**UNIVERSIDAD DE SANTIAGO DE CHILE**

FACULTAD DE INGENIERÍA
Departamento de Ingeniería Informática


# Paradigmas de Programación

# Laboratorio N°3: Paradigma Orientado a Objetos

## Lenguaje Java

# Simulador sistema de Chatbots

Realizado por: Javier Salas Mardones
Profesor: Víctor Flores Sánchez
Sección: B-2

Santiago, 11 de diciembre de 2023

Paradigmas de Programación
Laboratorio N°3
Simulador sistema de Chatbots

Índice de contenido


**1 Introducción** **2**
**2 Descripción del problema** **2**
**3 Descripción del paradigma** **3**
**4 Análisis del problema** **4**
**5 Diseño de la solución** **5**
**6 Aspectos de implementación** **7**
**7 Instrucciones de uso** **8**
**8 Resultados y autoevaluación** **9**
**9 Conclusiones** **10**
**10 Referencias** **10**
**Anexos** **11**
Tabla de resultados y autoevaluación: 11
Instrucciones de uso 12
Diagramas 16

# 1 INTRODUCCIÓN

Este informe presenta el trabajo realizado en el laboratorio semestral de la asignatura de Paradigmas de Programación en la Universidad de Santiago de Chile, donde el objetivo central es explorar el paradigma de programación orientado a objetos mediante el uso del lenguaje Java.

El proyecto consiste en desarrollar un sistema para la creación, despliegue y administración de chatbots simplificados. Este sistema tiene como finalidad no solo poner a prueba nuestras habilidades en programación orientada a objetos, sino también comprender cómo las distintas formas de programación pueden ser aplicadas en soluciones tecnológicas prácticas y relevantes en el mundo actual.

En este contexto, el proyecto tiene como reto la construcción de un software que a través de un menú se permita al usuario interactuar con chatbots. Estos chatbots son de Interacción de Respuesta a Texto (ITR), se caracterizan por su estructura basada en opciones específicas y una interacción limitada en cuanto al procesamiento del lenguaje natural, evitando el uso de inteligencia artificial avanzada.

El informe tiene la siguiente estructura: Primero, se tiene una descripción breve del problema, luego la descripción del paradigma, el análisis del problema, diseño de la solución, aspectos de implementación, instrucciones y ejemplos de uso, resultados y autoevaluación y finalmente una conclusión del proyecto de laboratorio realizado.

# 2 DESCRIPCIÓN DEL PROBLEMA

El problema central por abordar en este proyecto se enfoca en el desarrollo de un sistema eficaz y funcional para la creación, despliegue y administración de chatbots simplificados, utilizando el paradigma de programación orientado a objetos (POO) con el lenguaje Java. La dificultad principal radica en simular un sistema interactivo de chatbots que, a pesar de su estructura simplificada, debe ser lo suficientemente versátil para permitir a los usuarios realizar una variedad de operaciones, como crear, vincular e interactuar con estos chatbots.

En este contexto, el desafío principal es diseñar una estructura de clases y objetos que represente de manera efectiva la interacción entre el usuario y los chatbots, garantizando una interfaz intuitiva y fácil de usar. La programación orientada a objetos se convierte en la base fundamental para la creación de chatbots que pueden ser instanciados, configurados y manipulados mediante la creación de objetos y el uso de métodos y propiedades relacionados.

Por lo tanto, el problema a resolver se centra en cómo diseñar e implementar un sistema de chatbots en Java que aproveche al máximo las ventajas del paradigma orientado a objetos, proporcionando una experiencia intuitiva para los usuarios finales mientras se mantiene una estructura lógica clara y eficiente en el código.

# 3 DESCRIPCIÓN DEL PARADIGMA

El paradigma de programación orientado a objetos (POO), que se ejemplifica en este proyecto mediante el uso del lenguaje de programación Java, representa una metodología fundamentalmente diferente a los enfoques de programación lógica o funcional. Este paradigma se centra en la organización de código en torno a objetos y clases, donde un objeto es una instancia de una clase que encapsula datos y comportamiento relacionados. La programación orientada a objetos promueve la reutilización de código, la modularidad y la abstracción. A continuación se describen algunos conceptos clave del paradigma orientado a objetos:

- **Clases:** En POO, las clases son estructuras fundamentales que definen los atributos y métodos de los objetos. Representan un "molde" para la creación de objetos. En nuestro proyecto, las clases son utilizadas para definir la estructura y el comportamiento de los chatbots**.**

- **Objetos**: Los objetos son instancias de clases específicas. Representan entidades con datos y comportamiento. En el contexto del proyecto, por ejemplo los chatbots son objetos que pueden ser creados, configurados y utilizados por los usuarios.

- **Encapsulación**: La encapsulación es el principio de ocultar los detalles internos de una clase y exponer solo una interfaz pública. Esto promueve la modularidad y el acceso controlado a los datos y métodos de un objeto, solicitando información o verificando si ciertas condiciones se cumplen.

- **Herencia**: La herencia permite que una clase herede propiedades y métodos de otra clase, lo que fomenta la reutilización de código y la creación de jerarquías de clases. En el contexto del proyecto, la herencia se utiliza para la clase usuario que se divide en usuario normal y usuario administrador.

- **Polimorfismo**: El polimorfismo permite que objetos de diferentes clases respondan de manera diferente a los mismos métodos. En el contexto del proyecto, se tiene una lista de usuarios y gracias al polimorfismo se pueden almacenar objetos de tipo “usuario administrador” y “usuario normal”.

- **Clase abstracta**: se utiliza como un tipo de plantilla o modelo para otras clases. A diferencia de una clase regular, una clase abstracta no se puede instanciar directamente para crear objetos. En su lugar, se define para que otras clases la hereden y proporciona una estructura común y métodos que deben ser implementados por las clases derivadas. En el contexto del proyecto, la clase usuario es abstracta por lo que tiene métodos no implementados que se tuvieron que implementar en las clases que lo heredan (normal y administrador).

En resumen, el paradigma orientado a objetos proporciona un enfoque de diseño y desarrollo que se basa en la creación de objetos con comportamiento y relaciones definidos, lo que facilita la construcción de un sistema de chatbots intuitivo y eficiente en Java.

# 4 ANÁLISIS DEL PROBLEMA

Como se mencionó anteriormente, se necesita realizar un sistema de chatbots simplificado, que permita realizar operaciones para modificar e interactuar con el sistema. Para lograr esto debemos crear distintos tipos de datos que nos sirvan de base para hacer los requerimientos funcionales solicitados.

En primer lugar, debemos considerar que el sistema debe permitir un ambiente contenedor de chatbots, esto implica la capacidad de gestionar múltiples chatbots de manera simultánea y la creación de sus respectivos flujos de interacción. Además de la creación e identificación de los mismos chatbots en donde cada chatbot tiene un propósito específico. También el sistema debe facilitar la inclusión de preguntas y opciones dentro de cada chatbot, las cuales sirven para las posibles interacciones que un usuario pueda tener con el chatbot mediante palabras clave que se vinculan a opciones predefinidas, el sistema está representado de la siguiente forma:

**Sistema**: Una lista que contiene un nombre, un código inicial que conecta a un chatbot, una lista de chatbots, una lista de usuarios registrados, una lista de historial y un string que representa al usuario que está logueado o no.

Luego, algunos elementos importantes que están dentro del sistema:

- **Usuarios**: Corresponden a aquellos que interactúan con el sistema, registrándose para poder realizar las operaciones disponibles.
- **Chatbot**: Permiten simular la conversación humana, actúa como el punto de interacción principal, pueden ser personalizados para diferentes propósitos.
- **Flow**: Corresponde al flujo o secuencia estructurada de interacciones dentro de un chatbot. Cada flujo representa un camino de conversación especifico.
- **Option**: Corresponden a las posibles opciones que puede poseer un flujo que representan las posibles elecciones o respuestas que un usuario puede seleccionar durante la interacción con un chatbot.

Finalmente, algunas de las operaciones que se pueden realizar en el sistema:

- **Creación de Chatbot** (chatbot - constructor): Esta operación permite crear un nuevo chatbot dentro del sistema. Involucra definir atributos clave como el ID del chatbot, nombre, mensaje de bienvenida, ID del flujo inicial, y una lista de flujos asociados. Esta función es esencial para iniciar la estructura de interacción del chatbot con los usuarios.

- **Añadir Flujo a Chatbot** (chatbotAddFlow): Permite expandir la funcionalidad de un chatbot existente añadiendo nuevos flujos. Esta operación es crucial para aumentar la complejidad y la profundidad de las conversaciones que el chatbot puede manejar, permitiendo una mayor variedad en las interacciones con el usuario.

- **Crear Opción en un Flujo** (option - constructor): Esta función se utiliza para crear opciones dentro de un flujo específico. Cada opción incluye un código único, un mensaje asociado,

vínculos a chatbots y flujos específicos, y una lista de palabras clave. Las opciones son fundamentales para guiar la dirección de la conversación en el chatbot.

- **Modificar Flujo Añadiendo Opciones** (flowAddOption): Esta operación permite modificar un flujo existente añadiendo nuevas opciones. Es vital para la adaptabilidad y actualización de los chatbots, permitiendo modificar y enriquecer las interacciones disponibles para los usuarios a lo largo del tiempo.

- **Iniciar Sesión en el Sistema** (systemLogin): Permite a un usuario registrado iniciar una sesión en el sistema. Esta función es clave para personalizar la experiencia del usuario, manteniendo un historial de interacciones y preferencias individuales.

- **Cerrar Sesión en el Sistema** (systemLogout): Esta operación finaliza la sesión activa de un usuario, asegurando la privacidad y la seguridad de la información del usuario. Es esencial para gestionar el acceso y mantener la integridad del sistema.

# 5 Diseño de la solución

Luego de analizar y comprender el problema planteado comenzamos con la creación de las diferentes clases, las cuales son “system”, “chatbot”, “flow”, “option”, “user”, “menú”.

### 1. CLASE SYSTEM

- Representación: Tiene un nombre (String) que almacena el nombre del sistema. Un código inicial (Int) que representa el código de inicio que conecta a un chatbot específico. Una lista de chatbots, que es una colección que contiene objetos de tipo Chatbot que representan los chatbots disponibles en el sistema. Una lista de usuarios registrados, que es una colección que almacena objetos de tipo User que representan a los usuarios registrados en el sistema. La fecha de creación del sistema (Date). Una lista de historial, que es una colección que registra las interacciones previas entre los usuarios y los chatbots. Un usuario de tipo usuario que representa si hay alguien logueado en el sistema.

### 2. CLASE CHATBOT

- Representación: Tiene un código identificador (Int) que sirve como identificación única del chatbot. Un nombre (String) que almacena el nombre del chatbot. Un mensaje de bienvenida (String) que contiene el mensaje de bienvenida que muestra el chatbot cuando se inicia la conversación. Un código identificador de flujo inicial (Int) que representa el código de inicio que conecta al chatbot con su flujo inicial. Una lista de flujos, que es una colección que almacena objetos de tipo Flow que representan los flujos de conversación disponibles para el chatbot.

### 3. CLASE FLOW

- Representación**:** Tiene un identificador (Int) que sirve como identificación única del flujo. Un nombre del flujo (String) que almacena el nombre del flujo. Una lista de opciones, que es una colección que contiene objetos de tipo Option que representan las opciones de respuesta disponibles en el flujo.

### 4. CLASE OPTION

- Representación**:** Tiene código identificador (Int) que sirve como identificación única de la opción. Un mensaje (String) que almacena el mensaje que muestra la opción al usuario. Un código que conecta con un chatbot (Int) que representa el código de inicio que conecta con un chatbot específico si el usuario selecciona esta opción. Un código de flujo inicial (Int) que representa el código de inicio del flujo que se activa si el usuario selecciona esta opción. Una lista de palabras clave, que es una colección que contiene palabras clave relacionadas con la opción para facilitar la búsqueda y emparejamiento.

### 5. CLASE USER

- Representación: Tiene un nombre de usuario (String) que almacena el nombre de usuario. Un estado de logueado (Boolean) que representa si es que ese usuario está logueado en el sistema o no. Esta al ser una clase abstracta las clases hijas heredan de esta, las cuales corresponden a “Usuario Común” y “Usuario Administrador”.

### 6. CLASE MENÚ

- Representación: Representa una interfaz de usuario para interactuar con el sistema de chatbots. Utiliza la clase Scanner para recibir la entrada del usuario desde la consola.

Cabe destacar que dentro de la clase menú se encuentras distintos submenús (métodos), entre los cuales están: ejecutarMenu, ejecutarMenuRegistro, mostrarMenuPostLogin, mostarMenuAdmin, mostrarMenuNormal. Además, en la clase Main existe un sistema con datos ya cargados por defecto y la opción de crear un sistema, que pide como entrada un nombre para el sistema y el código que conecta con el chatbot inicial que a partir del diseño de la solución se asume con código cero, ya que el método systemTalk posee la lógica de que el chatbot inicial tiene código cero, el cual se debe tener en consideración para la posterior creación de nuevos chatbots. Otro punto a tener en cuenta es que dentro de las opciones una es capaz de contener gran parte de los requerimientos funcionales solicitados como lo hace la creación de un chatbot en la que se utilizan, la clase Option (constructor) (RF. 1), el método constructor de la clase Flow (RF. 2), el método modificador flowAddOption (RF. 3), el método constructor de un chatbot (RF. 4), el método modificador de un chatbot chatbotAddFlow (RF. 5) y finalmente el método modificador de un sistema systemAddChatbot (RF. 8).

# 6 ASPECTOS DE IMPLEMENTACIÓN

La implementación del proyecto está estructurada en 9 archivos de código: clase System, clase Chatbot, clase Flow, clase Option, clase User (abstracta), clase Usuario Común, clase Usuario Administrador, clase Menú en un package de Vista y por último la clase Main, en donde se ejecutará el menú.

En el apartado de anexos se podrá ver como las clases se relacionan entre sí.

Para realizar este laboratorio se utilizó el lenguaje de programación Java, usando el JDK versión 11, específicamente Temurin-11.0.19+7, mediante el IDE IntelliJ IDEA en su versión 2023.1.2. La mayoría de requerimientos fueron implementados mediante la biblioteca estándar de Java, utilizando principalmente el package java.util. Por último, se utilizó el sistema operativo Windows 11 Home en su versión 23H2.

# 7 INSTRUCCIONES DE USO

Dado que se integró la herramienta Gradle al proyecto, para ejecutarlo se debe abrir un cmd o PowerShell en la raíz del proyecto y ejecutar el comando:

"**gradlew run --console=plain**" (cmd)
"**./gradlew run --console=plain**" (PowerShell)

El añadido **"--console=plain**" es simplemente para eliminar las barras de carga que presenta Gradle, por lo tanto esto es opcional.

Una vez ejecutado el programa se desplegará, un menú con 3 opciones, la primera opción pregunta al usuario si desea interactuar con un sistema que posee información predeterminada la cual corresponde a; dos usuarios (un administrador "Admin" y un usuario común "Javier"), opciones, flujos y chatbots ya definidos en el Main, en la segunda opción se le ofrece al usuario interactuar con un sistema "vacío", que solo contiene un nombre, un código que conecta con el chatbot inicial (0) y una lista vacía en donde el usuario del sistema (se debe registrar previamente) puede ir creando chatbots, flujos y opciones, esto lo podrá hacer cuando se ingrese al sistema como usuario administrador y como tercera opción se dará la posibilidad de terminar la ejecución del programa.

A continuación, una explicación a grandes rasgos del menú interactivo:

Independientemente de que se escoja interactuar con el sistema cargado por defecto o el sistema "vacío" se desplegará otro menú que tiene relación con los usuarios, allí se puede iniciar sesión o registrar un usuario que puede ser de tipo administrador o un usuario normal, los cuales también tienen un menú propio dependiendo del tipo de usuario que ingrese al sistema, estos difieren en los métodos/funciones que pueden realizar en el sistema. Por ejemplo, el usuario administrador puede crear un chatbot, en cambio el usuario normal solo puede interactuar con los chatbots que estén disponibles en el sistema. Además, en cada parte de los menús creados hay opciones para salir del sistema o regresar si así se desea, así como en el primer menú se puede terminar la ejecución del programa.

En el apartado de anexos se podrá ver algunas imágenes utilizando el menú.

# 8 RESULTADOS Y AUTOEVALUACIÓN

Se espera crear una simulación de un sistema de chatbots, en donde el programa sea funcional. Las posibles fallas pueden suceder a partir de errores de tipeo ya que, si se ingresan valores erróneos en la creación de chatbots, flujos u opciones, estas terminan con la ejecución del programa, ya que, no todas las entradas están con manejo de excepciones.

En el apartado de anexos (**Tabla 1**) se puede observar la autoevaluación del laboratorio. Los requerimientos funcionales implementados fueron probados con diferentes ejemplos para verificar su correcto funcionamiento, luego según los resultados obtenidos son evaluados con la escala de autoevaluación proporcionada por la asignatura que va desde 0 hasta 1.

# 9 Conclusiones

Se considera que el laboratorio logró un alcance medianamente aceptable, ya que se logró implementar la mayoría de los requerimientos funcionales solicitados de manera exitosa (hasta "SystemSynthesis"). La implementación demostró la eficacia de la programación orientada a objetos (POO).

Uno de los principales desafíos fue la transición desde el enfoque declarativo de Prolog hacia el enfoque orientado a objetos de Java. El paradigma de POO se enfoca en la implementación de estructuras y patrones de diseño orientados a objetos para gestionar las interacciones y los estados de los chatbots. A diferencia del enfoque de Prolog, que se centra en "qué" debe hacerse en lugar del "cómo", la programación orientada a objetos se basa en la encapsulación, herencia y polimorfismo para estructurar respuestas basadas en opciones específicas. A su vez el contraste entre el paradigma orientado a objetos y el paradigma funcional, por ejemplo, en el paradigma POO los objetos son mutables, lo que significa que sus estados pueden cambiar después de su creación, en cambio en el paradigma funcional los datos son inalterables una vez creados, lo que simplifica la programación concurrente y la depuración.

A pesar de que el proyecto en Java no se basa en el procesamiento avanzado del lenguaje natural (NLP), ofrece una plataforma para estructurar respuestas basadas en opciones específicas, aprovechando las características de la POO. Esto requiere una programación orientada a objetos precisa y eficiente para manejar interacciones basadas en menús y opciones predefinidas, manteniendo una clara estructura de código y una lógica de interacción intuitiva para los usuarios finales. La adaptación a este nuevo paradigma implicó un cambio significativo en el enfoque de resolución de problemas, pero permitió una implementación efectiva de los requerimientos funcionales del proyecto en Java.

# 10 Referencias

1. Leiva, E. & Martínez, G. (2023). "*Proyecto semestral de laboratorio*". Paradigmas de programación.https://docs.google.com/document/d/1Psn6YqXfWA99n3AA-rLsvAJsc2-gqj6ot7j90ucLcog/edit

# ANEXOS

## TABLA DE RESULTADOS Y AUTOEVALUACIÓN:

*Tabla 1: "Tabla de resultados y autoevaluación".*

| Requerimiento | Alcance | Prueba | Fallos | Razón Fallo | Puntaje |
|---|---|---|---|---|---|
| Clases y estructuras | Realizado | Se crean las distintas clases y estructuras | 0 | NA | 1 |
| Menú interactivo por terminal | Realizado | Se crea menú interactivo funcional | 0 | NA | 1 |
| option | Realizado | Se crean distintas opciones | 0 | NA | 1 |
| flow | Realizado | Se crean distintos flujos | 0 | NA | 1 |
| flowAddOption | Realizado | Se añaden distintas opciones a un flujo | 0 | NA | 1 |
| chatbot | Realizado | Se crean distintos chatbots | 0 | NA | 1 |
| chatbotAddFlow | Realizado | Se añaden distintos flujos a un chatbot | 0 | NA | 1 |
| system | Realizado | Se crean distintos sistemas | 0 | NA | 1 |
| systemAddChatbot | Realizado | Se añaden distintos chatbot a un sistema | 0 | NA | 1 |
| systemAddUser | Realizado | Se añaden usuariosal sistema | 0 | NA | 1 |
| systemLogin | Realizado | Un usuario inicia sesión en el sistema | 0 | NA | 1 |
| systemLogout | Realizado | Un usuario cierra sesión en el sistema | 0 | NA | 1 |
| systemTalk | Realizado | Un usuario interactúa con chatbots | 0 | NA | 1 |
| systemSynthesis | Realizado | Se muestra un resumen de interacción del usuario con los chatbots | 0 | NA | 1 |
| systemSimulate | No Realizado | NA | NA | NA | 0 |

## Instrucciones de uso

- El menú principal se muestra de esta forma:

```
> Task :run
Usuario Admin registrado con exito.
Usuario Javier registrado con exito.
Desea interactuar con un sistem vacio o interactuar con el sistema creado por defecto?
1. DIGITAR NUMERO (1) PARA SISTEMA CARGADO POR DEFECTO:
2. DIGITAR NUMERO (2) PARA CREAR SISTEMA VACIO (SOLO SE INGRESA NOMBRE Y ENLACE DE CODIGO CON CHATBOT INICIAL):
3. DIGITAR NUMERO (3) PARA TERMINAR EJECUCION
INTRODUZCA SU OPCION:
1
```

- Si se selecciona la opción 1 se desplegará otro menú que tiene relación con los usuarios.

```
### Sistema de Chatbots - Inicio ###
1. Login de Usuario
2. Registro de Usuario
3. Salir
INTRODUZCA SU OPCION:
1
```

- Si se selecciona la opción 1 se debe ingresar el nombre del usuario para iniciar sesión (recordar que se usó el sistema con los datos ya cargados).

```
### Sistema de Chatbots - Login ###
Ingrese el nombre de usuario para iniciar sesion:
Admin
```

- Se inicia sesión en el sistema como administrador, y tiene una serie de opciones para realizar, en este caso se selecciona la opción 3 (Ejecutar System Talk) y se ingresa un saludo inicial para la correcta ejecución del método.

```
Usuario administrador: Admin ha iniciado sesion
Inicio de sesi├│n exitoso para Admin
### Sistema de Chatbots - Usuario Administrador ###
Bienvenido Admin, usted es administrador.
1. Crear un Chatbot
2. Modificar un Chatbot
3. Ejecutar System Talk
4. Ejecutar System Synthesis
5. Ejecutar System Simulate (METODO NO IMPLEMENTADO)
6. Salir y cerrar sesion
INTRODUZCA SU OPCION:
3
INGRESE UN SALUDO INICIAL PARA INTERACTUAR CON CHATBOTS:
Hola
```

- Luego del saludo inicial se selecciona la opción “comer”.

```
Para finalizar la ejecucion de systemTalk debe escribir 'terminar', esto se puede realizar en cualquier momento
Chatbot responde: Bienvenido, ┬┐qu├® deseas hacer?
Opciones disponibles:
Codigo: 1, Mensaje: 1) Cocinar (Palabras Clave: cocinar, comida, comer)
Codigo: 2, Mensaje: 2) Ejercitar (Palabras Clave: ejercitar, ejercicio, gimnasia)
Seleccione una opci├│n (ingrese el c├│digo o una palabra clave):
comer
```

- El chatbot responde, y muestra las opciones disponibles relacionadas al chatbot cocinero, luego se selecciona la opción “dulces” ingresando la palabra clave “postres”.

```
Has seleccionado: 1) Cocinar
Opciones disponibles:
Codigo: 1, Mensaje: 1) Dulces (Palabras Clave: postres, dulces, pasteles)
Codigo: 2, Mensaje: 2) Salados (Palabras Clave: salados, comida salada, plato salado)
Codigo: 3, Mensaje: 3) Tragos (Palabras Clave: copete, alcohol, algo para tomar)
Codigo: 4, Mensaje: 4) Volver (Palabras Clave: volver, regresar)
Seleccione una opci├│n (ingrese el c├│digo o una palabra clave):
postres
```

- Luego se despliega las opciones relacionadas a “Dulces”, finalmente se introduce la palabra “terminar” para terminar la ejecución de ese método.

```
Has seleccionado: 1) Dulces
Opciones disponibles:
Codigo: 1, Mensaje: 1) Alfajor de manjar (Palabras Clave: alfajor, manjar, alfajor de manjar)
Codigo: 2, Mensaje: 2) Torta pompadour (Palabras Clave: torta, pompadour, torta pompadour)
Codigo: 3, Mensaje: 3) Volver al men├╣ anterior (Palabras Clave: volver, regresar, men├╣ anterior)
Seleccione una opci├│n (ingrese el c├│digo o una palabra clave):
terminar
```

- Luego se selecciona la opción 6 (Salir y cerrar sesión).

```
Saliendo de la interacci├│n con el chatbot.
### Sistema de Chatbots - Usuario Administrador ###
Bienvenido Admin, usted es administrador.
1. Crear un Chatbot
2. Modificar un Chatbot
3. Ejecutar System Talk
4. Ejecutar System Synthesis
5. Ejecutar System Simulate (METODO NO IMPLEMENTADO)
6. Salir y cerrar sesion
INTRODUZCA SU OPCION:
6
```

- Luego se selecciona la opción 3, para salir del menú de los usuarios y volver al menú inicial

```
Admin ha cerrado sesion
El usuario Admin ha cerrado sesion.
### Sistema de Chatbots - Inicio ###
1. Login de Usuario
2. Registro de Usuario
3. Salir
INTRODUZCA SU OPCION:
3
```

- Al salir del sistema se muestra toda la información del sistema con las modificaciones que se le hayan realizado, y se selecciona la opción 3 para finar la ejecución del programa.

```
Saliendo del sistema ...
System_19080187_SalasMardones{name='Sistema Chatbots', initialChatbotCodeLink=0, chatbotList=[Chatbot_19080
187_SalasMardones{chatbotId=0, name='Chatbot Principal', welcomeMessage='Bienvenido, ┬┐qu├® deseas hacer?',
 startFlowId=0, flowList=[Flow_19080187_SalasMardones{id=0, nameMsg='Flujo principal Chatbot Principal', op
tionList=[Option_19080187_SalasMardones{code=1, message='1) Cocinar', chatbotCodeLink=1, initialFlowCodeLin
k=0, keywords=[cocinar, comida, comer]}, Option_19080187_SalasMardones{code=2, message='2) Ejercitar', chat
botCodeLink=2, initialFlowCodeLink=0, keywords=[ejercitar, ejercicio, gimnasia]}]}]}, Chatbot_19080187_Sala
sMardones{chatbotId=1, name='Chatbot Cocinero', welcomeMessage='Bienvenido, ┬┐qu├® te gustar├¡a cocinar?',
startFlowId=1, flowList=[Flow_19080187_SalasMardones{id=1, nameMsg='Flujo 1', optionList=[Option_19080187_S
alasMardones{code=1, message='1) Dulces', chatbotCodeLink=1, initialFlowCodeLink=2, keywords=[postres, dulc
es, pasteles]}, Option_19080187_SalasMardones{code=2, message='2) Salados', chatbotCodeLink=1, initialFlowC
odeLink=1, keywords=[salados, comida salada, plato salado]}, Option_19080187_SalasMardones{code=3, message=
'3) Tragos', chatbotCodeLink=1, initialFlowCodeLink=1, keywords=[copete, alcohol, algo para tomar]}, Option
_19080187_SalasMardones{code=4, message='4) Volver', chatbotCodeLink=0, initialFlowCodeLink=0, keywords=[vo
lver, regresar]}]}, Flow_19080187_SalasMardones{id=2, nameMsg='Recetas espec├¡ficas', optionList=[Option_19
080187_SalasMardones{code=1, message='1) Alfajor de manjar', chatbotCodeLink=1, initialFlowCodeLink=2, keyw
ords=[alfajor, manjar, alfajor de manjar]}, Option_19080187_SalasMardones{code=2, message='2) Torta pompado
ur', chatbotCodeLink=1, initialFlowCodeLink=2, keywords=[torta, pompadour, torta pompadour]}, Option_190801
87_SalasMardones{code=3, message='3) Volver al men├╗ anterior', chatbotCodeLink=1, initialFlowCodeLink=1, k
eywords=[volver, regresar, men├╗ anterior]}]}]}, Chatbot_19080187_SalasMardones{chatbotId=2, name='Chatbot
Entrenador', welcomeMessage='Bienvenido que te gustaria entrenar?', startFlowId=1, flowList=[Flow_19080187_
SalasMardones{id=1, nameMsg='Flujo 1', optionList=[Option_19080187_SalasMardones{code=1, message='1) Cardio
', chatbotCodeLink=2, initialFlowCodeLink=1, keywords=[cardio, cardiovascular, aerobico]}, Option_19080187_
SalasMardones{code=2, message='2) Fuerza', chatbotCodeLink=2, initialFlowCodeLink=1, keywords=[fuerza, pesa
s, musculo]}, Option_19080187_SalasMardones{code=3, message='3) Volver', chatbotCodeLink=0, initialFlowCode
Link=0, keywords=[volver, regresar]}]}]}], userList=[User_19080187_SalasMardones{username='Admin', loginSta
tus=false}, User_19080187_SalasMardones{username='Javier', loginStatus=false}], creationDate=11/12/23, chat
History=[Usuario: Hola, Chatbot: Bienvenido, ┬┐qu├® deseas hacer?, Usuario Admin: comer, Chatbot: Has selec
cionado: 1) Cocinar, Usuario Admin: postres, Chatbot: Has seleccionado: 1) Dulces, Usuario Admin: terminar]
, userOn=null}
Desea interactuar con un sistem vacio o interactuar con el sistema creado por defecto?
1. DIGITAR NUMERO (1) PARA SISTEMA CARGADO POR DEFECTO:
2. DIGITAR NUMERO (2) PARA CREAR SISTEMA VACIO (SOLO SE INGRESA NOMBRE Y ENLACE DE CODIGO CON CHATBOT INICI
AL):
3. DIGITAR NUMERO (3) PARA TERMINAR EJECUCION
INTRODUZCA SU OPCION:
3
Finalizando ejecucion  ...
```

Diagrama de análisis UML:

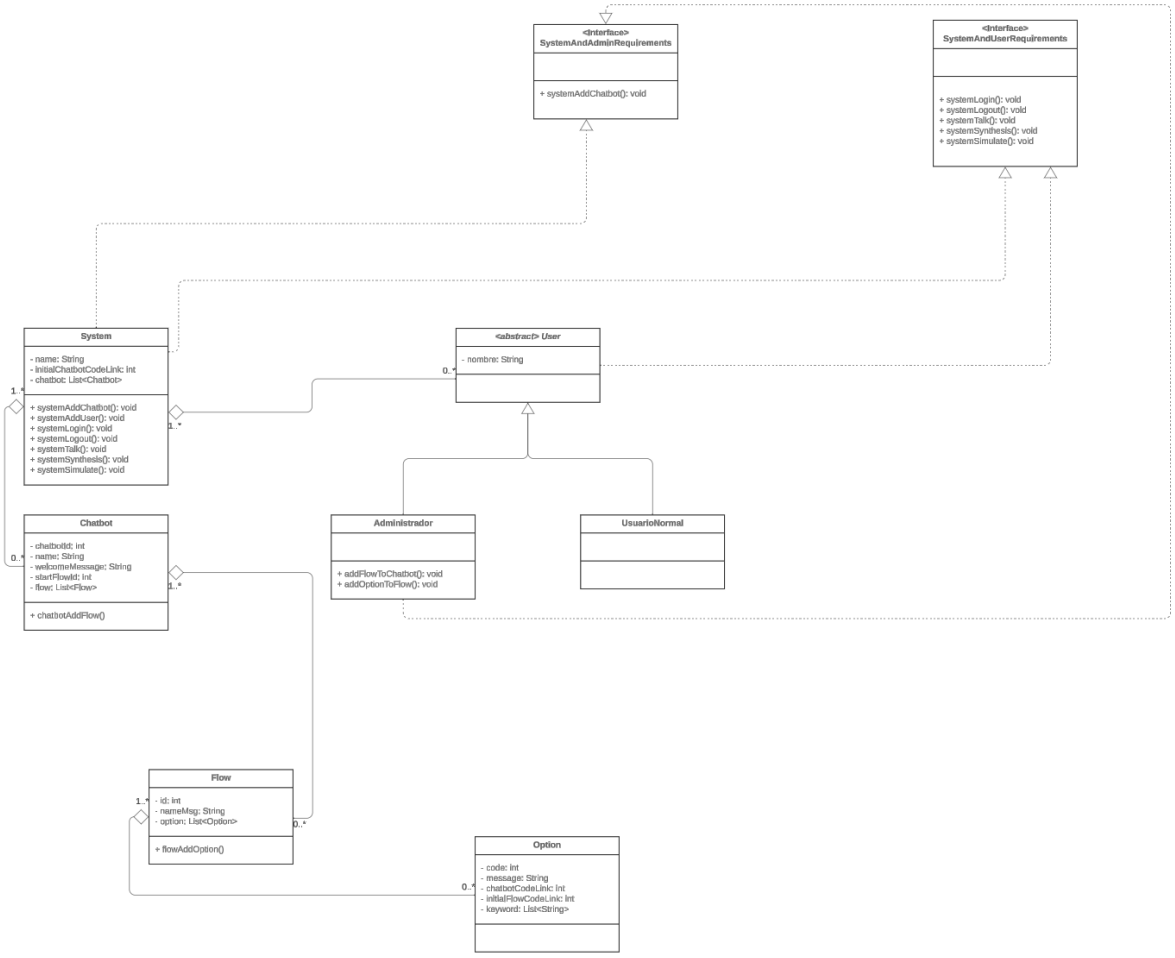


***Figura 1**: Diagrama de análisis UML*

Diagrama de diseño UML:

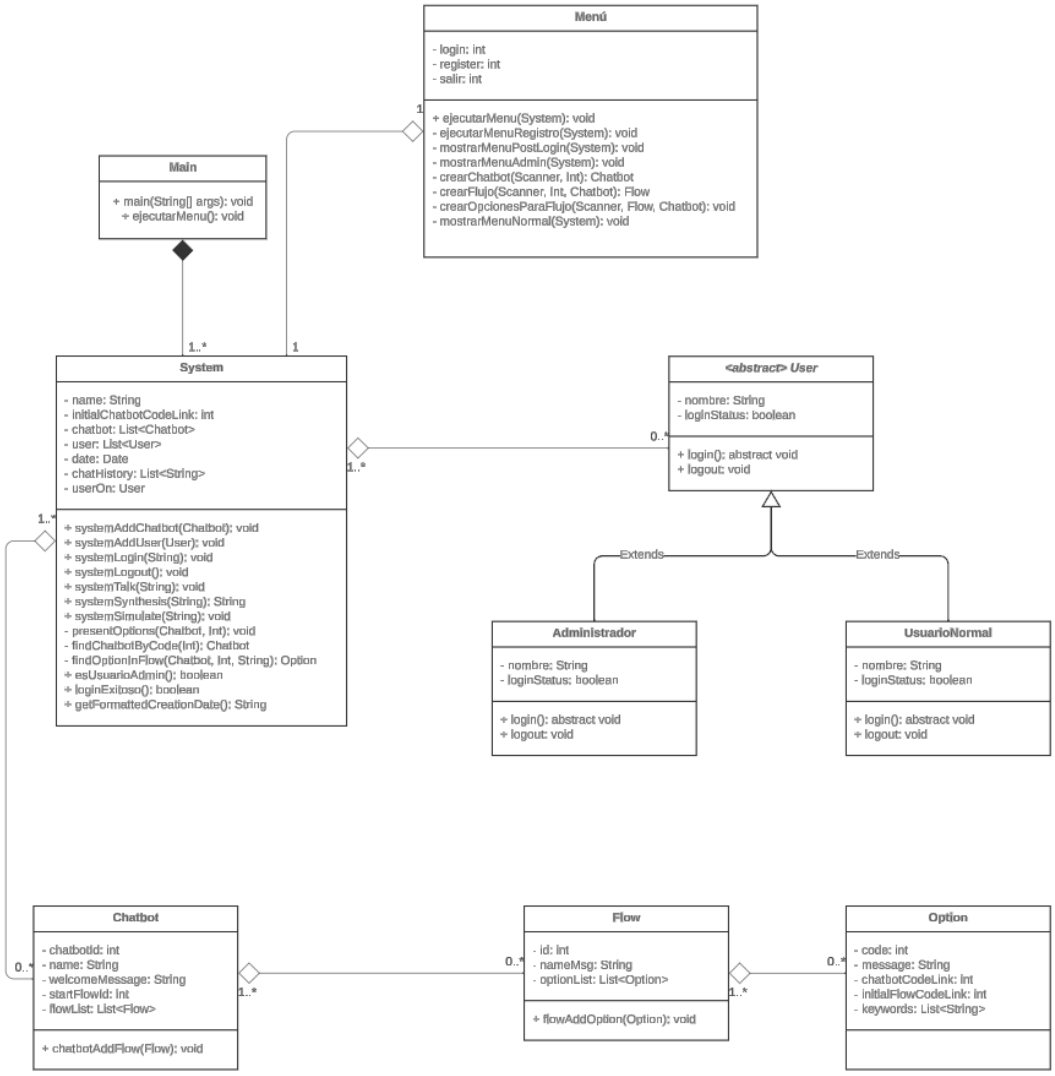


***Figura 2**: Diagrama de diseño UML*