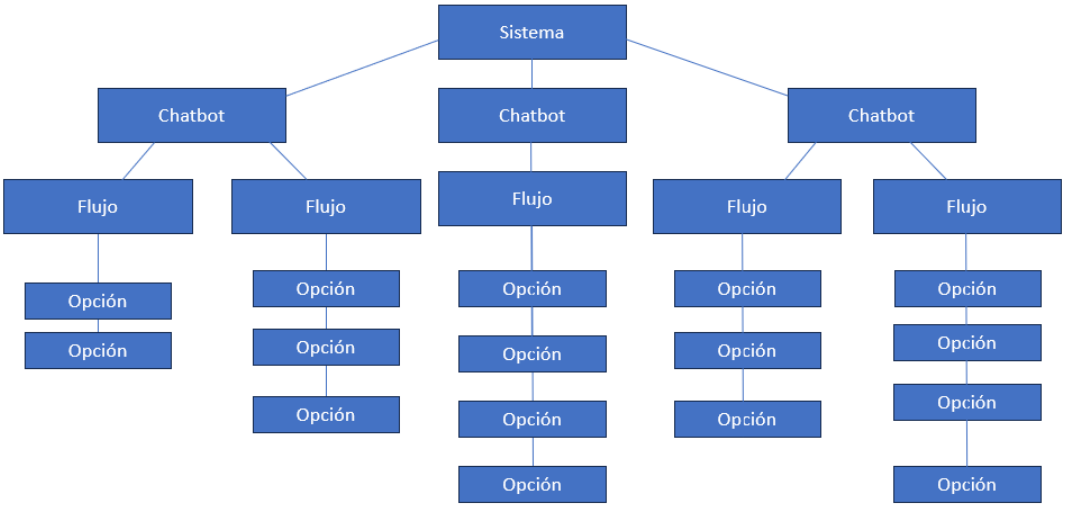
Los resultados alcanzados fueron bastante satisfactorios, ya que se logro implementar todas las funciones obligatorias de buena manera, sin ninguna dificultad mayor, considerando el grado de dificultad inherente a cada función y cumpliendo los requisitos específicos para cada una de estas, aparte se logró implementar funciones no obligatorias como funciones de pertenencia que permitieron poner en práctica tanto el paradigma como los conceptos aprendidos más en profundidad y a su vez facilitaron la implementación de otras funciones.(Tabla 1 de autoevaluación en la sección "Anexos")

## Conclusión

Finalmente a modo de conclusión se puede decir que se logró solucionar el problema de buena manera, como se mencionaba en el apartado "Resultados y Autoevaluación", es importante considerar que el alcance la solución no es muy grande, debido a que se simula de buena manera y funciona, pero de manera relativamente primitiva, netamente funciones,

lo cual no permite una interacción fluida con el usuario, aparte como limitación cabe destacar de que al trabajar en el lenguaje Racket y usar principalmente listas para la mayoría, implementar un sistema masivo, cercano a lo que podría ser algo más real este no funcionaría de buena manera debido a la recursión presente en las funciones nativas del lenguaje, Con respecto al paradigma en un principio fue difícil en un comienzo comenzar a desarrollar la solución ya que descomponer en una función hasta los más básico que era necesario implementar no es algo fácil y menos el no uso de variables, debido a la costumbre de programar bajo otros paradigmas de programación.

## Anexos




*(Imagen 1: Representación del sistema)*

```
(define my-op34(option 1 "1)Cambia el valor del dolar" 3 2  "cambio" "DOLAR" "Valor"))
(define my-op35(option 2 "2)Proyecto para reducir la inflación" 3 2 "reducir" "INFLACION"))
(define my-op36(option 3 "3)Volver" 3 1  "volver" "ATRAS" "Regresar"))

(define my-op37(option 1 "1)Mundial de Futbol 2026" 3 3 "MUNDIAL" "futbol" "2026"))
(define my-op38(option 2 "2)Juegos Olimpicos 2024" 3 3"juegos" "OLIMPIADAS" "Olimpicos" "JJOO" "2024"))
(define my-op39(option 3 "3)Volver" 3 1  "volver" "ATRAS" "Regresar"))

(define my-flow01 (flow 1 "Flujo Inicial Chatbot0\n Bienvenido\n ¿Qué tipo de noticia quieres ver?" my-op01 my-op01 my-op02 my-op01 my-op03)

(define my-flow11(flow 1 "Flujo 1 Chatbot1\n Bienvenido\n ¿De dónde quieres ver noticias?" ))

(define my-flow12(flow 2 "Flujo 2 Chatbot1\n Bienvenido\n ¿Que noticia quieres ver?" my-op16 my-op17 my-op18 ))

(define my-flow21(flow 1 "Flujo 1 Chatbot2\n Bienvenido\n ¿De dónde quieres ver noticias?" my-op21 my-op22 my-op23 my-op24))

(define my-flow22(flow 2 "Flujo 2 Chatbot2\n Bienvenido\n ¿Que noticia quieres ver?" my-op25 my-op26 my-op27 ))

(define my-flow31(flow 1 "Flujo 1 Chatbot3\n Bienvenido\n ¿Que tema te interesa?" my-op31 my-op32 my-op33 ))

(define my-flow32(flow 2 "Flujo 2 Chatbot3\n Bienvenido\n ¿Que noticia quieres ver?" my-op34 my-op35 my-op36 ))

(define my-flow33(flow 3 "Flujo 3 Chatbot3\n Bienvenido\n ¿Que noticia quieres ver?" my-op37 my-op38 my-op39 ))

(define my-flow010 (flow-add-option my-flow01 my-op01))

(define my-flow111 (flow-add-option my-flow11 my-op11))
(define my-flow112 (flow-add-option my-flow111 my-op12))
(define my-flow113 (flow-add-option my-flow112 my-op13))
(define my-flow114 (flow-add-option my-flow113 my-op14))
(define my-flow115 (flow-add-option my-flow114 my-op15))

(define my-cb0 (chatbot 0 "Inicial" "Bienvenido\n" 1 my-flow010 my-flow010))

(define my-cb1 (chatbot 1 "Noticias Nacionales" "Bienvenido\n" 1 my-flow115 my-flow12 my-flow010))
```

*(Imagen 2: Algunas funciones en uso con sus parámetros correspondientes)*

| N° Requerimiento funcional | Puntaje |
|---|---|
| 1 | 1 |
| 2 | 1 |
| 3 | 1 |
| 4 | 1 |
| 5 | 1 |
| 6 | 1 |
| 7 | 1 |
| 8 | 1 |
| 9 | 1 |
| 10 | 1 |
| 11 | 1 |
| 12 | 1 |
| 13 | 1 |
| 14 | 1 |
| 15 | 1 |

*(Tabla 1: Puntajes de autoevaluación)*