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

LABORATORIO 4

PARADIGMAS DE PROGRAMACIÓN

Alberto Rodríguez Z.

Profesores:	Roberto González
	Daniel Gacitúa
	Víctor Flores
Fecha de Entrega:	07-09-2018

Santiago de Chile

1- 2018

TABLA DE CONTENIDOS

TABLA DE FIGURAS

CAPÍTULO 1. INTRODUCCIÓN

El presente informe tiene como objetivo principal ser una referencia a la línea de pensamiento y al contexto de desarrollo del código fuente que lo acompaña para la presentación del laboratorio 4 del curso de Paradigmas de Programación. En esta ocasión se hará uso del principalmente del paradigma de programación Orientado a Objetos y tambien un pequeño uso del paradigma de programación Orientado a Eventos con el lenguaje de programación C# (en este caso es solo C#, pero podían usarse otros lenguajes).

En primer lugar, el problema a resolver es la creación e implementación de un chatbot. De este mismo modo, es relevante mencionar que un chatbot se puede definir como programas computacionales que pueden mantener una conversación con un ser humano, sin embargo, esta interacción queda limitada por un contexto/temática y una serie de parámetros. Asimismo, los chatbot resultan ser de utilidad para manejar algunas fases de la conversación como pueden ser saludos de bienvenida, solicitud de datos y responder a algunas preguntas específicas hechas por el usuario. Toda interacción con el chatbot se desarrolla mediante la interfaz gráfica del programa.

Por consiguiente, en este informe la solución para el programa con interfaz gráfica implementada para el chatbot trabaja en base a la temática de atención al cliente en un negocio de comida rápida, en donde se crean flujos conversacionales protocolares, estructuras gramaticales simples y vocabulario conocido por el chatbot. De tal forma dando los parámetros y temática se puede mantener una conversación con el chatbot.

Por lo tanto, se establece como objetivo de este laboratorio, la aplicación de las técnicas aprendidas en clases, poder desarrollar un programa con interfaz gráfica, además ir más allá e investigar cuando corresponda, con el fin de crear un algoritmo eficaz que funcione bajo el paradigma orientado a objetos.

CAPÍTULO 2. MARCO TEÓRICO

2.1 PARADIGMA ORIENTADO A OBJETOS

La programación orientada a objetos (POO por sus siglas) es un paradigma de programación que usar objetos y sus interacciones para diseñar aplicaciones y programas de computadora. Esta basado en varias técnicas incluyendo herencia, modularidad (sigue conceptos de acoplamiento y cohesión), polimorfismo y encapsulamiento.

Uno de los conceptos claves de la POO es la abstracción, que es un proceso de interpretación que implica reconocer y enfocarse en las características importantes de una situación u objeto, deja a un lado los detalles del objeto y definir las características específicas de éste, que los distingan de los demás objetos, tambien hay que centrarse en lo que es y lo que hace el objeto, antes de decidir cómo debería ser implementado.

En la POO se trabaja con el concepto de clase, que no es mas que un TDA con propiedades adicionales a la hora de implementarse. La clase es una plantilla de código para crear objetos, la cual posee variables de estados iniciales (atributos) y un cierto comportamiento (llamados métodos). Al igual que un TDA, una clase puede modelar entidades de la realidad de forma que puedan ser procesadas por una maquina computacional.

Un atributo es una característica de un objeto, que ayuda a definir su estructura y permite diferenciarlo de otros objetos. Modela el estado actual de un objeto, normalmente están compuestos por variables primitivas y TDA.

Los métodos son los comportamientos de un objeto y permite identificar la forma en que actúa respecto a su entorno o respecto a otros objetos. Además, representa una operación o función que un objeto realiza. El conjunto de métodos de un objeto determina el comportamiento general del objeto.

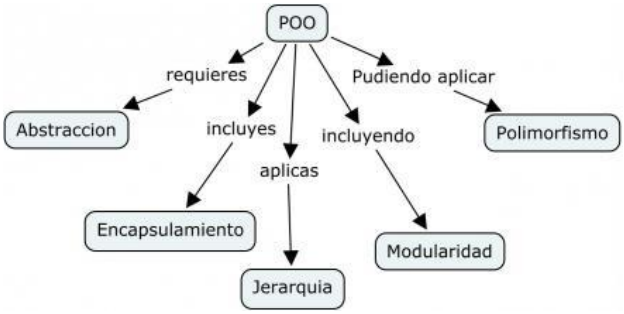

Figura 1.- Diagrama de las características principales un POO.

2.2 PARADIGMA ORIENTADO A EVENTOS

Es un paradigma de programación en que la estructura y ejecución dependen por lo sucesos que ocurra en el sistema o que haga el usuario. El flujo del programa no es definido por el programador, sino que por las acciones que realiza el sistema o el usuario. Estas acciones son llamadas eventos. Los eventos es una acción u ocurrencia que pueden ser detectada por un programa, se origina tras una acción del sistema o del usuario de manera asíncrona a la ejecución del programa.

Las aplicaciones desarrolladas con programación dirigida por eventos implementan un bucle principal o main loop donde se ejecutan las dos secciones principales de la aplicación. El selector de eventos y el manejador de eventos.

En la programación dirigida a eventos, al comenzar la ejecución del programa se llevarán a cabo las inicializaciones y demas código inicial y a continuación el programa quedara bloqueado hasta que se produzca algún evento. Cuando alguno de los eventos esperados por el programa tenga lugar, el programa pasará a ejecutar el código del correspondiente administrador del evento.

Un manejador de eventos es una subrutina llamada tras la entrada de un evento con una determinada característica. La subrutina ejecuta el código dentro de ella cada vez que detecta un evento de su tipo, ejemplo: click del rato, expiración de timer(timeout), datos disponibles tras una estructura de archivo, etc.

Un ejemplo claro es el Framework o marco común de librerías de Visual Studio, en los que en cada momento del programa (objetos,controles,etcétera) se le asignan una serie de eventos que generará dicho elemento, como la pulsación del ratón sobre él o el redibujado control.

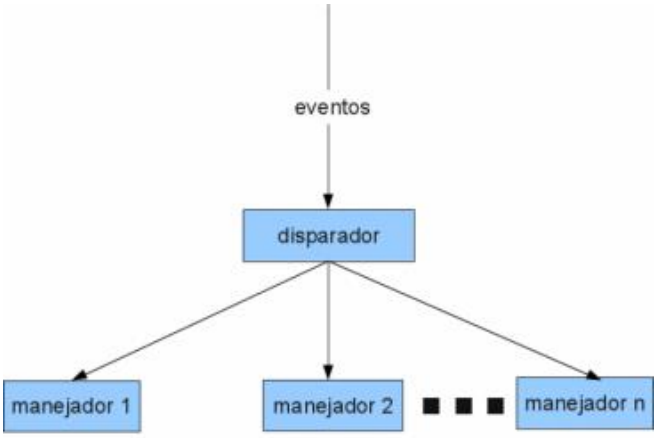

Figura 2.- Diagrama que ilustra cada manejador de eventos a la entrada de un evento especifico.

CAPÍTULO 3. DESCRIPCIÓN DEL PROBLEMA

Se solicita crear un algoritmo en el lenguaje de programación C# que simule la conversación entre el usuario y un chatbot, un chatbot que contiene personalidad y un vocabulario. La conversación debe funcionar en base a una temática manteniendo una conversación fluida con un inicio de la conversación y un final. La conversación se va almacenando en un registro llamado log, como tambien una semilla seed que da variabilidad a las respuestas. Toda la interacción con el chatbot es mediante una interfaz grafica donde el usuario a través de los botones podrá selecciones funciones especificas que quiera que realice el chatbot y con una caja de texto puede enviar mensajes con al chatbot. Al ser POO se trabaja con clases y estructuras para el programa, las cuales tienen las siguientes caracetristicas.

1.**Chatbot**: Clase que interactuará con el usuario respondiendo sus mensajes. Esta clase constituye las posibles respuestas del chatbot.
2.**Log**: Estructura en la cual se almacena toda la conversación entre el usuario y el chatbot. Cada mensaje debe incluir fecha y hora de cuando se realiza el mensaje.
3.**Usuario:** Clase que obtiene la información útil entre el usuario y el chatbot. Para este laboratorio el usuario determina la personalidad con la que se desarrolla el chatbot en la conversación.

3.1 ANÁLISIS

El principal problema radica en la capacidad de darle “sentido” a una conversación, ya que cuando el usuario envía un mensaje, el chatbot debe ser capaz de responderle lógicamente en relación con lo preguntado por este. Por lo tanto, debe existir una gran posibilidad de respuestas para que el chatbot tenga la capacidad de seleccionar la indicada.

De otro modo ya que se trabaja con POO, se debe crear las clases correspondientes y poder relacionarlas entre sí, o sea se instancian objetos de las clases anteriores y se deben establecer relaciones para que una conversación pueda funcionar. Estas clases se deben trabajar mediante interfaces graficas donde, algunas deben ser una clase y según lo que desee el usuario ir relacionándolas con las demás clases (que pueden ser interfaces o no).

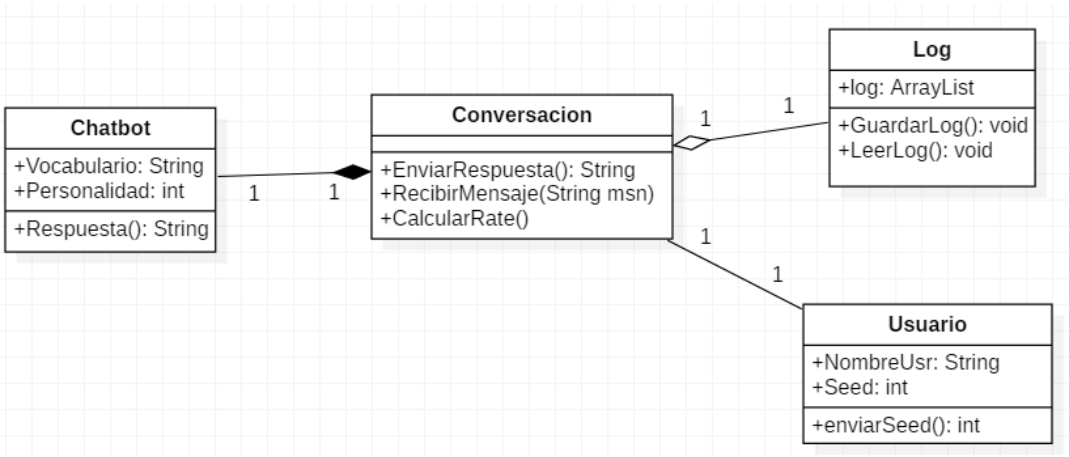

Figura 3.- Diagrama de clases UML del problema.

CAPÍTULO 4. DESCRIPCIÓN DE LA SOLUCIÓN

Puesto que se deben cumplir las clases y estructuras mencionadas en el capitulo anterior, lo primero que se debe hacer es definir la interfaz gráfica principal. Para lograrlo verificamos que funcionalidades mínimas debe tener nuestro chatbot, estas son: Comenzar conversación, enviar mensajes al chatbot, guardar Log, cargar Log, finalizar conversación y evaluar al chatbot. Ya teniendo en cuenta estas funcionalidades se puede diseñar la interfaz grafica con sus botones y cajas de textos correspondientes.

Antes de empezar la conversación, se realiza una interfaz anterior, esta es la clase usuario la cual le solicita al usuario la semilla seed con la que se va a desarrollar la conversación, enviando esta semilla se desplegará la clase principal llamada “ChatbotOficial” mediante una interfaz gráfica con todas las opciones que puede realizar un usuario. Si no se ingresa esta semilla no será posible comenzar la conversación ya que el chatbot no tendrá personalidad.

Ya en la clase principal el usuario puede realizar dos opciones primero, comenzar la conversación o cargar log. Comenzar la conversación da inicio a un nuevo chat con el chatbot, donde este da el mensaje de bienvenida. Para poder el chatbot enviar el mensaje de bienvenida, instancia a la clase Chatbot (que no es una interfaz gráfica) la cual le brindara una serie de métodos con los cuales podrá obtener el mensaje correcto. La clase Chatbot contiene el método “obtenerTiempo” con el cual se podra determinar la fecha y la hora con la cual el chatbot está solicitando ese evento. Junto con la función “obtenerSaludoBienvenida”, la clase podrá determinar cual es el mensaje correcto que debe enviar el chatbot, contando la hora y la personalidad ingresada. El otro evento que puede solicitar el usuario es el de cargar Log, al presionar el botón se desplegara una ventana del computador donde el usuario puede elegir cual es el archivo que desea leer, siempre y cuando sea un Log y sea de tipo “.txt”. Al elegir el log este se desplegará en la pantalla y se podrá

seguir la conversación desde alli. Si no se elige una de estas opciones y el usuario optar por presionar otro botón o enviar un mensaje, el chatbot indicará que debe comenzar la conversación primero.

Cuando se inicia la conversación, el usuario puede elegir las demás opciones. Lo más lógico sería que quiera enviarle mensaje al chatbot, para eso tiene una caja de texto donde puede escribir el mensaje para enviar. Cuando se envía el mensaje, el chatbot debe ser capaz de leer este mensaje y saber que responderle al usuario, para eso se instancia nuevamente la clase Chatbot la cual gracias al método "darRespuestaChatbot" podrá elegir mediante el mensaje enviado cual es la respuesta mas indicada para el usuario, esta puede ser de muchos tipo: Puede ser de afirmación, de pregunta o de guardar información en caso de que se guarde la cuenta que ira acumulando el cliente con su compra o su nombre. Para cada mensaje enviado, aparecerá por la pantalla de la interfaz el mensaje enviado por el usuario y la respuesta del chatbot los dos con su tiempo enviado (obtenido por la función antes mencionada).

Otra opción tambien posible para el usuario es realizar la función guardar log. Si se quiere guardar la conversación actual en un archivo de texto, esta instanciará a la función guardarArchvivo la cual desplegará una interfaz gráfica con 2 cajas de texto donde se debe Ingresar la ruta donde se quiere guardar el archivo y el nombre de este. Ya ingresados si todo es correcto se crea el archivo, sino indicara cual es el error especifico. Para poder crear el archivo la función guardarArchivo llama a su método "saveArchive" con el cual se solicita el log de la conversacion que es de tipo ArrayList el cual irá recorriendo para escribir el archivo de texto.

Si no se quiere enviar más mensajes al chatbot, se puede terminar la conversación mediante el botón del mismo nombre quien envía un mensaje de despedida, este mensaje lo obtiene gracias al método de la clase Chatbot "obtenerDespedida", después de unos 6 segundos aparece otra interfaz solicitando calificar al chatbot, esta es la clase Rate, la cual solo puede ser instanciada cuando se finaliza la conversación. Es clase muestra una interfaz grafica donde pregunta al usuario con que nota califica al chatbot, esta nota debe ser entre 1 y 5 (de muy malo a excelente). Ya ingresada la nota le chatbot se despide y finaliza el programa cerrando todas las interfaces.

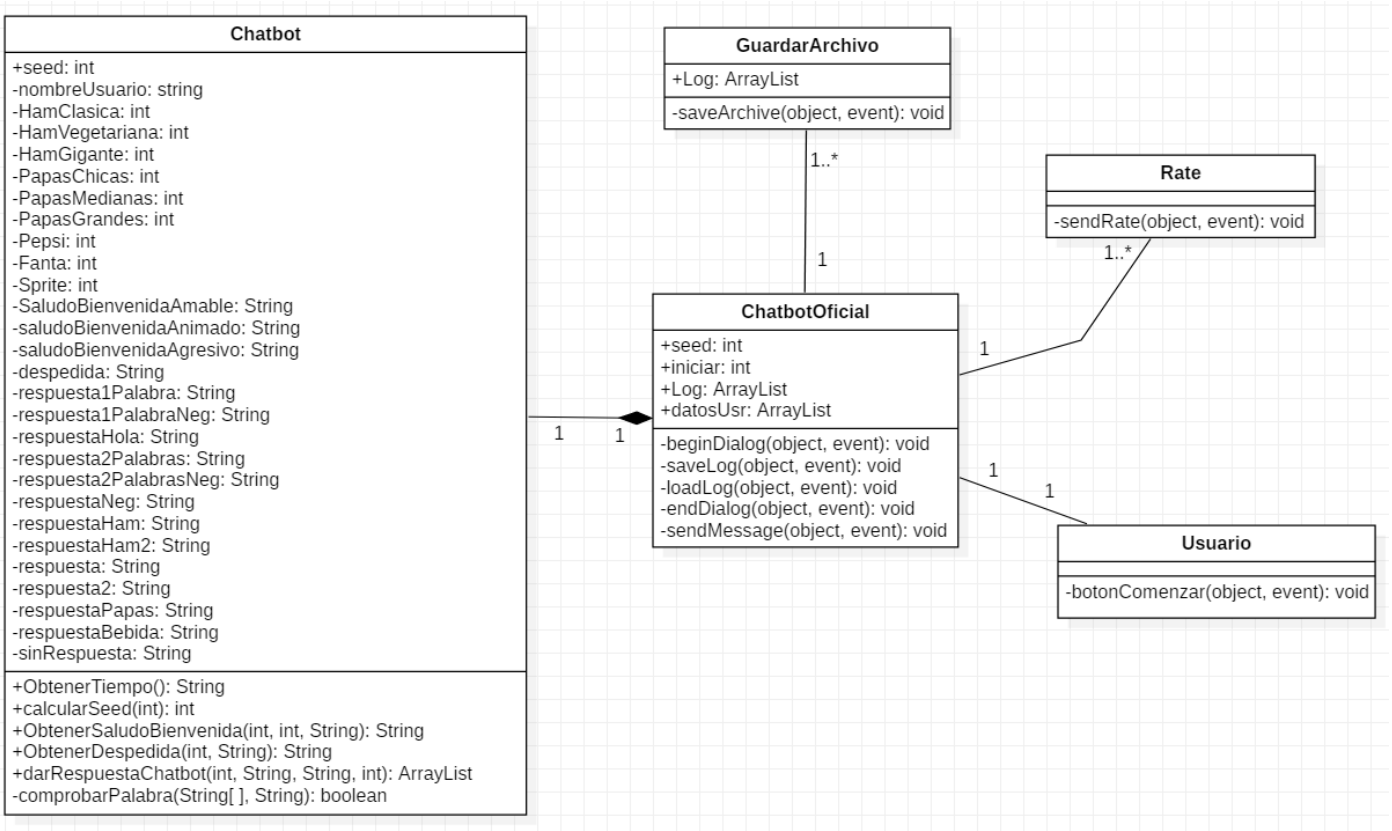

Figura 4.- Diagrama de clases UML de la solución.

CAPÍTULO 5. RESULTADOS

Tiempo de respuesta aceptable, es decir, el usuario no nota o queda en espera de resultados, ya sea cuando envía el mensaje en el que el chatbot tiene que responder corriendo todas las variadas posibilidades para verificar, a pesar de que la ejecución de los programas orientadas a objetos es más lenta y el tamaño de los programas es mayor.

Se realizaron todas las funcionalidades mínimas incluidas en el enunciado, todas funcionaban de manera exitosa en las pruebas que se le hicieron. Se verifica que todos los eventos realizados por el usuario cumplan con la solicitud. Las pruebas que se hicieron era al inicio cambiar la semilla para ver si el chatbot cambiaba su personalidad, además de siempre preguntarle respecto al contexto.

Como muestra la figura 5, ilustra una conversación con el chatbot desde que se comienza la conversación hasta que da la cuenta final de la compra, la personalidad del seed fue ingresada anteriormente.

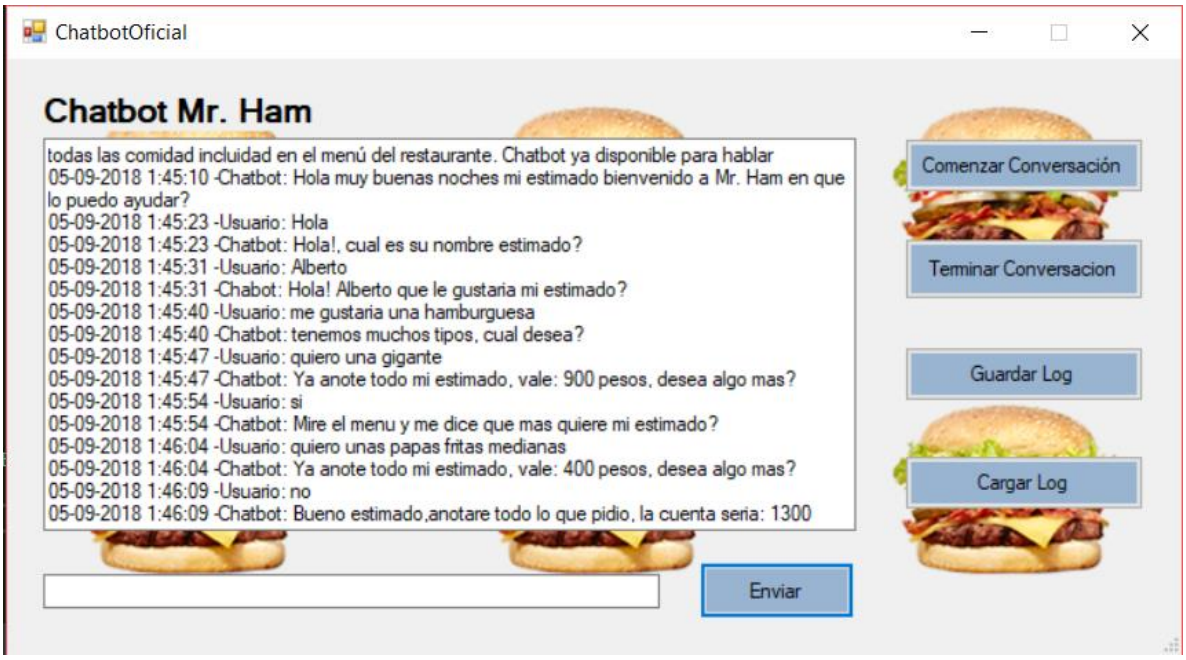

Figura 5.- Conversación entre chatbot y usuario.

Al terminar la conversación, pudimos verificar que después de 6 segundos el programa ingresa una interfaz donde solicita calificar al chatbot, para después terminar el programa.

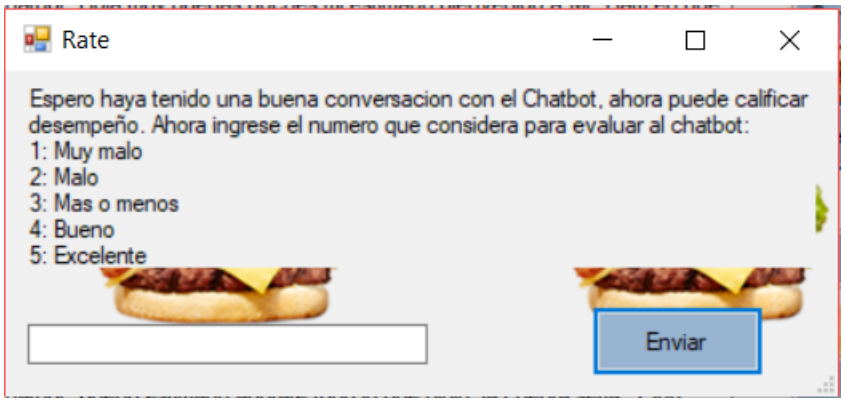

Figura 6.- Interfaz donde solicita nota para el chatbot.

CAPÍTULO 6. CONCLUSIONES

Realizando este laboratorio se pudo concluir que el paradigma de programación Orientada a Objetos tiene las ventajas de ser un paradigma simple de desarrollar, una buena abstracción de las clases, objetos y atributos nos brinda una implementación mas detallada puntual y coherente. Como se puede trabajar estas clases mediante interfaces graficas con las cuales son más fácil acceder al usuario, como la reutilización de código es algo que se puede observar en este paradigma siempre y cuando se desarrollen adecuadamente las clases. Se puedo entender cómo funciona el paradigma Orientado a Eventos y como este es multiparadigma ya que se implementa con el POO, para poder desarrollarse adecuadamente.

En comparación al laboratorio anterior, que tambien era POO, pero todo se ejecutaba por consola. Pudimos comprobar la facilidad que tiene para el usuario implementarlo mediante una interfaz gráfica y como el mismo usuario tiene una interacción más directa comparado con el laboratorio anterior, analizando que los dos pedían funcionalidades similares se puedo comparar entre los dos laboratorios.

En segundo lugar, respecto al problema del laboratorio, ha sido interesante plantear el problema de como poder desarrollar una conversación con un chatbot, a través de distintas clases y técnicas que tiene el POO, junto con la implementación de una interfaz gráfica y la implementación de la biblioteca estándar de C#. De modo que, se puede afirmar que se realiza un gran trabajo con C# al poder trabajar directamente con las clases mediante la interfaz gráfica y como relacionar los objetos mediante los eventos ingresados por el usuario.

Finalmente, puesto que se ha logrado desarrollar un chatbot con cierta temática, con el cual uno puede mantener una conversación, es posible concluir que se ha logrado cumplir con el objetivo principal del laboratorio.

REFERENCIAS

- Josué Turcios. *Programación orientada a objetos y programación orientada a eventos.* (España) Recuperado de: https://es.slideshare.net/josueivan/programacion-orientada-a-objetos-y-programacion-orientada-a-eventos

- Curso de Paradigmas de programación. *Presentaciones paradigmas catedra(2018-1).* (Chile) Recuperado de https://drive.google.com/drive/folders/1HbFEr1f5MCntqecuqNRPfEsoVowyYssy

- Victor García. *Programación Orientada a Eventos.* (Colombia) Recuperado de: https://programarjava.wordpress.com/2011/12/13/programacion-orientada-a-eventos/

- LearnWTutorials. *Programación en Visual C#.* (España) Recuperado de: https://www.youtube.com/playlist?list=PLhbcXfA7vHX64kzwPvdHGOsiPh9q3IOhF