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

Informe de laboratorio 3 - Aplicación de la programación orientada a objetos en el desarrollo de un sistema de chatbots

Alumno: Mohamed Al-Marzuk
Curso: Paradigmas de programación
Sección A-1
Profesor: Roberto González

Contenido

Introducción

El propósito de este informe es presentar el diseño de un simulador de un sistema de chatbots, mostrando la interacción entre el usuario y las distintas opciones que contiene cada flujo de un chatbot, a través de un menú interactivo por consola. Para esto, se utilizarán los conceptos del paradigma de la programación orientada a objetos, y se implementará la solución en el lenguaje de programación Java, utilizando el IDE ”Intellij”, y OpenJDK en su versión 11. A lo largo de este documento se analizarán los antecedentes del proyecto, al igual que las motivaciones que llevan a realizarlo, y se expondrán los detalles del proceso de solución del problema, utilizando la programación orientada a objetos. Se procederá a realizar primero una descripción del problema a abordar, y del paradigma en sí. tras esto, se verán aspectos importantes, como el diseño de la solución y las instrucciones de uso. Finalmente, se realizará un análisis de los resultados, junto a una autoevaluación acerca del trabajo realizado.

Descripción del problema

Como se mencionó antes, el propósito de esta entrega es crear un sistema de ITR (Respuesta de Interacción a Texto) con varios chatbots que, en base a la palabra clave, o del número que se le entregue a alguno de ellos, se derive la atención del usuario al chatbot que se relacione con lo que este busca. Estas palabras se encuentran almacenadas con anterioridad en la estructura del chatbot, por lo que son lo único que este puede entender. Lo que se busca es crear un sistema que contenga a varios chatbots, cada uno de ellos está asociado a una actividad en concreto. En el caso de esta entrega, la interacción con el chatbot se realiza a través de un menú.

¿Qué es la programación orientada a objetos?

La programación orientada a objetos pertenece al grupo de los paradigmas imperativos. Este paradigma se basa en la definición de “objetos”, para modelar las cosas de la vida real, cada uno de estos objetos tiene características y comportamientos propios. Para entender este paradigma, hay que tener en mente ciertos conceptos relevantes: Primero se va a hablar de lo que es una ***clase***, esta es la forma en la que se organizan los programas orientados a objetos, siendo estas un molde para la creación de diversos objetos. Mientras que el ***objeto*** en sí es una instancia de una clase, es decir, tiene los ***atributos***, es decir, sus características propias, y ***métodos***, que hacen referencia a los comportamientos de la clase a la que pertenece. Cada clase tiene un ***constructor***, que es un método especial que sirve para instanciar un objeto, e inicializar cada uno de sus atributos. Los objetos se comunican entre sí a través de ***mensajes***, los cuales hacen referencia a la invocación de alguno de los métodos del objeto en cuestión. Una clase se puede crear a partir de otra ya existente, a este fenómeno se le conoce como ***herencia***, la subclase, es decir, la clase hija, posee los métodos y atributos de la superclase, la clase padre, y también puede poseer métodos y atributos propios. Por último, una ***interface*** sirve para declarar los métodos que pueden ser implementados por una clase.

Desarrollo

Análisis del problema

Antes que nada, para poder realizar un correcto análisis del problema es necesario identificar correctamente las clases e interfaces fundamentales para poder realizar este simulador de un sistema de chatbots. Estos son: el sistema que contiene a los chatbots, el chatbot en sí, los flujos de un chatbot y las opciones de un flujo. Además de estos cuatro, es necesario tener una buena representación de un usuario, además de contar con una buena construcción del chat-history y de los mensajes que lo componen. A continuación, se brindará más detalle sobre estas clases.

- **Usuario**: Pueden ser comunes o administradores. Un usuario común puede interactuar con un chatbot y ver su historial propio, y un administrador, además de esto, puede manipular los chatbots del sistema.
- **Mensaje**: Un mensaje corresponde a la representación de una interacción entre un usuario con un chatbot y viceversa. Tiene como atributos la fecha de envío, el emisor del mensaje, sea este un usuario o un chatbot, y el mensaje en sí.
- **Chat-History:** Cada usuario tiene un historial en el sistema. Un historial tiene como atributos un usuario y una lista de mensajes. El historial se crea dentro de un sistema.
- **Opción**: La clase opción tiene como atributos el ID de la opción, su nombre, el ID del chatbot al que se asocia, el ID del flujo al que se asocia y una lista con una o más palabras claves asociadas. La clase opción implementa los métodos de la interface opción.
- **Flujo**: La clase flujo contiene su ID, su nombre, y una lista que contiene a varias opciones. Algunos comportamientos relevantes de esta clase son: encontrar una opción por ID, y la de agregar una opción al flujo. La clase flujo implementa los métodos de la interface flujo.
- **Chatbot:** La clase flujo contiene el ID, el nombre y el mensaje de bienvenida, el ID del flujo actual, y una lista con varios flujos. La clase chatbot implementa los métodos de la interface chatbot.
- **Sistema:** La clase sistema contiene diversos atributos, entre los cuales se encuentra el nombre del sistema, el ID del chatbot actual, una lista de formateo, que se encarga de devolver el sistema a su chatbot y flujos iniciales, que se asume que son el chatbot 0, y el flujo 1. Luego hay un atributo público de tipo booleano para saber si el sistema está inicializado. Luego, está la lista de usuarios registrados, la lista del usuario loggeado y la lista del historial, estas 3 listas se inicializan como vacías en el constructor. Y por último, la lista de los chatbots del sistema.

Teniendo esto en cuenta, queda hablar de cómo se relacionan estos elementos, con respecto a la interacción, esta se basa en uno o más flujos estructurados dentro de un chatbot, que contienen opciones. Las palabras claves o los IDs de las opciones son las que facilitan esta interacción. El sistema puede gestionar uno o más chatbots, y también guarda la información de uno o de varios usuarios. El chat-history permite representar de manera clara la conversación de un usuario. Ahora, para poder hacer una buena

implementación de estas clases, se deben definir correctamente los métodos necesarios y suficientes para asegurar una correcta relación entre cada uno de los objetos creados. La **figura 1** muestra el análisis inicial del problema.

Diseño de la solución

El diseño de la solución, utilizando el paradigma de la programación orientada a objetos, consiste en la creación de distintos objetos para cumplir con lo requerido por el problema. Primero, se tiene que hacer la implementación de las clases mencionadas en el análisis del problema con todos los atributos mencionados, junto a los métodos esenciales para que estos funcionen correctamente. Podemos ver que la clase principal es el sistema, pues de alguna forma, esta clase es la que contiene a todas las demás. El sistema incluye diversa información útil para el correcto funcionamiento del programa, tales como la lista de usuarios registrados, el usuario loggeado y el chat-history de los usuarios, además de los datos para formatear el sistema en caso de que el usuario actual decida cerrar su sesión. Al momento de registrarse, los usuarios por defecto se construyen como usuarios comunes. Sin embargo, a través del menú, se utiliza el método setAdmin() para asignarle admin al usuario correspondiente. Cada usuario tiene un historial asociado, y este historial se guarda dentro del sistema. En el caso de que no haya interactuado nunca con un chatbot, su historial será vacío. Solamente los usuarios loggeados pueden interactuar con un chatbot. Un usuario administrador puede ver el historial de cualquier otro usuario, mientras que un usuario común solo puede ver el propio. Dentro del sistema, también está la lista de todos los chatbots. El sistema tiene un id de chatbot inicial, el cual será el primero en aparecer frente al usuario, cuando el sistema todavía no esté inicializado. Ese chatbot inicial mostrará las opciones que se encuentran contenidas en su flujo inicial. Cuando el usuario hace la interacción con el método systemTalk, se accede al flujo actual del chatbot actual y se busca la keyword o la opción dentro de ese flujo, cuando la encuentre, se recogen las IDs del chatbot y flujos asociados a la opción, se actualiza el ID del flujo dentro del chatbot, y el ID del chatbot dentro del sistema. De esta forma, se puede asegurar una correcta interacción entre el usuario y el sistema. Una vez que un usuario termine de interactuar con el sistema y se cierre la sesión, el sistema vuelve a estar en su modo no inicializado, y los chatbot y flujos iniciales vuelven a sus variables originales. La **figura 2** muestra el diagrama UML de la solución.

Aspectos de implementación

Para la implementación de este problema, se utilizó el lenguaje Java, utilizando JDK 11 en su versión estándar, es decir, no se importó ninguna librería externa al lenguaje, y se utilizaron únicamente las clases estándar de Java, además de las clases propias que fueron implementadas para el desarrollo del programa.

Instrucciones de uso

Para poder ejecutar el programa, se debe crear una carpeta en donde se encuentren los archivos de cada uno de las clases e interfaces implementadas para solucionar el problema planteado. El código fuente tiene 12 archivos. Cada uno de estos archivos contiene clases e interfaces, con sus métodos y atributos correspondientes a cada uno de los TDA, con sus constructores, modificadores y selectores, mientras que el archivo main contiene el menú,

junto a la construcción del sistema que se va a utilizar en dicho menú. Teniendo esto, se debe abrir Intellij, y compilar el programa utilizando Gradle. Inmediatamente, se mostrará un menú por pantalla, en donde el usuario se tendrá que registrar en el sistema, ya sea como administrador o como usuario común, para posteriormente iniciar sesión e interactuar con el sistema, en la **figura 3** se ve este menú.

- **Ejemplos de uso**: En el menú se puede interactuar con el sistema de varias formas. Al iniciar sesión con un usuario común, se mostrará un menú básico, en donde se da la opción de hablar con un chatbot o de mostrar el historial propio del usuario, esto se ve en la **figura 4**. En el caso de seleccionar hablar con un chatbot, se tiene que escribir el mensaje, ya sea este el número de la opción en cuestión o alguna palabra clave. Los flujos del chatbot se van mostrando continuamente por pantalla, tal y como se ve en la **figura 5**. En el caso de registrar a un usuario administrador, se tendrá un menú distinto, el cual se ve en la **figura 6**, en donde además de poder hablar con un chatbot y ver el historial, se podrá ver todos los chatbots, flujos y opciones creadas, y manipular los chatbots del sistema, ya sea eliminándolos o modificándolos. En la **figura 7** se ve cómo se elimina del sistema el chatbot de ID 1.
- **Resultados esperados**: Se espera que el sistema de chatbots esté implementado correctamente en un 95%, pues todos los RF cumplen su cometido de manera precisa sin mostrar errores. El menú tampoco tiene fallos notables, y funciona de manera correcta. Lo único que faltó por hacer es la simulación de una conversación con un chatbot.
- **Posibles errores**: Se esperan errores en la opción de crear chatbots, flujos u opciones, pues los strings ingresados por el menú no pueden tener espacios ni saltos de línea, es así que cualquier string ingresado por el menú debe ser una palabra ininterrumpida. Otros posibles errores vienen dados si se entrega por consola un tipo de dato incorrecto. En la **figura 8** se puede ver este error, pues se le entrega un string como ID de chatbot, debiendo ser este un entero. El programa entregará el mensaje de error que se ve en la **figura 9**.

Resultados

Todos los requerimientos funcionales, desde el RF1 hasta el RF13, fueron implementados correctamente. El menú también fue implementado a la perfección, mostrando opciones distintas dependiendo de si un usuario es administrador o si es común. Las clases se relacionan perfectamente entre sí, y el programa funciona de forma correcta. El RF14 no fue implementado.

Autoevaluación

Los RF funcionan correctamente. No hubo ningún problema en su implementación. El menú también funciona de forma correcta y sin mostrar errores, por lo que se podría decir que todos estos requisitos están al 100%. El RF14, al no ser implementado, está al 0%. En el archivo de autoevaluación hay más detalles al respecto.

Conclusión

Tras haber implementado la mayoría de los requerimientos funcionales del laboratorio, se puede concluir que se aplicaron correctamente los principios de la programación orientada a objetos para el desarrollo de un simulador de un sistema de chatbots. La principal complicación que se tuvo para desarrollar esta entrega viene ligada al menú de interacción, pues realmente, la implementación de este requirió muchísimo tiempo de desarrollo, ya que había que pensar en demasiados casos distintos y programar cada uno de ellos, para que finalmente funcione a la perfección. Además, al ser la primera vez que se pide implementar un menú, no tenía una idea muy clara de como es que debía hacerlo. Sin embargo, a pesar de las dificultades, el resultado obtenido fue realmente satisfactorio. Con respecto a las dificultades de la programación orientada a objetos, se puede destacar la confusión que da a veces saber con qué objeto se está trabajando, pues realmente, de no ser por la ayuda que entrega Intellij, habría cometido muchos más errores. Aún así, si se compara la programación orientada a objetos con los paradigmas anteriores, se puede afirmar que el paradigma de esta ocasión fue el más fácil de entender, pues pertenece al grupo de paradigmas imperativos que ya conocía con anterioridad. Me gustó recuperar por fin algunas sentencias, tales como los ciclos for, while, o la declaración de variable, y quitando lo nuevo de esta entrega respecto a las anteriores, que sería el menú, el resto de funcionalidades se realizaron de forma mucho más rápida que en los otros paradigmas. En resumen, fue menos complicado aprender a utilizar la programación orientada a objetos en Java que la programación lógica en Prolog, y que la programación funcional en Racket.

En conclusión, se lograron adquirir conceptos importantes sobre la programación orientada a objetos y sobre el lenguaje de programación Java, y lo más importante, se pudo ampliar la mente hacia nuevos conocimientos al estar sujetos a programar bajo estas “nuevas reglas”. No cabe duda de que los conocimientos adquiridos serán de suma utilidad tanto para la vida laboral como para el resto de la vida universitaria.

Bibliografía

Gonzalez, R. (2023, septiembre 30). *Proyecto semestral paradigmas*. https://docs.google.com/document/d/1L-B2b3J71Baqa_IuZt6EmRwDlxC0qzWn9YJAm6FIiJk/edit

González, R. (2023, 19 noviembre). *Proyecto semestral de laboratorio - Paradigma orientado a objetos*. https://docs.google.com. Recuperado 11 de diciembre de 2023, de https://docs.google.com/document/d/1Psn6YqXfWA99n3AA-rLsvAJsc2-gqj6ot7j90ucLcog/edit

Qué es la programacion orientada a objetos. (s. f.). DesarrolloWeb.com. https://desarrolloweb.com/articulos/499.php

Anexos

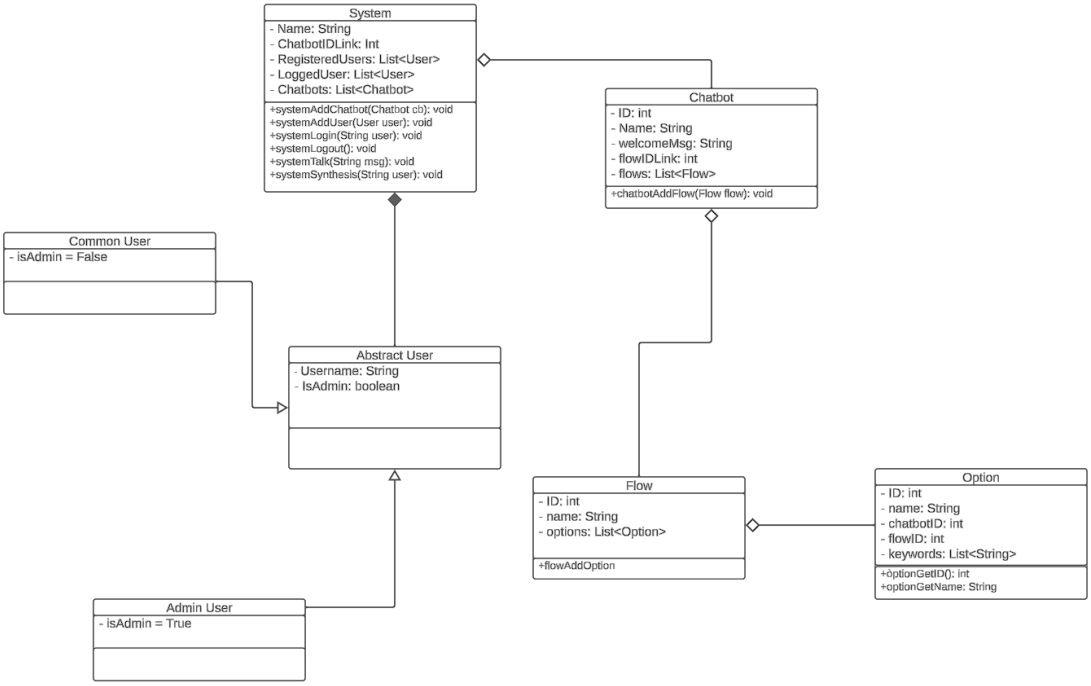

Figura 1 – Diagrama de análisis

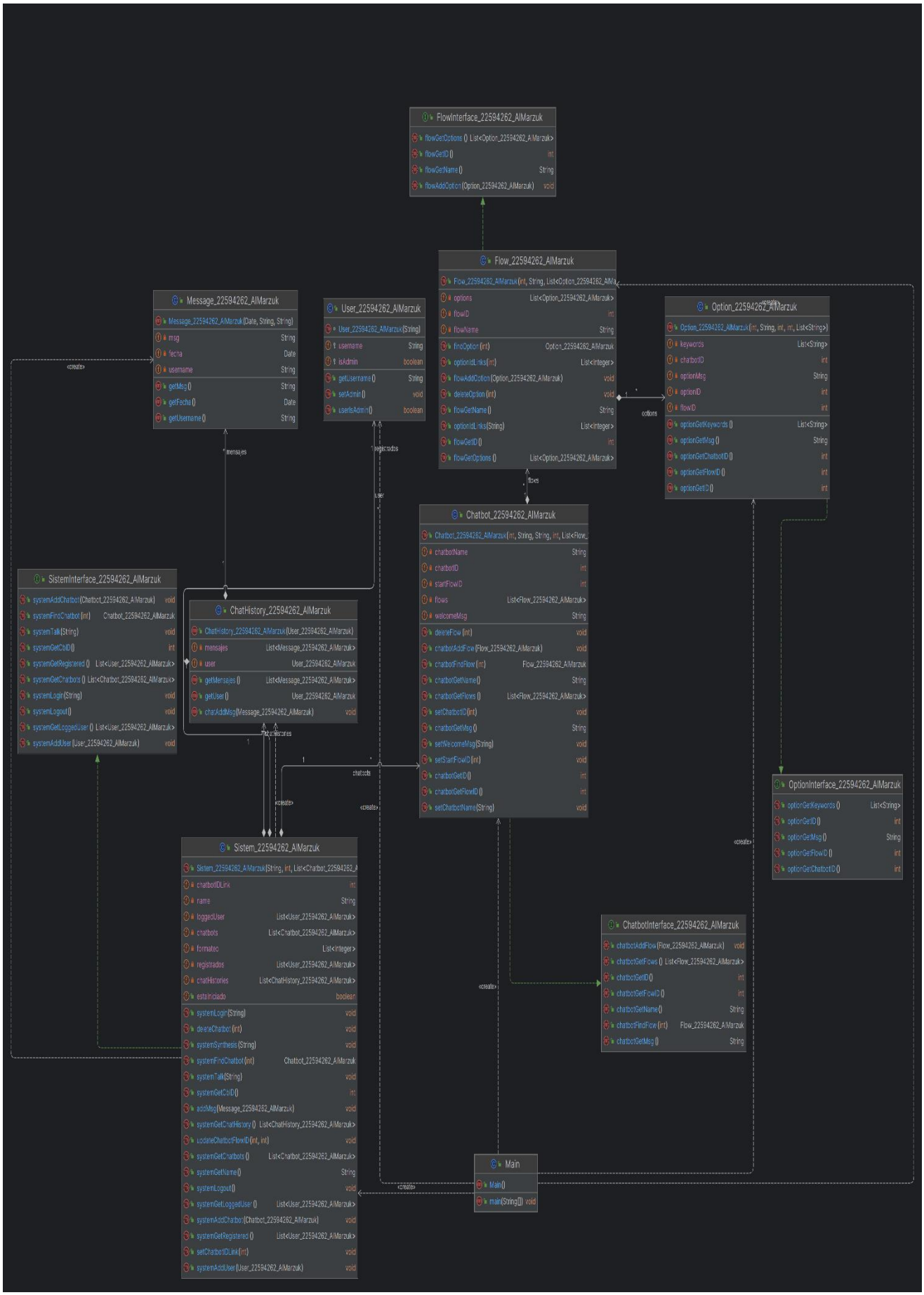

Figura 2 – Diagrama UML del problema

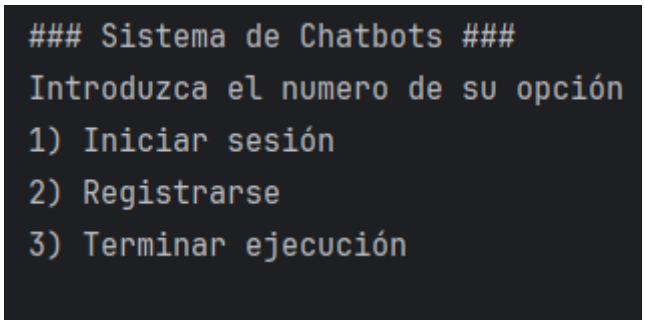

Figura 3 – Menú principal

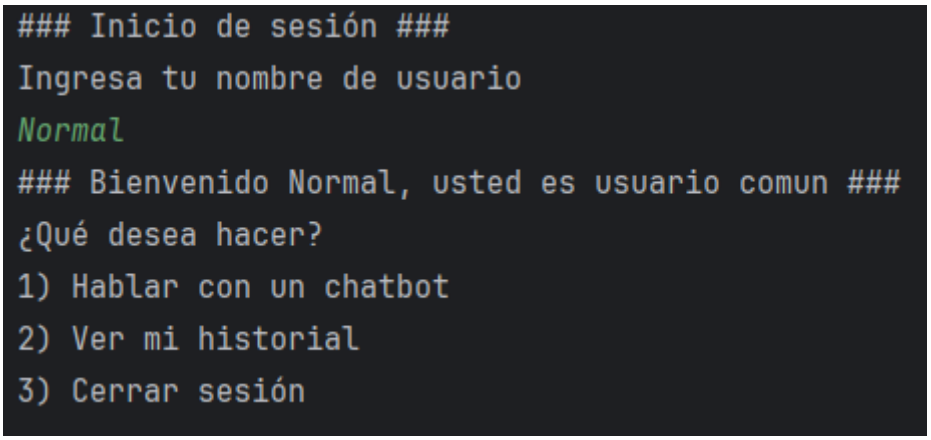

Figura 4 – Menú del usuario común

```
hola
Bienvenido
¿Qué planeas hacer?
1) Jugar
2) Trabajar

### Bienvenido Normal, usted es usuario comun ###
¿Qué desea hacer?
1) Hablar con un chatbot
2) Ver mi historial
3) Cerrar sesión
1
Escribe aquí tu mensaje
jugar

Flujo1 Chatbot1
¿Qué Quieres Jugar?
1) Minecraft
2) Fortnite
3) Stardew Valley
4) No quiero jugar

### Bienvenido Normal, usted es usuario comun ###
¿Qué desea hacer?
1) Hablar con un chatbot
2) Ver mi historial
3) Cerrar sesión
```

Figura 5 – Usuario hablando con un chatbot

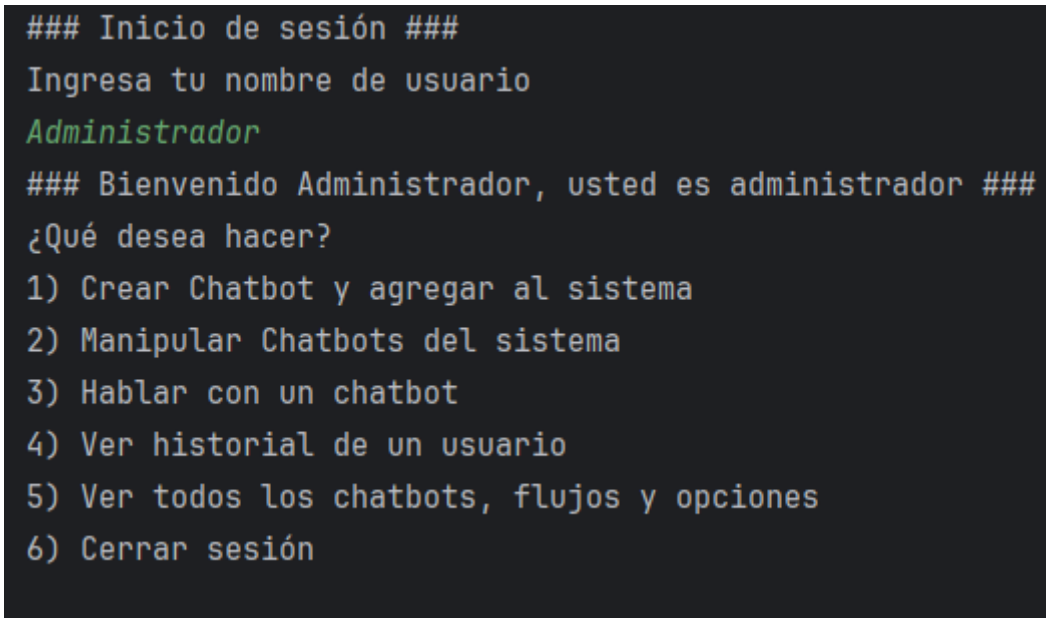

Figura 6 – Menú usuario administrador

```
2
El sistema tiene los siguientes ID de chatbot: [0, 1, 2]
### Modificar chatbots del sistema ###
¿Qué desea hacer?
1) Eliminar chatbot
2) Modificar un chatbot
3) Volver
1
Ingresa el id del chatbot a eliminar
1
### Bienvenido Administrador, usted es administrador ###
¿Qué desea hacer?
1) Crear Chatbot y agregar al sistema
2) Manipular Chatbots del sistema
3) Hablar con un chatbot
4) Ver historial de un usuario
5) Ver todos los chatbots, flujos y opciones
6) Cerrar sesión
2
El sistema tiene los siguientes ID de chatbot: [0, 2]
### Modificar chatbots del sistema ###
¿Qué desea hacer?
1) Eliminar chatbot
2) Modificar un chatbot
3) Volver
```

Figura 7 – Eliminación del chatbot 1 del sistema

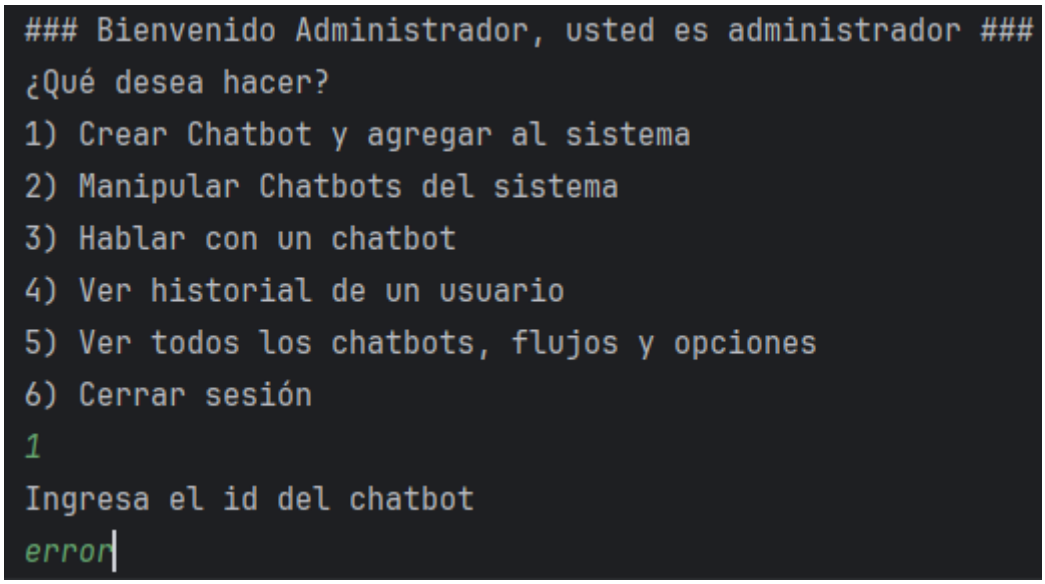

Figura 8 – Mal ingreso de los datos

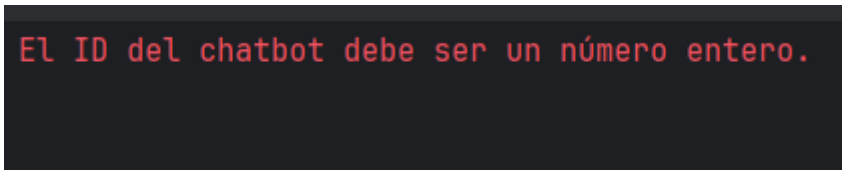

Figura 9 – Mensaje de error