Universidad de Santiago de Chile
Facultad de Ingeniería
Departamento de Ingeniería Informática

Informe Laboratorio 3 Paradigmas de Programación Orientado a Objetos

Nombre: Amaru Monje Rojas

Rut: 21.130.814-1

Profesor: Roberto González Ibañez

Índice

Introducción:

Este informe detalla el proceso de trabajo e ideación que se empleó en este proyecto. Esto se describe en; Descripción del problema, Descripción del Paradigma, Análisis del Problema, Diseño de la Solución, Aspectos de Implementación, Instrucciones de uso, Resultados Obtenidos, Evaluación Completa y Conclusiones.

El propósito de este proyecto es crear un ambiente de chatbots ITR (Respuesta de Interacción a Texto), a través del lenguaje de programación Java, que permita crear sistemas de chatbots e interactuar con ellos, todo esto a través de un menú mostrado por consola.

Descripción del Problema:

Se debe crear mediante programación y utilizando el paradigma orientado a objetos, un ambiente que permita crear sistemas contenedores de chatbots, crear chatbots, interactuar con uno, ofrecer una síntesis de las interacciones realizadas con un sistema y realizar simulaciones de éstas. Para esto, se debe implementar un menú por consola que permita ingresar como usuario común o administrador y entregar opciones de acuerdo con los permisos.

Descripción del Paradigma:

La programación orientada a objetos es un tipo de paradigma de programación que se basa en el concepto de clases y objetos. Permite organizar el código en piezas flexibles y reutilizables utilizando el concepto de instancias. Las clases pueden tener relaciones como la agregación, en la que una se añade a otra, o la composición que es similar a la ya mencionada pero la vida útil de una instancia de la clase compuesta es dependiente de la que compone. Además, utilizando conceptos como herencia e interfaces es posible establecer “funciones” llamadas métodos que varias clases comparten y permiten comprender mejor el funcionamiento o conexión de un programa. (Martínez, 2020)

Análisis del Problema:

Para empezar, se deben considerar las partes del problema, estos son, diseñar un ambiente que contenga chatbots, estos se deben poder modificar y manipular, también debe ser capaz de interactuar con ellos y dejar un registro para la síntesis, es decir, poder almacenar o mantener un historial. Además, se deben poder crear y estructurar tal que se pueda acceder a los datos en el código.

Considerando el paradigma en uso, se pueden definir clases para cada estructura según se necesite, también se pueden utilizar métodos para crear y modificar estas estructuras, finalmente, utilizando relaciones se puede lograr que cada clase se comuniquen con otras de forma eficiente y completa.

Diseño de la Solución:

Se optó por diseñar TDAs (Tipo de dato abstracto), para cada parte del problema, siendo estos; Menu, System, Chatbot, Flow, Option, User. Para esto se definieron clases para cada uno, además de clases especializadas las cuales son: AdminUser, CommonUser, DemoSystem, MenuSystem, UserSystem y una clase abstracta llamada BaseStruct.

Cada System contiene Chatbots y Users, un Chatbot contiene Flows y un Flow contiene Options.

El Menu contiene un DemoSystem y un MenuSystem, ambos son Systems con chatbots predefinidos y no están disponibles para modificación, la función del DemoSystem es servir como elemento de interacción base (Ver Anexo 7) y la de MenuSystem es contener las preguntas que se le realizan al Usuario por consola.

AdminUser y CommonUser son tipos de usuario que heredan los atributos y métodos de User y su única diferencia es el nivel de permiso que poseen, AdminUser puede crear, modificar e interactuar con Systems mientras que CommonUser sólo puede interactuar.

Por último, BaseStruct fue hecho para contener y heredar elementos base a System, Chatbot, Flow y Option, por ejemplo, define la id como un entero mayor o igual que 0, además define nombre y mensaje para ser usados y modificados por las otras clases.

Aspectos de Implementación:

Solo se utilizaron las librerías nativas de Java, trabajando en el Java Runtime Environment Temurin-11.0.21+9 (build 11.0.21+9) (Anexo 1) y con el JDK temurin 11 (Anexo 2).

Se separó el código en archivos para cada clase agrupándolos en 4 paquetes; clases, que contiene cada TDA, interfaz, en el que se encuentra la Interfaz Duplicidad, menu, que contiene la clase Menu y MenuSystem, y main, que contiene la clase Main la cual es la raíz y es la que corre el código.

El proyecto se diseñó y testeó con gradle-7.5.1 integrando en el archivo build.gradle.

Para la edición, se utilizó el IDE Intellij IDEA 2022.3 (JET BRAINS, s.f.)

En cuanto a código, el Menu posee el método run() que va mostrando el MenuSystem myMenu, en cada estado y en base a la respuesta entregada por consola va moviéndose y ejecutando distintas acciones. Cada UserSystem se instancia en el Menu y se le añade un dueño de tipo AdminUser que es el único que puede interactuar o modificarlo. Los CommonUser sólo pueden interactuar con el DemoSystem, no pueden crear sus sistemas ni acceder a los creados por administradores.

Instrucciones de uso:

En la carpeta CodigoLab3_21130814_MonjeRojas, que contiene el archivo gradlew.bat se debe abrir en Windows (no se probó en Linux), el powershell y se debe utilizar el comando ".\gradlew build" para compilar el proyecto, luego se puede ingresar ".\gradlew run" para correrlo, sin embargo, se recomienda fuertemente el utilizar ".\gradlew run –console plain" para que sea visualmente más estético. En

cmd también es posible, la única diferencia en los comandos es que no es necesario añadir ".\" al principio. (Ver Anexo 8)

En la interacción por consola, a menos que se indique lo contrario es posible cometer errores de ingreso, como el Menu utiliza un System por detrás para preguntar, se aceptan como respuesta palabras claves o números, sin embargo, como se indica en el programa, no se aceptan errores de ingreso al crear Systems, Chatbots, Flows o Options, tampoco es aceptable un error al escoger un System o Chatbot para modificar, si hay un error se cae el programa. Todo está bien indicado por consola.

Resultados Obtenidos:

Las funcionalidades fueron implementadas en su totalidad, funcionando el 100% de las veces.

A diferencia de proyectos anteriores, se implementó la capacidad de reiniciar un System a su estado original al terminar de hablar con él, para que así sea reutilizable.

No se añadieron funciones para eliminar o modificar estructuras ya creadas, sólo se pueden agregar elementos.

El ingreso funciona correctamente permitiendo registrarse y verificando que no exista de antes el usuario ingresado (Anexo 3). Luego el Menu redirige automáticamente a un CommonUser a un menú de interacción (Anexo 6) mientras que a un AdminUser le pregunta qué tipo de menú desea (Anexo 4), de interacción o de modificación y además le pregunta sobre que System desea trabajar, en el de modificación le ofrece la opción de crear uno nuevo (Anexo 5).

Conclusiones:

Los diagramas de análisis y diseño se encuentran en el Anexo 9 y Anexo 10, respectivamente.

El paradigma orientado a objetos provee una manera cómoda de trabajar, separando los TDAs en Clases y controlando el proceso a través de métodos, además se sintió similar al paradigma lógico en que, al encadenar métodos, se pueden obtener elementos muy específicos en una sola línea de código. También, Java es un lenguaje muy atractivo para trabajar puesto que es fácil de ordenar y además se siente como un lenguaje rápido.

Tomando todo en consideración, el trabajo realizado logró implementar lo solicitado, pudiendo profundizar en el entendimiento del paradigma orientado a objetos.

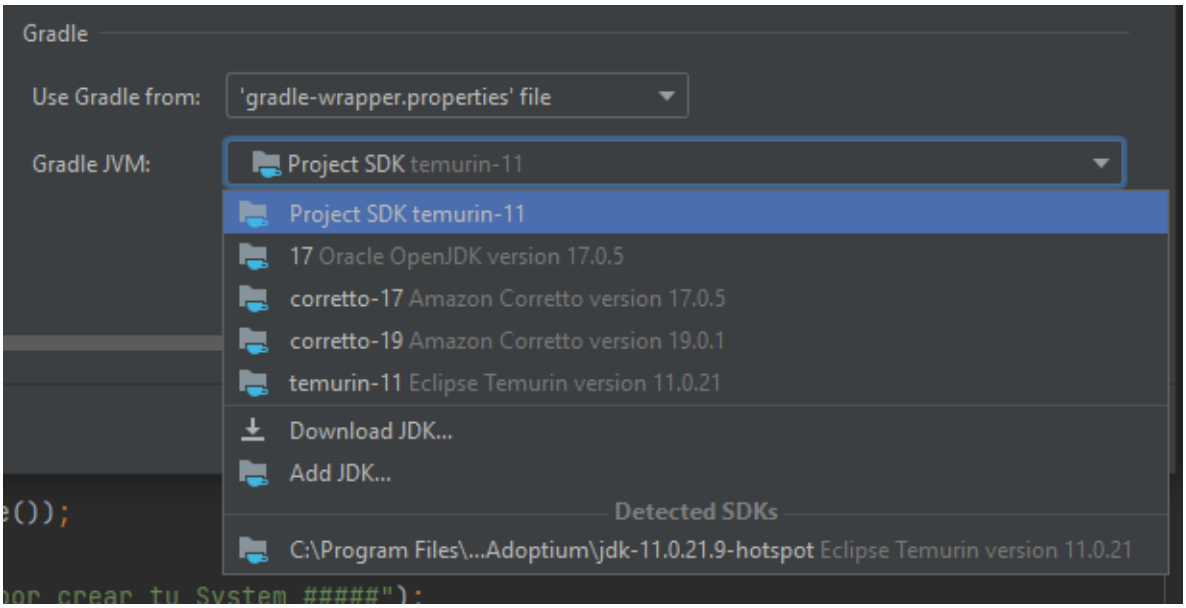
En una siguiente instancia, un objetivo será agregar más funcionalidades al menú y en lo posible, implementar una interfaz gráfica utilizando Swing.

Anexo:

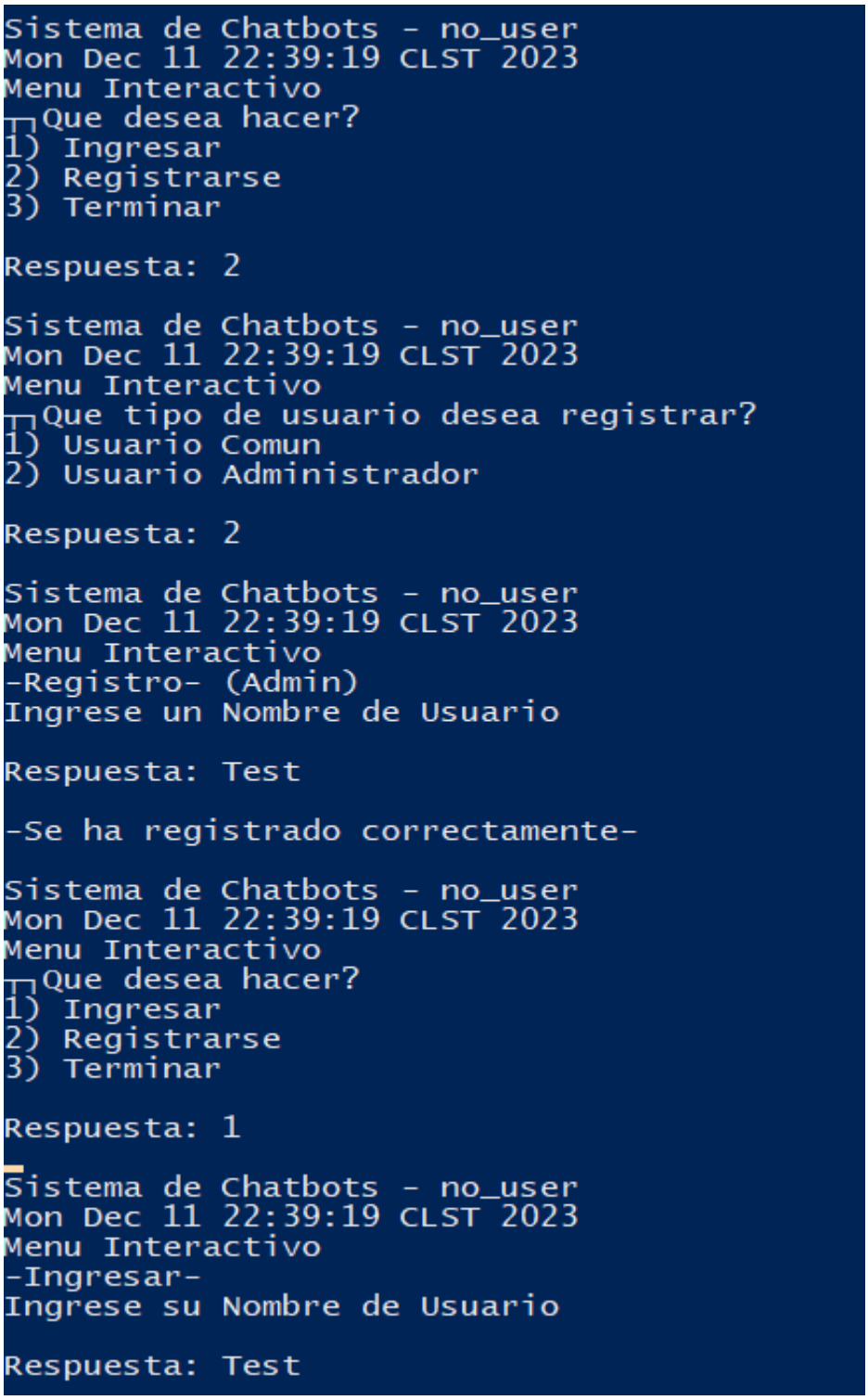
Anexo 1: JRE utilizado

```
C:\Users\amaru>java --version
openjdk 11.0.21 2023-10-17
OpenJDK Runtime Environment Temurin-11.0.21+9 (build 11.0.21+9)
OpenJDK 64-Bit Server VM Temurin-11.0.21+9 (build 11.0.21+9, mixed mode)
```

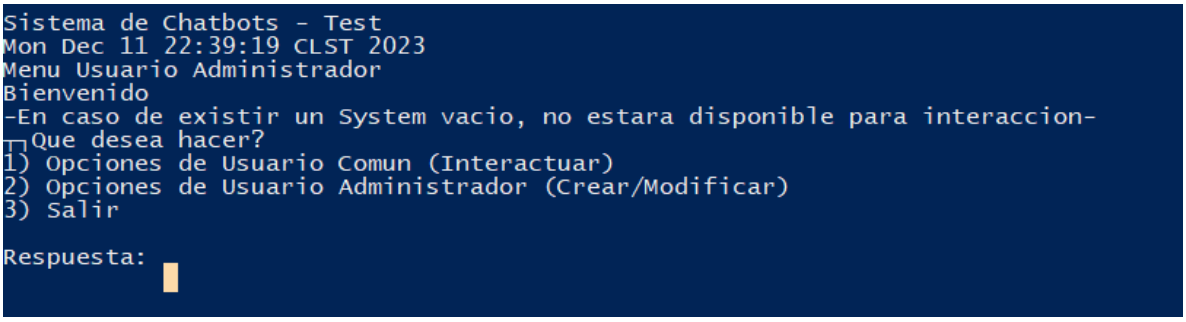
Anexo 2: JDK utilizado



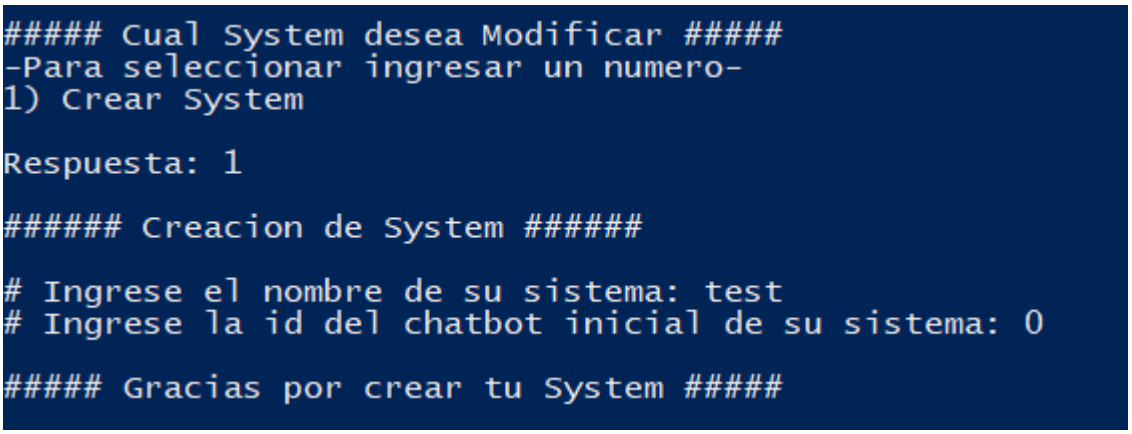
Anexo 3: Registro e Ingreso



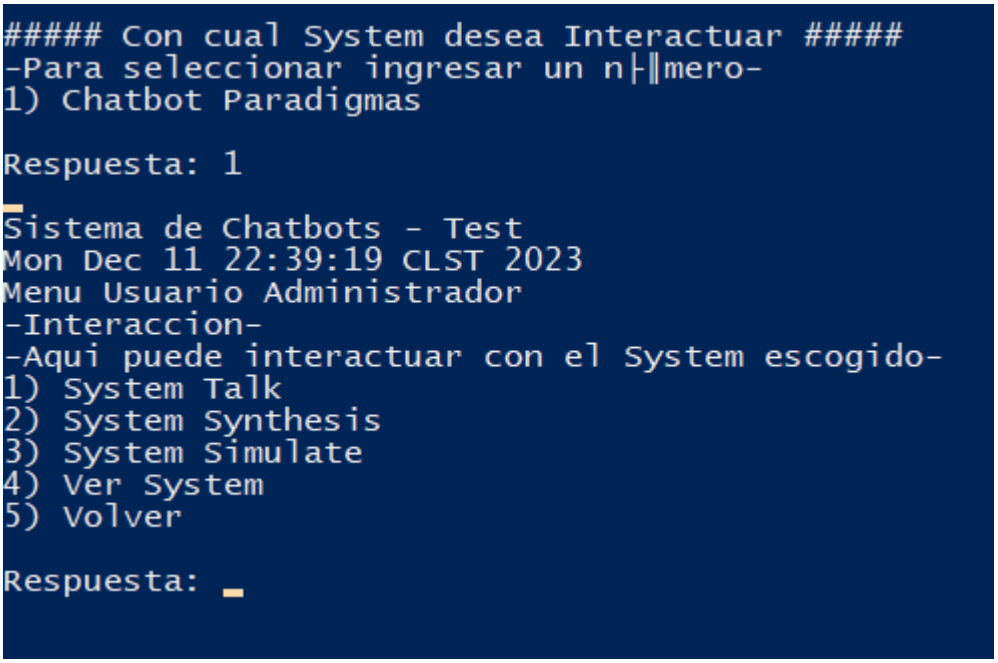
Anexo 4: Opciones AdminUser



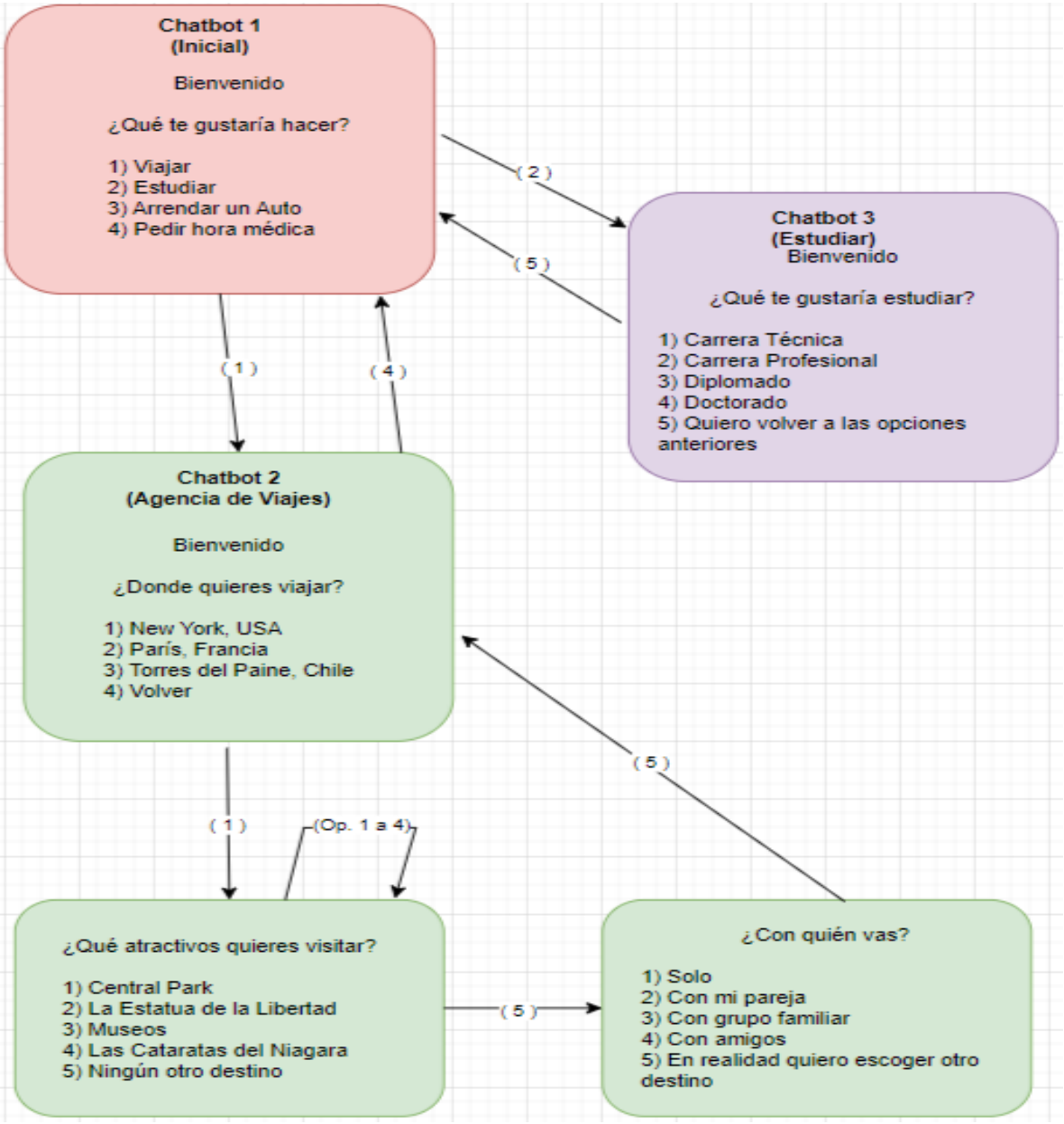
Anexo 5: Crear System



Anexo 6: Opciones Común AdminUser igual para CommonUser



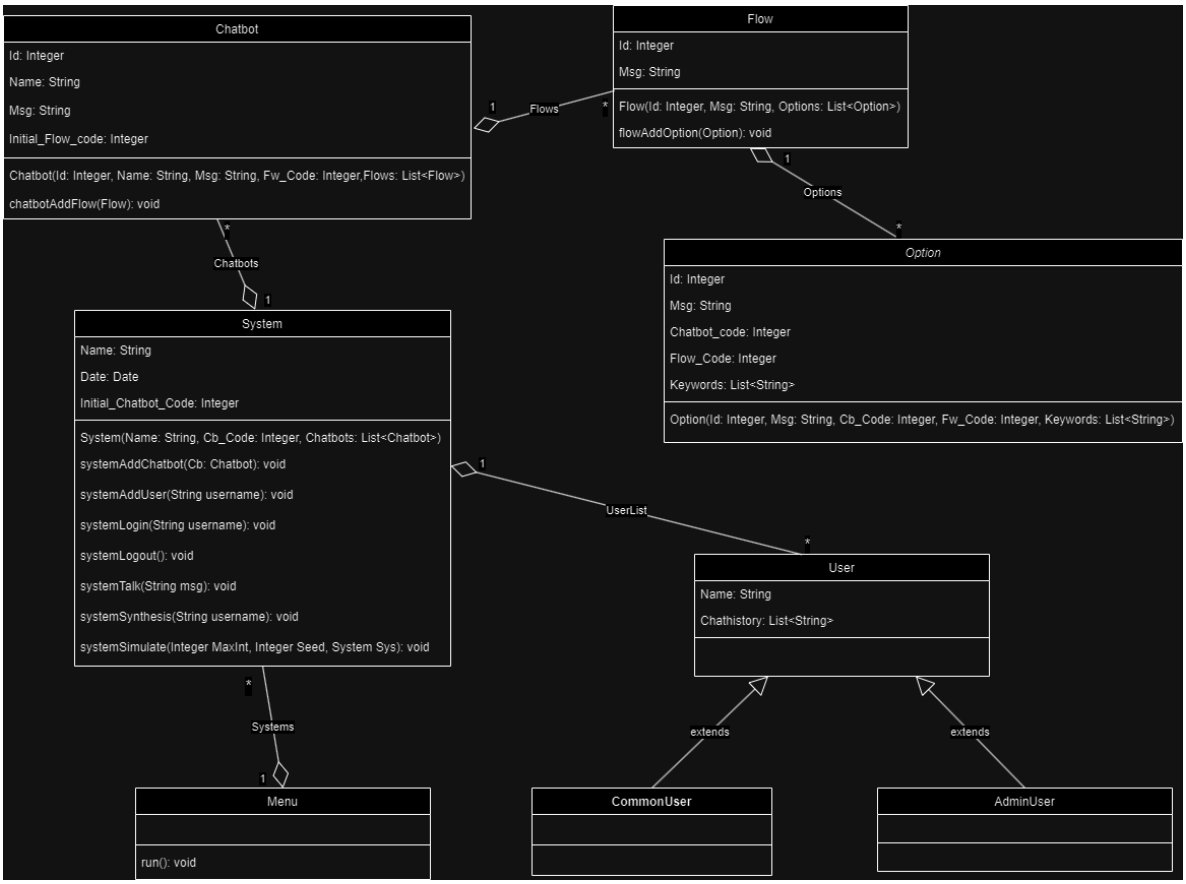
Anexo 7: Esquema en el que se basó el DemoSystem.



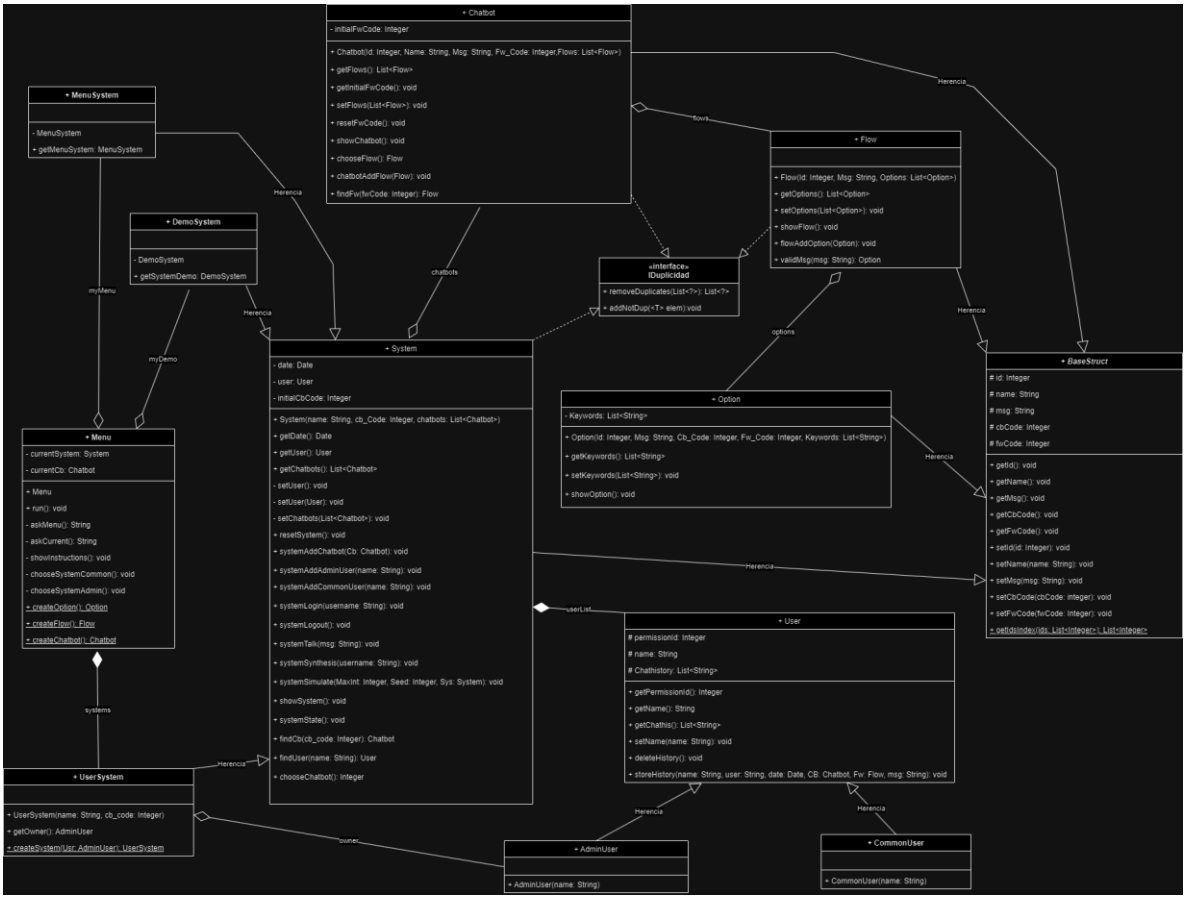
Anexo 8: Compilación y Ejecución con Gradle

```
PS C:\Users\amaru\Desktop\Proyectos Paradigmas\lab3_21130814_MonjeRojas\CodigoLab3_21130814_MonjeRojas> .\gradlew build

Deprecated Gradle features were used in this build, making it incompatible with Gradle 8.0.

You can use '--warning-mode all' to show the individual deprecation warnings and determine if they come from your own scripts or plugins.

See https://docs.gradle.org/7.5.1/userguide/command_line_interface.html#sec:command_line_warnings

BUILD SUCCESSFUL in 2s
5 actionable tasks: 5 up-to-date
PS C:\Users\amaru\Desktop\Proyectos Paradigmas\lab3_21130814_MonjeRojas\CodigoLab3_21130814_MonjeRojas> .\gradlew run --console plain
> Task :compileJava UP-TO-DATE
> Task :processResources NO-SOURCE
> Task :classes UP-TO-DATE

> Task :run

Sistema de Chatbots - no_user
Mon Dec 11 23:13:19 CLST 2023
Menu Interactivo
¬¬Que desea hacer?
1) Ingresar
2) Registrarse
3) Terminar

Respuesta:
```

Anexo 9: Diagrama de análisis



Anexo 10: Diagrama de Diseño



Referencias

JET BRAINS. (s.f.). *Intellij IDEA*. Obtenido de https://www.jetbrains.com/idea/

Martínez, M. (02 de 11 de 2020). *profile*. Obtenido de https://profile.es/blog/que-es-la-programacion-orientada-a-objetos/