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

INFORME DE LABORATORIO 3: SIMULACIÓN DE UN SISTEMA E INTERACCIÓN DE CHATBOTS EN JAVA

Nombre: Reinaldo Pacheco Parra
Profesor: Gonzalo Martínez Ramírez
Asignatura: Paradigmas de programación (2/2023)

Índice

Introducción

En el siguiente informe, se presenta el desarrollo de un sistema de interacción de chatbots utilizando el Paradigma Orientado a Objetos en el lenguaje Java a través del IDE IntelliJ. Este informe se estructura en varias secciones clave, que abarcan la descripción del problema y los objetivos, la explicación del paradigma orientado a objetos, aspectos de análisis, diseño e implementación de la solución.
Se discute la razón por la cual se optó por una implementación en particular en lugar de otras alternativas y se destacan las ventajas de esta elección. A continuación, se exploran algunos de los métodos específicos que se han implementado y los enfoques que se tuvieron que considerar durante el proceso de desarrollo. Posteriormente, se proporcionan las instrucciones necesarias para verificar el correcto funcionamiento del programa. Para concluir, se realiza un análisis de los resultados obtenidos a partir de la implementación y se lleva a cabo una autoevaluación detallada de cada uno de los métodos implementados en el sistema de interacción de chatbots.

Descripción breve del problema:

Se busca realizar una simulación de un sistema de chatbots de ITR (interacción respuesta a texto) los cuales responden a preguntas en base a opciones especificas ya definidas por algún número o palabra clave que permite diferenciarlos entre sí. Este simulador de chatbots debe permitir realizar diferentes operaciones convencionales tales como: crear un flujo de opciones, crear un chatbot, agregar un flujo dentro de un chatbot, crear un sistema de chatbots, registrar usuarios para poder iniciar y cerrar sesión en el sistema, simular una conversación con un chatbot, entre otros.

Descripción breve del paradigma:

El paradigma orientado a objetos (POO) se basa en la definición de objetos, los cuales se modelan según características o comportamientos. Algunos de los conceptos clave del paradigma son los siguientes:

- **Clase:** Es una plantilla que define las características o comportamientos que puede tener un conjunto de objetos. En la clase, se especifican los atributos y los métodos que los objetos de esta clase pueden tener.
- **Objeto:** Es una instancia de una clase y representa una entidad la cual posee datos y acciones llamadas atributos y métodos respectivamente.
- **Atributo:** Es un dato o datos que definen características específicas de un objeto.
- **Método:** Es un comportamiento que puede realizar un objeto. Están definidas dentro de una clase y especifican las acciones que se pueden realizar. Permiten modificar el estado de un objeto o interactuar con otros objetos.
- **Herencia:** Es la forma de crear nuevas clases en base a la información que ya posee una clase ya existente, esto permite reutilizar código que ya fue creado anteriormente.
- **Interfaz:** Es la forma de definir un contrato, en una interfaz se especifica que métodos debe cumplir un tipo de objeto, pero no se define como deben ser implementados.

Desarrollo

Análisis del problema

Para desarrollar un programa de simulación de chatbots, se han identificado cinco objetos esenciales
Sistema: posee un name, un initialChatbotCodeLink y una lista de chatbots.
Chatbot: posee un id, un name, un mensaje de bienvenida, un startFlowId y una lista de flujos.
Opción: posee un id, un name, un mensaje, un chatbotCodeLink y una lista de palabras clave.
Flujo: posee un id, un mensaje y una lista de opciones.
User: posee un nombre de usuario y un verificador que indica si es administrador o no.
Menú: se encarga de manejar la lógica e interacciones de los métodos con el usuario.
Main: se encarga de ejecutar el programa.

Además, se deben realizar las siguientes operaciones entre los objetos:
option: crea una opción
flow: crea un flujo (conjunto de opciones)
flowAddOption: permite agregar una opción dentro de un flujo
chatbot: crea un chatbot.
chatbotAddFlow: permite agregar un flujo dentro de un chatbot
system: crear un sistema
systemAddChatbot: permite agregar un chatbot al sistema
systemAddUser: permite agregar un usuario al sistema
systemLogin: inicia una sesión de un usuario en el sistema (el usuario debe estar registrado para poder iniciar sesión)
systemLogout: cierra una sesión de un usuario en el sistema.
systemTalk: permite interactuar con un chatbot.
systemSynthesis: ofrece una síntesis del chatbot a partir del chat-history.
systemSymulate: simula un diálogo entre dos chatbots en el sistema.

Estas operaciones se deben presentar al usuario haciendo uso de un menú interactivo que permita seleccionar las opciones y mostrar los cambios realizados en el sistema.

Diseño de la solución

El diseño de la solución implementado, utilizando el Paradigma Orientado a Objetos (POO), consiste en representar los elementos del sistema de chatbots en distintas clases. Esto se hace dado que esta estructura es propia del paradigma lo cual facilita su uso. Para ello, se tiene una clase principal llamada Sistema, la cual alberga una lista de chatbots, de la misma forma la clase Chatbot almacenará una lista de flujos, la clase Flujo almacenará una lista de opciones y la clase Opción se encargará de guardar una lista de palabras clave.
Cabe destacar que también se implementa la clase Main la cual permite ejecutar el inicio y termino del programa y la clase Menú que se encarga de administrar la correcta llamada de los métodos e interacciones con el usuario.
Además, uno de los enfoques principales de la problemática es que algunos de los elementos del sistema de chatbots se representan mediante un id único, el cual puede ser un entero o un string, este enfoque evita tener elementos duplicados que puedan afectar en la lógica del sistema.

Algunos de los métodos esenciales utilizados durante la solución son los siguientes:

Los métodos esenciales para la construcción de la solución se muestran a través de menús y submenús que se desplazan y ofrecen las opciones disponibles al usuario. Para ello, cada uno de los menús creados utilizan switch-case en donde, según la opción escogida por el usuario, se deriva a otro menú u otro método según corresponda. Uno de los ejemplos es el método **"procesoRegistro"** (ver Figura 1 en Anexos) en el cual se tiene la opción de registrar un usuario administrador o un usuario normal.
Para cada uno de los menús en donde se tienen casos u opciones diferentes para seleccionar se utiliza switch-case, (ver Figura 2 en Anexos) en donde se muestra el menú con mayor cantidad de opciones, el cual es el método menuAdministrador.

Otro de los métodos importantes es **"visualizarChatbotsConFlujos"** (ver Figura 3 en Anexos) el cual corresponde a una de las opciones disponibles para el usuario normal o usuario administrador y permite visualizar el sistema completo.

Por último, uno de los métodos clave para la construcción del requerimiento funcional "systemTalk" fue la elaboración el método **"findOptionCode"** (ver Figura 4 en Anexos) el cual permite buscar un código específico de una opción, utilizando la misma lógica, se construyeron los métodos **"findFlowId"** y **"findChatbotId"** para buscar el id de un flujo o chatbot correspondiente.

Aspectos de implementación

Para la resolución del problema, se utiliza el Paradigma Orientado a Objetos mediante el lenguaje de programación Java en el IDE-IntelliJ de versión IntelliJ IDEA 2023.2.5 y JDK 11.0.21. Por último, para la ejecución del programa se utiliza Gradle 8.2.

Las clases utilizadas en la implementación son:

- **Clase Menu**
- **Clase Chatbot**
- **Clase Flow**
- **Clase Option**
- **Clase User**
- **Clase Sistema**
- **Clase CargaDatos**
- **Interface Chatbot**
- **Interface Flow**
- **Interface Sistema**

(ver Figura 5 en Anexo donde se muestra el diagrama final implementado)

Instrucciones de uso

Ejemplos de uso

Antes de ver el correcto uso y ejecución de este programa, primero se debe tener instalado JDK 11 y Gradle. Además, es necesario mencionar que la solución fue realizada y ejecutada en un Notebook NITRO 5 con sistema operativo Windows 11.
Se debe ejecutar la terminal y acceder al siguiente directorio:
\lab3_211333732_PachecoParra\ lab_3_211333732_PachecoParra
Luego, escribir el comando "gradlew.bat build" y después "gradlew.bat run" (ver Figura 5) Finalmente, luego de realizar estos pasos, es desplegado el menú de inicio del sistema de chatbots en donde ya se pueden realizar las interacciones mediante la consola tales como registrar un usuario o iniciar sesión.

Resultados Esperados

Se espera que el programa logre funcionar correctamente con todos los métodos implementados, permitir registrar un usuario normal o usuario administrador e iniciar sesión para desplegar el menú correspondiente para el tipo de usuario. En tal menú se tienen submenús para crear o modificar opciones, flujos y chatbots. Además de permitir interacciones entre un usuario y un chatbot o entre 2 chatbots distintos. Luego de cada una de las interacciones es posible ver el sistema y los chatbots dentro del sistema para corroborar el correcto funcionamiento del sistema de chatbots.
Otro de los resultados esperados es que el programa maneje correctamente los errores de lógica que se puedan producir, por ejemplo, al momento de registrar un usuario que ya se encuentra registrado o al iniciar sesión con un usuario que aún no ha sido registrado se entregan mensajes indicando el problema.

Posibles errores

Al momento de ejecutar el programa, si se entrega una entrada no válida (por ejemplo, si se pide ingresar un número entero y se entrega un string) el programa podría entregar error, ya que, se intentaron corregir todos los posibles errores de entrada, pero puede que algunos casos específicos hayan quedado sin resolver. Además, se tomó el supuesto en el método systemTalk de que la primera interacción inicia el primer chatbot, esto con fin de que la interacción con el chatbot tenga linealidad y no se pierda el sentido de la conversación con el usuario. Por último, en el método SystemSymulate se tomó una cantidad finita de frases por lo cual puede que la conversación entre los chatbots carezca un poco de sentido, ya que, los mensajes se obtienen de manera aleatoria.

Resultados

Los métodos implementados fueron probados de múltiples formas para verificar el correcto funcionamiento de cada uno. Los resultados fueron los esperados para cada opción probada.

Autoevaluación

Los métodos option, flow, flowAddOption, chatbot, chatbotAddFlow, system, systemAddChatbot, systemAddUser, systemLogin, systemLogout, systemTalk, systemSinthesis y systemSimulate fueron implementados en su totalidad con correcto funcionamiento para todos.

Conclusión

Después de llevar a cabo el programa de simulación de un sistema de chatbots utilizando el Paradigma Orientado a Objetos en Java, es evidente que se lograron satisfacer la mayoría de los requisitos planteados en la problemática. Inicialmente, se enfrentó el desafío de comprender y adaptarse al paradigma orientado a objetos, dado que previamente se había trabajado con el paradigma funcional y el paradigma lógico en los laboratorios anteriores. Sin embargo, a medida que se avanzó en la implementación de los requisitos se observó que, mediante la construcción de objetos y manejo de este paradigma, algunos de los métodos eran más cómodos de implementar debido a lo estructurado del paradigma.

En comparación al paradigma funcional, este paradigma ofreció ventajas tales como reusabilidad del código a través del uso de herencia, composición e interfaz. Además, es importante saber que, en el paradigma orientado a objetos, estos se pueden ir modificando. En cambio, en el paradigma funcional no se permite realizar modificaciones como tal, sino que se va creando cada elemento de nuevo.

En cuanto al paradigma lógico, este último presenta mayores características para problemas de búsqueda o patrones, por esta misma razón, se tiene una gran

desventaja en el tiempo de ejecución, siendo menos eficiente que el paradigma orientado a objetos y el paradigma funcional.
En cuanto a desventajas, en el paradigma orientado a objetos, se debe adquirir una mayor cantidad de conocimientos antes de poder manejarlo correctamente, ya que, que cuenta con muchos conceptos los cuales son nuevos y no están presentes en otros paradigmas, es por esto, que también su curva de aprendizaje y manejo de conceptos es más extensa que, por ejemplo, el paradigma funcional o el paradigma lógico.

Fue crucial abordar el problema de manera adecuada y trabajar representando cada elemento del sistema de chatbots como una clase, además de implementar los conceptos del POO. Esto ya que, con otro tipo de representación, como por ejemplo usando listas o árbol, la construcción de la solución se habría complicado significativamente. Además, el uso de la capa de Tipos de Datos Abstractos (TDA) permitió una mejor organización de la problemática permitiendo la creación de constructores, selectores y modificadores que podían ser llamadas en otros métodos.

Bibliografía y referencias

Java Documentation, Oracle extraído de: https://docs.oracle.com/en/java/

Martínez Ramírez, V. (2023). Enunciado Laboratorio 3 Paradigmas de Programación 2023/2 https://docs.google.com/document/d/1Psn6YqXfWA99n3AA-rLsvAJsc2-gqj6ot7j90ucLcog/edit

González Ibáñez, R. (2023). Proyecto Semestral de Laboratorio General 2023/2 https://docs.google.com/document/d/1L-B2b3J71Baqa_IuZt6EmRwDlxC0qzWn9YJAm6FIiJk/edit

Anexos

```java
/**
 * Maneja el proceso de registro de nuevos usuarios.
 * Permite a los usuarios registrarse como usuarios normales o administradores.
 */
1 usage   Reinaldo Pacheco
private static void procesoRegistro() {
    System.out.println("### Sistema de Chatbots - Registro ###");
    System.out.print("INTRODUZCA NOMBRE DE USUARIO:");
    String nombreUsuario = scanner.nextLine().toLowerCase();

    if (usuariosRegistrados.containsKey(nombreUsuario)) {
        System.out.println("El nombre de usuario ya esta registrado. Por favor, elija otro nombre.");
        return;
    }
    System.out.println("1. Registrar usuario normal");
    System.out.println("2. Registrar usuario administrador");
    System.out.print("INTRODUZCA SU OPCION:");
    int opcionRegistro = scanner.nextInt();
    scanner.nextLine();
    switch (opcionRegistro) {
        case 1:
            usuariosRegistrados.put(nombreUsuario, false);
            System.out.println("Usuario normal registrado exitosamente.");
            break;
        case 2:
            usuariosRegistrados.put(nombreUsuario, true);
            System.out.println("Usuario administrador registrado exitosamente.");
            break;
        default:
            System.out.println("Opcion no válida.");
            break;
    }
```

Figura 1: Método procesoRegistro

```java
switch (opcion) {
    case 1:
        menuOption();
        break;
    case 2:
        menuFlow();
        break;
    case 3:
        menuChatbot();
        break;
    case 4:
        visualizarChatbots();
        break;
    case 5:
        visualizarChatbotsConFlujos();
        break;
    case 6:
        ejecutarSimulacion();
        break;
    case 7:
        sintesisChatbot();
        break;
    case 8:
        simulacionEntreChatbots();
        break;
    case 9:
        continuar = false;
        break;
    default:
```

Figura 2: uso de switch-case menuAdministrador

```java
/**
 * Muestra todos los chatbots existentes en el sistema, con sus flujos y opciones.
 * Si no hay chatbots registrados, muestra un mensaje de error.
 * @return true si el usuario desea salir al menú anterior, false en caso contrario
 * @see Sistema_21133732_PachecoParra#getChatbots()
 * @see Chatbot_21133732_PachecoParra#getFlows()
 * @see Flow_21133732_PachecoParra#getOptions()
 * @see Option_21133732_PachecoParra#getMessage()
 */
2 usages  Reinaldo Pacheco
private static void visualizarChatbotsConFlujos() {
    if (sistema == null || sistema.getChatbots().isEmpty()) {
        System.out.println("No hay chatbots registrados en el sistema.");
        return;
    }

    for (Chatbot_21133732_PachecoParra chatbot : sistema.getChatbots()) {
        System.out.println("Chatbot: " + chatbot.getName());
        for (Flow_21133732_PachecoParra flujo : chatbot.getFlows()) {
            System.out.println("  Flujo: " + flujo.getNameMsg());
            for (Option_21133732_PachecoParra opcion : flujo.getOptions()) {
                System.out.println("Opcion: " + opcion.getMessage());
            }
        }
```

Figura 3: Método visualizarChatbotsConFlujos

```
INTRODUZCA SU OPCION:5
Chatbot: Inicial
  Flujo: Flujo Principal Chatbot 1
Bienvenido
¿Que te gustaria hacer?
Opcion: 1) Viajar
Opcion: 2) Estudiar
Chatbot: Agencia Viajes
  Flujo: Flujo 1 Chatbot1
¿Donde te gustaria ir?
Opcion: 1) New York, USA
Opcion: 2) Paris, Francia
Opcion: 3) Torres del Paine, Chile
Opcion: 4) Volver
  Flujo: Flujo 2 Chatbot1
¿Que atractivos te gustaria visitar?
Opcion: 1) Central Park
Opcion: 2) Museos
Opcion: 3) Ningun otro atractivo
Opcion: 4) Cambiar destino
  Flujo: Flujo 3 Chatbot1
```

Figura 3.1: Ejecución visualizarChatbotsConFlujos

```java
/**
 * Busca una opción por su código.
 *
 * @param flow el flujo asociado con la opción
 * @param code el código de la opción
 * @return la opción encontrada o null si no se encuentra
 */
1 usage  Reinaldo Pacheco *
private Option_21133732_PachecoParra findOptionCode(Flow_21133732_PachecoParra flow, String code) {
    try {
        int optionCode = Integer.parseInt(code.trim());
        return flow.getOptions().stream() Stream<Option_21133732_PachecoParra>
                .filter(option -> option.getCode() == optionCode)
                .findFirst() Optional<Option_21133732_PachecoParra>
                .orElse( other: null);
    } catch (NumberFormatException e) {
        return null;
    }
}
```

Figura 4: Método findOptionCode

```
C:\Users\Reinaldo Pacheco>cd C:\Usach\2-2023\Paradigmas\lab3_21133732_PachecoParra\lab3_21133732_PachecoParra

C:\Usach\2-2023\Paradigmas\lab3_21133732_PachecoParra\lab3_21133732_PachecoParra>gradlew.bat build
Starting a Gradle Daemon, 1 busy Daemon could not be reused, use --status for details

Deprecated Gradle features were used in this build, making it incompatible with Gradle 9.0.

You can use '--warning-mode all' to show the individual deprecation warnings and determine if they come from yo

For more on this, please refer to https://docs.gradle.org/8.2/userguide/command_line_interface.html#sec:commanc

BUILD SUCCESSFUL in 9s
5 actionable tasks: 4 executed, 1 up-to-date
C:\Usach\2-2023\Paradigmas\lab3_21133732_PachecoParra\lab3_21133732_PachecoParra>gradlew.bat run

> Task :run
### Sistema de Chatbots - Inicio ###
1. Login de Usuario
2. Registro de Usuario
3. Salir
INTRODUZCA SU OPCION:
<=========----> 75% EXECUTING [8s]
> :run
```

Figura 5: Ejecución del programa mediante Gradle

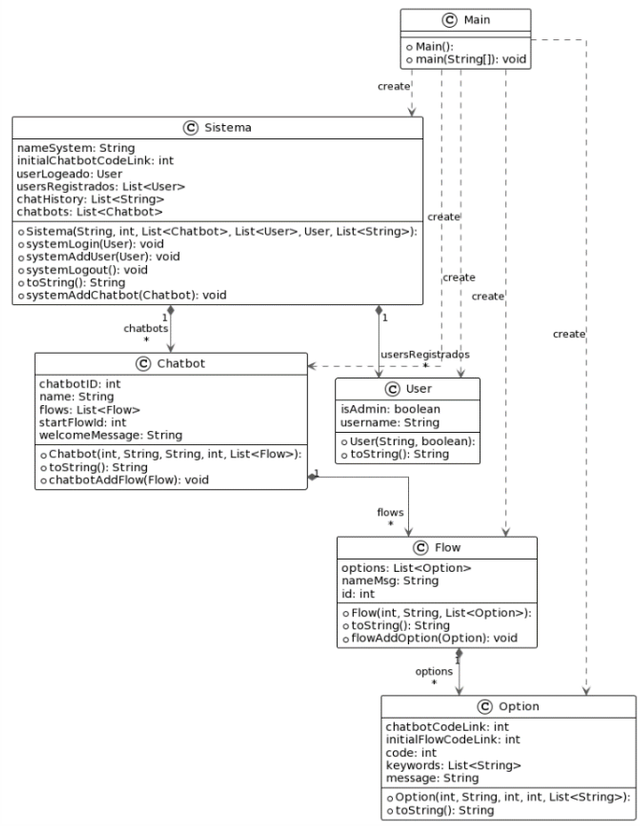

Figura 6: Diagrama de análisis inicial

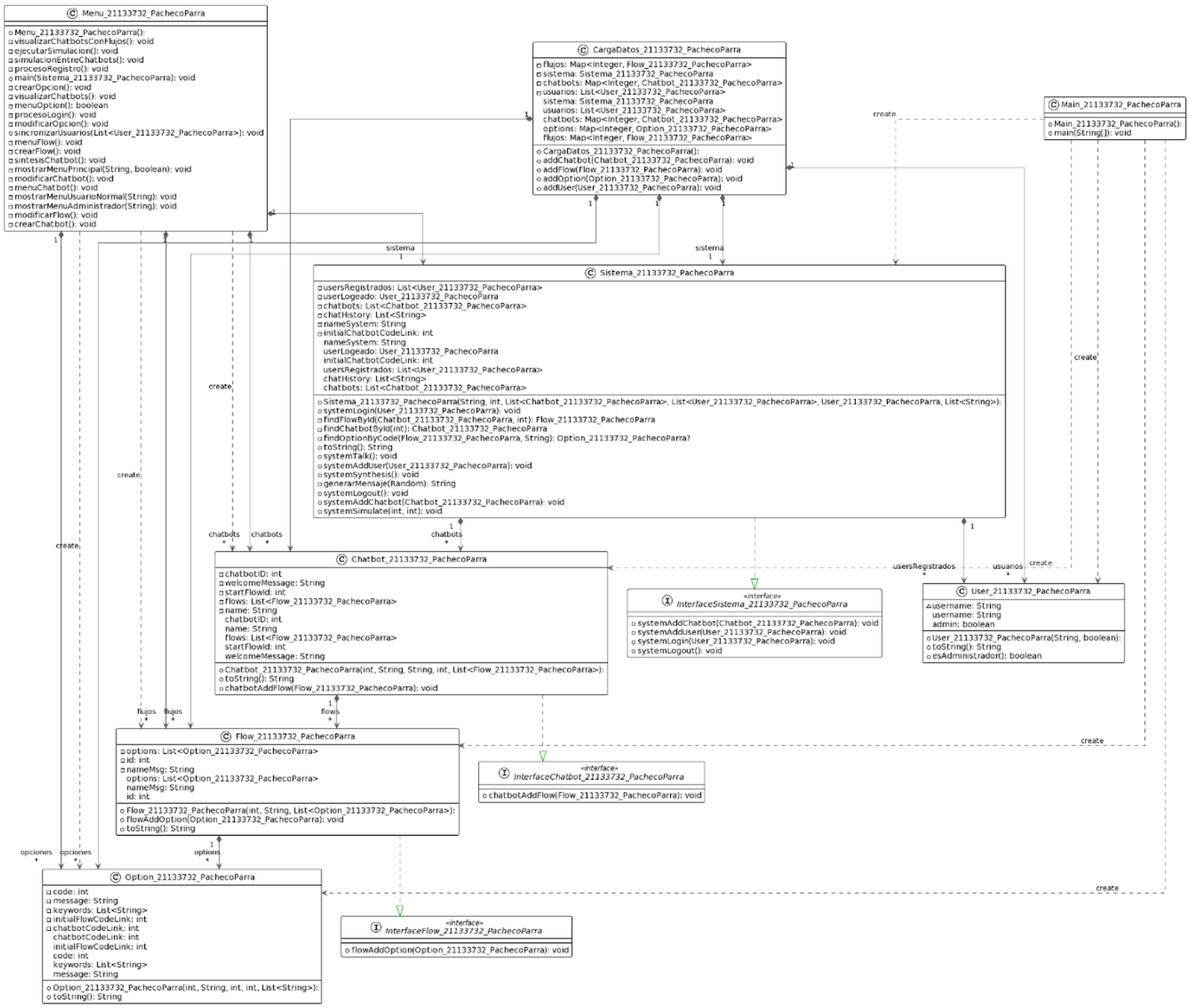

Figura 7: Diagrama de Diseño Final