FACULTAD DE INGENIERÍA Y CIENCIAS APLICADAS

IMPLEMENTACIÓN DE UN CHATBOT INTEGRADO A UN LMS, COMO ASISTENTE PARA LA GESTIÓN EDUCATIVA

AUTOR

JULIO DAVID PAZMIÑO RICAURTE

AÑO

2020

FACULTAD DE INGENIERÍA Y CIENCIAS APLICADAS

IMPLEMENTACIÓN DE UN CHATBOT INTEGRADO A UN LMS, COMO ASISTENTE PARA LA GESTIÓN EDUCATIVA

Trabajo de Titulación presentado en conformidad con los requisitos

establecidos para optar por el título de Ingeniero en

Electrónica y Redes de Información

Profesor Guía:

William Eduardo Villegas

Autor:

Julio David Pazmiño Ricaurte

Año:

2020

DECLARACIÓN DEL PROFESOR GUÍA

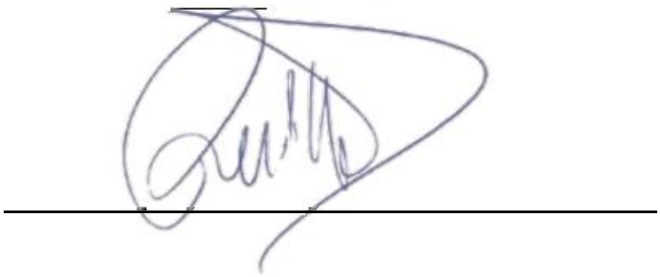
"Declaro haber dirigido el trabajo, Implementación de un chatbot integrado a un LMS, como asistente para la gestión educativa, a través de reuniones periódicas con el estudiante Julio David Pazmiño Ricaurte, en el semestre Marzo – Julio 2020, orientando sus conocimientos y competencias para un eficiente desarrollo del tema escogido y dando cumplimiento a todas las disposiciones vigentes que regulan los Trabajos de Titulación".

William Eduardo Villegas

1715338263

DECLARACIÓN DEL PROFESOR CORRECTOR

"Declaro haber revisado este trabajo, Implementación de un chatbot integrado a un LMS, como asistente para la gestión educativa, del estudiante Julio David Pazmiño Ricaurte, en el semestre Marzo – Julio 2020, dando cumplimiento a todas las disposiciones vigentes que regulan los Trabajos de Titulación".

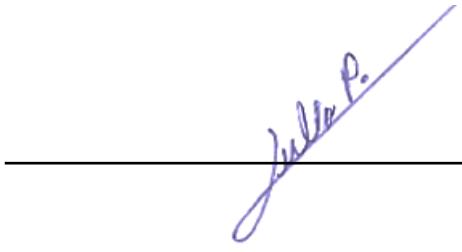


Iván Patricio Ortiz Garcés

CI: 0602356776

DECLARACIÓN DE AUTORÍA DEL ESTUDIANTE

“Declaro que este trabajo es original, de mi autoría, que se han citado las fuentes correspondientes y que en su ejecución se respetaron las disposiciones legales que protegen los derechos de autor vigentes.”



Julio David Pazmiño Ricaurte

1721884474

AGRADECIMIENTOS

A mi madre, que me brindó su apoyo en todo momento; gracias a su sacrificio, paciencia y perseverancia me fue posible llegar a cumplir esta meta tan importante.

A mi familia con su apoyo incondicional y amigos con los que se compartieron gratos momentos en el avance de la carrera.

DEDICATORIA

Dedico este trabajo de titulación a mi madre, abuelita y tías; por forjar a la persona que soy actualmente. A Dios que es el guía del destino de mi vida. Maestros y amigos que constantemente me motivaron a alcanzar mis objetivos en el transcurso de esta carrera universitaria.

RESUMEN

El presente trabajo de titulación se enfoca en el desarrollo de un chatbot como asistente para le gestión educativa, el cual esté implementado en una plataforma LMS; en este caso, dentro del aula virtual de la Universidad de las Américas (Quito-Ecuador). Para lo cual se investigó información pertinente a desarrollos previos de chatbots dentro del entorno educativo, a fin de tener una base sobre la cual establecer esta implementación.

Este chatbot se desarrolló utilizando herramientas especializadas como lo son el software Visual Studio 2019 y el SDK Microsoft Bot Framework, los cuales ayudaron en la creación del proyecto. A su vez, también se utilizó un servicio cognitivo externo llamado “LUIS” (Language Understanding Intelligent Service), el cual se encargó de dotar de inteligencia artificial y comprensión del leguaje natural a este chatbot.

Para la implementación de este proyecto, se utilizó el servicio de Microsoft Azure, el cual ayudó con la publicación del chatbot como un servicio web capaz de agregarse en el diseño del aula virtual de la universidad. Esta aplicación puede brindar ayuda en el manejo del aula virtual y servicio Banner, tanto a profesores como alumnos.

Finalmente, se evaluó el desenvolvimiento de esta herramienta en un entorno educativo, siendo utilizado por varios alumnos a los cuales posteriormente se les encuestó con respecto a la experiencia obtenida, con lo cual se reafirmó la ayuda y utilidad que este chatbot puede brindar en la gestión educativa que se lleva a cabo dentro de los diferentes portales web de la universidad.

ABSTRACT

This degree work focuses on the development of a chatbot as an assistant for educational management, which is implemented in an LMS platform; in this case, within the virtual classroom of the University of the Americas (Quito-Ecuador). For which information pertinent to previous developments of chatbots within the educational environment was investigated, in order to have a basis on which to establish this implementation.

This chatbot was developed using specialized tools such as Visual Studio 2019 software and the Microsoft Bot Framework SDK, which helped in the creation of the project. In turn, an external cognitive service called “LUIS” (Language Understanding Intelligent Service) was also used, which was in charge of providing artificial intelligence and natural language understanding to this chatbot.

For the implementation of this project, the Microsoft Azure service was used, which helped with the publication of the chatbot as a web service capable of being added to the design of the university's virtual classroom. This application can provide help in the management of the virtual classroom and Banner service, both to teachers and students.

Finally, the development of this tool in an educational environment was evaluated, being used by several students who were subsequently surveyed regarding the experience obtained, thus reaffirming the help and usefulness that this chatbot can provide in management educational that takes place within the different web portals of the university.

ÍNDICE DEL CONTENIDO

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

INTRODUCCIÓN

Actualmente, los chatbots marcan una tendencia en diversos campos como la medicina, la industria de productos y servicios, entre otros. En la educación actual, los chatbots son un campo emergente donde aún no se ha aprovechado todo su potencial.

Hoy en día, existen estudios que sugieren el potencial que tienen los chatbots para mejorar los procesos y resultados de aprendizaje. Además, tienen una presencia creciente en la sociedad moderna, convirtiéndose en partes integrales de la cotidianidad. Desde asistentes personales en dispositivos móviles para asistencia técnica a través de líneas telefónicas, hasta ser utilizados para consultas de salud. (Serban et al., 2017).

En 2015, el tamaño del mercado de chatbot comprendía 113 millones de dólares de ganancias y se proyecta que para el 2024 serán de 994.5 millones de dólares, lo que demuestra la creciente frecuencia de uso de esta tecnología. (McKinsey & Company, 2016; Statista, 2017).

El uso de aplicaciones de mensajería también ha aumentado exponencialmente en los últimos años y fomentado el desarrollo de los chatbots. En 2016, se estima que aproximadamente el 75% de todos los usuarios de teléfonos inteligentes usaron algún tipo de aplicaciones de mensajería (Maruti Techlabs, 2017).

Además, los analistas predicen que, para este año 2020, el 50% de todas las búsquedas se realizarán mediante comandos de voz, y los clientes gestionarán el 85% de sus relaciones empresariales sin interactuar con un ser humano. Por lo que el uso de esta tecnología emergente se vuelve cada vez más frecuente con el pasar de los años (Gartner, 2016; Olson, 2016).

1. CAPÍTULO I

En este capítulo se definirá la motivación del desarrollo de este trabajo de titulación, el alcance del mismo, los diferentes objetivos planteados y una breve reseña sobre la evolución de los chatbots en el estado del arte.

1.1. MOTIVACIÓN

El incentivo para el desarrollo de este trabajo de titulación es el poder ayudar a mejorar la gestión educativa tanto para estudiantes y docentes al momento de manejar el aula virtual de la universidad. También se espera con el desarrollo de este tema, los profesores puedan implementar recursos de e-learning (como test existentes en Moodle u otros LMS) y puedan adecuar su uso para integrarlos en un chatbot.

La educación está basada en la comunicación e interacción. Por lo que los chatbots tienen un potencial muy importante en este campo, gracias a su habilidad de comunicación mediante lenguaje natural.

Los chatbots poseen una interfaz de usuario basada en una conversación y pueden estar presentes entre la comunicación de estudiantes y maestros, aportando con información y contenido actualizado. Pueden ser mediadores en las interacciones y en un ambiente de aprendizaje virtual, pueden aportar información de manera inmediata (Bii, 2013).

Esta tecnología es capaz de proporcionar, tanto a profesores como alumnos, información continua acerca de su progreso (qué áreas de la asignatura están generando mayores dificultades, cómo se desenvuelve cada estudiante, cuántas pruebas han hecho, cuáles son las más fáciles, problemas de alumnos para mantener el ritmo, etc.). Estos datos son útiles cuando se lleva a cabo una evaluación continua y controlada por parte de los maestros.

A su vez, pueden analizar los datos que se generan en las conversaciones y enviar una notificación personalizada a cada estudiante. También advertir al profesor sobre los estudiantes con dificultades en alguna asignatura o sugerir retos específicos dependiendo el caso.

Para el uso de los chatbot, no es necesario una instalación previa ya que vienen incorporados dentro de otros servicios. Tampoco es necesario un aprendizaje previo a su uso, debido a que la mayoría de las personas alguna vez han utilizado una aplicación de mensajería.

Un chatbot puede imitar una conversación en lenguaje natural por un canal de comunicación común. Por lo que los usuarios, por lo general, saben la forma de interactuar en dichas conversaciones sin requerir capacitación previa.

El uso de los chatbots enfocados en educación todavía es un área emergente, sobre todo en el Ecuador, y el desarrollo de este tema puede ayudar al país a entrar y adaptarse en esta nueva era de aprendizaje online.

1.2. ALCANCE

El alcance de este trabajo de titulación es presentar el diseño y la implementación de un prototipo de chatbot integrado a una plataforma LMS, en este caso el aula virtual de la Universidad de las Américas (Quito-Ecuador); y mostrar cómo estos pueden convertirse en una herramienta de ayuda para la gestión educativa. Específicamente evidenciar cómo este prototipo puede contribuir con el manejo de las funcionalidades que brinda el aula virtual a estudiantes y maestros.

El prototipo estará integrado a la página web del aula virtual de la universidad, para que pueda demostrar su propósito de ayudar en la gestión educativa. Este proyecto se desarrollará con la ayuda de herramientas especializadas las cuales garanticen un correcto funcionamiento y desenvolvimiento del chatbot. Para ello el chatbot se implementará en Microsoft Azure, utilizando para su desarrollo el SDK Bot Framework de la plataforma Visual Studio 2019.

1.3. OBJETIVOS

1.3.1. OBJETIVO GENERAL

Implementar un chatbot integrado a una plataforma LMS como asistente para la gestión educativa en un entorno universitario.

1.3.2. OBJETIVOS ESPECÍFICOS

- Investigar información sobre el uso e implementación de los chatbots en el ámbito de la educación.
- Implementar un chatbot con inteligencia artificial, capaz de interactuar en lenguaje natural con los usuarios.
- Evaluar el desempeño del chatbot dentro de un entorno educativo.

1.4. ESTADO DEL ARTE

Las nuevas tecnologías e-Learning, son herramientas que ayudan a mejorar y facilitar la calidad del aprendizaje. Agilitan y simplifican varios procesos que se llevan a cabo en la educación, lo que significa un beneficio para los usuarios. Estas nuevas tecnologías junto con herramientas de gestión, como los LMS (sistemas de gestión de aprendizajes), han posibilitado una enseñanza mucho más flexible.

Esto permite a los estudiantes tener un aprendizaje más personalizado y enfocado en el uso las TICs (tecnologías de información y comunicación). A su

vez, también apoyan a los docentes a interactuar con los estudiantes de forma interactiva y sencilla, abriendo la posibilidad de compartir una gran variedad de contenidos actualizados. (Farkash, 2018).

Entre estas herramientas de e-Learning, la última innovación que se presenta para mejorar la experiencia de aprendizaje, además de aumentar la calidad educativa, es el chatbot. Esta nueva tecnología surge como una necesidad dentro del proceso de enseñanza que se lleva a cabo dentro de un LMS, para promover el aprendizaje personalizado y satisfacer requerimientos de los usuarios en cualquier momento. Los chatbots junto con el uso de inteligencia artificial (IA), la cual se usa en varios campos corporativos y educativos, pueden ayudar al docente a adaptarse rápidamente a las necesidades de los estudiantes de una manera más personalizada. (García, Fuentes, y Molas; 2018).

Los chatbots son programas conversacionales que se utilizan para realizar diálogos auditivos o textuales. Evidencia del uso de algunos chatbots en la educación son el caso de los chatbots: “Jill Watson”, “Alice”, etc. El análisis de desenvolvimiento de estas implementaciones sugiere que estos programas tienen el potencial de cambiar la forma en que los estudiantes aprenden y buscan información. También pueden resolver el problema del apoyo individual a los estudiantes, especialmente en escenarios de aprendizaje a gran escala. (García, Fuentes, y Molas; 2018).

Los chatbots son capaces de simular conversaciones e integrarse a servicios existentes para convertirse en un nuevo tipo de aplicación de e-Learning. La cual es capaz de aprovechar ventajas de aplicaciones comerciales como Facebook, Twitter, WhatsApp, etc.; sin esfuerzo de instalación y con total disponibilidad. (Pereira, Medina y Díaz, 2016, p. 97).

1.4.1. HISTORIA DE LOS CHATBOTS

Los chatbots han visto la luz desde mediados de los años 60 con la aparición de “Eliza” (1966), también denominado “chatter bot”, el cuál fue el primer chatbot creado en el MIT (Instituto de Tecnología de Massachusetts) por Joseph Weizenbaum. Su objetivo era habilitar interfaces de comunicación entre las máquinas y el hombre como se muestra en la Figura 1. Fue una gran innovación para su época, por lo que ganó gran popularidad imitando respuestas de un psicoterapeuta en una entrevista psiquiátrica. (Sharoon, 2019).

Dada una oración de entrada, “Eliza” identificaría palabras clave cuyo patrón coincidiría con un conjunto de reglas preprogramadas para generar respuestas apropiadas. (Weizenbaum, 1966) Desde “Eliza” se han desarrollado chatbots cada vez más inteligentes.

Figura 1: Chatbot “Elizza”.

Tomado de (Caravana, 2017)

Después, en 1972, Kenneth Colby de la Universidad de Stanford creó un chatbot denominado “Parry”. Este chatbot, podría decirse, fue una versión modificada de “Eliza”, además de tener actitudes emocionales. Entre ambos chatbots existió una conexión a través de ARPANET, lo que les permitía comunicarse entre sí. Durante una prueba realizada, solo se pudieron distinguir el 48% de las respuestas entre un humano real y “Parry”. (Sharoon, 2019)

Posteriormente en 1988 se da el lanzamiento de Jabberwacky, Figura 2. Este fue el primer chatbot que imitaba la voz humana. Su objetivo era pasar la prueba de Turing, el cual es un test para medir la capacidad de una máquina de tener un comportamiento similar al de un ser humano. (Sharoon, 2019)

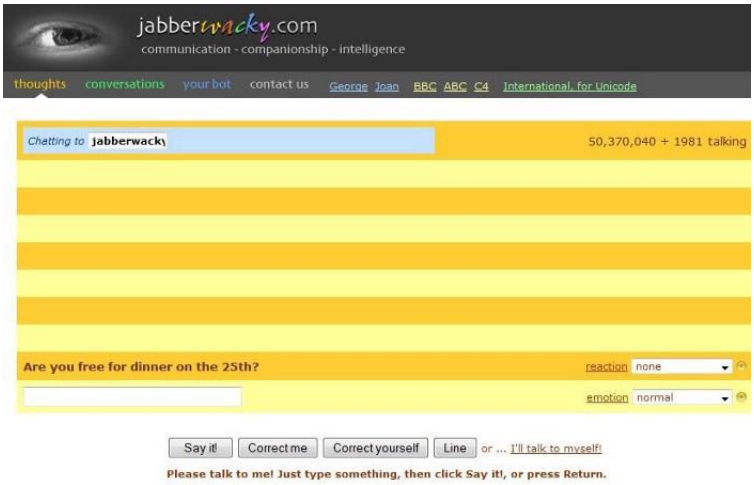


Figura 2: Chatbot “Jabberwacky”.

Tomado de (Sharoon, 2019)

En 1992 se desarrolló el chatbot “Dr. Sbaitso”, mostrado en la Figura 3. El cual poseía interfaz de usuario e intentó imitar respuestas de un psicólogo.

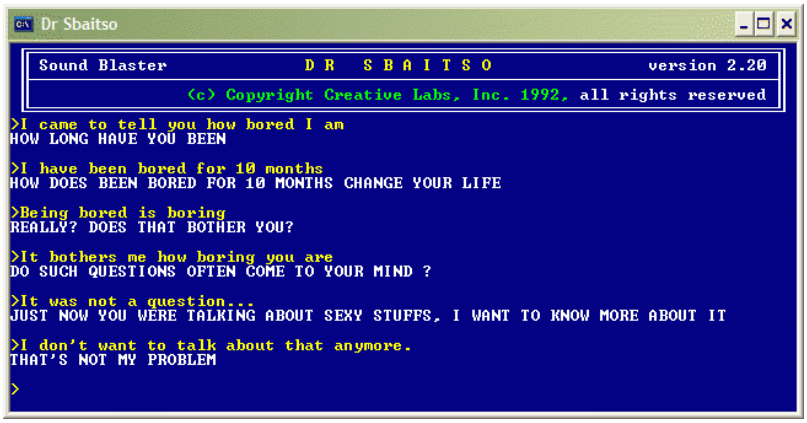


Figura 3: Chatbot “Dr. Sbaitso”.

Tomado de (Sharoon, 2019)

En 1995 se da el desarrollo de “Alice”, Figura 4, por parte de Richard Wallace. Este chatbot fue el más famoso de su época debido a su capacidad para comunicarse de manera eficiente con los humanos, gracias a sus funciones experimentales. Fue una inspiración para la creación de “Siri” de Apple y encaminó las futuras innovaciones en la ciencia de los chatbots.

Este chatbot funcionaba generando respuestas mediante la coincidencia de patrones pares llamados “input” y “output”, los cuales estaban almacenados en documentos en una base de datos. Estos documentos se escribieron en lenguaje de marcado de inteligencia artificial (AIML) con una extensión XML, la cual todavía se usa en la actualidad. “ALICE” es tres veces ganadora del premio Loebner, la competencia que intenta ejecutar la prueba de Turing y otorga el título de chatbot más inteligente. (Wallace, s.f.)

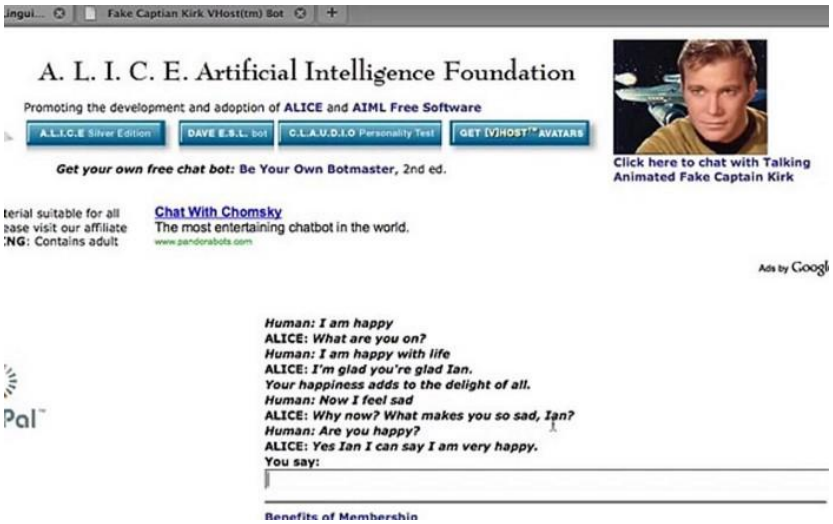


Figura 4: Chatbot “ALICE”.

Tomado de (Cason, 2015)

El primer agente de diálogo que fue implementado por Windows se llamó “Clippy”, el cual se lanzó en 1997, el cual se muestra en la Figura 5. Este asistente ayudaba a utilizar las herramientas de Microsoft Office a los usuarios, sin

embargo, este bot y sus otras personalidades (mago, gato, etc.) desaparecieron en las versiones posteriores de Office 2003. (Cerdas, 2017)

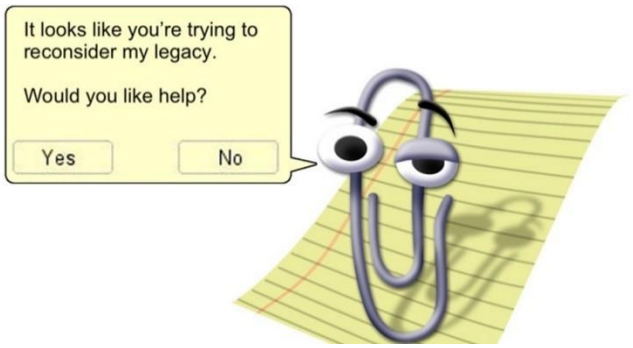


Figura 5: Agente de diálogo “Clippy”.

Tomado de (Palacios, 2018)

Con la llegada de los smartphones modernos, también llegan los chatbots modernos como “SIRI”, Figura 6. Este chatbot fue creado en 2011 por la empresa Apple, siendo el primer chatbot virtual diseñado para smartphones. Esto se logró con la ayuda de la empresa Nuance, la cual fue la encargada del desarrollo del reconocimiento por voz. Esta empresa y los desarrolladores de Apple crearon un chatbot cuya conversación era amigable con los usuarios. Estas conversaciones pueden ser de tipo chat y ser capaces de responder consultas relacionadas con música, cálculos, historia, clima, etc. Gracias al Machine Learning, “Siri” aumenta los conocimientos de su base de datos a medida que aumentan sus usuarios. También posee datos aportados de fuentes externas como lo son significados, localidades, lenguaje y pronunciaciones; para mejorar la experiencia del usuario. (Aron, 2011) .

Figura 6: “Siri de Apple”.

Tomado de (Bernal, 2020)

Luego, en el 2014, Microsoft lanza a “Cortana”, Figura 7. Un asistente que se expandió a lo largo de varios dispositivos como lo son computadoras, consolas de videojuegos o teléfonos que utilizan Windows 10. Su IA se basa en “Halo”, un videojuego de Microsoft Studios. Este asistente es capaz de adaptarse y aprender gracias a su tecnología Machine Learning. También puede aprovechar el motor de Microsoft Bing Satori, que procesa millones de datos a la vez. (Jo-Foley, 2013)

Figura 7: “Cortana”.

Tomado de (Sabán, 2019)

El mismo año, Amazon también lanza su primer asistente virtual llamado “Alexa”, cuya característica es que utiliza un parlante inteligente llamado Amazon Echo

para la interacción con los usuarios, como se muestra en la Figura 8. Este asistente permite reconocer información sobre productos, compras, recordatorios, clima, videollamadas, etc. (Ebling, 2016)

Figura 8: “Alexa”.

Tomado de (Núñez, 2018)

Por su parte Google lanzó su asistente en el 2016 llamado “Google Assistant” y está disponible para teléfonos celulares y para una herramienta de nombre Google Home. Este asistente puede asociarse con otros chatbots en sus conversaciones y ser proactivo, por ejemplo, cuando utiliza al ayudante de direcciones o cuando pide información de negocios. (Cerdas, 2017)

En el Ecuador, el Banco del Pacífico promociona a “Sophi”, el cual es un chatbot utilizado en la atención de clientes con el cual se pueden realizar consultas puntuales y recibir atención al momento que se solicite. Senescyt por otro lado, ha implementado a “Otto”, un chatbot amigable con múltiples opciones, además de reconocimiento de palabras clave. Estos chatbots están implementados en Facebook únicamente.

Algunos de estos bots modernos aprovechan los avances en el aprendizaje automático para proporcionar procesos avanzados de recuperación de

información, en los que las respuestas se generan analizando resultados de búsquedas en diferentes bases de datos.

Otros, han adoptado modelos generativos para responder las preguntas, utilizan técnicas de traducción automática estadística (SMT) para traducir frases de entrada en respuestas de salida a los usuarios. También pueden utilizar cualquier otra técnica que ayude a codificar y decodificar entradas en respuestas de usuarios.

En la actualidad, no existen implementaciones de chatbots en plataformas LMS a nivel nacional, a pesar de ser una buena alternativa para desarrollo de la calidad educativa dentro del País.

1.4.2. USO DE CHATBOTS EN LA DOCENCIA UNIVERSITARIA

Cuando se habla de chatbots en el ámbito académico, el más sobresaliente es el caso del chatbot “Jill Watson”. El cual, en el trascurso de varios meses del 2016, fingió ser una profesora ayudante del Instituto Tecnológico de Georgia. Se encargó de responder mensajes de varios alumnos del grado de informática dentro de un foro online, sin que los participantes notaran de que se trataba de un programa de computadora (Michael, 2016, pp. 112–117).

En los Estados Unidos se está desarrollando un proyecto que involucra a la Fundación Bill y Melissa Gates, Facebook y la escuela pública Summit; en el cual se utilizan chatbots para dar conferencias básicas. El objetivo es mostrar cómo los chatbots pueden servir de asesores virtuales mientras se adaptan a las habilidades y necesidades de cada estudiante, sobre todo a su ritmo de aprendizaje. Se pueden obtener muchas ventajas con el uso de los chatbots, ya

que los maestros no tendrían que centrarse en la memorización de lecciones y exámenes de calificación, sino que pueden expandir la enseñanza en otras formas más dinámicas. (Pereira, 2016).

Los chatbots pueden complementar el aprendizaje estudiantil a través del despliegue de información, también podrían almacenar datos para su uso posterior. En casos particulares, pueden ser entrenados para realizar entrevistas o encuestas e inclusive, corregir tareas u otro tipo de evaluaciones complicadas objetivamente hablando. La inclusión de esta tecnología es un enorme paso para la evolución de la educación y es importante que siga de la mano con el desarrollo de las TICs. (ChatCompose, 2019)

2. CAPÍTULO II

Este capítulo se centra en el desarrollo del marco teórico, la metodología a utilizar, un análisis general y un análisis de riesgos.

2.1. MARCO TEÓRICO

Se abordan los conceptos referentes a los chatbots y las herramientas necesarias para su desarrollo.

2.1.1. CONCEPTOS GENERALES

Se detallan los principales conceptos que sobresalen cuando se habla de la creación y uso de los chatbots.

2.1.1.1. CHATBOTS

“Los chatbots son sistemas de diálogo entre humano y computadora utilizando lenguaje natural”. (Jia, 2003) La conceptualización del chatbot se atribuye a Alan Turing, quien preguntó "¿Pueden pensar las máquinas?", en 1950. En un artículo, él presentó un juego de imitación de tres personas (A, B y un interrogador). Para incorporar IA, el chatbot debe de tomar el papel de A y tratar de engañar al interrogador de ser la persona B. (Turing, 1950).

El test de Turing, gracias a su facilidad, se puede ejecutar en diversos esquemas, con preguntas y actores diferentes. Con este test se pretendió que los chatbots sean capaces de responder, junto con el conocimiento adquirido, dentro de tres principales ejes: aprendizaje, razonamiento y entendimiento. (Reddy, 1976). El premio Loebner Prize (concurso para programas considerados los más

inteligentes), aplica el test de Turing como herramienta principal para medir la inteligencia y capacidad de Deep Learning (aprendizaje autónomo) de los Bots. Por lo tanto, aprobar este test es el logro más significativo para los chatbots. (Cerdas, 2017).

Desde Turing, la tecnología de chatbot ha mejorado con los avances en lenguaje natural, procesamiento y aprendizaje automático. Del mismo modo, el uso del chatbot también ha aumentado, especialmente con el lanzamiento de plataformas que lo utilizan, como por ejemplo Facebook, Kik, Slack, Skype, WeChat, WhatsApp, etc.

Un marco básico que describe las funciones que se esperan de los chatbots modernos es el de Sansonnet, Leray & Martin (Sansonnet, Leray & Martin; 2006), el cual indica que los chatbots deben cumplir con lo siguiente:

Agente dialógico: Los chatbots deben comprender a los usuarios, es decir, proporcionar la función de comprensión de lenguaje natural. Se proporcionan con una entrada textual u oral, que se analizan con herramientas de procesamiento de lenguaje, y se utilizan para generar respuestas apropiadas.

Agente racional: Debe tener acceso a una base externa de conocimiento y sentido común (por ejemplo, a través de bases de datos) de modo que pueda responder preguntas del usuario. Debe almacenar información específica del contexto, por ejemplo, el nombre del usuario, etc.

Agente encarnado: Debe proporcionar la función de presencia, una función crucial en el caso de los usuarios comunes. Los primeros bots recibieron nombres (ELIZA, ALICE, CHARLIE, etc.), para satisfacer esta condición.

Actualmente, los desarrolladores se centran en el uso de trucos de lenguaje para crear personalidades para los chatbots, las cuales puedan generar confianza con los usuarios y dar la impresión de una persona real, como se muestra en la representación de la Figura 9. (Sansonnet, et al; 2006).

Figura 9: Chatbot y el ser humano.

2.1.1.2. INTELIGENCIA ARTIFICIAL (IA)

La Inteligencia Artificial (IA) es el dominio de la ciencia e ingeniería relacionada al desarrollo de sistemas con características asociadas a la inteligencia detrás del comportamiento humano, el procesamiento del lenguaje natural, la resolución y planificación de problemas, el aprendizaje, la adaptación y la interacción sobre el medio. Su principal objetivo es comprender los principios que permiten el comportamiento inteligente en humanos, animales y agentes artificiales. También respalda directamente varios objetivos de ingeniería, como desarrollar agentes inteligentes, formalizar el conocimiento y mecanizar el razonamiento en todas las áreas del esfuerzo humano. Se encarga de hacer que el trabajo con computadoras sea tan fácil como trabajar con personas, desarrollando sistemas humano-máquina que exploten la complejidad del razonamiento humano.

La inteligencia artificial es un campo amplio y de muchos dominios, no solo todas las disciplinas informáticas, sino también matemáticas, lingüística, psicología, neurociencia, ingeniería mecánica, estadística, economía, teoría de control y

cibernética, filosofía entre otras. La mayoría de estos sistemas se desarrollan como componentes de aplicaciones complejas a las que agregan inteligencia de varias maneras, por ejemplo, permitiéndoles razonar con conocimiento, procesar el lenguaje natural o aprender y adaptarse. (Tecuco, 2012).

Finalmente, la inteligencia artificial ha venido evolucionando en los últimos años, lo que ha permitido utilizar y convertir expresiones en lenguaje natural a datos útiles para los programas. Estos sistemas procesan la información y son capaces de resolver peticiones, facilitando las conversaciones naturales entre aplicaciones y usuarios. Al integrar Machine Learning a aplicaciones que utilizan IA, han mejorado potencialmente su uso, lo que permite que estas apps tengan mayor velocidad de procesamiento de información y puedan dar respuestas mucho más precisas. (Reina, 2018).

2.1.1.3. MACHINE LEARNING

El Machine Learning es una rama importante dentro de la inteligencia artificial (IA) que permite a los sistemas informáticos aprender directamente de ejemplos, datos y experiencias. Gracias a la ayuda que brindan a las computadoras para que realicen tareas específicas de manera inteligente, los sistemas de aprendizaje automático (machine learning) pueden llevar a cabo procesos complejos aprendiendo de los datos en lugar de seguir reglas preprogramadas.

Antes de Machine Learning, la única forma de que un sistema realice una acción era desarrollando un algoritmo que defina los detalles y el contexto de esa acción. Ahora, gran parte de estos procesos los realizan por su cuenta, recopilan información y generan sus propios cálculos.

Es de gran importancia que las bases de datos de las aplicaciones que utilizan Machine Learning tengan gran cantidad de información, para acciones más precisas. Los programas que utilizan esta tecnología pueden modificarse a sí mismos utilizando algoritmos y pueden diseñar respuestas informáticas dependiendo la información suministrada por los usuarios. Cada dato de entrada pasa a ser parte del algoritmo y cuando existe gran cantidad de datos, se logra obtener mayor efectividad en los cálculos o respuestas del sistema.

Machine Learning debe contar con una gran cantidad de datos relevantes para dar respuestas válidas a gran velocidad. Existe un mínimo de estradas de datos recomendada para formular cada nueva respuesta en los sistemas con Machine Learning, el cual es de 6 entradas. Esto debe repetirse para cada respuesta del sistema. (Redacción APD, 2019).

Machine Learning puede ser de tres tipos:

Aprendizaje Supervisado:

Este tipo de aprendizaje consiste en entrenar al sistema, proporcionándole una cantidad significativa de datos, definiéndolos detalladamente y etiquetándolos. Una vez que la cantidad de información es suficiente, el sistema podrá introducir nuevos datos sin necesidad de etiquetas en base a los distintos patrones que se han registrado durante el entrenamiento. Estos sistemas son conocidos como de clasificación.

Otra forma con la que se lleva a cabo el aprendizaje automático es la predicción de valores continuos. Los sistemas usan parámetros combinados que, cuando se introduce nueva información, ayudan a predecir un resultado determinado y lo expresan en forma de respuesta. Este método es conocido como regresión.

Dentro de este método existen tres vías relevantes, las cuales son aprendizaje por corrección de errores, por refuerzo y estocástico. (Redacción APD, 2019).

Aprendizaje no Supervisado:

Este tipo de aprendizaje tiene como finalidad la comprensión y abstracción en la información de una manera más directa. Este modelo se conoce como clustering, el cual es un método de entrenamiento parecido a la manera en que los humanos procesan la información. (Redacción APD, 2019).

Aprendizaje por Refuerzo:

Son algoritmos que ayudan a los sistemas a aprender el comportamiento de una manera precisa, como lo hacen los seres humanos. Este tipo de aprendizaje tiene como objetivo aprender del entorno interactuando con él y recibiendo recompensas o premios acumulados por efectuar acciones positivas. (Redacción APD, 2019).

El uso de Machine Learning es relevante para resolver tareas complejas o problemas que involucren grandes cantidades de datos normales o compuestos. Tanto el uso del Machine Learning y la IA ha permitido que los chatbots se hayan transformado gradualmente en asistentes virtuales.

2.1.1.4. DEEP LEARNING

El deep learning es una técnica de aprendizaje automático que se define como un algoritmo jerárquico con el fin de emular el aprendizaje humano de ciertos conocimientos. No requiere reglas, el sistema tiene la capacidad de aprender y efectuar tareas por sí mismo a través de su fase previa de entrenamiento. Su principal característica es que está compuesto de redes neuronales artificiales

entrelazadas y se emplea en la automatización de análisis predictivos. Gracias al Deep Learning, los chatbots pueden aprender representaciones de datos, mejorar el reconocimiento al habla, reconocimiento de objetos y NPL. (Smart Panel, 2020).

2.1.1.5. NATURAL LANGUAGE PROCESSING (NLP)

También llamado como computación lingüística o procesamiento del lenguaje natural, permite a los programas hacer la interpretación del lenguaje humano. Es un modelo mejorado para la traducción del lenguaje entre la computadora y el ser humano, con datos empíricos. El procesamiento del lenguaje natural se encarga del desarrollo de métodos computacionales que ayudan con la producción, aprendizaje y entendimiento de contenidos por parte de los sistemas. Para el entendimiento del lenguaje, NPL desarrolló tres técnicas:

Machine Traslation: Es la forma en la que se traduce el lenguaje y se codifica en una estructura entendible para una computadora. Según Hirschberg, “esta tecnología ha evolucionado gracias al Deep Learning”, el cual optimiza las traducciones preparando modelos con diferentes representaciones. También acepta el uso de Skype Traslator y Google Traslate. (Saygin, Cicekli, Akman; 2000).

Speech Recognition: Se encarga de transformar una señal en palabras por medio de algoritmos de computadora. Esta tecnología ha posibilitado que los programas puedan manifestarse por comandos de voz y a las personas comprender el lenguaje de las computadoras. Speech recognition tiene tres formas: palabras conectadas, dialogo espontáneo y palabras aisladas.

Speech Synthesis: Gracias a este software los programas pueden transformar texto a dialogo y viceversa. Es capaz de comprender la pronunciación, duración y entonación de una conversación antes de la transformación. (Glasgow y Browse; 2002).

2.1.2. HERRAMIENTAS UTILIZADAS

Se describen las herramientas de software utilizadas en el desarrollo de este trabajo de titulación.

2.1.2.1. VISUAL STUDIO ENTERPRICE 2019

El software que se va a utilizar para el desarrollo de este chatbots es el Visual Studio Enterprice 2019, debido a su intuitiva interfaz de desarrollo de aplicaciones y a la herramienta Microsoft Bot Framework, que tiene compatibilidad con este software. Visual Studio es una plataforma creativa para el desarrollo de aplicaciones que se puede usar para editar, depurar y crear código, para posteriormente publicarlo. (Microsoft, 2019) Es un IDE (entorno de desarrollo integrado), que posee varias funciones que ayudan en muchos aspectos de la creación de un software. Por ejemplo, incluye editores, depuradores estándar, compiladores, herramientas de finalización de código, diseñadores gráficos y muchas más funciones que facilitan el proceso de creación de aplicaciones (Strauss, D. 2020).

Para los usuarios que han utilizado Visual Studio durante varios años, el IDE que ofrece a los programadores, es familiar al de versiones anteriores. Con una gran cantidad de herramientas y características, las cuales sirven para el desarrollo

de aplicaciones. Visual Studio es una potencia absoluta cuando se trata de herramientas para implementar softwares de clase mundial.

Lo nuevo de Visual Studio 2019 con respecto a versiones anteriores

- Interfaz gráfica renovada: Presenta un menú renovado para elegir la solución con la que vamos a trabajar
- Abrir soluciones sin cargar los proyectos
- Navegación entre advertencias y errores
- Portapapeles integrado: Copiar y pegar contenido
- Búsqueda en las variables locales: permite buscar el nombre y contenido de las variables.
- Visual Studio Live Share integrado: Hace posible trabajar en equipo en un mismo código (Turrado, J. 2019)

Versión Enterprise

Es una solución integrada y completa para equipos de cualquier tamaño que tiene importantes requisitos de calidad.

Características:

Entre sus principales características se encuentran:

- Interfaz gráfica renovada.
- Permite navegar por el código.
- Depurar, generar perfiles y emitir diagnósticos fácilmente.
- Permite escribir un código de alta calidad con herramientas de pruebas integrales.
- Control de versiones.
- Extensiones para personalizar el IDE.

- Desarrollar soluciones en macOS y Windows.
- Productividad de desarrollo máxima.
- Depuración y diagnósticos avanzados. (Microsoft, 2020).

2.1.2.2. MICROSOFT AZURE

Es una plataforma de computación que se encuentra en la nube pública de Microsoft, también llamado Windows Azure, que provee una serie de servicios que incluyen computación, redes, almacenamiento y análisis. Los usuarios pueden escoger estos servicios para el desarrollo e implementación de nuevas aplicaciones o ejecutar aplicaciones existentes en la nube pública; en lugar de construir una instalación de servidor local o alquilar servidores físicos de centros de datos tradicionales.

Ofrece herramientas de compatibilidad con tecnologías de código abierto, lo que les permite a los usuarios utilizar sus tecnologías preferidas. Además, brinda cuatro formas diferentes de computación: sin servidor, plataforma como servicio (PaaS), software como servicio (SaaS) e infraestructura como servicio (IaaS).

Microsoft Azure se usa comúnmente para alojar bases de datos a la nube; también para alojar recursos informáticos y componentes de infraestructura, por ejemplo: DNS, Servicios de Windows Server, Internet Information Services (IIS), aplicaciones de terceros o sistemas operativos de terceros.

Una vez que se realiza la suscripción a Azure, se obtendrá acceso a todos los servicios que incluye el portal. Los clientes pueden usar estos servicios para crear recursos basados en la nube.

Microsoft cobra por Azure, sobre la base de pago por uso, lo que quiere decir que a los suscriptores se les cobra por los recursos específicos que han utilizado. Los precios varían entre los diferentes tipos de servicios, tipos de almacenamiento y la ubicación física desde la que se alojan sus instancias de Azure. (Rouse, 2020).

2.1.2.3. MICROSOFT BOT FRAMEWORK

Un Bot es un programa que funciona de manera automática, permitiendo realizar una o múltiples tareas. Para su desarrollo, la empresa Microsoft implementó una herramienta en 2016 denominada Bot Framework. Esta herramienta posee un conjunto de métodos y clases predefinidas y preinstaladas creadas por desarrolladores para la creación de bots. El objetivo de esta herramienta es que el programador pueda utilizar estos métodos, los cuales ayudan a escribir el código de los bots de manera rápida y mejor. En términos simples, los desarrolladores podrán utilizar marcos existentes en esta herramienta para crear bots de chat desde cero, utilizando lenguaje de programación. Las plataformas más famosas y modernas que permiten a los desarrolladores crear sus propios bots desde cero son: Dialogflow desarrollado por Google, Facebook Bot Engine y Microsoft Bot Framework.

Ventajas Microsoft Bot Framework

Microsoft ha creado todo un framework para poder desarrollar bots de una manera mucho más fácil. Bot Framework es compatible con las multiplataformas que Microsoft utiliza, junto con sus herramientas, las cuales podemos utilizar tanto para desarrollo de JavaScript, como para lenguaje C#. Los bots pueden estar diseñados en cualquier lenguaje de programación, además de poder trabajar en servidores, clientes, ser un agente móvil, etc.

Este framework se adecúa muy bien a las tecnologías Web. Gracias a ello los bots serán una aplicación exteriorizada, por medio de una API bastante flexible que escucha cualquier respuesta de los usuarios por un “endpoint” común. Además, lo positivo que tiene Bot framework y lo diferencia de cualquier librería o framework dedicado a peticiones de respuesta, es que contiene un gestor de patrones y expresiones regulares muy potente, el cual ayudará a entender mejor lo que el usuario pretende y la acción que se deberá desencaminar por parte del bot.

Servicios cognitivos y Bot framework de Microsoft

Son un conjunto de APIs, SDKs y demás servicios destinados para agregar la capa de inteligencia a las aplicaciones de software. Proponen un esquema de acceso vía servicios web y una key para realizar mediciones del uso y acceso a estos servicios, ya que estos se encuentran en la nube de Microsoft AZURE de manera paga. Aunque también se pueden ofrecer gratuitamente con límites máximos. Estos servicios a su vez pueden unirse en grupos con fines específicos como entendimiento del lenguaje, extracción de la base de conocimiento, reconocimiento de voz y conversión, búsqueda web, entendimiento de imagen y video, etc. (Delgado, 2017).

Desarrollo de Bot Framework

“Microsoft bot framework“, facilita los recursos que un desarrollador necesita para construir un chatbot conversacional, inteligente y que interactúe naturalmente. Implementar Microsoft Bot Framework es una buena opción, debido a que posee muchas herramientas con tecnología para la creación de bots y programación en C#. (Microsoft Bot Framework. 2017).

Los componentes principales de Bot Framework son:

Bot Connector: Es un servicio que permite a un bot intercambiar mensajes con canales que están disponibles en Microsoft Bot Framework. Utilizando REST API y JSON, sobre el protocolo seguro HTTPS. Cuando un usuario envía un mensaje, Bot conector envía una solicitud POST al punto final que se especifica durante el registro del bot en canal respectivo. (Microsoft Bot Framework. 2017).

Bot Builder: Es un SDK para desarrolladores de .NET Framework, es útil para la creación de bots usando Visual Studio y Windows. Este SDK admite programación C# y Node.js. También posee un grupo de plantillas de aplicación de bot, controladores de bot y diálogo de bot. Bot Builder, también posee la plantilla de un proyecto, con todos los componentes para bot simple. Incluye un método POST para aceptar mensajes y un generador de diálogo para enviar una respuesta. (Microsoft Bot Framework. 2017).

2.1.2.4. LANGUAGE UNDERSTANDING INTELLIGENT SERVICE (LUIS)

Es una API que se encuentra en la nube, la cual brinda servicios basados en el aprendizaje automático. Ayuda a integrar el procesamiento del lenguaje natural en aplicaciones o dispositivos IoT. A su vez, aplica IA para predecir el significado de una conversación o un texto y extraer información de forma adecuada, la cual se utiliza para realizar acciones dentro de los programas computarizados.

Las aplicaciones clientes de “LUIS”, son aplicaciones conversacionales que se comunican con los usuarios en lenguaje natural. Cuando la aplicación que utiliza “LUIS” es publicada envía expresiones (Texto) a “LUIS” el cual recibe los resultados como un objeto en formato JSON.

“LUIS” se maneja con un sistema de dominios, el cual pude ser previamente creado o se puede utilizar alguno de los modelos propios de la herramienta. Estos modelos suministran los datos necesarios para la identificación de las “intenciones” de los usuarios y las expresiones que utilizan para su formulación, a fin de identificar claramente la acción que debe realizar el bot.

Modelo Creado Previamente:

Son modelos que “LUIS” tiene incluidos en la API; contienen expresiones, entidades e intenciones las cuales fueron previamente creadas con el objetivo de facilitar la interacción con los usuarios de manera rápida.

Modelo Personalizado:

Es posible crear modelos personalizados con entidades, intenciones y expresiones que pueden acoplarse a las necesidades de cada aplicación, a fin de optimizar el funcionamiento de estas y la interacción con los usuarios.

Para la creación de modelos personalizados se comienza con la categorización de las intenciones. Para cada intención se agregan ejemplos de expresiones, los cuales pueden proporcionar datos necesarios que se deben extraer. Estas expresiones pueden aumentar, dependiendo la variedad de expresiones con las que se solicita una “intención”. Esta etapa de asignación de expresiones se la conoce como fase de entrenamiento.

Una vez el bot ha sido entrenado, se procede al envío de mensajes por parte de la aplicación cliente. Las expresiones de estos mensajes se procesan y analizan dando como resultado la predicción de la “intención” del usuario. Los resultados son enviados en formato tipo JSON, el cual contiene la expresión de consulta y la “intención” con mayor porcentaje de coincidencia.

“LUIS” también proporciona un aprendizaje constante y activo de las expresiones, con el fin de mejorar las predicciones. A su vez, posee herramientas para el control de versiones. (Microsoft, 2020)

2.2. METODOLOGÍA

El presente proyecto utiliza una metodología incremental, la cual se basa en aumentar de una manera iterativa una implementación y desarrollarla hasta cumplir los requerimientos y objetivos planteados.

Los incrementos son funcionalidades del proyecto o mejoramiento de versiones de funciones existentes. Con esto se logra un análisis progresivo del producto final, sabiendo exactamente que funcionalidad está implementada correctamente.

En la Figura 10, se muestra un esquema del modelo incremental y las fases en las que se basa cada incremento. Las cuales son: análisis, diseño, código, prueba, integración y operación.

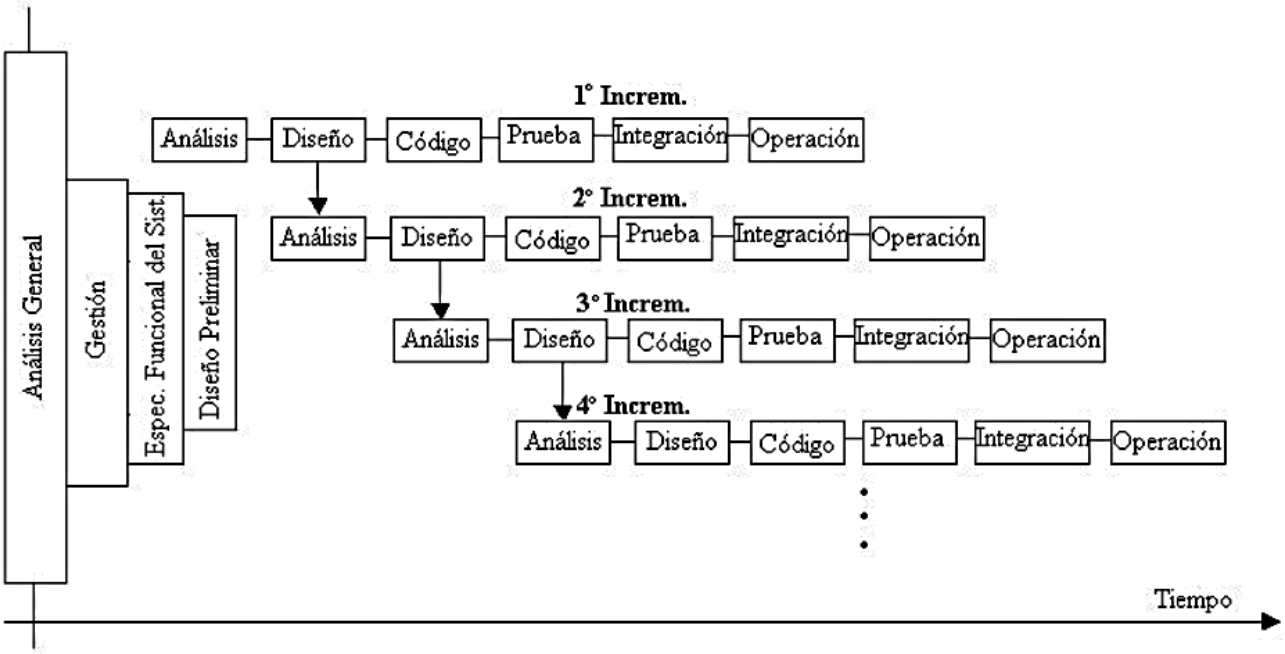


Figura 10: Metodología Incremental.

Tomado de (Wikipedia, 2008)

2.3. ANÁLISIS GENERAL

Este análisis tiene como objetivo la especificación de los requerimientos funcionales necesarios para el correcto funcionamiento del chatbot, para la elaboración del diseño iterativo del proyecto. Estos requisitos se elaboraron en base a los posibles usuarios a los que va enfocado el chatbot. Para poder desarrollar un proyecto, es necesario saber el tipo de usuarios al que va destinado; ¿qué nivel tienen?, ¿qué lenguaje se utiliza?, ¿qué esperan del producto?, etc. En este caso los usuarios serán estudiantes y docentes de la Universidad de las Américas (UDLA), Quito – Ecuador.

Este proyecto, sobre todo, está centrado en usuarios que manejan el aula virtual de las carreras presenciales de la universidad. Por lo que, el idioma imprescindible para el manejo de este chatbot es el español.

Este chatbot busca ser intuitivo y de fácil manejo, no es necesario ningún conocimiento previo para el uso del chatbot, solo tener una cuenta vinculada a la universidad y estar loggeado dentro del aula virtual.

Los requisitos funcionales para este chatbot son los siguientes:

- Recogida de nombre de usuario
- Identificación del tipo de usuario
- Acceso a información sobre el horario semanal
- Acceso a información de la malla curricular de la carrera
- Acceso a información sobre las materias inscritas
- Acceso a información sobre los participantes en cada asignatura
- Registro de asistencia por materia (Solo Profesores)
- Acceso a información sobre el control de faltas del usuario (Solo Alumnos)
- Acceso a la biblioteca virtual de la UDLA
- Registro de calificaciones por materia (Solo Profesores)
- Acceso a información sobre las calificaciones por materia (Solo Alumnos)
- Realizar búsqueda de información en la web
- Acceso a información del calendario de actividades de los usuarios
- Acceso a los foros informativos de cada materia
- Acceso a la información de perfil
- Terminar la conversación
- Tratamiento de errores

2.4. ANÁLISIS DE RIESGOS

El principal riesgo que puede tener este proyecto es la inoperabilidad. Debido a cambios o modificaciones que puedan existir tanto en la URL de despliegue o en datos de terceros, los cuales son utilizados por el programa. En este caso el proyecto depende del aula virtual de la universidad, donde se realizan peticiones para cumplir funciones o requerimientos de los usuarios. Por lo que, ciertos servicios quedarían inservibles hasta que se proceda con su arreglo o se realicen las modificaciones necesarias en el programa.

3. CAPÍTULO III

Este capítulo contiene información acerca de la etapa iterativa, los requisitos funcionales, la arquitectura utilizada, el diseño y el desarrollo de este trabajo de titulación.

3.1. ETAPA ITERATIVA

Para esta etapa es necesario ordenar todas las funcionalidades que se van a implementar, se toma en cuenta la importancia que tienen en el proyecto y se empieza a implementar por prioridad.

La característica de esta metodología es que las iteraciones dan como producto un objeto operacional, en el cual se observa la interacción de los usuarios con el chatbot. Para esto es necesario desarrollar la base del proyecto sobre el cual se va a trabajar.

A continuación, en la Tabla 1, podemos observar cada iteración, una descripción de la funcionalidad y un tiempo estimado para su implantación.

Tabla 1. Etapas del Desarrollo del Chatbot

N°	Descripción	Tiempo (horas)
1	Base de la aplicación	20
2	Conexión con servicio “LUIS”	20
3	Función Horario Semanal	10
4	Función Malla Curricular	10

5	Función Materias	10
6	Función Lista de Alumnos	10
7	Función de Registrar Asistencia (Profesor)	10
8	Función Control de Faltas (Alumnos)	10
9	Función Ir a Biblioteca UDLA	10
10	Función Registrar Calificaciones (Profesores)	10
11	Función Ver Calificaciones (Alumnos)	10
12	Función Realizar Búsqueda de Información	20
13	Función Calendario de Actividades	10
14	Función Ira a Foro de Materia	10
15	Función Ir a Editar Información de Perfil	10
16	Función Terminar la Conversación	10
17	Tratamiento de errores	10
18	Creación de Intenciones y Entidades en “LUIS”	20
19	Publicación en Microsoft Azure	5
20	Implementación en Aula Virtual UDLA	2

1° Iteración

Al inicio del desarrollo, se deben elegir las herramientas necesarias para la construcción del chatbot. En este caso, el software Visual Studio 2019 y el servicio Bot Framework, que a su vez está integrado al servicio de Microsoft Azure y al servicio cognitivo “LUIS” (Language Understanding Intelligent Service). También diseñar un diagrama de casos de uso para ver el posible desenvolvimiento del chatbot y junto con estos datos, desarrollar la base del proyecto para que cualquier usuario del Aula Virtual de la UDLA pueda utilizarlo.

2° Iteración

En esta iteración se debe crear un nuevo proyecto en el servicio de “LUIS” y crear la conexión al mismo agregando las credenciales en nuestro proyecto base de Visual Studio 2019.

3° Iteración

En esta iteración se implementó la función de ver el horario semanal, con la cual cada usuario puede ver y acceder a su horario de clases particular.

4° Iteración

En esta iteración se implementó la función de ver revisar la malla curricular, con la cual cada usuario puede ver y acceder a la malla curricular de su carrera particular.

5° Iteración

En esta iteración se implementó la función de ver materias, con la cual cada usuario puede ver y acceder a sus materias particulares a las que está inscrito.

6° Iteración

En esta iteración se implementó la función de ver lista de alumnos, con la cual cada usuario puede ver y acceder a la lista de participantes de las materias particulares a las que está inscrito.

7° Iteración (Profesor)

En esta iteración se implementó la función de registrar asistencia, con la cual cada usuario que es profesor puede acceder al registro de asistencia de cada clase particular a la que está inscrito.

8° Iteración (Alumno)

En esta iteración se implementó la función de control de faltas, con la cual cada usuario que es alumno puede acceder a su control de faltas particular.

9° Iteración

En esta iteración se implementó la función de ir a biblioteca UDLA, con la cual cada usuario puede acceder a la biblioteca de la UDLA.

10° Iteración (Profesor)

En esta iteración se implementó la función de registrar calificaciones, con la cual cada usuario que es profesor puede acceder al registro de calificaciones de cada materia particular a la que está inscrito.

11° Iteración (Alumno)

En esta iteración se implementó la función de ver calificaciones, con la cual cada usuario que es alumno puede acceder a las calificaciones de cada una de sus materias.

12° Iteración

En esta iteración se implementó la función de realizar búsqueda de información, con la cual cada usuario puede realizar una consulta a la web de cualquier tema en la plataforma de Google.

13° Iteración

En esta iteración se implementó la función de calendario de actividades, con la cual cada usuario puede ver y acceder a su calendario de actividades particular.

14° Iteración

En esta iteración se implementó la función de ir a foro de materia, con la cual cada usuario puede ver y acceder al foro o cartelera informativa de cada una de las materias particulares a las que está inscrito.

15° Iteración

En esta iteración se implementó la función de ir a editar información de perfil, con la cual cada usuario puede ver y acceder su perfil particular.

16° Iteración

En esta iteración se implementó la función de terminar la conversación, con la cual cada usuario puede salir de la conversación actual que tiene con el chatbot.

17° Iteración

En esta iteración se implementó el tratamiento de errores, con lo cual se validan los posibles errores que podrían aparecer al momento de que los usuarios

ingresan información; mostrando mensajes de error que ayudan estos a continuar utilizando el chatbot.

18° Iteración

En esta iteración se crearon las “intenciones” y “entidades” necesarias para que el programa desarrollado en “LUIS” pueda enviar la información apropiada a la API en Visual Studio 2019. Para que la API pueda obtener apropiadamente la “intención” de los usuarios.

19° Iteración

En esta iteración se agregan las configuraciones necesarias para la publicación del chatbot en la plataforma Microsoft Azure.

20° Iteración

En esta iteración, una vez publicado el chatbot en Microsoft Azure, se procede a implementar el mismo en el aula virtual de la Universidad de las Américas (UDLA), modalidad presencial.

3.2. REQUISITOS

En primer lugar, se le asignó un “nombre” al proyecto, para que el chatbot posea una identidad al momento de la interacción con los usuarios. La identidad que se asignó a este proyecto para su comunicación con el mundo exterior es “**CHAPIE**”; las cuales son las siglas de **Chatbot – Asistente – Personal – Inteligente – Estudiantil.** Este nombre proporciona una identidad fácil de recordar para los usuarios y a su vez, menciona el principal objetico de este prototipo en sus siglas.

Para analizar los requisitos se procedió a realizar un diagrama de casos de uso, como se ve en la Figura 11, para poder mostrar las acciones principales que el usuario puede desempeñar al momento de tener una conversación con nuestro chatbot.

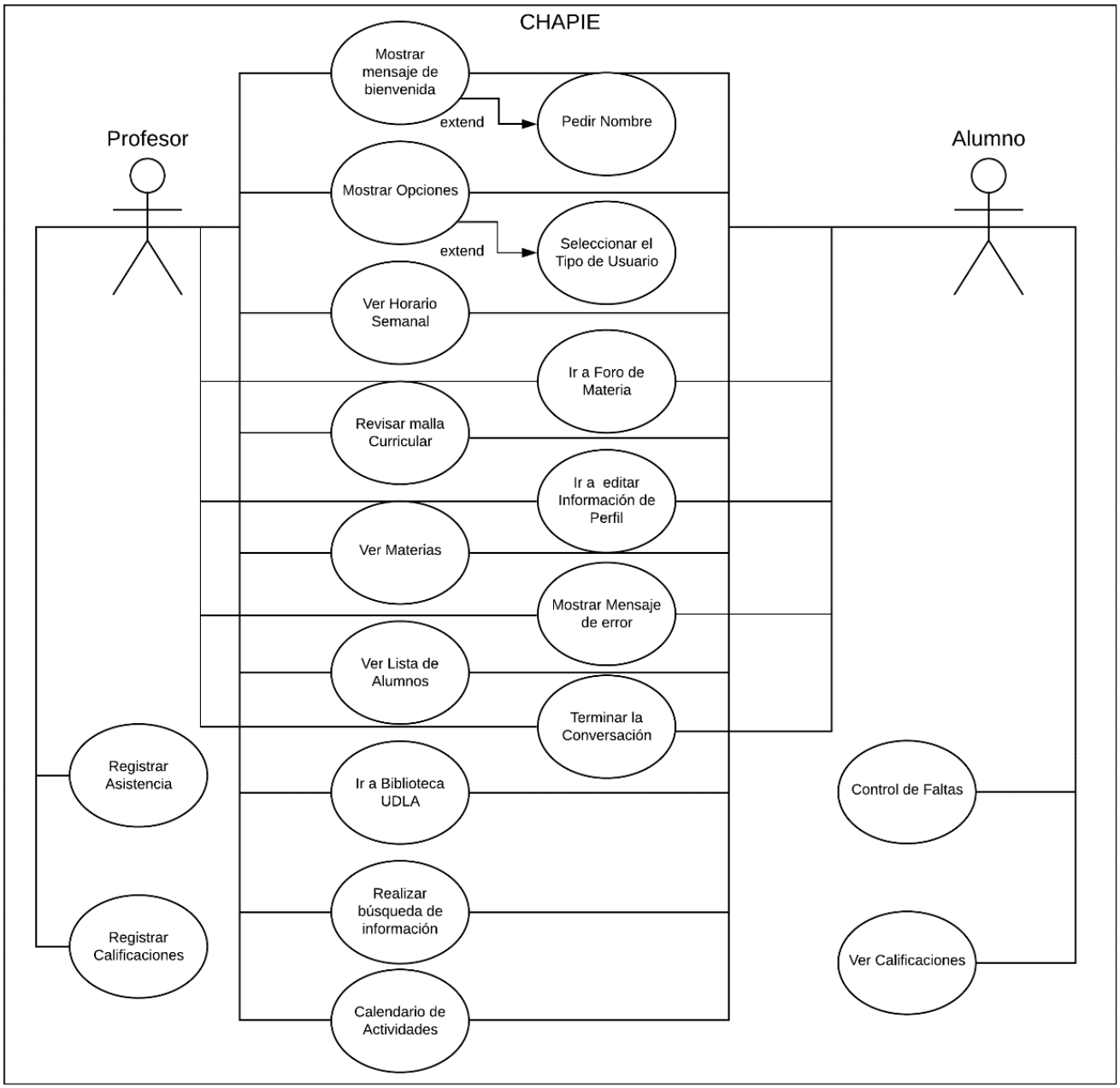


Figura 11: Diagrama de casos de uso de “CHAPIE”

Mostrar mensaje de bienvenida:

Cuando se inicia una nueva conversación, el usuario recibe un mensaje de bienvenida y la posibilidad de ingresar su nombre. Para agregar el nombre y continuar la conversación, el usuario escribe su respuesta y presiona el botón “Enter”.

Mostrar opciones:

Después de ingresar su nombre, el usuario recibe un mensaje en el cual se le solicita escoger el tipo de usuario. De esto depende el menú que se va a observar posteriormente, ya sea “Profesor” o “Alumno”. El usuario debe pulsar el botón correspondiente que se hablita al momento de la petición y una vez seleccionado el tipo de usuario, se muestra el respectivo menú en pantalla para continuar con la conversación.

Ver horario semanal:

El usuario tiene la posibilidad de ver su horario semanal introduciendo la petición adecuada. Puede escribir “ver horario semanal”, “quiero ver mi horario semanal”, “horario”, “deseo ver mi horario”, “ver mi horario”, etc.

Revisar malla curricular:

El usuario tiene la posibilidad de revisar la malla curricular de su carrera, introduciendo la petición adecuada. Puede escribir “malla curricular”, “ver malla curricular”, “quiero ver mi malla curricular”, “deseo ver mi malla curricular”, “ver mi malla curricular”, etc.

Ver materias:

El usuario tiene la posibilidad de ver las materias a las cuales está inscrito actualmente, introduciendo la petición adecuada. Puede escribir “Ver materias”, “materias”, “quiero ver mis materias”, “deseo ver materias”, “ver mis materias”, etc.

Ver lista de alumnos:

El usuario tiene la posibilidad de ver la lista de alumnos de las materias a las cuales está inscrito actualmente, introduciendo la petición adecuada. Puede escribir “Ver lista de alumnos”, “lista de alumnos”, “quiero ver la lista de alumnos”, “deseo ver la lista de alumnos”, “ver lista de alumnos”, etc.

Ir a biblioteca UDLA:

El usuario tiene la posibilidad de ir a la biblioteca de la UDLA, introduciendo la petición adecuada. Puede escribir “Ir a biblioteca UDLA”, “biblioteca UDLA”, “quiero ir a biblioteca UDLA”, “deseo ir a biblioteca UDLA”, “ver biblioteca UDLA”, etc.

Realizar búsqueda de información:

El usuario tiene la posibilidad de realizar una búsqueda de información en la web, introduciendo la petición adecuada. Puede escribir “Deseo buscar información de (Lo que se desea Buscar)”, “Buscar información de (Lo que se desea Buscar)”, “Buscar (Lo que se desea Buscar)”, “quiero buscar información de (Lo que se desea Buscar)”, etc.

Calendario de actividades:

El usuario tiene la posibilidad de ver su calendario de actividades, introduciendo la petición adecuada. Puede escribir “Ir a calendario de actividades”, “calendario de actividades”, “quiero ir a calendario de actividades”, “deseo ir a calendario de actividades”, “ver calendario de actividades”, etc.

Ir a foro de materia:

El usuario tiene la posibilidad de ir al foro o cartelera informativa de cada materia a la cual está inscrito, introduciendo la petición adecuada. Puede escribir “Ir a foro de materia”, “foro de materias”, “quiero ir a foro de materias”, “deseo ir a foro de materias”, “ver cartelera informativa”, etc.

Ir a editar información de perfil:

El usuario tiene la posibilidad de ir a su perfil personal, introduciendo la petición adecuada. Puede escribir “Ir a editar información de perfil”, “información de perfil”, “quiero ir a información de perfil”, “deseo ir a información de perfil”, “ver información de perfil”, etc.

Registrar asistencia (Profesores):

El usuario profesor tiene la posibilidad de registrar la asistencia de los estudiantes cuyas clases imparte, introduciendo la petición adecuada. Puede escribir “registrar asistencia”, “quiero registrar asistencia”, “deseo ir a registrar asistencia”, “deseo ir al registro de asistencia”, etc.

Registrar calificaciones (Profesores):

El usuario profesor tiene la posibilidad de registrar las calificaciones de los estudiantes en las clases que imparten. Puede escribir “registrar calificaciones”,

“quiero ir a registrar calificaciones”, “deseo ir a registrar calificaciones”, “ir al registro de calificaciones”, etc.

Control de faltas (Alumnos):

El usuario alumno tiene la posibilidad de ver las faltas o el control de faltas de las materias en las cuales está inscrito. Puede escribir “control de faltas”, “quiero ver mi control de faltas”, “deseo ir a control de faltas”, “ir a control de faltas”, etc.

Ver calificaciones (Alumnos):

El usuario alumno tiene la posibilidad de ver las calificaciones de las materias en las cuales está inscrito. Puede escribir “ver las calificaciones”, “quiero ver mis calificaciones”, “deseo ver las calificaciones”, “ir a calificaciones”, etc.

Mostrar mensaje de error:

Muestra un mensaje de error en caso de que el usuario ingrese una petición que no pueda ser entendida por “CHAPIE”.

Terminar la conversación:

Muestra un mensaje de despedida y termina la conversación actual entre el usuario y “CHAPIE”.

Nota: Al momento de escribir o solicitar alguna de las funcionalidades del chatbot, lo importante es que se dé a entender claramente la intención del usuario en lenguaje natural.

3.1. ARQUITECTURA

Este proyecto se implementó utilizando el modelo cliente – servidor, donde además se tiene conexión con el servicio cognitivo “LUIS” para la identificación de las “intenciones” de los usuarios y para que el chatbot procese el lenguaje natural.

Cuando los usuarios establecen una conversación con nuestro chatbot, se realizan intercambios de mensajes mediante peticiones y respuestas HTTP, donde se envían objetos de tipo JSON. El sistema envía peticiones HTTP al servidor de Microsoft Azure, en el cual está almacenado el chatbot.

En la Figura 12, se puede observar el protocolo de comunicación entre el cliente y el servidor donde se implementará el chatbot. También se muestran todos los elementos que interfieren en el intercambio de mensajes entre el usuario y el chatbot de una manera más detallada.

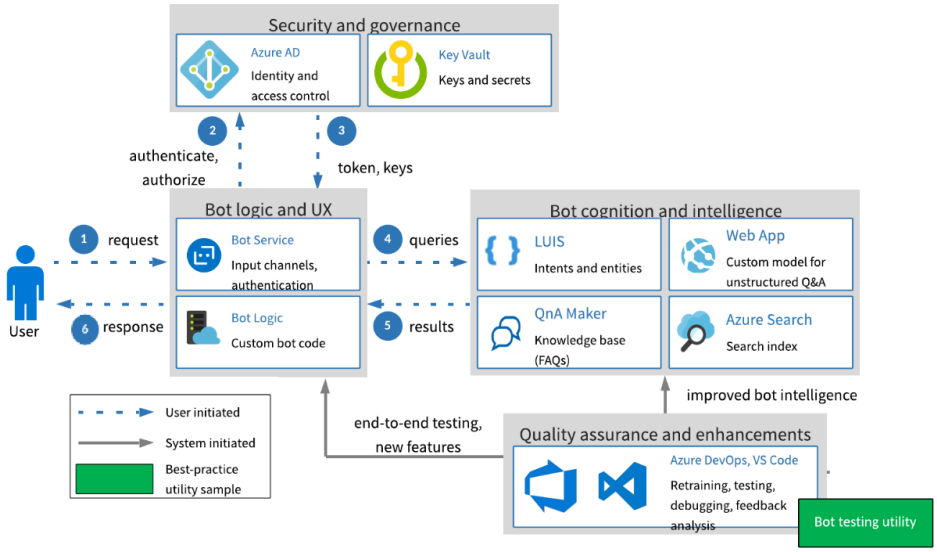


Figura 12: Comunicación Usuario - Bot

Tomado de (Microsoft Azure, 2019)

Este chatbot funciona recibiendo solicitudes (“request”) por parte de los usuarios y enviando respuestas (“response”) con las resoluciones a dichas solicitudes, en un contexto de peticiones HTTP.

Cuando un usuario envía un “request” (1), la solicitud ingresa al bloque Bot Logic and UX, el cual está compuesto por Bot Service y Bot Logic. Los cuales se encargan de las siguientes funciones:

Inicialmente se encargan de la autenticación y autorización para que la aplicación pueda utilizar un canal de Azure, el cual es creado previamente para procesar los “requests” de este proyecto. Para ello se agregó el ID y el password del canal asignado en Microsoft Azure al proyecto de Visual Studio 2019 antes de su publicación. Con esto se logra subir el proyecto realizado al canal de Azure asignado, para que el chatbot se pueda utilizar en cualquier aplicación Web.

El bloque Bot Logic and UX se comunica con el bloque Security and governance realizando la autenticación del id y el nombre correspondiente del canal de Azure (2). Posteriormente comprueba que la key del proyecto sea la misma que la del canal, para poder autorizar la implementación de proyecto (3).

Después, una vez terminado el proceso de autenticación, Bot Logic and UX se comunica con el bloque Bot cognition and intelligence, para poder llamar al servicio cognitivo “LUIS” mediante consultas o “queries” (4). Este servicio se encarga de identificar la “intención” de los usuarios con cada consulta y da como resultado o “result” un objeto tipo JSON serializado, el cual contiene información que es útil para el programa, como se observa en la Figura 13 (5).

La “intención” del usuario y las “entidades” que esta contiene son identificadas y guardadas en variables, las cuales que son leídas y utilizadas por el programa; además de la información textual como tal. Para lograr la conexión con el servicio cognitivo “LUIS”, se agregaron los respectivos “AppId”, “APIKey” y “HostName” del servicio cognitivo de “LUIS” al proyecto de Visual Studio 2019 antes de la publicación en Microsoft Azure.

El programa utiliza la información que devuelve el servicio cognitivo “LUIS” y dependiendo la “intención” detectada resuelve una acción determinada, la cual se muestra en modo de respuesta o “response” al cliente (6).

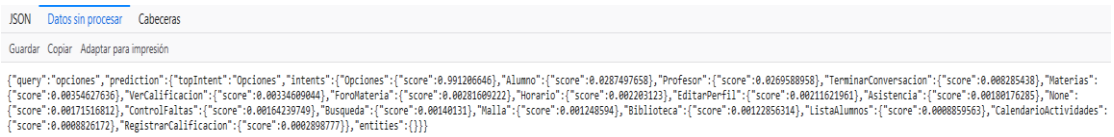


Figura 13: Estructura del objeto tipo JSON.

3.2. DISEÑO DEL CHATBOT

Para poder apreciar las posibles interacciones entre los usuarios y el sistema, se ha desarrollado un diagrama de flujo, Figura 14. En este diagrama se especificaron las acciones que pueden realizar los usuarios y las posibilidades de manejo de la conversación.

El sistema comienza con un saludo inicial en el cual también se identifica con el usuario, posteriormente nos pide ingresar nuestro nombre y luego escoger el tipo de usuario que va a utilizar el chatbot (Profesor - Alumno).

Una vez identificado el tipo de usuario el sistema muestra el menú correspondiente y espera por el ingreso de un “comando” apropiado para la ejecución.

Cuando el usuario introduce el “comando” de alguna de las opciones, se despliega una respuesta para dicho “comando” y el sistema espera por el ingreso del siguiente.

Entre los comandos que se pueden utilizar está “ver opciones”, con el cual se despliega el menú de opciones para cada tipo de usuario. También está el comando “Ver Horario Semanal”, con el cual se despliega el horario de los usuarios y un enlace de acceso para mejor visibilidad.

El comando “Revisar malla curricular” muestra una imagen de la malla curricular de la carrera a la que el usuario pertenece y un enlace para poder descargar el mismo. Si el usuario ingresa el comando “Ver materias”, el chatbot muestra una imagen de las materias inscritas actualmente y un enlace para una mejor visualización.

El comando “Ver Lista de Alumnos” muestra botones para acceder a la lista de alumnos de las materias a las cuales el usuario está inscrito. El comando “Ir a Biblioteca UDLA” muestra un botón para acceder a la biblioteca de la Universidad de las Américas (UDLA).

El comando “Realizar búsqueda de información” nos muestra la forma de realizar una búsqueda web de cualquier tema. El chatbot extrae la información a buscar y muestra un botón para acceder al resultado de la búsqueda web. El comando

“Calendario de Actividades” nos muestra un botón de acceso para el calendario de actividades de los usuarios.

El comando “Ir a Foro de Materia” muestra un botón para acceder al foro de cada una de las materias a las cuales el usuario está inscrito. El comando “Ir a editar información de Perfil” muestra un botón para acceder al perfil personal de cada usuario y poder modificarlo.

También existen comandos que son específicos para cada tipo de usuario. El comando “Registrar Asistencia” muestra a los usuarios “Profesores” un botón para acceder al registro de asistencia de cada una de las materias a las que está inscrito. De igual forma, el comando “Registrar Calificaciones”, muestra a los usuarios “Profesores” botones para acceder al registro de calificaciones por materia inscrita.

Entre los comandos que son específicos para usuarios “Alumnos” está el comando “Ver Calificaciones”, el cual muestra al usuario botones para acceder a la visualización de calificaciones por materia inscrita. También para los usuarios “Alumnos” existe el comando “Control de Faltas”, el cual nos muestra un botón para acceder al control de faltas de todas las materias a las que está inscrito.

Finalmente, el comando “Terminar Conversación” permite todos los usuarios terminar la conversación que se lleva a cabo con el chatbot, se muestra un mensaje de despedida a los usuarios y termina la comunicación.

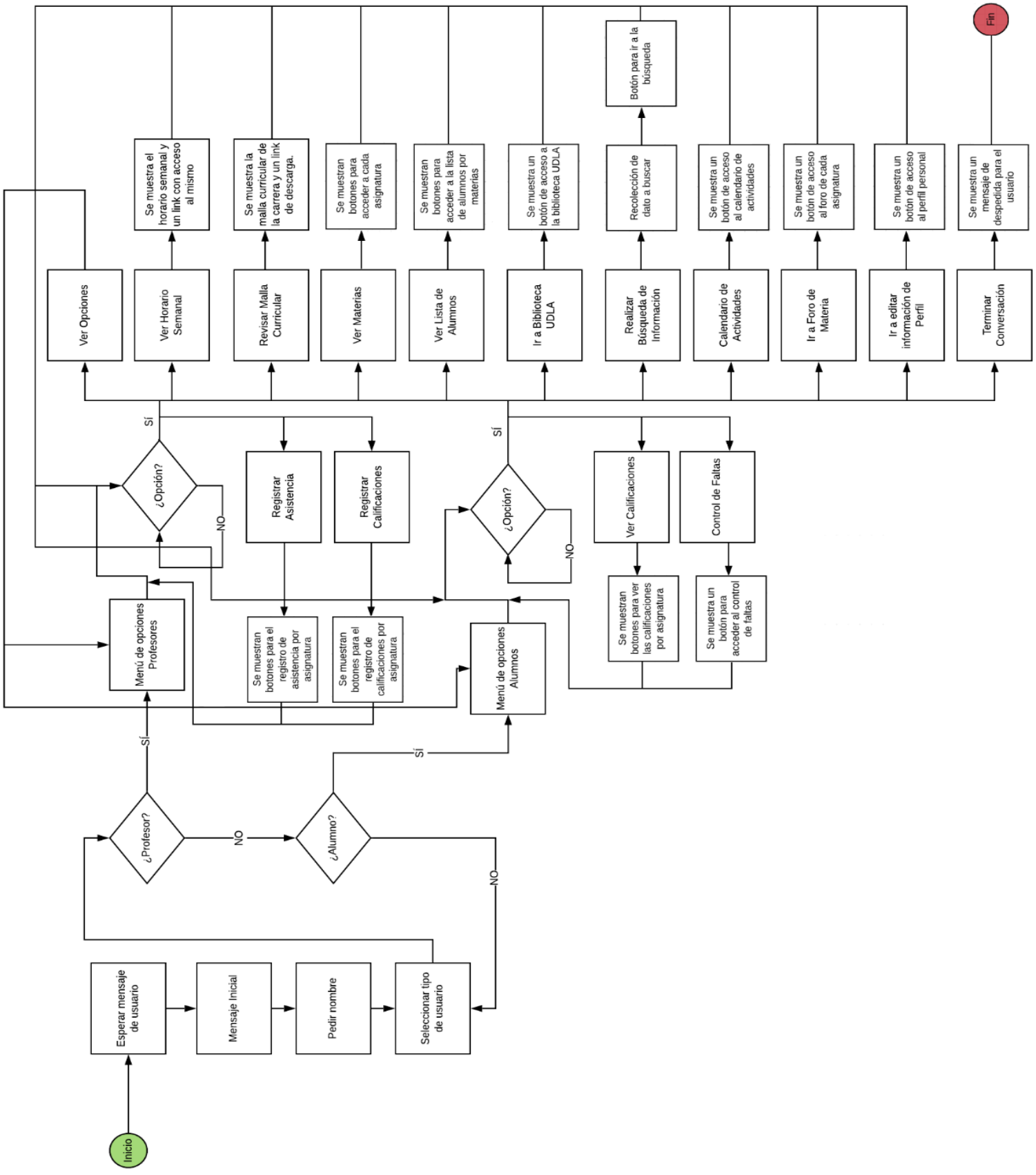


Figura 14: Diagrama de flujo, “CHAPIE”.

3.3. DESARROLLO DEL CHATBOT

En esta etapa se programaron las configuraciones del chatbot, las funcionalidades de este y cómo va a responder a los requerimientos de los usuarios.

3.3.1. TECNOLOGÍAS

Las principales tecnologías utilizadas para el desarrollo de “CHAPIE” son:

Aula Virtual UDLA

El servicio web de la Universidad de las Américas donde se probará la implementación de “CHAPIE”.

C#

El lenguaje de programación por el cual se ha optado para el desarrollo de este proyecto.

Visual Studio 2019

Entorno de programación el cual interpreta el código C#. Es el editor de código sobre el cual se desarrolla el proyecto. El tener la posibilidad de conectarse con servicios cognitivos y su facilidad para publicar proyectos en la nube de Azure han facilitado el desarrollo e implementación del chatbot.

JavaScript Object Notation (JSON)

Es el formato de estructuración de datos con el cual la información entre el servicio cognitivo “LUIS” y la aplicación es intercambiada cuando existe interacción entre el cliente y el servidor.

Bot Framework Emulator (v4)

Es el software de emulación sobre el cual se fue probando el chatbot mediante se iba desarrollado, antes de la publicación en Azure y la implementación.

Microsoft Azure

Nube pública de Microsoft donde se publicó “CHAPIE” para su posterior implementación.

LUIS (Language Understanding Intelligent Service)

Servicio cognitivo utilizado para la identificación de las “intenciones” de los usuarios y el proceso de aprendizaje autónomo de “CHAPIE” con respecto a las conversaciones en lenguaje natural.

3.3.2. PREPARACIÓN DEL ENTORNO

Para la preparación del entorno sobre el cual se desarrolló “CHAPIE” se procedió primero con la instalación de Visual Studio 2019, Figura 15; Bot Framework Emulator (v4), Figura 16. Además de la creación de las cuentas y licencias respectivas en Microsoft Azure, Figura 17; y en “LUIS” (Language Understanding Intelligent Service), Figura 18.

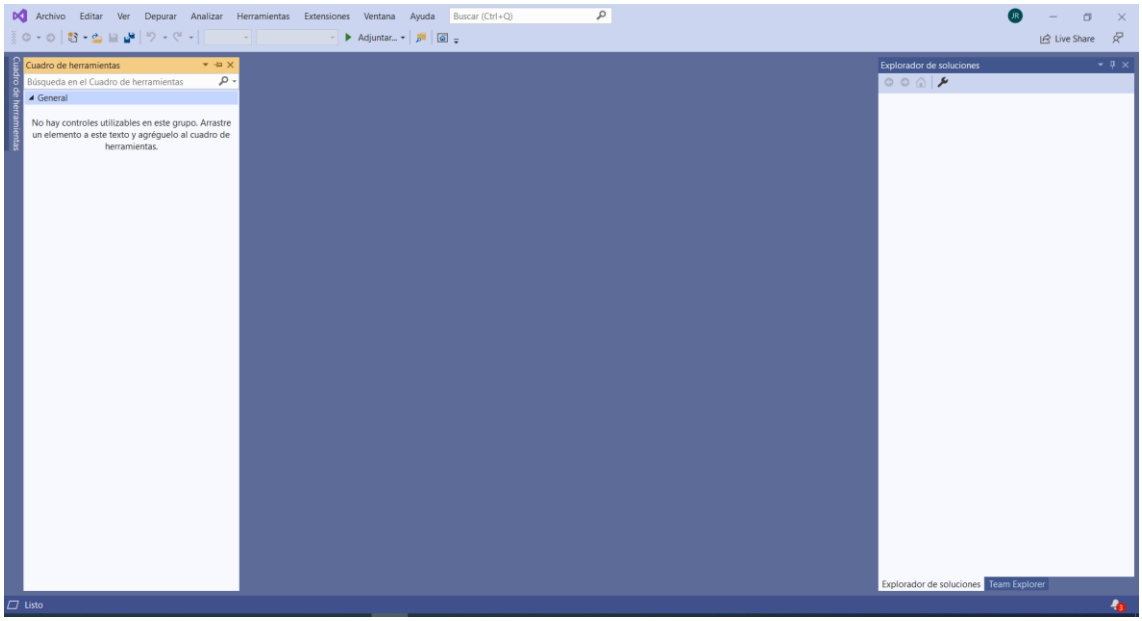


Figura 15: Interfaz Visual Studio 2019

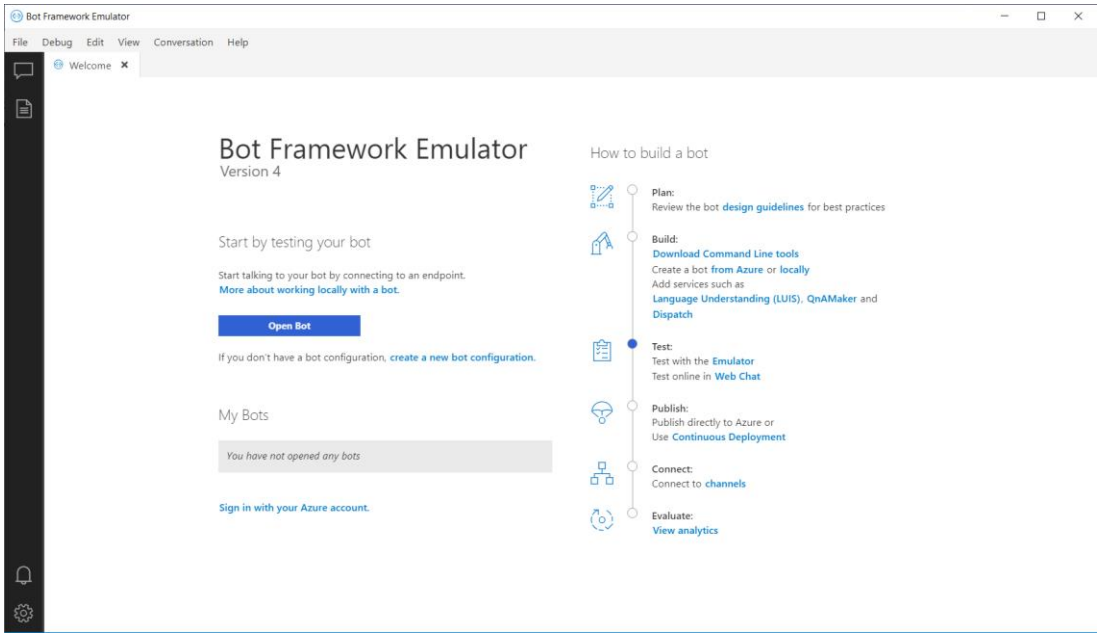


Figura 16: Interfaz Bot Framework Emulator

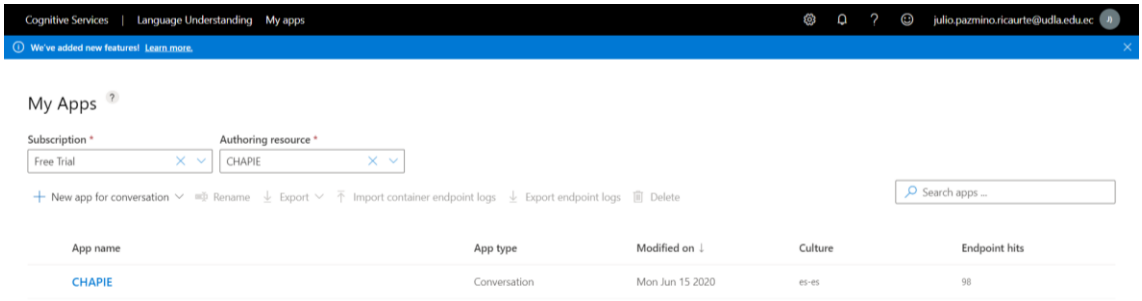


Figura 17: “LUIS” (Language Understanding Service), tipo de subscripción.

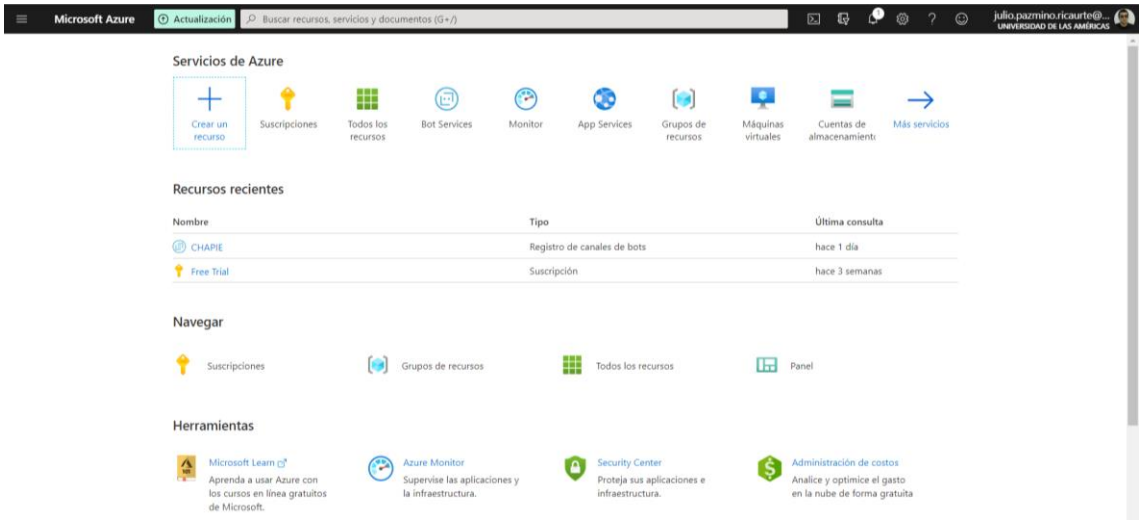


Figura 18: Microsoft Azure, Subscripción y Registro de canal del bot.

3.3.3. CREACIÓN DEL PROYECTO EN VISUAL STUDIO 2019

Lo primero que se llevó a cabo fue la creación del proyecto en Visual Studio 2019 partiendo de la plantilla “Empty Bot” que proporciona el SDK Bot Framework v4 (.NET Core 2.1), como se muestra en la Figura 19, instalado junto con Visual Studio 2019.

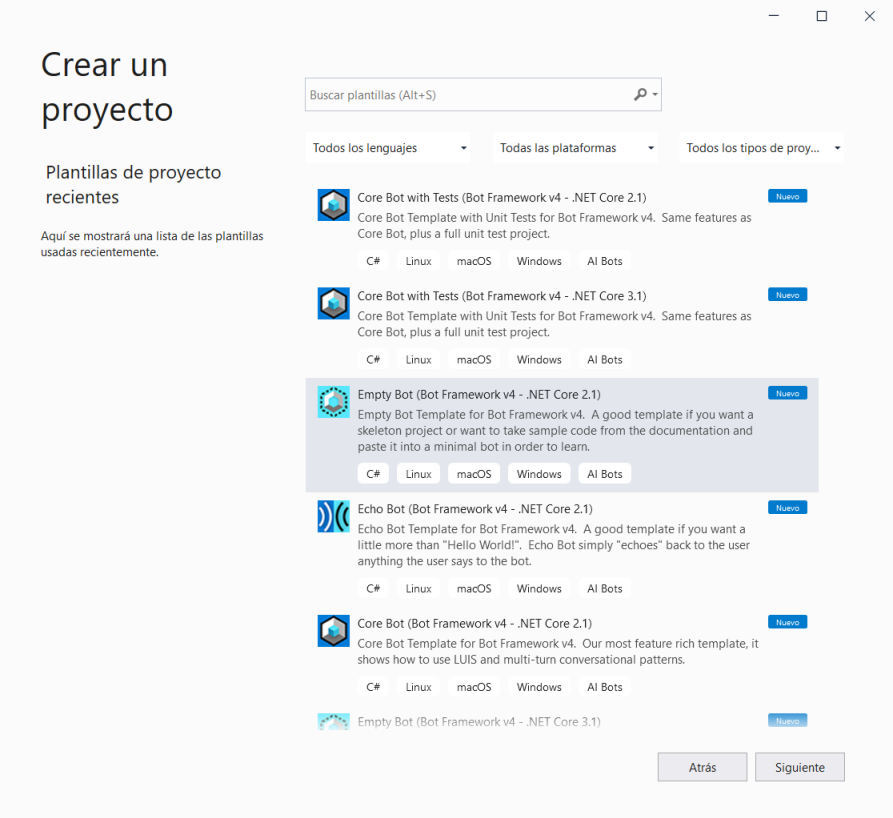


Figura 19: Plantilla “Empty Bot” de Visual Studio 2019

Una vez analizada la estructura de la plantilla “Empty Bot” y los elementos que la conforman, se procedió con la programación de las configuraciones y funcionalidades necesarias para cumplir con los objetivos planteados.

3.3.4. ESTRUCTURA DE “CHAPIE” EN VISUAL STUDIO 2019

Para la programación de “CHAPIE” se revisaron proyectos similares previos y se adquirieron conocimientos relacionados a la creación de agentes inteligentes en Visual Studio.

La estructura de “CHAPIE”, una vez finalizado el desarrollo dentro del programa Visual Studio 2019, se ve de la siguiente manera, Figura 20:

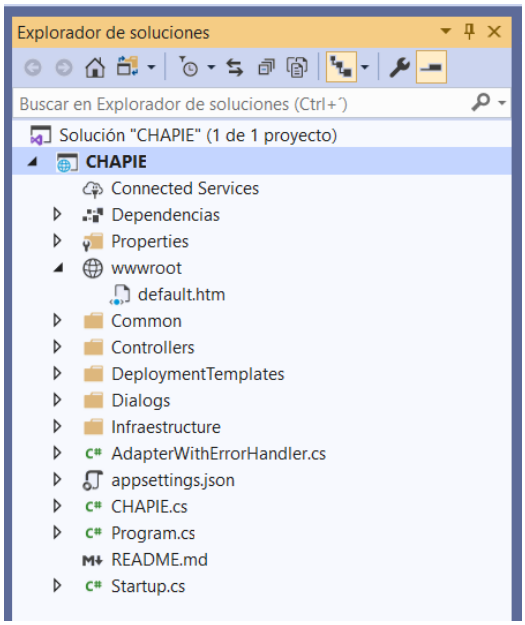


Figura 20: Estructura de “CHAPIE” en Visual Studio 2019.

Dentro del proyecto “CHAPIE”, se tiene la carpeta wwwroot, la cual contiene una página llamada “default.htm”, que se despliega cuando se ejecuta el proyecto de Visual Studio 2019, Figura 21. Esta página nos proporciona una dirección URL, la cual es necesaria para que el software Bot Framework Emulator, pueda realizar las pruebas previas a la implementación.

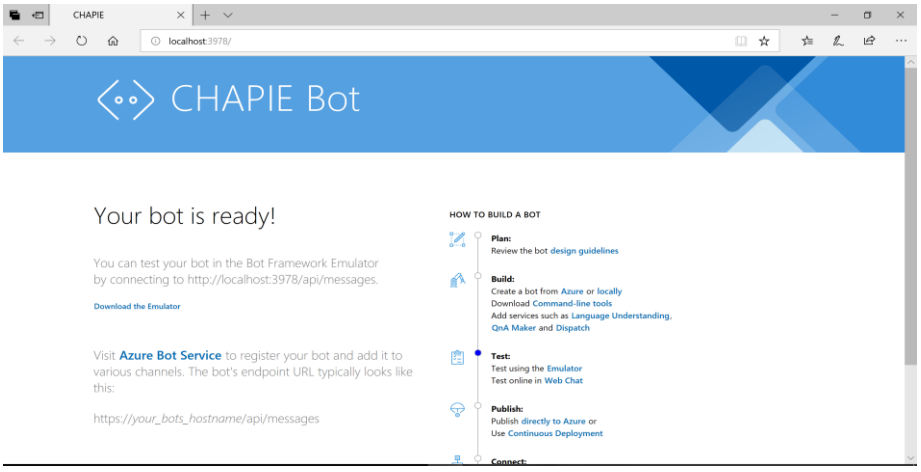


Figura 21: Página http que se despliega al ejecutar “CHAPIE”.

Dentro de la carpeta “Controllers”, Figura 22, se encuentra una clase llamada “BotController”, la cual es el controlador principal de “CHAPIE”, Figura 23. Dentro de esta clase controlador existe un método llamado “PostAsync”, el cual se encarga del envío y recepción de mensajes entre el sistema y el usuario, Figura 24.

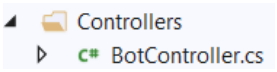


Figura 22: Carpeta “Controllers”

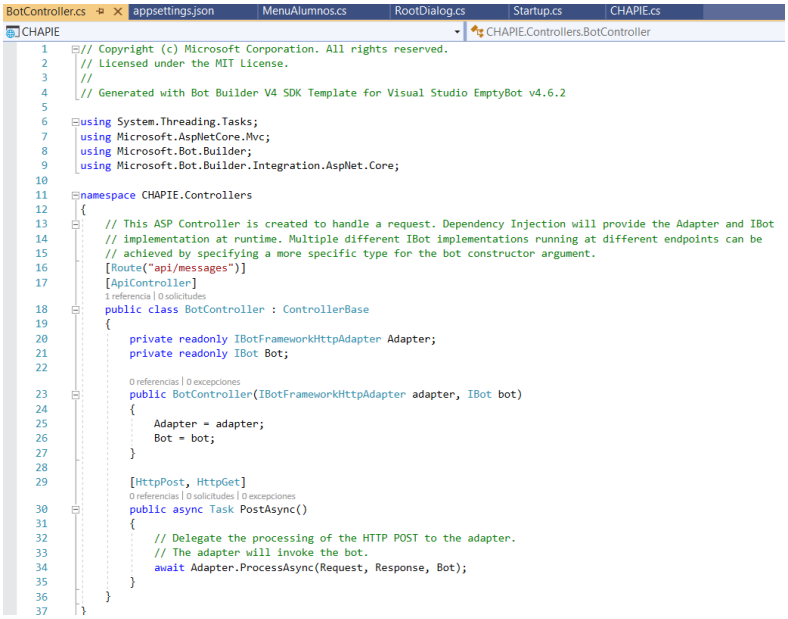


Figura 23: BotController (Controlador Principal de “CHAPIE”)

```
public async Task PostAsync()
{
    // Delegate the processing of the HTTP POST to the adapter.
    // The adapter will invoke the bot.
    await Adapter.ProcessAsync(Request, Response, Bot);
}
```

Figura 24: Método “PostAsync”

La carpeta “DeploymentTemplates”, Figura 25, contiene algunos archivos que el programa utiliza para la implementación del proyecto en Microsoft Azure.

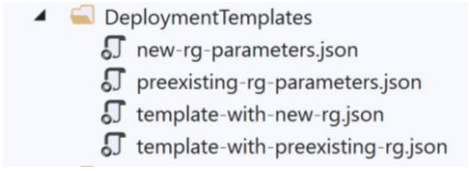


Figura 25: Carpeta “DeploymentTemplates”

La carpeta “Dialogs”, Figura 26, contiene la clase “RootDialog.cs”. Esta clase se encarga de la ejecución de las funcionalidades de “CHAPIE”, correspondiendo a las “intenciones” de los usuarios. En esta clase se programaron los mensajes que se muestran en pantalla a los usuarios, las resoluciones de las interacciones y el llamado al menú principal para cada tipo de usuario.

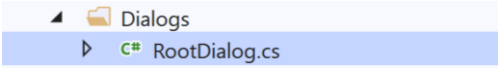


Figura 26: Carpeta “Dialogs”

Dentro de la carpeta “Common”, Figura 27, se encuentra la programación de los botones, menús y algunas funcionalidades de “CHAPIE”, por lo que algunos métodos harán referencia a clases que se encuentran dentro de esta carpeta.

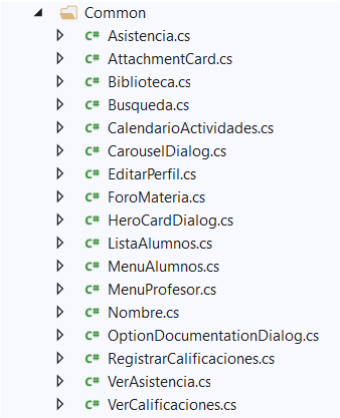


Figura 27: Carpeta “Common”.

En la carpeta “Infraestructure”, Figura 28, se encuentran las configuraciones para la conexión con el servicio cognitivo “LUIS”.

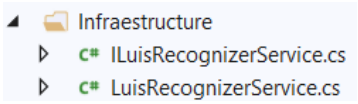


Figura 28: Carpeta “Infraestructure”.

“CHAPIE” también posee una clase muy importante, utilizada para el control de errores, llamada “AdapterWithErrorHandler”, Figura 29. Esta clase permite capturar los posibles errores que pueden ocurrir durante el desarrollo del bot. Errores tales como desconexión o falta de respuesta del servicio cognitivo o si se suscitan eventos no programados.

```
using Microsoft.Bot.Builder.Integration.AspNet.Core;
using Microsoft.Bot.Builder.TraceExtensions;
using Microsoft.Extensions.Configuration;
using Microsoft.Extensions.Logging;

namespace CHAPIE
{
    public class AdapterWithErrorHandler : BotFrameworkHttpAdapter
    {
        public AdapterWithErrorHandler(IConfiguration configuration, ILogger<BotFrameworkHttpAdapter> logger)
            : base(configuration, logger)
        {
            OnTurnError = async (turnContext, exception) =>
            {
                // Log any leaked exception from the application.
                logger.LogError(exception, $"[OnTurnError] unhandled error : {exception.Message}");

                // Send a message to the user
                await turnContext.SendActivityAsync("The bot encountered an error or bug.");
                await turnContext.SendActivityAsync("To continue to run this bot, please fix the bot source code.");

                // Send a trace activity, which will be displayed in the Bot Framework Emulator
                await turnContext.TraceActivityAsync("OnTurnError Trace", exception.Message, "https://www.botframework.com/schemas/error", "TurnError");
            };
        }
    }
}
```

Figura 29: Clase “AdapterWithErrorHandler” de “CHAPIE”.

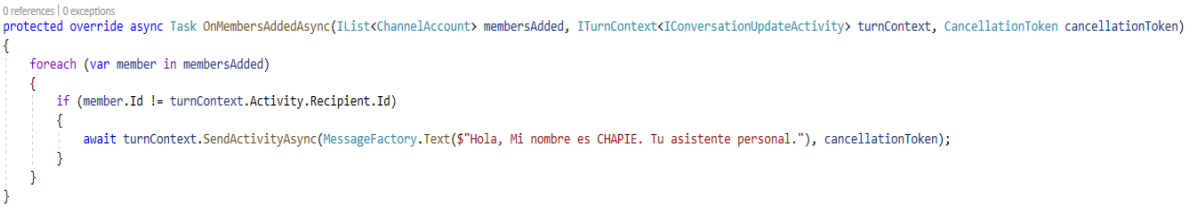
3.3.5. PROGRAMACIÓN DE FUNCIONES INICIALES DE “CHAPIE”

La clase principal del proyecto lleva el mismo nombre de este (CHAPIE.cs), Figura 30. Esta clase se genera automáticamente al momento de crear el proyecto. En esta clase se implementaron los principales métodos para la interacción del chatbot con los usuarios.

```
namespace CHAPIE
{
    3 references
    public class CHAPIE<T> : ActivityHandler where T: Dialog
    {
        protected readonly Dialog _dialog; //
        protected readonly BotState _conversacionState; //Guardar estado de conversacion
        protected readonly ILogger _logger;  // Guardar registros de clase

        0 references | 0 exceptions
        public CHAPIE(T dialog, ConversationState conversationState, ILogger<CHAPIE<T>> logger)
        {
            _dialog = dialog;
            _conversacionState = conversationState;
            _logger = logger;
        }


        0 references | 0 exceptions
        protected override async Task OnMembersAddedAsync(IList<ChannelAccount> membersAdded, ITurnContext<IConversationUpdateActivity> turnContext, CancellationToken cancellationToken)
        {
            foreach (var member in membersAdded)
            {
                if (member.Id != turnContext.Activity.Recipient.Id)
                {
                    await turnContext.SendActivityAsync(MessageFactory.Text($"Hola, Mi nombre es CHAPIE. Tu asistente personal."), cancellationToken);
                }
            }
        }


        0 references | 0 exceptions
        protected override async Task OnMessageActivityAsync(ITurnContext<IMessageActivity> turnContext, CancellationToken cancellationToken)
        {
            //Verificar si se ha recibido una actividad
        await  _dialog.RunAsync(
                turnContext,
                _conversacionState.CreateProperty<DialogState>(nameof(DialogState)),
                cancellationToken
                );

            //var message = turnContext.Activity.Text;
            //await turnContext.SendActivityAsync($"Hola {message}. Bienvenido", cancellationToken: cancellationToken);
        }

        0 references | 0 exceptions
        public override async Task OnTurnAsync(ITurnContext turnContext, CancellationToken cancellationToken = default)
        {
            //procesa una actividad entrante
            await base.OnTurnAsync(turnContext, cancellationToken);
            await _conversacionState.SaveChangesAsync(turnContext, false, cancellationToken); //Guarda estados de una conversacion
        }
    }
```

Figura 30: Clase “CHAPIE.cs”.

El método “OnMembersAddedAsync”, Figura 31, permite capturar la información de los nuevos usuarios que interactúan con el bot y los diferencia con un ID. Esto permite identificar cuando existe un nuevo usuario que va a utilizar el chatbot.



```
0 references | 0 exceptions
protected override async Task OnMembersAddedAsync(IList<ChannelAccount> membersAdded, ITurnContext<IConversationUpdateActivity> turnContext, CancellationToken cancellationToken)
{
    foreach (var member in membersAdded)
    {
        if (member.Id != turnContext.Activity.Recipient.Id)
        {
            await turnContext.SendActivityAsync(MessageFactory.Text($"Hola, Mi nombre es CHAPIE. Tu asistente personal."), cancellationToken);
        }
    }
}
```

Figura 31: Método “OnMembersAddedAsync”.

El método correspondiente a la captura de información que entra por parte del usuario se llama “OnMessageActivityAsync”, Figura 32. Este método identifica si el sistema ha recibido una actividad o dato de entrada por parte del usuario, capturando la información ingresada.

```
0 references | 0 exceptions
protected override async Task OnMessageActivityAsync(ITurnContext<IMessageActivity> turnContext, CancellationToken cancellationToken)
{
    //Verificar si se ha recibido una actividad
await  _dialog.RunAsync(
        turnContext,
        _conversacionState.CreateProperty<DialogState>(nameof(DialogState)),
        cancellationToken
        );

    //var message = turnContext.Activity.Text;
    //await turnContext.SendActivityAsync($"Hola {message}. Bienvenido", cancellationToken: cancellationToken);
}
```

Figura 32: Método “OnMessageActivityAsync”.

El siguiente método importante en el funcionamiento del chatbot se llama “OnTurnAsync”, Figura 33. El cual procesa las actividades entrantes por parte del usuario y guarda los estados de la conversación que se tiene con el chatbot.

```
0 references | 0 exceptions
public override async Task OnTurnAsync(ITurnContext turnContext, CancellationToken cancellationToken = default)
{
    //procesa una actividad entrante
    await base.OnTurnAsync(turnContext, cancellationToken);
    await _conversacionState.SaveChangesAsync(turnContext, false, cancellationToken); //Guarda estados de una conversacion
}
```

Figura 33: Método “OnTurnAsync”.

La clase “Startup.cs”, Figura 34, contiene las configuraciones para el funcionamiento del chatbot y se ejecuta al momento del arranque del programa. En esta clase también llama a ejecución a la clase “RootDialog.cs”.

```
namespace CHAPIE
{
    2 references
    public class Startup
    {
        0 references | 0 exceptions
        public Startup(IConfiguration configuration)
        {
            Configuration = configuration;
        }

        1 reference | 0 exceptions
        public IConfiguration Configuration { get; }

        // This method gets called by the runtime. Use this method to add services to the container.
        0 references | 0 exceptions
        public void ConfigureServices(IServiceCollection services)
        {
            services.AddMvc().SetCompatibilityVersion(CompatibilityVersion.Version_2_1);

            // Create the Bot Framework Adapter with error handling enabled.
            services.AddSingleton<IBotFrameworkHttpAdapter, AdapterWithErrorHandler>();
            services.AddSingleton<IStorage, MemoryStorage>();
            services.AddSingleton<ConversationState>();
            services.AddSingleton<RootDialog>();


            // Create the bot as a transient. In this case the ASP Controller is expecting an IBot.
            services.AddTransient<IBot, CHAPIE<RootDialog>>(); //pasa a root dialog
        }

        // This method gets called by the runtime. Use this method to configure the HTTP request pipeline.
        0 references | 0 exceptions
        public void Configure(IApplicationBuilder app, IHostingEnvironment env)
        {
            if (env.IsDevelopment())
            {
                app.UseDeveloperExceptionPage();
            }
            else
            {
                app.UseHsts();
            }

            app.UseDefaultFiles();
            app.UseStaticFiles();
            app.UseWebSockets();
            app.UseMvc();
        }
    }
}
```

Figura 34: Clase “Startup.cs”.

La clase “RootDialog.cs”, Figura 35, es la clase que se encarga de la interacción de “CHAPIE” con los usuarios. Dentro del método principal de la clase se encuentra el llamado a los demás métodos del programa. Estos métodos se encargan de la interacción de “CHAPIE”.

```
namespace CHAPIE.Dialogs
{
    3 referencias
    public class RootDialog : ComponentDialog
    {
        private string usuario = "";
        private string nombreUsuario = "";
        protected readonly ILuisRecognizerService _luisRecognizerService;   //Conexion Luis

        0 referencias | 0 excepciones
        public RootDialog(ILuisRecognizerService luisRecognizerService)
        {

            _luisRecognizerService = luisRecognizerService;

            var waterfallStep = new WaterfallStep[]
            {
                // variables

              SetName,
              ShowOpcionInicio,
              ShowMenuSeleccionUsuario,
              ShowRequerimientosUsuarios,
              ShowOpcionesChatbot,
              ShowOpcionesChatbot,
              ShowOpcionesChatbot,
              ShowOpcionesChatbot,
              ShowOpcionesChatbot,
              ShowOpcionesChatbot,
              ShowOpcionesChatbot,
              ShowOpcionesChatbot,
              ShowOpcionesChatbot,
              ShowOpcionesChatbot,
              ShowOpcionesChatbot,
              ShowOpcionesChatbot,
              ShowOpcionesChatbot,
              ShowOpcionesChatbot,
              ShowOpcionesChatbot,
              ShowOpcionesChatbot,
              ShowOpcionesChatbot,
              ShowOpcionesChatbot,
              ShowOpcionesChatbot,
              ShowOpcionesChatbot,
              ShowOpcionesChatbot,
              ShowOpcionesChatbot,
              ShowOpcionesChatbot,
              ShowOpcionesChatbot,
              ShowOpcionesChatbot,
              ShowOpcionesChatbot,
              ShowOpcionesChatbot,
              ShowOpcionesChatbot,
              ShowOpcionesChatbot,
              ShowOpcionesChatbot,
              ShowOpcionesChatbot,
              ShowOpcionesChatbot,
              ShowOpcionesChatbot,
              ShowOpcionesChatbot
            };

            AddDialog(new WaterfallDialog(nameof(WaterfallDialog), waterfallStep)); //llamada a variables
           // AddDialog(new ConfirmPrompt(nameof(ConfirmPrompt),ConfirmValidate));
            AddDialog(new TextPrompt(nameof(TextPrompt)));
            AddDialog(new ChoicePrompt(nameof(ChoicePrompt)));
           // AddDialog(new NumberPrompt<int>(nameof(NumberPrompt<int>), ValidateAge));
        }
```

Figura 35: Clase “RootDialog”, variables globales y método principal.

El primer método en ejecutarse de esta clase, después del saludo inicial, es el método “SetName”, Figura 36. El cual se encarga de hacer la petición del nombre de usuario y esperar a que se ingrese dicha solicitud.

```
private async Task<DialogTurnResult> SetName(WaterfallStepContext stepContext, CancellationToken cancellationToken)//
{
    await Task.Delay(1000);

    var name = stepContext.Context.Activity.Text;
    return await stepContext.PromptAsync(
        nameof(TextPrompt),
        new PromptOptions
        {
            Prompt = MessageFactory.Text($"Por favor dime tu nombre"), //Obtencion de dato
        },
        cancellationToken
        );

}
```

Figura 36: Método “SetName”

Después, el método “ShowOpcionInicio”, Figura 37, se encarga de guardar la información que ingresa el usuario en una variable, la cual se utilizará posteriormente. También muestra las opciones para la identificación del tipo de usuario que está utilizando “CHAPIE” (Profesor - Estudiante). Para lo cual invoca al método “ShowOptions”.

```
private async Task<DialogTurnResult> ShowOpcionInicio(WaterfallStepContext stepContext, CancellationToken cancellationToken)
{
    //
    await Task.Delay(2000);
    nombreUsuario = stepContext.Context.Activity.Text;

    return await ShowOptions(stepContext, cancellationToken);


}
```

Figura 37: Método “ShowOpcionInicio”.

El método “ShowOptions”, Figura 38, se encarga de llamar a la clase que contiene la programación del diseño y acción de los botones de selección de usuario. La clase “OptionDocumentationDialog.cs”, Figura 39, que se encuentra en la carpera “Common”, es la que se utiliza para ello.

```
private async Task<DialogTurnResult> ShowOptions(WaterfallStepContext stepContext, CancellationToken cancellationToken)
{

    //MOstrar botones
    return await OptionDocumentationDialog.ShowOptions(stepContext, cancellationToken);
}
```

Figura 38: Método “ShowOptions”.

```
namespace CHAPIE.Common
{
    1 referencia
    public class OptionDocumentationDialog
    {
        1 referencia | 0 excepciones
        public static async Task<DialogTurnResult> ShowOptions(WaterfallStepContext stepContext, CancellationToken cancellationToken)
        {
            await Task.Delay(2000);
            var options =  await stepContext.PromptAsync(
                nameof(TextPrompt),
                new PromptOptions
                {
                  Prompt = CreateSuggestedActions(),
                  Style = ListStyle.HeroCard
                },
                cancellationToken
                );
            return options;
        }


        1 referencia | 0 excepciones
        private static Activity CreateSuggestedActions()
        {
            var reply = MessageFactory.Text("Por favor dime si eres profesor o alumno");
            reply.SuggestedActions = new SuggestedActions()
            {
                Actions = new List<CardAction>()
                {
                    new CardAction(){Title = "Profesor", Value="Profesor", Type=ActionTypes.ImBack},
                    new CardAction(){Title = "Alumno", Value="Alumno", Type=ActionTypes.ImBack},
                }
            };
            return reply;
        }
    }
}
```

Figura 39: Clase “OptionDocumentationDialog.cs”.

Posteriormente le método “ShowMenuSeleccionUsuario”, Figura 40, dependiendo la selección anterior, guarda el tipo de usuario en otra variable y muestra un mensaje de entrada correspondiente. Para mostrar el mensaje, existe un método para cada tipo de usuario (Profesor, Figura 41 – Alumno, Figura 42). También existe un método que valida si el usuario ingresado no existe (None), Figura 43.

```
private async Task<DialogTurnResult> ShowMenuSeleccionUsuario(WaterfallStepContext stepContext, CancellationToken cancellationToken)
{
    var recognizerResult = await _luisRecognizerService._recognizer.RecognizeAsync(stepContext.Context, cancellationToken);
    var topIntent = recognizerResult.GetTopScoringIntent();
    switch (topIntent.intent)
    {

        case "Alumno":
            usuario = "Alumno";
            await IntentAlumno(stepContext, recognizerResult, cancellationToken);
            break;
        case "Profesor":
            usuario = "Profesor";
            await IntentProfesor(stepContext, recognizerResult, cancellationToken);
            break;
        case "None":
            await IntentNone(stepContext, recognizerResult, cancellationToken);
            break;
        default:
            await stepContext.Context.SendActivityAsync($"No entiendo lo que dices", cancellationToken: cancellationToken);
            await Task.Delay(2000);
            await stepContext.Context.SendActivityAsync($"Debes seleccionar una de las opciones que te muestro, de acuerdo?", cancellationToken: cancellationToken);

            await stepContext.EndDialogAsync(cancellationToken: cancellationToken);

            break;
    }
    return await stepContext.EndDialogAsync(cancellationToken:cancellationToken);

}
```

Figura 40: Método “ShowMenuSeleccionUsuario”

```
private async Task<DialogTurnResult> IntentProfesor(WaterfallStepContext stepContext, RecognizerResult recognizerResult, CancellationToken cancellationToken)
{
    await Task.Delay(1000);
    await stepContext.Context.SendActivityAsync($"Bienvenido {nombreUsuario} ", cancellationToken: cancellationToken);

    return await stepContext.PromptAsync(
        nameof(TextPrompt),
        new PromptOptions
        {
            Prompt = MessageFactory.Text($""), //Obtencion de dato
        },
        cancellationToken
        );

}
```

Figura 41: Método “IntentProfesor”

```
private async Task<DialogTurnResult> IntentAlumno(WaterfallStepContext stepContext, RecognizerResult recognizerResult, CancellationToken cancellationToken)
{
    await Task.Delay(1000);
    await stepContext.Context.SendActivityAsync($"Bienvenido {nombreUsuario} ", cancellationToken: cancellationToken);

    return await stepContext.PromptAsync(
        nameof(TextPrompt),
        new PromptOptions
        {
            Prompt = MessageFactory.Text($""), //Obtencion de dato
        },
        cancellationToken
        );
}
```

Figura 42: Método “IntentAlumno”

```
private async Task IntentNone(WaterfallStepContext stepContext, RecognizerResult recognizerResult, CancellationToken cancellationToken)
{
    await stepContext.Context.SendActivityAsync($"No entiendo lo que dices", cancellationToken: cancellationToken);
    await Task.Delay(2000);
    await stepContext.Context.SendActivityAsync($"Debes seleccionar una de las opciones que te muestro, de acuerdo?", cancellationToken: cancellationToken);
    await stepContext.EndDialogAsync(cancellationToken: cancellationToken);
}
```

Figura 43: Método “IntentNone”

El siguiente método que se ejecuta es “ShowRequerimientosUsuarios”, Figura 44, el cual se encarga de llamar a las clases “MenuProfesor” y “MenuAlumnos” de la carpeta “common”, las cuales contienen los menús de usuario.

```
private async Task<DialogTurnResult> ShowRequerimientosUsuarios(WaterfallStepContext stepContext, CancellationToken cancellationToken)
{
    await Task.Delay(1000);
    if (usuario.Equals("Profesor"))
    {
      return  await MenuProfesor.ShowOptions(stepContext, cancellationToken);
    }

    if (usuario.Equals("Alumno"))
    {
        return await MenuAlumnos.ShowOptions(stepContext, cancellationToken);
    }

    await Task.Delay(1000);

    return await stepContext.PromptAsync(
        nameof(TextPrompt),
        new PromptOptions { Prompt = MessageFactory.Text($"") },  //peticion de datos
        cancellationToken
        );
}
```

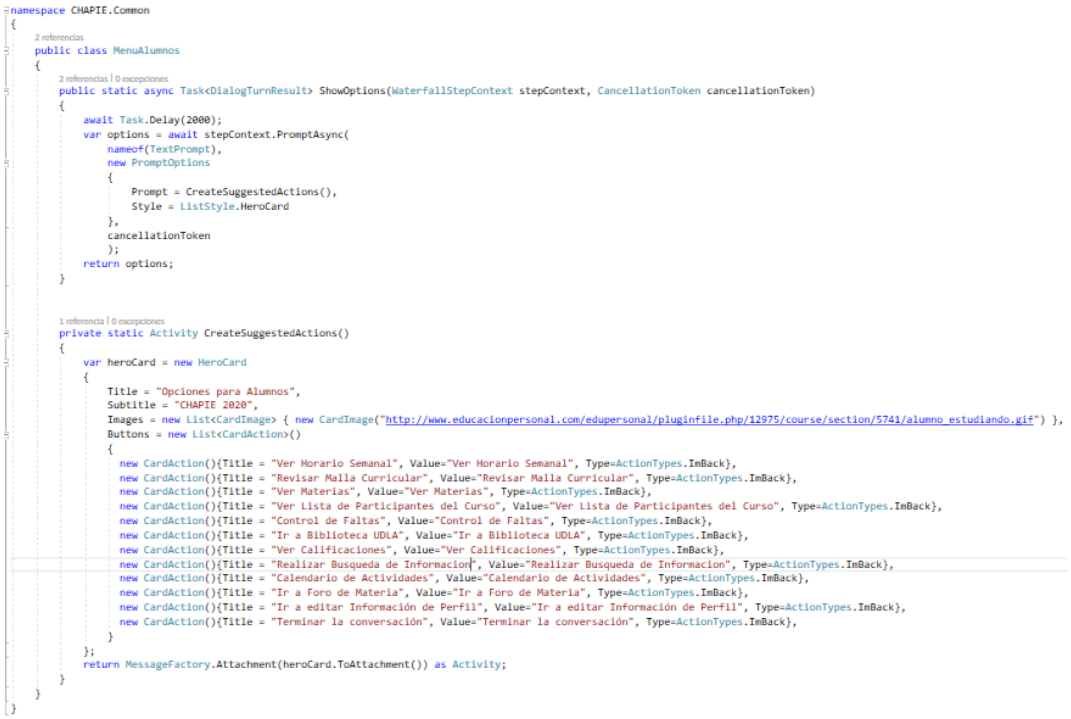
Figura 44: Método “ShowRequerimientosUsuarios”.

La clase “MenuProfesor”, Figura 45, contiene la programación del diseño del menú para el usuario “Profesor”.

```
namespace CHAPIE.Common
{
    2 referencias
    public class MenuProfesor
    {
        2 referencias | 0 excepciones
        public static async Task<DialogTurnResult> ShowOptions(WaterfallStepContext stepContext, CancellationToken cancellationToken)
        {
            await Task.Delay(2000);
            var options = await stepContext.PromptAsync(
                nameof(TextPrompt),
                new PromptOptions
                {
                    Prompt = CreateSuggestedActions(),
                    Style = ListStyle.HeroCard
                },
                cancellationToken
                );
            return options;
        }


        1 referencia | 0 excepciones
        private static Activity CreateSuggestedActions()
        {
            var heroCard = new HeroCard
            {
                Title = "Opciones para Profesores",
                Subtitle = "CHAPIE 2020",
                Images = new List<CardImage> { new CardImage("https://i.pinimg.com/originals/be/19/69/be1969906e09cc2cb7ac8abf5a985827.gif") },
                Buttons = new List<CardAction>()
                {
                  new CardAction(){Title = "Ver Horario Semanal", Value="Ver Horario Semanal", Type=ActionTypes.ImBack},
                  new CardAction(){Title = "Revisar Malla Curricular", Value="Revisar Malla Curricular", Type=ActionTypes.ImBack},
                  new CardAction(){Title = "Ver Materias", Value="Ver Materias", Type=ActionTypes.ImBack},
                  new CardAction(){Title = "Ver Lista de Alumnos", Value="Ver Lista de Alumnos", Type=ActionTypes.ImBack},
                  new CardAction(){Title = "Registrar Asistencia", Value="Registrar Asistencia", Type=ActionTypes.ImBack},
                  new CardAction(){Title = "Ir a Biblioteca UDLA", Value="Ir a Biblioteca UDLA", Type=ActionTypes.ImBack},
                  new CardAction(){Title = "Registrar Calificaciones", Value="Registrar Calificaciones", Type=ActionTypes.ImBack},
                  new CardAction(){Title = "Realizar Busqueda de Informacion", Value="Realizar Busqueda de Informacion", Type=ActionTypes.ImBack},
                  new CardAction(){Title = "Calendario de Actividades", Value="Calendario de Actividades", Type=ActionTypes.ImBack},
                  new CardAction(){Title = "Ir a Foro de Materia", Value="Ir a Foro de Materia", Type=ActionTypes.ImBack},
                  new CardAction(){Title = "Ir a editar Información de Perfil", Value="Ir a editar Información de Perfil", Type=ActionTypes.ImBack},
                  new CardAction(){Title = "Terminar la conversación", Value="Terminar la conversación", Type=ActionTypes.ImBack},
                }
            };
            return MessageFactory.Attachment(heroCard.ToAttachment()) as Activity;
        }
    }
}
```

Figura 45: Clase “MenuProfesor”

La clase “MenuAlumnos”, Figura 46, contiene la programación del diseño del menú para el usuario “Alumno”.



```
namespace CHAPIE.Common
{
    2 referencias
    public class MenuAlumnos
    {
        2 referencias | 0 excepciones
        public static async Task<DialogTurnResult> ShowOptions(WaterfallStepContext stepContext, CancellationToken cancellationToken)
        {
            await Task.Delay(2000);
            var options = await stepContext.PromptAsync(
                nameof(TextPrompt),
                new PromptOptions
                {
                    Prompt = CreateSuggestedActions(),
                    Style = ListStyle.HeroCard
                },
                cancellationToken
                );
            return options;
        }


        1 referencia | 0 excepciones
        private static Activity CreateSuggestedActions()
        {
            var heroCard = new HeroCard
            {
                Title = "Opciones para Alumnos",
                Subtitle = "CHAPIE 2020",
                Images = new List<CardImage> { new CardImage("http://www.educacionpersonal.com/edupersonal/pluginfile.php/12975/course/section/5741/alumno_estudiando.gif") },
                Buttons = new List<CardAction>()
                {
                  new CardAction(){Title = "Ver Horario Semanal", Value="Ver Horario Semanal", Type=ActionTypes.ImBack},
                  new CardAction(){Title = "Revisar Malla Curricular", Value="Revisar Malla Curricular", Type=ActionTypes.ImBack},
                  new CardAction(){Title = "Ver Materias", Value="Ver Materias", Type=ActionTypes.ImBack},
                  new CardAction(){Title = "Ver Lista de Participantes del Curso", Value="Ver Lista de Participantes del Curso", Type=ActionTypes.ImBack},
                  new CardAction(){Title = "Control de Faltas", Value="Control de Faltas", Type=ActionTypes.ImBack},
                  new CardAction(){Title = "Ir a Biblioteca UDLA", Value="Ir a Biblioteca UDLA", Type=ActionTypes.ImBack},
                  new CardAction(){Title = "Ver Calificaciones", Value="Ver Calificaciones", Type=ActionTypes.ImBack},
                  new CardAction(){Title = "Realizar Busqueda de Informacion", Value="Realizar Busqueda de Informacion", Type=ActionTypes.ImBack},
                  new CardAction(){Title = "Calendario de Actividades", Value="Calendario de Actividades", Type=ActionTypes.ImBack},
                  new CardAction(){Title = "Ir a Foro de Materia", Value="Ir a Foro de Materia", Type=ActionTypes.ImBack},
                  new CardAction(){Title = "Ir a editar Información de Perfil", Value="Ir a editar Información de Perfil", Type=ActionTypes.ImBack},
                  new CardAction(){Title = "Terminar la conversación", Value="Terminar la conversación", Type=ActionTypes.ImBack},
                }
            };
            return MessageFactory.Attachment(heroCard.ToAttachment()) as Activity;
        }
    }
}
```

Figura 46: Clase “MenuAlumnos”

Finalmente, se ejecuta el método “ShowOpcionesChatbot”, Figura 47- Figura 48 de forma continua, el cual se encarga de responder a las solicitudes de los usuarios conforme a la “intención” que se detecta. Las “intenciones” tienen que ser expresiones basadas en las opciones del menú. Una vez detectada la “intención”, llega al programa en forma de variable. Esta variable dentro del método “ShowOpcionesChatbot” sirve para detectar el tipo de acción que debe ejecutarse.

```
private async Task<DialogTurnResult> ShowOpcionesChatbot(WaterfallStepContext stepContext, CancellationToken cancellationToken)
{
    var recognizerResult = await _luisRecognizerService._recognizer.RecognizeAsync(stepContext.Context, cancellationToken);
    var topIntent = recognizerResult.GetTopScoringIntent();
    double score = recognizerResult.GetTopScoringIntent().score;
    if (score > 0.4)
    {
        if (usuario.Equals("Profesor"))
        {
            switch (topIntent.intent)
            {

                case "Opciones":
                    return await IntentOpciones(stepContext, recognizerResult, cancellationToken);

                case "Malla":
                    return await IntentMalla(stepContext, recognizerResult, cancellationToken);

                case "Horario":
                    return await IntentVerHorario(stepContext, recognizerResult, cancellationToken);

                case "Materias":
                    return await IntentVerMaterias(stepContext, recognizerResult, cancellationToken);

                case "listaAlumnos":
                    return await IntentVerListaAlumnos(stepContext, recognizerResult, cancellationToken);

                case "Asistencia":
                    return await IntentRegistrarAsistencia(stepContext, recognizerResult, cancellationToken);

                case "Biblioteca":
                    return await IntentBiblioteca(stepContext, recognizerResult, cancellationToken);

                case "RegistrarCalificacion":
                    return await IntentRegistrarCalificacion(stepContext, recognizerResult, cancellationToken);

                case "Busqueda":
                    return await IntentBusqueda(stepContext, recognizerResult, cancellationToken);

                case "CalendarioActividades":
                    return await IntentCalendarioActividades(stepContext, recognizerResult, cancellationToken);

                case "ForoMateria":
                    return await IntentForoMateria(stepContext, recognizerResult, cancellationToken);

                case "EditarPerfil":
                    return await IntentEditarPerfil(stepContext, recognizerResult, cancellationToken);

                case "TerminarConversacion":
                    await Task.Delay(2000);
                    await stepContext.Context.SendActivityAsync($"Espero haberte ayudado {nombreUsuario}, puedes volverme a escribir si necesitas de mi asistencia", cancellationToken: cancellationToken);
                    await Task.Delay(2000);
                    await stepContext.Context.SendActivityAsync($"Que te valla bien. Hasta Pronto....", cancellationToken: cancellationToken);

                    return await stepContext.EndDialogAsync(cancellationToken: cancellationToken);

                case "None":
                    await IntentNone2(stepContext, recognizerResult, cancellationToken);
                    break;


                default:
                    await stepContext.Context.SendActivityAsync($"No entiendo lo que dices", cancellationToken: cancellationToken);
                    await Task.Delay(2000);
                    await stepContext.Context.SendActivityAsync($"Debes pedirme una de las opciones que te muestro, de acuerdo?", cancellationToken: cancellationToken);
                    break;
            }
        }
        else if (usuario.Equals("Alumno"))
        {
            switch (topIntent.intent)
            {

                case "Opciones":
                    return await IntentOpciones(stepContext, recognizerResult, cancellationToken);

                case "Malla":
                    return await IntentMalla(stepContext, recognizerResult, cancellationToken);

                case "Horario":
                    return await IntentVerHorario(stepContext, recognizerResult, cancellationToken);

                case "Materias":
                    return await IntentVerMaterias(stepContext, recognizerResult, cancellationToken);

                case "listaAlumnos":
                    return await IntentVerListaAlumnos(stepContext, recognizerResult, cancellationToken);

                case "ControlFaltas":
                    return await IntentVerAsistencia(stepContext, recognizerResult, cancellationToken);

                case "Biblioteca":
                    return await IntentBiblioteca(stepContext, recognizerResult, cancellationToken);

                case "VerCalificacion":
                    return await IntentVerCalificacion(stepContext, recognizerResult, cancellationToken);

                case "Busqueda":
                    return await IntentBusqueda(stepContext, recognizerResult, cancellationToken);

                case "CalendarioActividades":
                    return await IntentCalendarioActividades(stepContext, recognizerResult, cancellationToken);

                case "ForoMateria":
                    return await IntentForoMateria(stepContext, recognizerResult, cancellationToken);

                case "EditarPerfil":
                    return await IntentEditarPerfil(stepContext, recognizerResult, cancellationToken);

                case "TerminarConversacion":
                    await Task.Delay(2000);
                    await stepContext.Context.SendActivityAsync($"Espero haberte ayudado {nombreUsuario}, puedes volverme a escribir si necesitas de mi asistencia", cancellationToken: cancellationToken);
                    await Task.Delay(2000);
                    await stepContext.Context.SendActivityAsync($"Que te valla bien. Hasta Pronto....", cancellationToken: cancellationToken);

                    return await stepContext.EndDialogAsync(cancellationToken: cancellationToken);

                case "None":
                    await IntentNone2(stepContext, recognizerResult, cancellationToken);
                    break;


                default:
                    await stepContext.Context.SendActivityAsync($"No entiendo lo que dices", cancellationToken: cancellationToken);
                    await Task.Delay(2000);
                    await stepContext.Context.SendActivityAsync($"Debes pedirme una de las opciones que te muestro, de acuerdo?", cancellationToken: cancellationToken);
                    break;
            }

        }
    }
```

Figura 47: Método “ShowOpcionesChatbot”, parte 1.

```
else
{
    if (stepContext.Context.Activity.Text.Trim() == "Realizar Busqueda de Informacion")
    {
        await stepContext.Context.SendActivityAsync($"Para Realizar una Búsqueda de Información en la Red, por favor escriba: ", cancellationToken: cancellationToken);
        await Task.Delay(1000);
        await stepContext.Context.SendActivityAsync($"\"Deseo buscar información de (Lo que se desea Buscar)\"", cancellationToken: cancellationToken);
        await Task.Delay(1000);
        await stepContext.Context.SendActivityAsync($"o \"Buscar información de (Lo que se desea Buscar)\"", cancellationToken: cancellationToken);
        return await stepContext.PromptAsync(
      nameof(TextPrompt),
      new PromptOptions { Prompt = MessageFactory.Text("") },  //peticion de datos
      cancellationToken
      );

    }

  await stepContext.Context.SendActivityAsync($"No entiendo lo que dices", cancellationToken: cancellationToken);
  await Task.Delay(2000);
  await stepContext.Context.SendActivityAsync($"Debes pedirme una de las opciones que te muestro, de acuerdo?", cancellationToken: cancellationToken);
}
return await stepContext.PromptAsync(
    nameof(TextPrompt),
    new PromptOptions { Prompt = MessageFactory.Text("") },  //peticion de datos
    cancellationToken
    );
}
```

Figura 48: Método “ShowOpcionesChatbot”, parte 2.

3.3.6. PROGRAMACIÓN DE FUNCIONALIDADES DEL MENÚ

Cuando el programa detecta la “intención” de un usuario de ejecutar alguna funcionalidad del menú, el método “ShowOpcionesChatbot” llama a otros métodos que, dependiendo el requerimiento solicitado, realizan diferentes acciones.

El método “IntentOpciones”, Figura 49, dependiendo el tipo de usuario, despliega el menú correspondiente invocando a los métodos “MenuProfesor” y “MenuAlumnos” de la carpeta “common”.

```
private async Task<DialogTurnResult> IntentOpciones(WaterfallStepContext stepContext, RecognizerResult recognizerResult, CancellationToken cancellationToken)
{
    if (usuario.Equals("Profesor"))
    {
        await Task.Delay(1000);
        await stepContext.Context.SendActivityAsync($"Entendido {nombreUsuario} ", cancellationToken: cancellationToken);
        await Task.Delay(1000);
        return await MenuProfesor.ShowOptions(stepContext, cancellationToken);
    }

    if (usuario.Equals("Alumno"))
    {
        await Task.Delay(1000);
        await stepContext.Context.SendActivityAsync($"Entendido {nombreUsuario} ", cancellationToken: cancellationToken);
        await Task.Delay(1000);
        return await MenuAlumnos.ShowOptions(stepContext, cancellationToken);
    }
    return await stepContext.PromptAsync(
        nameof(TextPrompt),
        new PromptOptions { Prompt = MessageFactory.Text("") },  //peticion de datos
        cancellationToken
        );

}
```

Figura 49: Método “IntentOpciones”.

El método “IntentVerHorario”, Figura 50, llama a la clase “AttachmentCard.cs” de la carpeta “Common”, específicamente al método “GetHorario”, Figura 51.

```
private async Task<DialogTurnResult> IntentVerHorario(WaterfallStepContext stepContext, RecognizerResult recognizerResult, CancellationToken cancellationToken)
{
    await stepContext.Context.SendActivityAsync(AttachmentCard.GetHorario(), cancellationToken);
    await stepContext.Context.SendActivityAsync("Para mayor detalle Visita: http://ssb.udla.edu.ec:9010/PROD/bwskfshd.P_CrseSchd", cancellationToken: cancellationToken);
    return await stepContext.PromptAsync (nameof(TextPrompt),
         new PromptOptions { Prompt = MessageFactory.Text("Dime si te puedo ayudar en otra cosa.....") },  //peticion de datos
         cancellationToken
         );
}
```

Figura 50: Método “IntentVerHorario”.

```
public static Activity GetHorario()
{
    var imagecard = new Attachment();
    imagecard.Name = "imagen";
    imagecard.ContentType = "image/png";
    imagecard.ContentUrl = "https://raw.githubusercontent.com/JhulioDavid/Chatbot/master/Horario.png";

    var reply = MessageFactory.Attachment(imagecard);
    return reply as Activity;
}
```

Figura 51: Método “GetHorario” de la clase “AttachmentCard.cs”.

El método “IntentVerHorario2”, Figura 52, llama a la clase “AttachmentCard.cs” de la carpeta “Common”, específicamente al método “GetHorario2”, Figura 53.

```
private async Task<DialogTurnResult> IntentVerHorario2(WaterfallStepContext stepContext, RecognizerResult recognizerResult, CancellationToken cancellationToken)
{
    await stepContext.Context.SendActivityAsync(AttachmentCard.GetHorario2(), cancellationToken);
    await stepContext.Context.SendActivityAsync("Para mayor detalle Visita: http://ssb.udla.edu.ec:9010/PROD/bwskfshd.P_CrseSchd", cancellationToken: cancellationToken);
    return await stepContext.PromptAsync(nameof(TextPrompt),
        new PromptOptions { Prompt = MessageFactory.Text("Dime si te puedo ayudar en otra cosa.....") },  //peticion de datos
        cancellationToken
        );
}
```

Figura 52: Método “IntentVerHorario2”.

```
public static Activity GetHorario2()
{
    var imagecard = new Attachment();
    imagecard.Name = "imagen";
    imagecard.ContentType = "image/png";
    imagecard.ContentUrl = "https://raw.githubusercontent.com/JhulioDavid/Chatbot/master/Horario.png";

    var reply = MessageFactory.Attachment(imagecard);
    return reply as Activity;
}
```

Figura 53: Método “GetHorario2” de la clase “AttachmentCard.cs”.

El método “IntentMalla”, Figura 54, llama a la clase “AttachmentCard.cs” de la carpeta “Common”, específicamente los métodos “GetMalla” y “GetMalla2”, Figura 55.

```
private async Task<DialogTurnResult> IntentMalla(WaterfallStepContext stepContext, RecognizerResult recognizerResult, CancellationToken cancellationToken)
{
    // await MallaCurricular.ShowOptions(stepContext,cancellationToken);
    await stepContext.Context.SendActivityAsync(AttachmentCard.GetMalla2(), cancellationToken);
    await stepContext.Context.SendActivityAsync(AttachmentCard.GetMalla(), cancellationToken);
    return await stepContext.PromptAsync(nameof(TextPrompt),
        new PromptOptions { Prompt = MessageFactory.Text("Dime si te puedo ayudar en otra cosa.....") },  //peticion de datos
        cancellationToken
        );
}
```

Figura 54: Método “IntentMalla”.

```
public static Activity GetMalla()
{
    var documentocard = new Attachment();
    documentocard.Name = "malla_curricular";
    documentocard.ContentType = "application/pdf";
    documentocard.ContentUrl = "http://www4.udla.edu.ec/MallaCurricular/pdf/mallas_2016/651_IngElectrRedes.pdf";

    var reply = MessageFactory.Attachment(documentocard);
    return reply as Activity;
}

1 referencia | 0 excepciones
public static Activity GetMalla2()
{
    var heroCard = new HeroCard
    {
        Title = "Malla Curricular",
        Subtitle = "Electrónica y Redes de Información",
        Images = new List<CardImage> { new CardImage("https://raw.githubusercontent.com/JhulioDavid/Chatbot/maste
    };
    return MessageFactory.Attachment(heroCard.ToAttachment()) as Activity;
}
```

Figura 55: Método “GetMalla” y “GetMalla2”, de la clase “AttachmentCard.cs”.

El método “IntentVerMaterias”, Figura 56, llama a la clase “AttachmentCard.cs” de la carpeta “Common”, específicamente al método “GetMaterias”, Figura 57.

```
private async Task<DialogTurnResult> IntentVerMaterias(WaterfallStepContext stepContext, RecognizerResult recognizerResult, CancellationToken cancellationT
{
    await stepContext.Context.SendActivityAsync(AttachmentCard.GetMaterias(), cancellationToken);
    await stepContext.Context.SendActivityAsync("Para mayor detalle Visita: https://autoservicio.udla.edu.ec/StudentSSB/ssb/studentProfile/studentProfile",
    return await stepContext.PromptAsync(nameof(TextPrompt),
         new PromptOptions { Prompt = MessageFactory.Text("Dime si te puedo ayudar en otra cosa.....") },  //peticion de datos
         cancellationToken
         );
}
```

Figura 56: Método “IntentVerMaterias”.

```
public static Activity GetMaterias()
{
    var imagecard = new Attachment();
    imagecard.Name = "imagen";
    imagecard.ContentType = "image/png";
    imagecard.ContentUrl = "https://raw.githubusercontent.com/JhulioDavid/Chatbot/master/Materias.png";

    var reply = MessageFactory.Attachment(imagecard);
    return reply as Activity;
}
```

Figura 57: Método “GetMaterias”, de la clase “AttachmentCard.cs”.

El método “IntentVerMaterias2”, Figura 58, llama a la clase “AttachmentCard.cs” de la carpeta “Common”, específicamente al método “GetMaterias2”, Figura 59.

```
private async Task<DialogTurnResult> IntentVerMaterias2(WaterfallStepContext stepContext, RecognizerResult recognizerResult, CancellationToken cancellationToken)
{
    await stepContext.Context.SendActivityAsync(AttachmentCard.GetMaterias2(), cancellationToken);
    await stepContext.Context.SendActivityAsync("Para mayor detalle Visita: http://ssb.udla.edu.ec:9010/PROD/bwlkifac.P_FacDaySched", cancellationToken: cancellationToken);
    return await stepContext.PromptAsync(nameof(TextPrompt),
        new PromptOptions { Prompt = MessageFactory.Text("Dime si te puedo ayudar en otra cosa.....") },  //peticion de datos
        cancellationToken
        );
}
```

Figura 58: Método “IntentVerMaterias2”.

```
public static Activity GetMaterias2()
{
    var imagecard = new Attachment();
    imagecard.Name = "imagen";
    imagecard.ContentType = "image/png";
    imagecard.ContentUrl = "https://raw.githubusercontent.com/JhulioDavid/Chatbot/master/Materias.png";

    var reply = MessageFactory.Attachment(imagecard);
    return reply as Activity;
}
```

Figura 59: Método “GetMaterias2”, de la clase “AttachmentCard.cs”.

El método “IntentVerListaAlumnos”, Figura 60, llama a la clase “ListaAlumnos.cs”, Figura 61, de la carpeta “Common”.

```
private async Task<DialogTurnResult> IntentVerListaAlumnos(WaterfallStepContext stepContext, RecognizerResult recognizerResult, CancellationToken cancellationToken)
{
    return await ListaAlumnos.ShowOptions(stepContext, cancellationToken);
}
```

Figura 60: Método “IntentVerListaAlumnos”.

```
namespace CHAPIE.Common
{
    1 referencia
    public class ListaAlumnos
    {
        1 referencia | 0 excepciones
        public static async Task<DialogTurnResult> ShowOptions(WaterfallStepContext stepContext, CancellationToken cancellationToken)
        {
            await Task.Delay(2000);
            var options = await stepContext.PromptAsync(
                nameof(TextPrompt),
                new PromptOptions
                {
                    Prompt = CreateSuggestedActions(),
                    Style = ListStyle.HeroCard
                },
                cancellationToken
                );
            await stepContext.Context.SendActivityAsync("Dime si te puedo ayudar en otra cosa.....", cancellationToken: cancellationToken);

            return options;
        }


        1 referencia | 0 excepciones
        private static Activity CreateSuggestedActions()
        {
            var heroCard = new HeroCard
            {
                Title = "Listado de Alumnos",
                Subtitle = "Seleccione una Materia",
                Buttons = new List<CardAction>()
                {
                  new CardAction(){Title = "Aplic. y Serv Convergentes", Value="https://aulasvirtuales.udla.edu.ec/udlapresencial/user/index.php?contextid=843577", Type=ActionTypes.OpenUrl},
                  new CardAction(){Title = "Certificación de Redes", Value="https://aulasvirtuales.udla.edu.ec/udlapresencial/user/index.php?contextid=1004929", Type=ActionTypes.OpenUrl},
                  new CardAction(){Title = "TIT", Value="https://aulasvirtuales.udla.edu.ec/udlapresencial/user/index.php?contextid=1087849", Type=ActionTypes.OpenUrl},

                }
            };
            return MessageFactory.Attachment(heroCard.ToAttachment()) as Activity;
        }
    }
}
```

Figura 61: Clase “ListaAlumnos”, de la carpera “Common”.

El método “IntentVerListaAlumnos2”, Figura 62, llama a la clase “ListaAlumnos2.cs”, Figura 63, de la carpeta “Common”.

```
private async Task<DialogTurnResult> IntentVerListaAlumnos2(WaterfallStepContext stepContext, RecognizerResult recognizerResult, CancellationToken cancellationToken)
{
    return await ListaAlumnos2.ShowOptions(stepContext, cancellationToken);
}
```

Figura 62: Método “IntentVerListaAlumnos2”.

```
public class ListaAlumnos2
{
    1 referencia | 0 excepciones
    public static async Task<DialogTurnResult> ShowOptions(WaterfallStepContext stepContext, CancellationToken cancellationToken)
    {
        await Task.Delay(2000);
        var options = await stepContext.PromptAsync(
            nameof(TextPrompt),
            new PromptOptions
            {
                Prompt = CreateSuggestedActions(),
                Style = ListStyle.HeroCard
            },
            cancellationToken
            );
        await stepContext.Context.SendActivityAsync("Dime si te puedo ayudar en otra cosa.....", cancellationToken: cancellationToken);

        return options;
    }

    1 referencia | 0 excepciones
    private static Activity CreateSuggestedActions()
    {
        var heroCard = new HeroCard
        {
            Title = "Listado de Alumnos",
            Subtitle = "Seleccione una Materia",
            Buttons = new List<CardAction>()
            {
              new CardAction(){Title = "Redes I", Value="https://aulasvirtuales.udla.edu.ec/udlapresencial/course/view.php?id=7680", Type=ActionTypes.OpenUrl},
              new CardAction(){Title = "Calidad de Servicio", Value="https://aulasvirtuales.udla.edu.ec/udlapresencial/course/view.php?id=7676", Type=ActionTypes.OpenUrl},

            }
        };
        return MessageFactory.Attachment(heroCard.ToAttachment()) as Activity;
    }
}
```

Figura 63: Clase “ListaAlumnos2”, de la carpera “Common”.

El método “IntentRegistrarAsistencia”, Figura 64, llama a la clase “Asistencia.cs”, Figura 65, de la carpeta “Common”.

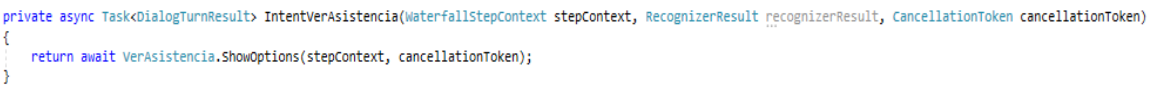
```
private async Task<DialogTurnResult> IntentRegistrarAsistencia(WaterfallStepContext stepContext, RecognizerResult recognizerResult, CancellationToken cancellationToken)
{
    return await Asistencia.ShowOptions(stepContext, cancellationToken);
}
```

Figura 64: Método “IntentRegistrarAsistencia”.

```
namespace CHAPIE.Common
{
    1 referencia
    public class Asistencia
    {
        1 referencia | 0 excepciones
        public static async Task<DialogTurnResult> ShowOptions(WaterfallStepContext stepContext, CancellationToken cancellationToken)
        {
            await Task.Delay(2000);
            var options = await stepContext.PromptAsync(
                nameof(TextPrompt),
                new PromptOptions
                {
                    Prompt = CreateSuggestedActions(),
                    Style = ListStyle.HeroCard
                },
                cancellationToken
                );
            await stepContext.Context.SendActivityAsync("Dime si te puedo ayudar en otra cosa.....", cancellationToken: cancellationToken);

            return options;
        }


        1 referencia | 0 excepciones
        private static Activity CreateSuggestedActions()
        {
            var heroCard = new HeroCard
            {
                Title = "Registro de Asistencia",
                Subtitle = "Seleccione una Materia",
                Buttons = new List<CardAction>()
                {
                  new CardAction(){Title = "Aplic. y Serv Convergentes", Value="https://aulasvirtuales.udla.edu.ec/udlapresencial/user/index.php?contextid=843577", Type=ActionTypes.OpenUrl},
                  new CardAction(){Title = "Certificación de Redes", Value="https://aulasvirtuales.udla.edu.ec/udlapresencial/user/index.php?contextid=1004929", Type=ActionTypes.OpenUrl},
                  new CardAction(){Title = "TIT", Value="https://aulasvirtuales.udla.edu.ec/udlapresencial/user/index.php?contextid=1007849", Type=ActionTypes.OpenUrl},

                }
            };
            return MessageFactory.Attachment(heroCard.ToAttachment()) as Activity;
        }
    }
```

Figura 65: Clase “Asistencia”, de la carpeta “Common”.

El método “IntentVerAsistencia”, Figura 66, llama a la clase “VerAsistencia.cs”, Figura 67, de la carpeta “Common”.



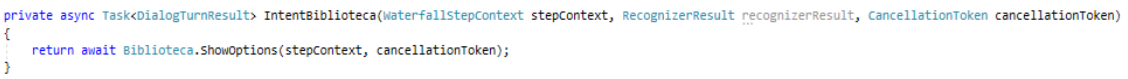
```
private async Task<DialogTurnResult> IntentVerAsistencia(WaterfallStepContext stepContext, RecognizerResult recognizerResult, CancellationToken cancellationToken)
{
    return await VerAsistencia.ShowOptions(stepContext, cancellationToken);
}
```

Figura 66: Método “IntentVerAsistencia”.

```
namespace CHAPIE.Common
{
    1 referencia
    public class VerAsistencia
    {
        1 referencia | 0 excepciones
        public static async Task<DialogTurnResult> ShowOptions(WaterfallStepContext stepContext, CancellationToken cancellationToken)
        {
            await Task.Delay(2000);
            var options = await stepContext.PromptAsync(
                nameof(TextPrompt),
                new PromptOptions
                {
                    Prompt = CreateSuggestedActions(),
                    Style = ListStyle.HeroCard
                },
                cancellationToken
                );
            await stepContext.Context.SendActivityAsync("Dime si te puedo ayudar en otra cosa.....", cancellationToken: cancellationToken);

            return options;
        }


        1 referencia | 0 excepciones
        private static Activity CreateSuggestedActions()
        {
            var heroCard = new HeroCard
            {
                Title = "Control de Faltas de Alumnos",
                Subtitle = "Click en el Botón",
                Buttons = new List<CardAction>()
                {
                  new CardAction(){Title = "Control de Faltas", Value="https://infoestu.udla.edu.ec/Estudiante/ControlFaltas", Type=ActionTypes.OpenUrl},
                }
            };
            return MessageFactory.Attachment(heroCard.ToAttachment()) as Activity;
        }
    }
}
```

Figura 67: Clase “VerAsistencia”, de la carpeta “Common”.

El método “IntentBiblioteca”, Figura 68, llama a la clase “Biblioteca.cs”, Figura 69, de la carpeta “Common”.



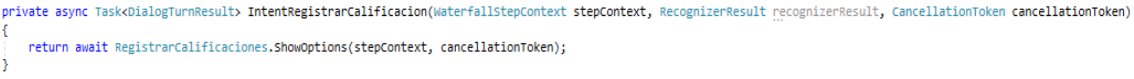
```
private async Task<DialogTurnResult> IntentBiblioteca(WaterfallStepContext stepContext, RecognizerResult recognizerResult, CancellationToken cancellationToken)
{
    return await Biblioteca.ShowOptions(stepContext, cancellationToken);
}
```

Figura 68: Método “IntentBiblioteca”.

```
amespace CHAPIE.Common

public class Biblioteca
{
    1 referencia | 0 excepciones
    public static async Task<DialogTurnResult> ShowOptions(WaterfallStepContext stepContext, CancellationToken cancellationToken)
    {
        await Task.Delay(2000);
        var options = await stepContext.PromptAsync(
            nameof(TextPrompt),
            new PromptOptions
            {
                Prompt = CreateSuggestedActions(),
                Style = ListStyle.HeroCard
            },
            cancellationToken
            );
        await stepContext.Context.SendActivityAsync("Dime si te puedo ayudar en otra cosa.....", cancellationToken: cancellationToken);

        return options;
    }

    1 referencia | 0 excepciones
    private static Activity CreateSuggestedActions()
    {
        var heroCard = new HeroCard
        {
            Title = "Biblioteca UDLA",
            Subtitle = "Click en el Botón",
            Buttons = new List<CardAction>()
            {
              new CardAction(){Title = "Biblioteca", Value="http://biblioteca.udla.edu.ec/client/es_EC/default/", Type=ActionTypes.OpenUrl},
            }
        };
        return MessageFactory.Attachment(heroCard.ToAttachment()) as Activity;
    }
}
```

Figura 69: Clase “Biblioteca”, de la carpeta “Common”.

El método “IntentRegistrarCalificacion”. Figura 70 llama a la clase “RegistrarCalificaciones.cs”, Figura 71, de la carpeta “Common”.



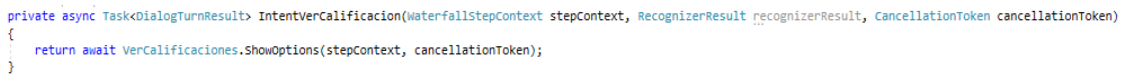
```
private async Task<DialogTurnResult> IntentRegistrarCalificacion(WaterfallStepContext stepContext, RecognizerResult recognizerResult, CancellationToken cancellationToken)
{
    return await RegistrarCalificaciones.ShowOptions(stepContext, cancellationToken);
}
```

Figura 70: Método “IntentRegistrarCalificacion”.

```
namespace CHAPIE.Common
{
    1 referencia
    public class RegistrarCalificaciones
    {
        1 referencia | 0 excepciones
        public static async Task<DialogTurnResult> ShowOptions(WaterfallStepContext stepContext, CancellationToken cancellationToken)
        {
            await Task.Delay(2000);
            var options = await stepContext.PromptAsync(
                nameof(TextPrompt),
                new PromptOptions
                {
                    Prompt = CreateSuggestedActions(),
                    Style = ListStyle.HeroCard
                },
                cancellationToken
                );
            await stepContext.Context.SendActivityAsync("Dime si te puedo ayudar en otra cosa.....", cancellationToken: cancellationToken);

            return options;
        }


        1 referencia | 0 excepciones
        private static Activity CreateSuggestedActions()
        {
            var heroCard = new HeroCard
            {
                Title = "Registro de Calificaciones",
                Subtitle = "Seleccione una Materia",
                Buttons = new List<CardAction>()
                {
                  new CardAction(){Title = "Aplic. y Serv Convergentes", Value="https://aulasvirtuales.udla.edu.ec/udlapresencial/course/user.php?mode=grade&id=6652&user=18449", Type=ActionTypes.OpenUrl},
                  new CardAction(){Title = "Certificación de Redes", Value="https://aulasvirtuales.udla.edu.ec/udlapresencial/course/user.php?mode=grade&id=8219&user=18449", Type=ActionTypes.OpenUrl},
                  new CardAction(){Title = "TIT", Value="https://aulasvirtuales.udla.edu.ec/udlapresencial/course/user.php?mode=grade&id=8612&user=18449", Type=ActionTypes.OpenUrl},

                }
            };
            return MessageFactory.Attachment(heroCard.ToAttachment()) as Activity;
        }
    }
}
```

Figura 71: Clase “RegistrarCalificaciones”, de la carpeta “Common”.

El método “IntentVerCalificacion”, Figura 72, llama a la clase “VerCalificaciones.cs”, Figura 73, de la carpeta “Common”.



```
private async Task<DialogTurnResult> IntentVerCalificacion(WaterfallStepContext stepContext, RecognizerResult recognizerResult, CancellationToken cancellationToken)
{
    return await VerCalificaciones.ShowOptions(stepContext, cancellationToken);
}
```

Figura 72: Método “IntentVerCalificacion”.

```
namespace CHAPIE.Common
{
    1 referencia
    public class VerCalificaciones
    {
        1 referencia | 0 excepciones
        public static async Task<DialogTurnResult> ShowOptions(WaterfallStepContext stepContext, CancellationToken cancellationToken)
        {
            await Task.Delay(2000);
            var options = await stepContext.PromptAsync(
                nameof(TextPrompt),
                new PromptOptions
                {
                    Prompt = CreateSuggestedActions(),
                    Style = ListStyle.HeroCard
                },
                cancellationToken
                );
            await stepContext.Context.SendActivityAsync("Dime si te puedo ayudar en otra cosa.....", cancellationToken: cancellationToken);

            return options;
        }


        1 referencia | 0 excepciones
        private static Activity CreateSuggestedActions()
        {
            var heroCard = new HeroCard
            {
                Title = "Ver Registro de Calificaciones",
                Subtitle = "Seleccione una Materia",
                Buttons = new List<CardAction>()
                {
                  new CardAction(){Title = "Aplic. y Serv Convergentes", Value="https://aulasvirtuales.udla.edu.ec/udlapresencial/course/user.php?mode=grade&id=6652&user=18449", Type=ActionTypes.OpenUrl},
                  new CardAction(){Title = "Certificación de Redes", Value="https://aulasvirtuales.udla.edu.ec/udlapresencial/course/user.php?mode=grade&id=8219&user=18449", Type=ActionTypes.OpenUrl},
                  new CardAction(){Title = "TIT", Value="https://aulasvirtuales.udla.edu.ec/udlapresencial/course/user.php?mode=grade&id=8612&user=18449", Type=ActionTypes.OpenUrl},

                }
            };
            return MessageFactory.Attachment(heroCard.ToAttachment()) as Activity;
        }
    }
}
```

Figura 73: Clase “VerCalificaciones”, de la carpeta “Common”.

El método “IntentBusqueda”, Figura 74, llama a la clase “Busqueda.cs”, Figura 75, de la carpeta “Common”.

```
private async Task<DialogTurnResult> IntentBusqueda(WaterfallStepContext stepContext, RecognizerResult recognizerResult, CancellationToken cancellationToken)
{

    var busqueda="";
    try
    {
        busqueda = recognizerResult.Entities.GetValue("buscar").ToString();

        busqueda = busqueda.Replace("[", "");
        busqueda = busqueda.Replace("]", "");
        busqueda = busqueda.Replace("\"", "");

        busqueda = busqueda.Trim();
    }
    catch (Exception)
    { }

    if (busqueda.Equals(""))
    {
        busqueda = stepContext.Context.Activity.Text.ToLower();
        busqueda=busqueda.Replace("buscar", "");
        busqueda = busqueda.Replace("sobre", "");
        busqueda = busqueda.Replace("acerca", "");

        busqueda = busqueda.Trim();
        if (busqueda.Equals("")|| busqueda.Equals("deseo información de"))
        {
            await stepContext.Context.SendActivityAsync($"Para Realizar una Búsqueda de Información en la Red, por favor escriba: ", cancellationToken: cancellationToken);
            await Task.Delay(1000);
            await stepContext.Context.SendActivityAsync($"\"Deseo buscar información de (Lo que se desea Buscar)\"", cancellationToken: cancellationToken);
            await Task.Delay(1000);
            await stepContext.Context.SendActivityAsync($"o \"Buscar información de (Lo que se desea Buscar)\"", cancellationToken: cancellationToken);
            return await stepContext.PromptAsync(
         nameof(TextPrompt),
         new PromptOptions { Prompt = MessageFactory.Text("") },  //peticion de datos
         cancellationToken
         );
        }
        await stepContext.Context.SendActivityAsync($"Listo...Te puedo ayudar en otra cosa?", cancellationToken: cancellationToken);


    }
    else
    {
        await stepContext.Context.SendActivityAsync($"Listo...Te puedo ayudar en otra cosa?", cancellationToken: cancellationToken);

    }
    return await Busqueda.ShowOptions(stepContext, cancellationToken,busqueda);


}
```

Figura 74: Método “IntentVerBusqueda”.

```
namespace CHAPIE.Common
{
    1 referencia
    public class Busqueda
    {
        1 referencia | 0 excepciones
        public static async Task<DialogTurnResult> ShowOptions(WaterfallStepContext stepContext, CancellationToken cancellationToken, string text)
        {
            var options = await stepContext.PromptAsync(
                nameof(TextPrompt),
                new PromptOptions
                {
                    Prompt = CreateSuggestedActions(text),
                    Style = ListStyle.HeroCard
                },
                cancellationToken
                );

            return options;
        }


        1 referencia | 0 excepciones
        private static Activity CreateSuggestedActions(string text)
        {
            var reply = MessageFactory.Text("");
            reply.SuggestedActions = new SuggestedActions()
            {
                Actions = new List<CardAction>()
                {
                    new CardAction(){Title = "Mostrar Búsqueda", Value="http://www.google.com/search?q="+text, Type=ActionTypes.OpenUrl},
                }
            };

            return reply;
        }
    }
}
```

Figura 75: Clase “Busqueda”, de la carpeta “Common”.

El método “IntentCalendarioActividades”, Figura 76, llama a las clases “CalendarioActividades.cs”, Figura 77, y “CalendarioActividades2.cs”, Figura 78, de la carpeta “Common”.

```
private async Task<DialogTurnResult> IntentCalendarioActividades(WaterfallStepContext stepContext, RecognizerResult recognizerResult, CancellationToken cancellationToken)
{
    if (usuario.Equals("Profesor"))
    {
        return await CalendarioActividades2.ShowOptions(stepContext, cancellationToken);
    }

    if (usuario.Equals("Alumno"))
    {
        return await CalendarioActividades.ShowOptions(stepContext, cancellationToken);
    }
    return await CalendarioActividades.ShowOptions(stepContext, cancellationToken);
}
```

Figura 76: Método “IntentCalendarioActividades”.

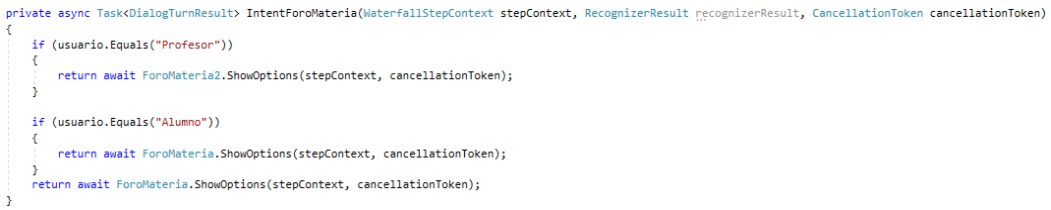
```
namespace CHAPIE.Common
{
    1 referencia
    public class CalendarioActividades
    {
        1 referencia | 0 excepciones
        public static async Task<DialogTurnResult> ShowOptions(WaterfallStepContext stepContext, CancellationToken cancellationToken)
        {
            await Task.Delay(2000);
            var options = await stepContext.PromptAsync(
                nameof(TextPrompt),
                new PromptOptions
                {
                    Prompt = CreateSuggestedActions(),
                    Style = ListStyle.HeroCard
                },
                cancellationToken
                );
            await stepContext.Context.SendActivityAsync("Dime si te puedo ayudar en otra cosa.....", cancellationToken: cancellationToken);

            return options;
        }


        1 referencia | 0 excepciones
        private static Activity CreateSuggestedActions()
        {
            var heroCard = new HeroCard
            {
                Title = "Calendario de Actividades",
                Subtitle = "Click en el Botón",
                Buttons = new List<CardAction>()
                {
                  new CardAction(){Title = "Calendario de Actividades", Value="https://aulasvirtuales.udla.edu.ec/udlapresencial/calendar/view.php?view=month", Type=ActionTypes.OpenUrl},
                }
            };
            return MessageFactory.Attachment(heroCard.ToAttachment()) as Activity;
        }
    }
}
```

Figura 77: Clase “CalendarioActividades”, de la carpeta “Common”.

```
namespace CHAPIE.Common
{
    1 referencia
    public class CalendarioActividades2
    {
        1 referencia | 0 excepciones
        public static async Task<DialogTurnResult> ShowOptions(WaterfallStepContext stepContext, CancellationToken cancellationToken)
        {
            await Task.Delay(2000);
            var options = await stepContext.PromptAsync(
                nameof(TextPrompt),
                new PromptOptions
                {
                    Prompt = CreateSuggestedActions(),
                    Style = ListStyle.HeroCard
                },
                cancellationToken
                );
            await stepContext.Context.SendActivityAsync("Dime si te puedo ayudar en otra cosa.....", cancellationToken: cancellationToken);

            return options;
        }


        1 referencia | 0 excepciones
        private static Activity CreateSuggestedActions()
        {
            var heroCard = new HeroCard
            {
                Title = "Calendario de Actividades",
                Subtitle = "Click en el Botón",
                Buttons = new List<CardAction>()
                {
                  new CardAction(){Title = "Calendario de Actividades", Value="https://aulasvirtuales.udla.edu.ec/udlapresencial/calendar/view.php?view=month", Type=ActionTypes.OpenUrl},
                }
            };
            return MessageFactory.Attachment(heroCard.ToAttachment()) as Activity;
        }
    }
}
```

Figura 78: Clase “CalendarioActividades2”, de la carpeta “Common”.

El método “IntentForoMateria”, Figura 79, llama a la clase “ForoMateria.cs”, Figura 80, y “ForoMateria2.cs”, Figura 81, de la carpeta “Common”.



```
private async Task<DialogTurnResult> IntentForoMateria(WaterfallStepContext stepContext, RecognizerResult recognizerResult, CancellationToken cancellationToken)
{
    if (usuario.Equals("Profesor"))
    {
        return await ForoMateria2.ShowOptions(stepContext, cancellationToken);
    }

    if (usuario.Equals("Alumno"))
    {
        return await ForoMateria.ShowOptions(stepContext, cancellationToken);
    }
    return await ForoMateria.ShowOptions(stepContext, cancellationToken);
}
```

Figura 79: Método “IntentForoMateria”.

```
namespace CHAPIE.Common
{
    1 referencia
    public class ForoMateria
    {

        1 referencia | 0 excepciones
        public static async Task<DialogTurnResult> ShowOptions(WaterfallStepContext stepContext, CancellationToken cancellationToken)
        {
            await Task.Delay(2000);
            var options = await stepContext.PromptAsync(
                nameof(TextPrompt),
                new PromptOptions
                {
                    Prompt = CreateSuggestedActions(),
                    Style = ListStyle.HeroCard
                },
                cancellationToken
                );
            await stepContext.Context.SendActivityAsync("Dime si te puedo ayudar en otra cosa.....", cancellationToken: cancellationToken);

            return options;
        }


        1 referencia | 0 excepciones
        private static Activity CreateSuggestedActions()
        {
            var heroCard = new HeroCard
            {
                Title = "Foro",
                Subtitle = "Seleccione una Materia",
                Buttons = new List<CardAction>()
                {
                  new CardAction(){Title = "Aplic. y Serv Convergentes", Value="https://aulasvirtuales.udla.edu.ec/udlapresencial/mod/forum/view.php?id=676437", Type=ActionTypes.OpenUrl},
                  new CardAction(){Title = "Certificación de Redes", Value="https://aulasvirtuales.udla.edu.ec/udlapresencial/mod/forum/view.php?id=826579", Type=ActionTypes.OpenUrl},
                  new CardAction(){Title = "TIT", Value="https://aulasvirtuales.udla.edu.ec/udlapresencial/mod/forum/view.php?id=904508", Type=ActionTypes.OpenUrl},

                }
            };
            return MessageFactory.Attachment(heroCard.ToAttachment()) as Activity;
        }
    }
}
```

Figura 80: Clase “ForoMateria”, de la carpeta “Common”.

```
namespace CHAPIE.Common
{
    1 referencia
    public class ForoMateria2
    {
        1 referencia | 0 excepciones
        public static async Task<DialogTurnResult> ShowOptions(WaterfallStepContext stepContext, CancellationToken cancellationToken)
        {
            await Task.Delay(2000);
            var options = await stepContext.PromptAsync(
                nameof(TextPrompt),
                new PromptOptions
                {
                    Prompt = CreateSuggestedActions(),
                    Style = ListStyle.HeroCard
                },
                cancellationToken
                );
            await stepContext.Context.SendActivityAsync("Dime si te puedo ayudar en otra cosa.....", cancellationToken: cancellationToken);

            return options;
        }


        1 referencia | 0 excepciones
        private static Activity CreateSuggestedActions()
        {
            var heroCard = new HeroCard
            {
                Title = "Foro",
                Subtitle = "Seleccione una Materia",
                Buttons = new List<CardAction>()
                {
                  new CardAction(){Title = "Redes I", Value="https://aulasvirtuales.udla.edu.ec/udlapresencial/mod/forum/view.php?id=676437", Type=ActionTypes.OpenUrl},
                  new CardAction(){Title = "Calidad de Servicio", Value="https://aulasvirtuales.udla.edu.ec/udlapresencial/mod/forum/view.php?id=826579", Type=ActionTypes.OpenUrl},

                }
            };
            return MessageFactory.Attachment(heroCard.ToAttachment()) as Activity;
        }
    }
}
```

Figura 81: Clase “ForoMateria2”, de la carpeta “Common”.

El método “IntentEditarPerfil”, Figura 82, llama a la clase “EditarPerfil.cs”, Figura 83, y la clase “EditarPerfil2.cs”, Figura 84, de la carpeta “Common”.

```
private async Task<DialogTurnResult> IntentEditarPerfil(WaterfallStepContext stepContext, RecognizerResult recognizerResult, CancellationToken cancellationToken)
{
    if (usuario.Equals("Profesor"))
    {
        return await EditarPerfil2.ShowOptions(stepContext, cancellationToken);
    }

    if (usuario.Equals("Alumno"))
    {
        return await EditarPerfil.ShowOptions(stepContext, cancellationToken);
    }
    return await EditarPerfil.ShowOptions(stepContext, cancellationToken);
}
```

Figura 82: Método “IntentEditarPerfil”.

```
namespace CHAPIE.Common
{
    1 referencia
    public class EditarPerfil
    {
        1 referencia | 0 excepciones
        public static async Task<DialogTurnResult> ShowOptions(WaterfallStepContext stepContext, CancellationToken cancellationToken)
        {
            await Task.Delay(2000);
            var options = await stepContext.PromptAsync(
                nameof(TextPrompt),
                new PromptOptions
                {
                    Prompt = CreateSuggestedActions(),
                    Style = ListStyle.HeroCard
                },
                cancellationToken
                );
            await stepContext.Context.SendActivityAsync("Dime si te puedo ayudar en otra cosa.....", cancellationToken: cancellationToken);

            return options;
        }


        1 referencia | 0 excepciones
        private static Activity CreateSuggestedActions()
        {
            var heroCard = new HeroCard
            {
                Title = "Editar Perfil Personal",
                Subtitle = "Click en el Botón",
                Buttons = new List<CardAction>()
                {
                  new CardAction(){Title = "Editar Perfil Personal", Value="https://aulasvirtuales.udla.edu.ec/udlapresencial/user/profile.php?id=18449", Type=ActionTypes.OpenUrl},
                }
            };
            return MessageFactory.Attachment(heroCard.ToAttachment()) as Activity;
        }
    }
}
```

Figura 83: Clase “EditarPerfil”, de la carpeta “Common”.

```
namespace CHAPIE.Common
{
    1 referencia
    public class EditarPerfil2
    {
        1 referencia | 0 excepciones
        public static async Task<DialogTurnResult> ShowOptions(WaterfallStepContext stepContext, CancellationToken cancellationToken)
        {
            await Task.Delay(2000);
            var options = await stepContext.PromptAsync(
                nameof(TextPrompt),
                new PromptOptions
                {
                    Prompt = CreateSuggestedActions(),
                    Style = ListStyle.HeroCard
                },
                cancellationToken
                );
            await stepContext.Context.SendActivityAsync("Dime si te puedo ayudar en otra cosa.....", cancellationToken: cancellationToken);

            return options;
        }


        1 referencia | 0 excepciones
        private static Activity CreateSuggestedActions()
        {
            var heroCard = new HeroCard
            {
                Title = "Editar Perfil Personal",
                Subtitle = "Click en el Botón",
                Buttons = new List<CardAction>()
                {
                  new CardAction(){Title = "Editar Perfil Personal", Value="https://aulasvirtuales.udla.edu.ec/udlapresencial/user/profile.php?id=18449", Type=ActionTypes.OpenUrl},
                }
            };
            return MessageFactory.Attachment(heroCard.ToAttachment()) as Activity;
        }
    }
}
```

Figura 84: Clase “EditarPerfil2”, de la carpeta “Common”.

El método “IntentNone2”, Figura 85, muestra un mensaje de error a los usuarios en caso de no escribir correctamente una petición a “CHAPIE”.

```
private async Task IntentNone2(WaterfallStepContext stepContext, RecognizerResult recognizerResult, CancellationToken cancellationToken)
{
    await stepContext.Context.SendActivityAsync($"No entiendo lo que dices", cancellationToken: cancellationToken);
    await Task.Delay(2000);
    await stepContext.Context.SendActivityAsync($"Debes pedirme una de las opciones que te muestro, de acuerdo?", cancellationToken: cancellationToken);

}
```

Figura 85: Método “IntentNone2”.

Finalmente, dentro de cada menú de usuario existen 2 “cases”, que no requieren ningún método programado.

El “case” “TerminarConversacion”, Figura 86, muestra un mensaje de despedida para los usuarios e invoca al método “SendActivityAsync” de la biblioteca “DialogContext”, el cual se encarga de terminar la conversación con el usuario.

```
case "TerminarConversacion":
    await Task.Delay(2000);
    await stepContext.Context.SendActivityAsync($"Espero haberte ayudado {nombreUsuario}, puedes volverme a escribir si necesitas de mi asistencia", cancellationToken: cancellationToken);
    await Task.Delay(2000);
    await stepContext.Context.SendActivityAsync($"Que tengas un buen día. Hasta Pronto....", cancellationToken: cancellationToken);

    return await stepContext.EndDialogAsync(cancellationToken: cancellationToken);
```

Figura 86: Case “TerminarConversacion”.

Por último, el case “Default”, Figura 87, muestra un mensaje de error para los usuarios del sistema en caso de no ingresar alguna opción anterior.

```
default:
    await stepContext.Context.SendActivityAsync($"No entiendo lo que dices", cancellationToken: cancellationToken);
    await Task.Delay(2000);
    await stepContext.Context.SendActivityAsync($"Debes pedirme una de las opciones que te muestro, de acuerdo?", cancellationToken: cancellationToken);
    break;
```

Figura 87: Case “default”.

3.3.7. SERVICIO COGNITIVO “LUIS”

Como se mencionó anteriormente, para que nuestro programa de Visual Studio 2019 pueda interactuar con los usuarios finales y a su vez pueda procesar la información que estos le proporcionan en forma de lenguaje natural, se procedió a utilizar la herramienta “LUIS” (Language Understanding Intelligent Service). El cuál es un servicio de aprendizaje automático en lenguaje natural que puede ser utilizado en distintos bots y dispositivos IoT.

Con la ayuda de esta herramienta, que obtiene información de las conversaciones, se puede llegar a identificar el requerimiento (“intención”) de los usuarios. Esta información puede ser obtenida de una conversación por “LUIS” gracias a su sistema de procesamiento de lenguaje, que, a su vez puede identificar las diferentes “entidades” que se necesitan en Visual Studio 2019 para continuar con la ejecución del código.

Este servicio cognitivo es una herramienta de procesamiento de lenguaje natural basado en Machine Learning que se encuentra disponible en la nube y funciona enviando las “Intenciones” y “Entidades” de los usuarios en forma de texto plano a la aplicación en Visual Studio 2019. Siendo esta información necesaria para las acciones dentro del programa.

“LUIS” envía la información captada como un archivo tipo JSON. Este archivo es reconocido por la aplicación de Visual Studio 2019 que a su vez extrae la acción que se desea realizar, lo cual representa la “intención”. Junto con las “entidades”, las cuales son las variables de esa intención y son necesarias para que el programa pueda detectar y solventar solicitudes.

En la Figura 88 se muestra la estructura del objeto tipo JSON que el servicio “LUIS” envía al programa para que lo interprete. Se puede observar la “intención” con mayor porcentaje de interpretación y las “entidades” de esta.

Figura 88: Objeto tipo JSON de “LUIS”.

Con la ayuda de esta herramienta se crearon las “intenciones” correspondientes para el desenvolvimiento deseado de “CHAPIE”; y las “entidades” necesarias para poder ejecutar correctamente estas “intenciones” dentro del programa de Visual Studio 2019. En la Figura 89, se muestra todas las intenciones que fueron creadas.

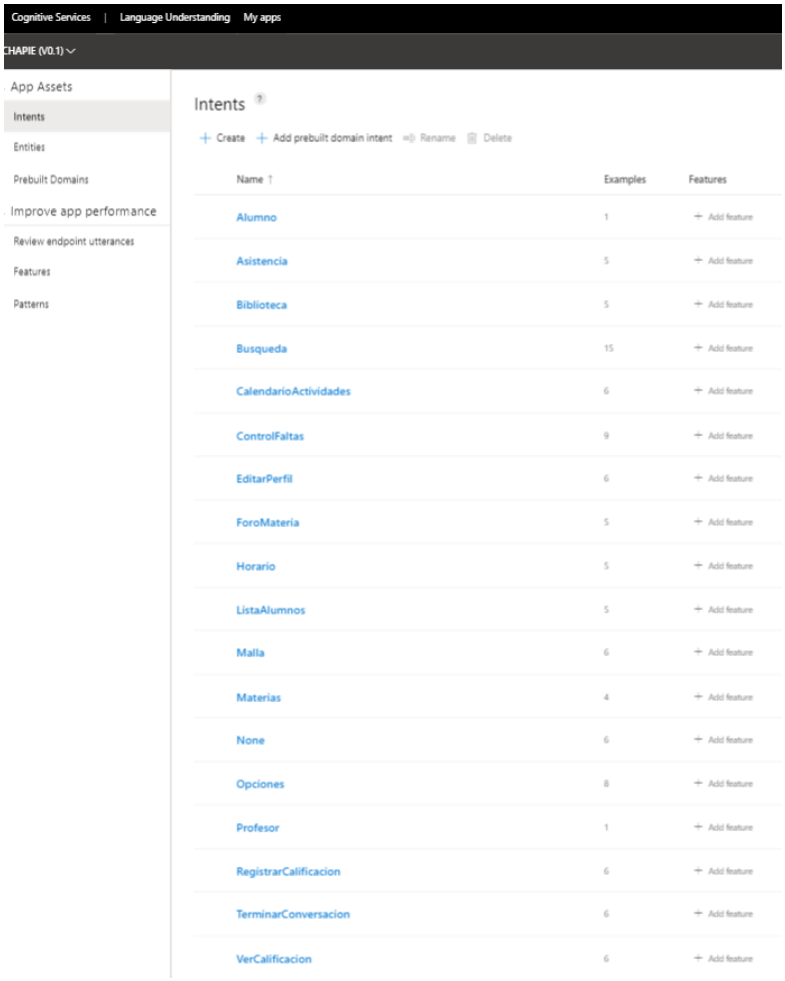


Figura 89: Intenciones creadas en “LUIS”.

Las “intenciones” creadas corresponden con una funcionalidad de nuestro chatbot como sus nombres lo indican: “Alumno”, “Asistencia”, “Biblioteca”, “Busqueda”, “CalendarioActividades”, “ControlFaltas”, “EditarPerfil”, “ForoMateria”, “Horario”, “ListaAlumnos”, “Malla”, “Materias”, “None”, “Opciones”, “Profesor”, “RegistrarCalificacion”, “TerminarConversacion”, “VerCalificacion”.

A su vez en este proyecto solo fue necesario la creación de una “entidad” llamada “buscar”, Figura 90.

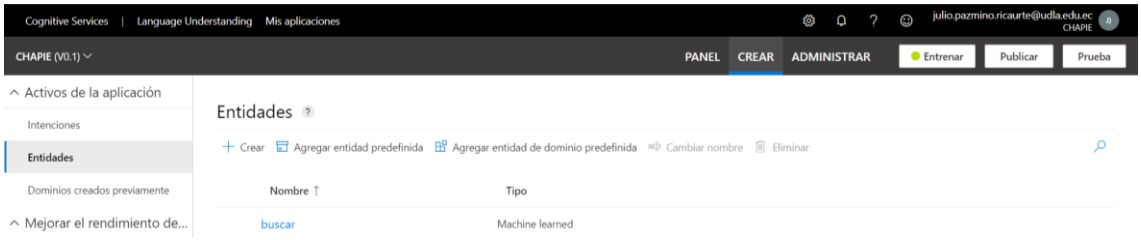


Figura 90: Entidades creadas en “LUIS”.

Una vez creadas las “intenciones”, se deben agregar ejemplos de expresiones en lenguaje natural. Las cuales, los usuarios podrían utilizar para solicitar la ejecución de alguna funcionalidad de “CHAPIE”. Estos ejemplos son expresiones que se presentan en conversaciones naturales, lo que le da al proyecto una mayor facilidad de uso. A su vez a cada ejemplo se le agregan las “entidades” necesarias dependiendo de lo que se necesita, como se detalla en la Figura 91.

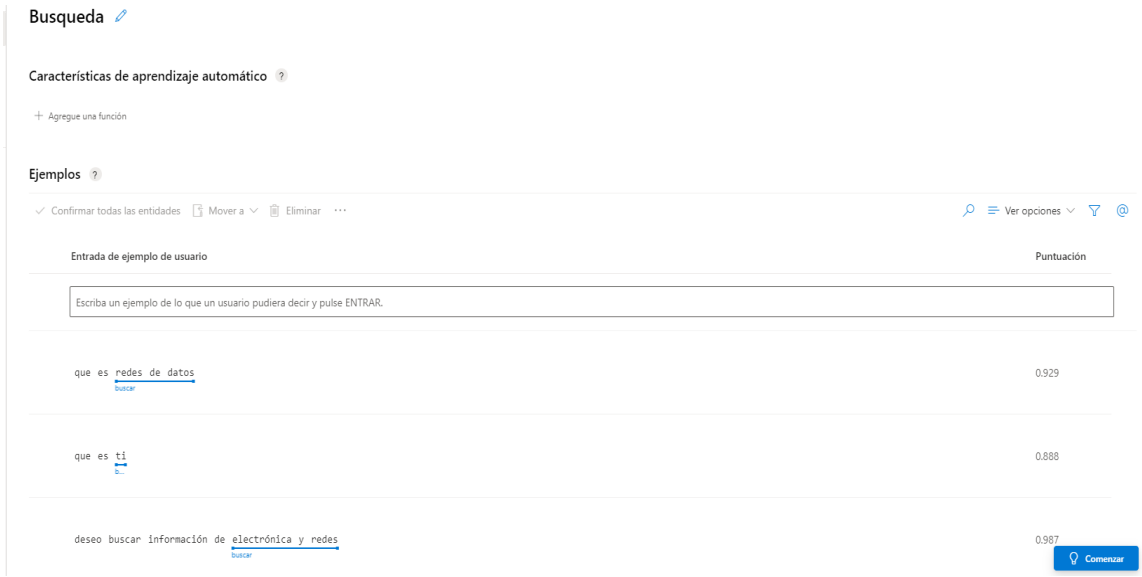


Figura 91: Expresiones de una “intención” y su “entidad”.

Una vez que fueron agregadas todas las expresiones necesarias en las “intenciones”, se procede al entrenamiento del bot y a su posterior publicación.

Esto ayuda a que los cambios agregados en “LUIS” sean entendidos y procesados por el programa en Visual Studio 2019.

Para entrenar al bot en “LUIS”, simplemente se presiona en el botón “Entrenar”. Posteriormente para la publicación el botón “Publicar”, Figura 92.

Figura 92: Botones para entrenar y publicar el bot en “LUIS”.

El botón “Publicar” permite seleccionar el espacio de publicación y la configuración para la implementación del chatbot. En este caso se seleccionó el espacio de producción, como se muestra en la Figura 93. Gracias a ello los cambios realizados por “LUIS” son leídos por “CHAPIE”.

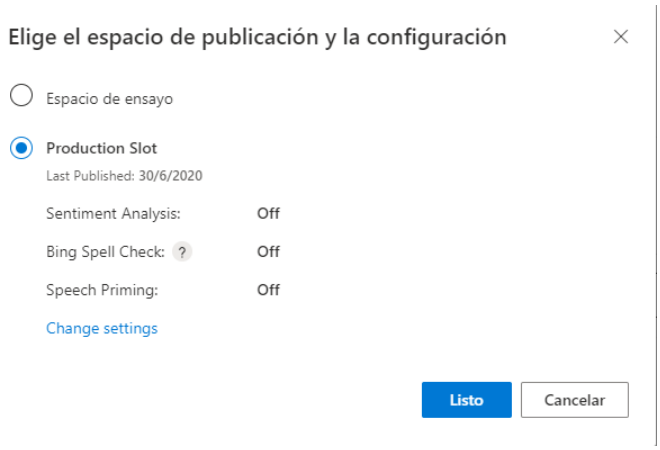


Figura 93: Publicación de cambios de “LUIS”.

3.3.8. CONFIGURACIONES DE “LUIS” EN VISUAL STUDIO 2019

Para que el programa en Visual Studio 2019 pueda hacer uso del servicio cognitivo “LUIS”, fueron necesarias las clases creadas en la carpeta “Infraestructure”, Figura 94, “ILuisRecognizerService.cs” y “LuisRecognizerService.cs”. Estas clases contienen las configuraciones necesarias para que el servicio de “LUIS” intercambie información con “CHAPIE”.

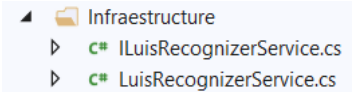


Figura 94: Carpeta “Infraestructure” de “CHAPIE”.

En la clase pública “ILuisRecognizerService.cs”, Figura 95, se encuentra un método, el cual se encarga de obtener el objeto tipo JSON del servicio cognitivo “LUIS”.

```
namespace CHAPIE.Infraestructure
{
    6 referencias
    public interface ILuisRecognizerService
    {
        4 referencias | 0 excepciones
        public LuisRecognizer _recognizer { get; }
    }
}
```

Figura 95: Clase “ILuisRecognizerService.cs”, de la carpeta “Infraestructure”.

La clase “LuisRecognizerService.cs”, Figura 96, posee el llamado a las credenciales para que el servicio cognitivo autorice el paso de información. Las credenciales necesarias son: "LuisAppId", "LuisAPIKey" y "LuisHostName". Estas credenciales se encuentran como tal en el archivo “appsettings.json” del proyecto de Visual Studio 2019 y se revisarán en la siguiente sección.

```
namespace CHAPIE.Infraestructure
{
    2 referencias
    public class LuisRecognizerService: ILuisRecognizerService
    {
        4 referencias | 0 excepciones
        public LuisRecognizer _recognizer { get; private set; }
        0 referencias | 0 excepciones
        public LuisRecognizerService(IConfiguration configuration)
        {
            // llamo variables de conexion con LUIS
            var luisApplication = new LuisApplication(
                configuration["LuisAppId"],
                configuration["LuisAPIKey"],
                configuration["LuisHostName"]
                );


            var recognizerOptions = new LuisRecognizerOptionsV3(luisApplication)
            {
                PredictionOptions = new Microsoft.Bot.Builder.AI.LuisV3.LuisPredictionOptions()
                {
                    IncludeInstanceData = true
                }

            };

            _recognizer = new LuisRecognizer(recognizerOptions);
        }
    }
}
```

Figura 96: Clase “LuisRecognizerService.cs”, de la carpeta “Infraestructure”.

3.3.9. ARCHIVO DE CONFIGURACIÓN DE SERVICIOS EXTERNOS DE “CHAPIE”

El proyecto posee un archivo llamado “appsettings.json”, Figura 97,el cual posee las configuraciones para la conexión de “CHAPIE” con herramientas y APIs externas. En este caso el servicio cognitivo “LUIS” y el servicio en la nube de Microsoft Azure, el cual almacenará el proyecto, son los servicios incorporados a nuestro chatbot. Con estas configuraciones “CHAPIE” podrá ser publicado en la nube de Microsoft Azure, además de lograr interconexión con el servicio cognitivo “LUIS”.

```
og.cs | appsettings.json | LuisRecognizerService.cs
: https://json.schemastore.org/appsettings
{
  "MicrosoftAppId": "1b910256-6319-4341-82f8-4ee7b9dea6b0",
  "MicrosoftAppPassword": "~fFXErpifs2qSYbkB123_w-Bh_dd.13F98",


  "LuisAppId": "4eaa39fb-1a6d-4161-9a64-3140cda4de36",
  "LuisAPIKey": "533a3850d3cc4fa1b519787e2d542bab",
  "LuisHostName": "https://westus.api.cognitive.microsoft.com/"
}
```

Figura 97: Archivo “appsettings.json” de “CHAPIE”.

3.3.10. PRUEBAS PREVIAS A LA IMPLEMENTACIÓN

En esta sección se revisará cómo se realizaron las pruebas de funcionamiento previas a la implementación del proyecto. Para ello se utilizó el software Bot Framework Emulator, el cual ayudó al despliegue del chatbot.

Para poder utilizar este software primero debemos agregar la URL que nos da la página “default.htm” al momento de su ejecución. Se debe agregar la URL en el formato requerido como muestra la Figura 98.

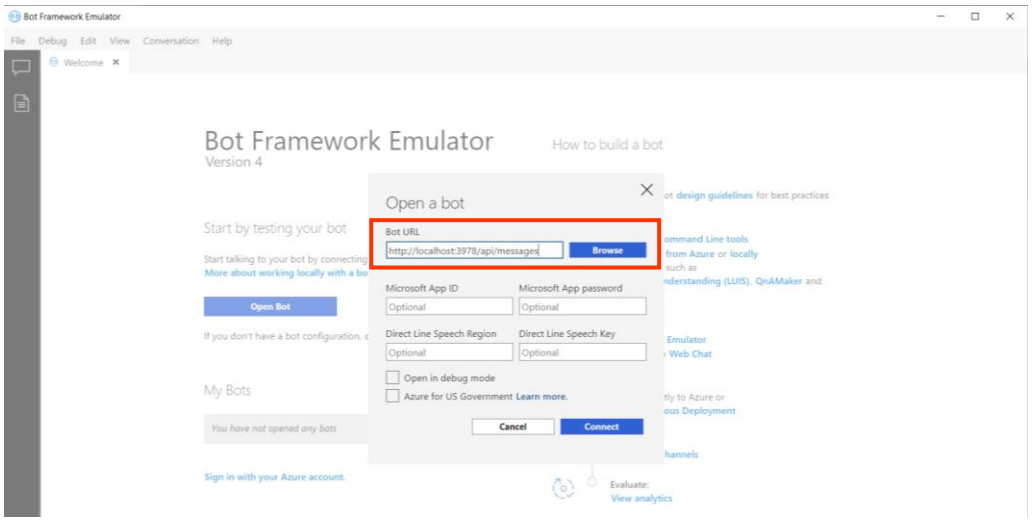


Figura 98: Inclusión de URL en Bot Framework Emulator.

A continuación, se muestra un ejemplo de la interacción que se lleva a cabo entre “CHAPIE” y un usuario, Figura 99. Además del archivo tipo JSON de “LUIS”, el cual identifica las “intenciones” y “entidades” de la solicitud del usuario.

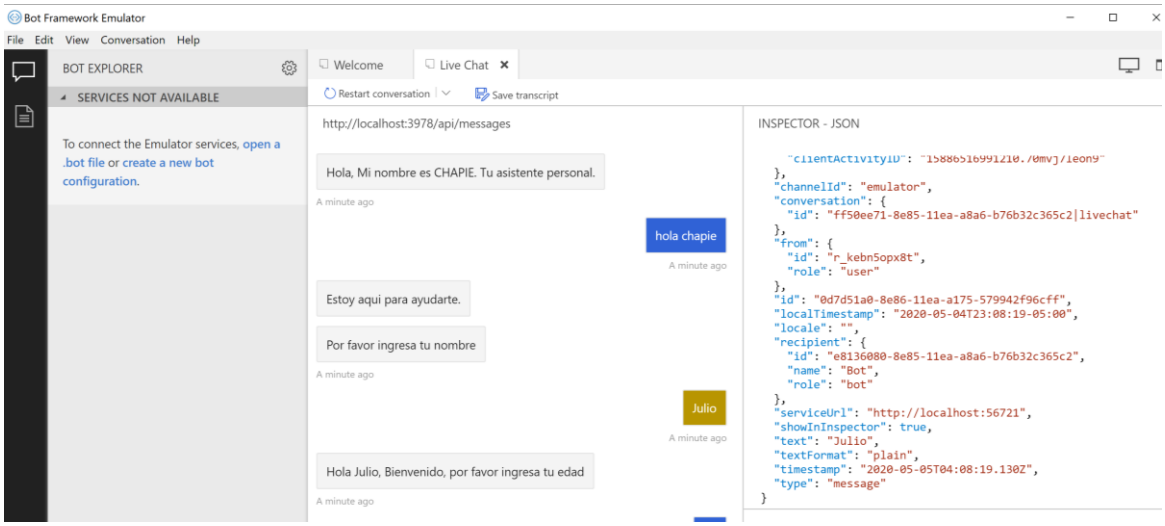


Figura 99: Eje. Prueba de funcionamiento de “CHAPIE”.

Se comprobaron cada una de las funcionalidades de “CHAPIE” y se hicieron los ajustes necesarios para un correcto funcionamiento y entendimiento del chatbot con los usuarios. Una vez aplicadas las debidas correcciones, se procedió con la etapa de implementación que se describe en el siguiente capítulo.

4. CAPÍTULO IV

En este capítulo se revisará la información correspondiente a la implementación de “CHAPIE” y sus respectivas pruebas de funcionamiento, además del análisis de los resultados obtenidos de su interacción con alumnos de la universidad.

4.1. IMPLEMENTACIÓN

Esta sección se enfoca en las fases que se realizaron para la implementación de “CHAPIE” dentro de un aula virtual, las cuales se describen a continuación:

4.1.1. PUBLICACIÓN DEL PROYECTO EN MICROSOFT AZURE

Para la implementación de “CHAPIE”, primero se procedió con la publicación del proyecto en Microsoft Azure. Para ello, una vez agregadas las credenciales del canal correspondiente, como se muestra en la Figura 100, se presiona sobre la opción “publicar” en el proyecto de Visual Studio 2019. El programa automáticamente se encarga de publicar el proyecto y se despliega una página con una URL pública que comprueba que el chatbot está en línea, Figura 101.

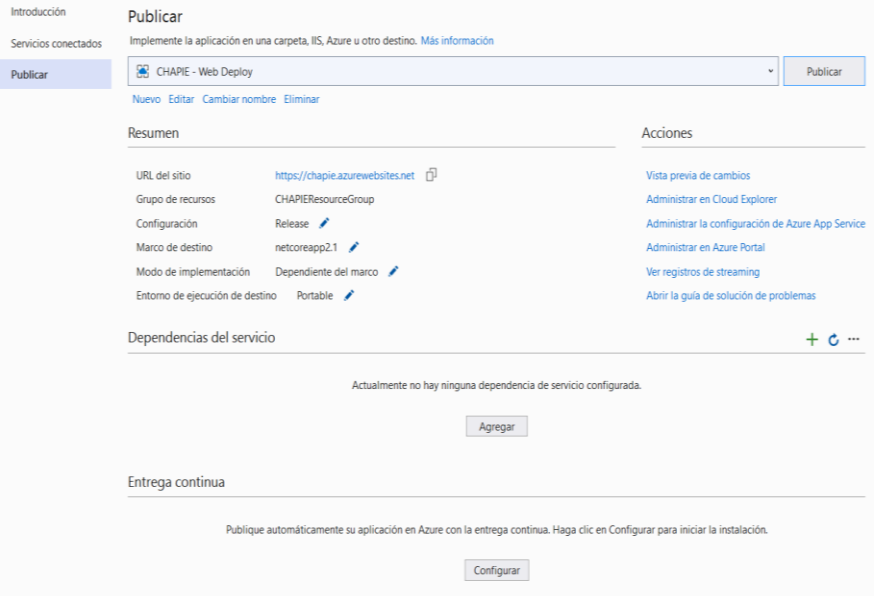


Figura 100. Publicación de “CHAPIE” en Microsoft Azure.

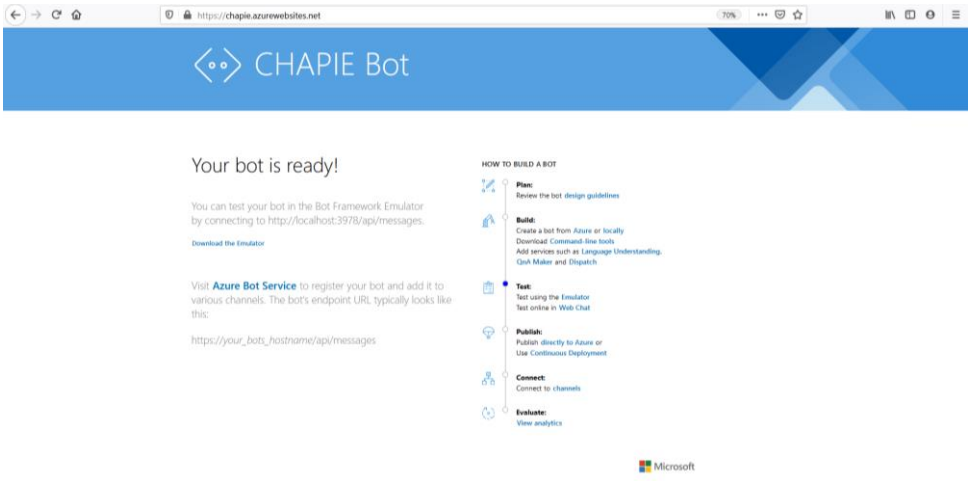


Figura 101: Página del chatbot en línea.

También en nuestra cuenta de Microsoft Azure previamente creada, se puede observar que el chatbot ha sido publicado exitosamente, Figura 102.

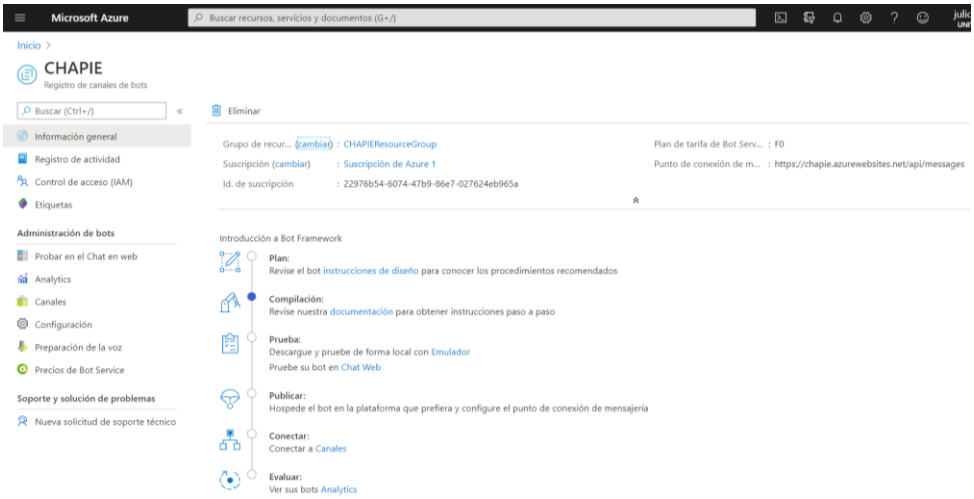


Figura 102: Chatbot publicado en Microsoft Azure.

4.1.2. HABILITACIÓN DE CANALES

Para que el chatbot pueda ser implementado en un LMS, como un aula virtual, es necesario habilitar el canal adecuado para que el servicio pueda ser utilizado por los usuarios, Figura 103.

En este proyecto fue necesario habilitar un canal web, Figura 104, debido a que será implementado en el aula virtual de la Universidad de las Américas (UDLA), que se encuentra en línea. Sin embargo, Microsoft Azure ofrece una variedad de canales para diferentes tipos de implementación.

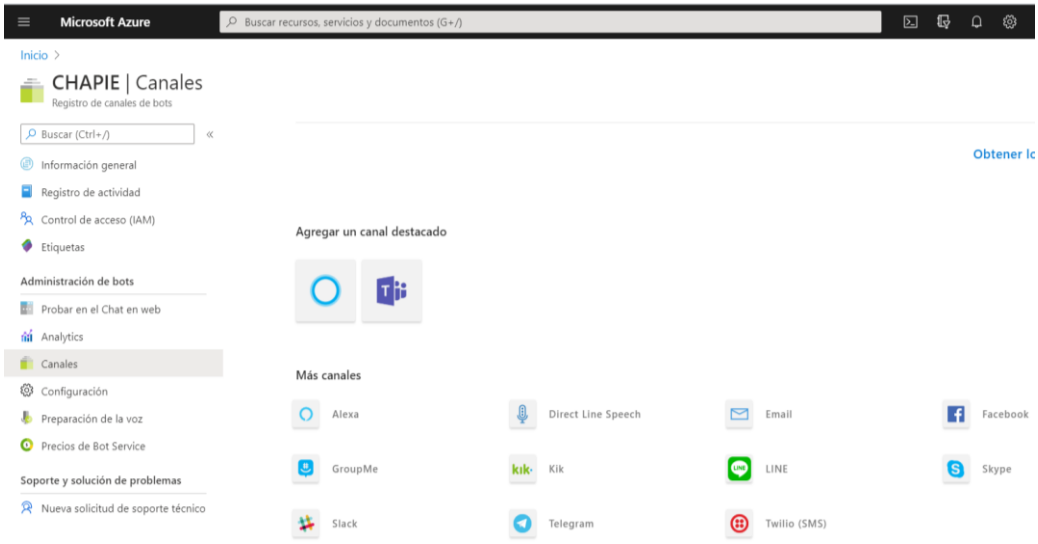


Figura 103: Canales de Microsoft Azure.

Figura 104: Canales habilitados para “CHAPIE”.

4.1.3. IMPLEMENTACION EN EL AULA VIRTUAL

Una vez que se ha habilitado el canal web, la página de Microsoft Azure nos proporciona un código HTML, el cual, se colocó en la parte de diseño del “front end” del aula virtual de la universidad para la implementación del chatbot y su despliegue. El código HTML se ve de la siguiente manera, Figura 105:

```
<iframe src='https://webchat.botframework.com/embed/CHAPIE?s=uqyKW--drFU.jHgKUV78pNVHRRiuCO4A1xGm91iCasXJmFjFn0wLZbs'
  style='min-width: 400px; width: 50%; min-height: 500px;'></iframe>
```

Figura 105: Código HTML para la implementación.

A continuación, se muestra la implementación de “CHAPIE”, el cual fue agregado en el diseño de la página del aula virtual de la Universidad de las Américas (Quito-Ecuador), Figura 106.

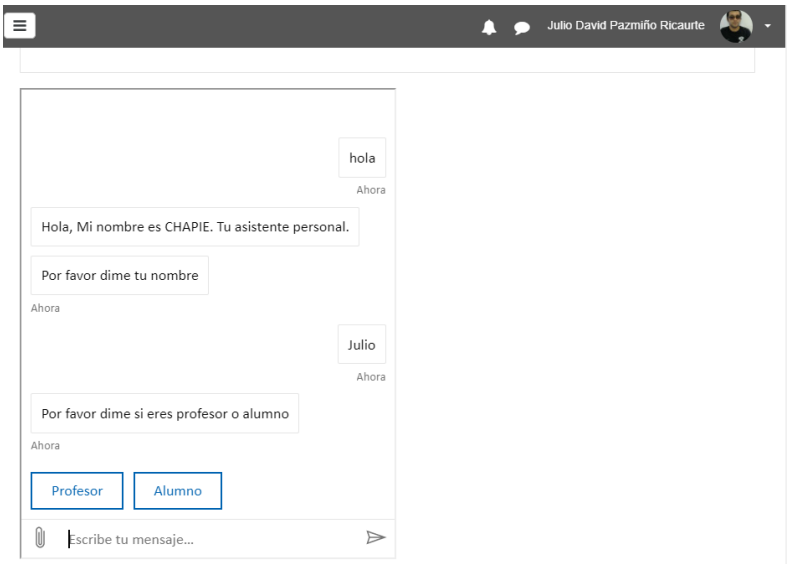


Figura 106: Implementación de “CHAPIE”.

4.2. PRUEBAS DE FUNCIONAMIENTO

En esta sección se mostrarán las diferentes pruebas de ejecución realizadas a fin de verificar el correcto funcionamiento de “CHAPIE”. Estas pruebas se realizaron dentro del aula virtual de la Universidad de las Américas (UDLA).

4.2.1. FUNCIONALIDADES

Lo primero que se probó, fue la conversación inicial que tiene “CHAPIE” con los usuarios. En esta primera interacción el chatbot despliega un saludo inicial y

hace una petición del nombre de usuario para poder continuar, Figura 107 – Figura 108.

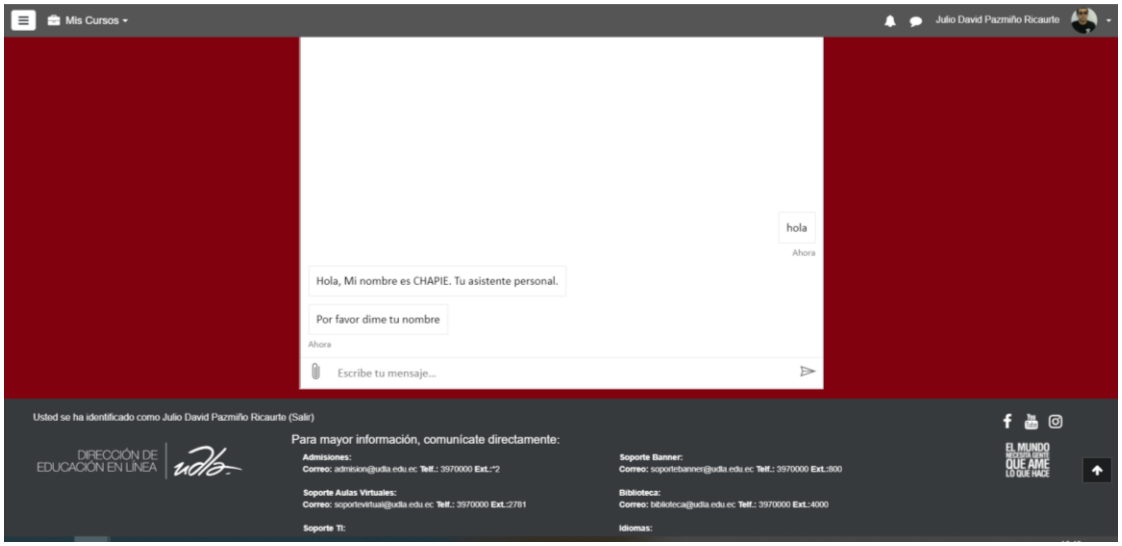


Figura 107: Despliegue de “CHAPIE” en aula virtual.

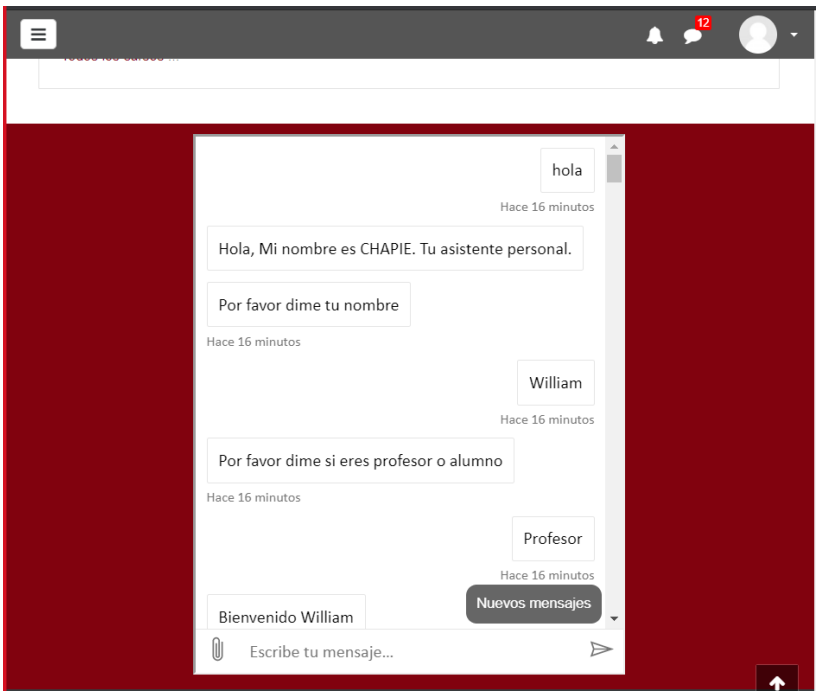


Figura 108: Conversación Inicial de “CHAPIE”.

Después, el programa continúa con la petición del tipo de usuario. Para ello despliega dos botones que se pueden seleccionar o directamente se puede escribir el tipo de usuario que está utilizando el chatbot, Figura 109.

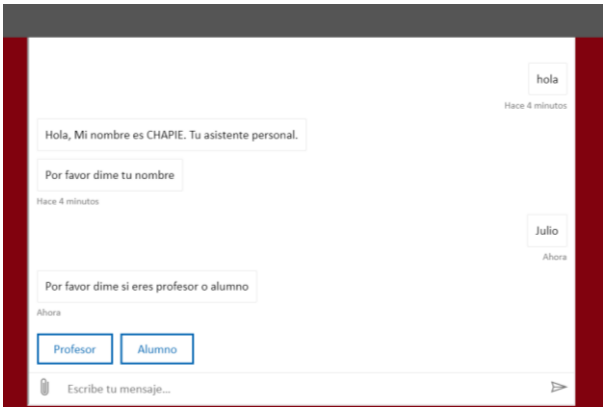


Figura 109: Prueba: Elección del tipo de usuario en “CHAPIE”.

El programa, despendiendo el tipo de usuario que lo está utilizando, despliega el menú correspondiente, Figura 110 – Figura 111.

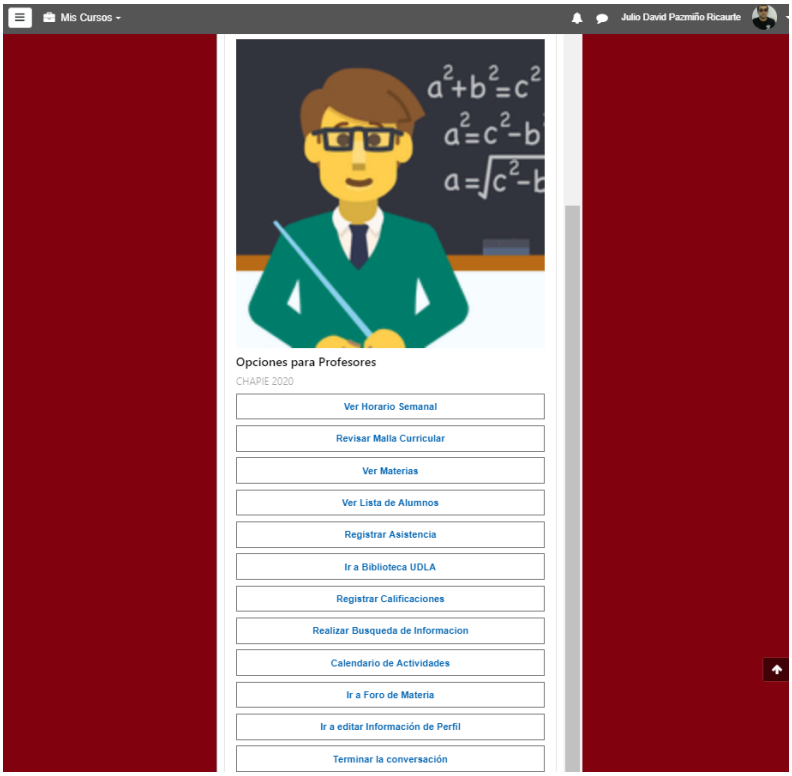


Figura 110: Prueba: Menú de Profesores de “CHAPIE”.

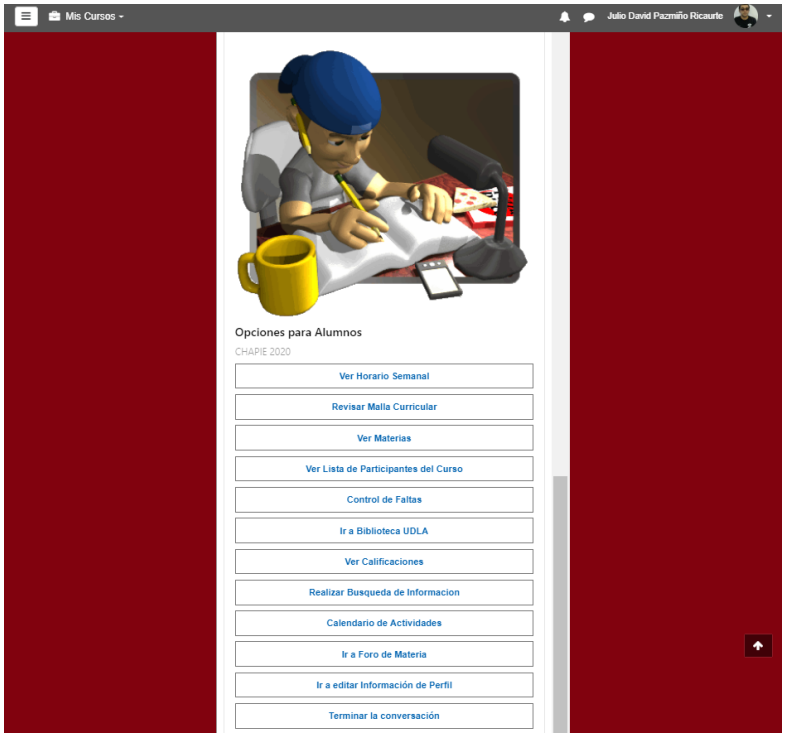


Figura 111: Prueba: Menú de Alumnos de “CHAPIE”.

Una vez terminada la fase inicial de la conversación, el programa espera por alguna petición que sea referente a sus opciones del menú. En esta primera interacción se pide el despliegue de opciones, lo cual ayuda a que se vuelva a mostrar el menú correspondiente a cada usuario, Figura 112.

Figura 112: Prueba: Función “Ver Opciones” en “CHAPIE”.

Posteriormente se fueron probando cada una de las opciones del menú. La función “Ver Horario Semanal” nos permite observar el horario de los usuarios, Figura 113.

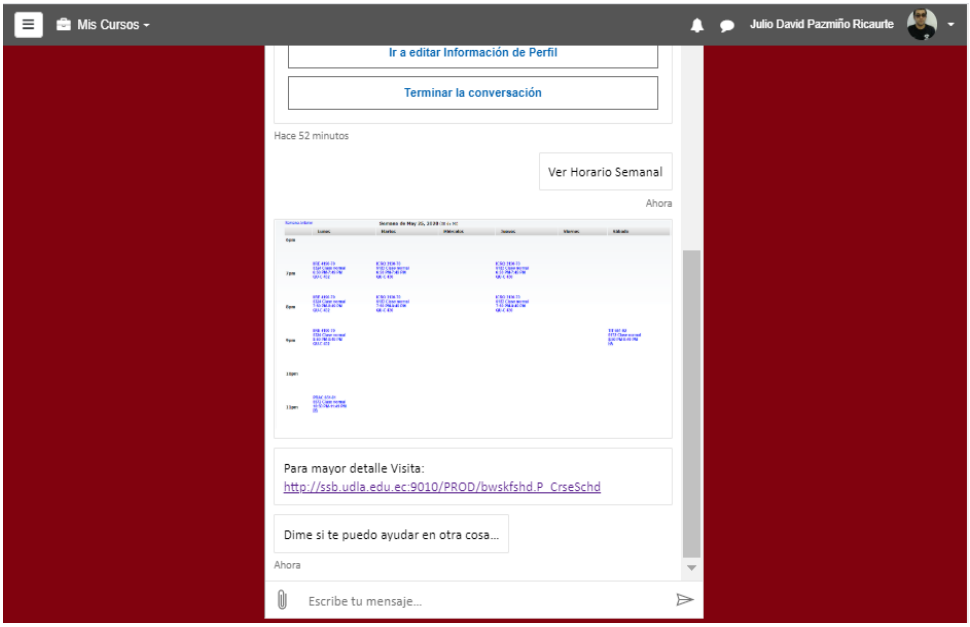


Figura 113: Prueba: Función “Ver Horario Semanal”

La función “Revisar Malla Curricular” permite a los usuarios visualizar la malla correspondiente a la carrera en curso, también ofrece un botón de acceso para descargar el archivo en PDF, Figura 114 – Figura 115.

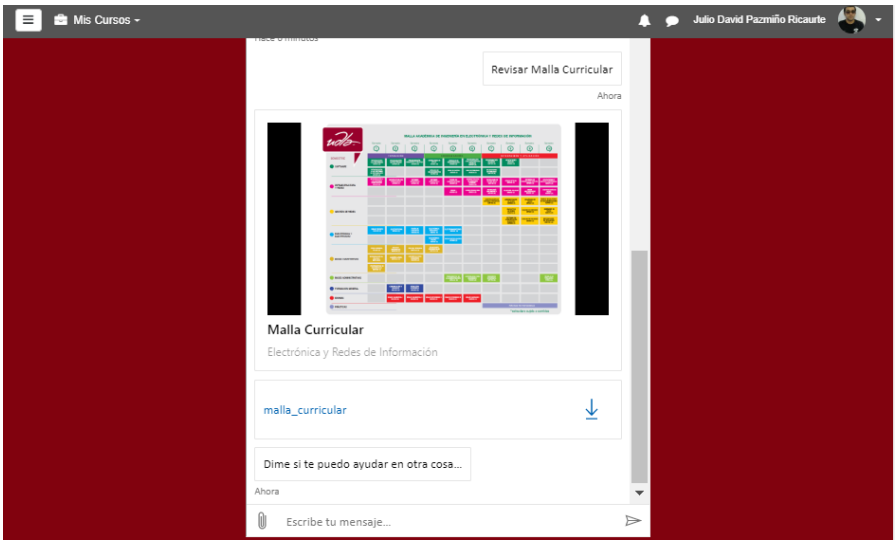


Figura 114: Prueba: Función “Revisar Malla Curricular”.

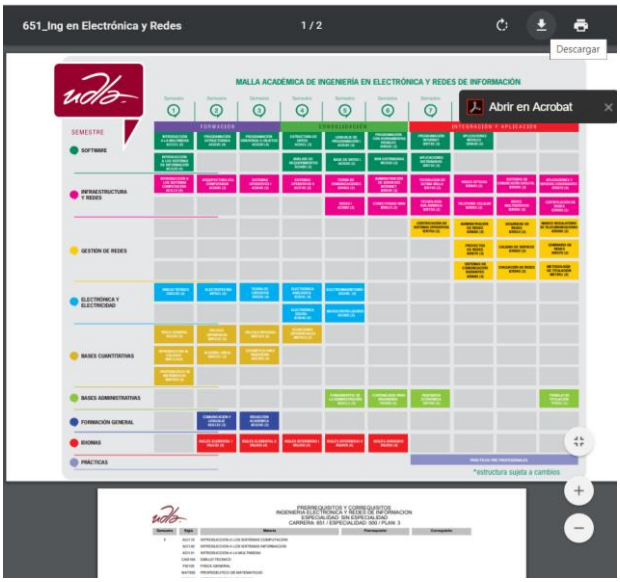


Figura 115: Prueba: Botón de acceso de la función “Revisar malla curricular”.

La función “Ver Materias” permite a los usuarios visualizar las correspondientes materias en curso. También ofrece un link de acceso para visualizar las materias en el servicio “Banner” de la Universidad de las Américas, Figura 116.

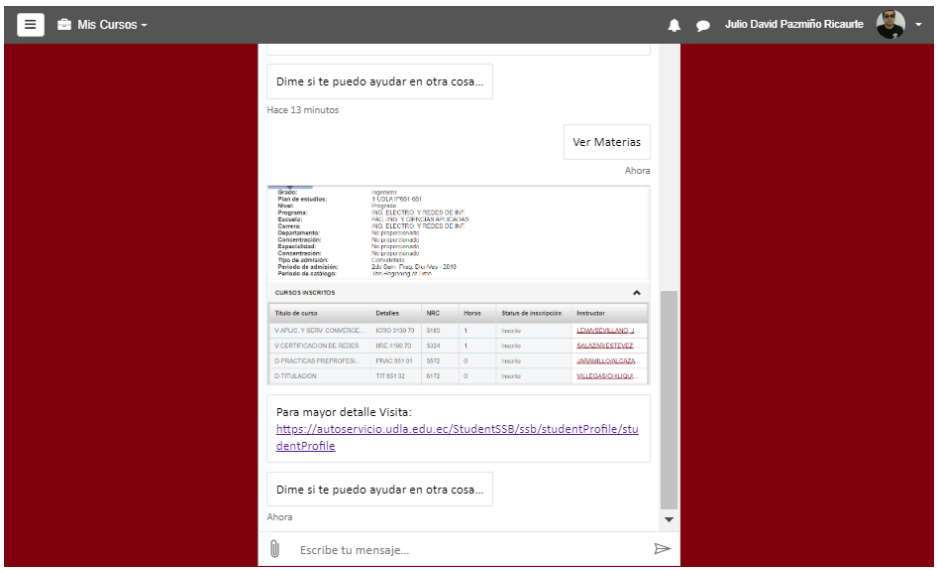


Figura 116: Prueba: Función “Ver Materias”.

La función “Ver Lista de Alumnos” permite a los usuarios visualizar a los demás alumnos de cada materia en curso. Despliega un botón de acceso para visualizar a los usuarios por materia correspondiente, accediendo a la respectiva dirección en el aula virtual de la Universidad de las Américas, Figura 117 – Figura 118.

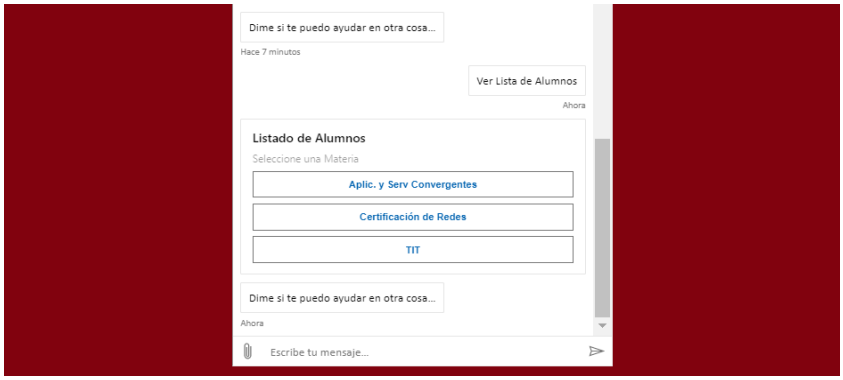


Figura 117: Prueba: Función “Ver Lista de Alumnos”.

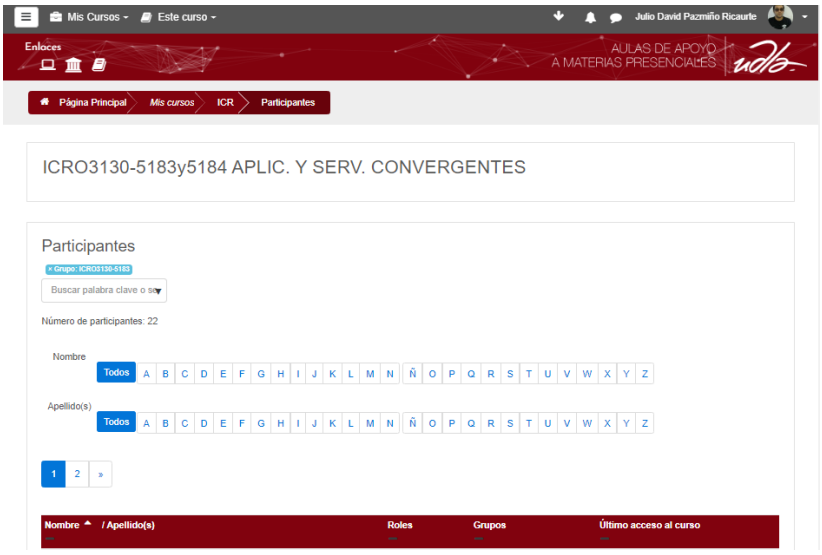


Figura 118: Prueba: Botón Función “Ver Lista de Alumnos”.

La función “Registrar Asistencia” permite a los usuarios que son profesores registrar la asistencia de los estudiantes de cada materia en curso. Despliega un

botón de acceso para el portal de registro de asistencia de la Universidad de las Américas, Figura 119.

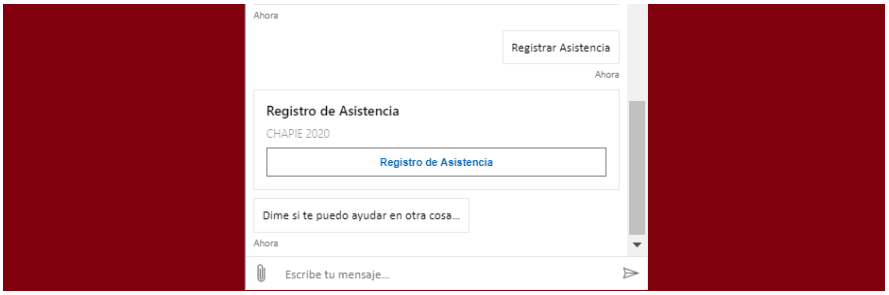


Figura 119: Prueba: Función “Registro de Asistencia”.

La función “Control de Faltas” permite a los usuarios que son alumnos visualizar el control de asistencia de cada materia en curso. Despliega un botón de acceso al portal de control de asistencia de la Universidad de las Américas, Figura 120 – Figura 121.

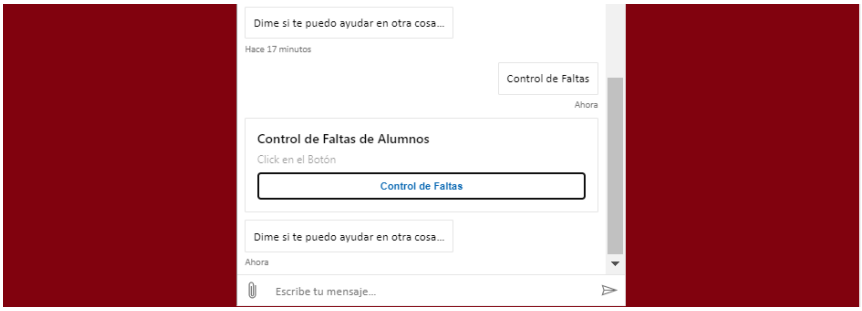


Figura 120: Prueba: Función “Control de Faltas”.

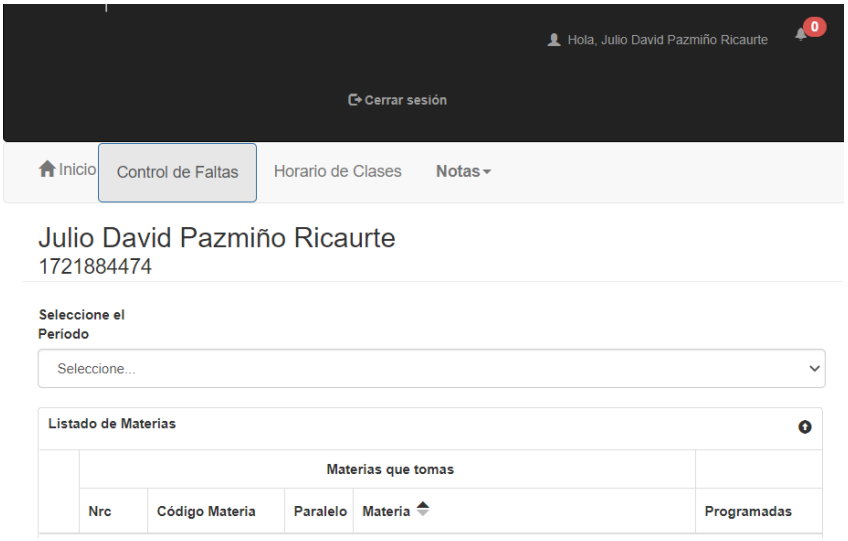


Figura 121: Prueba: Botón Función “Control de Faltas”.

La función “Ir a Biblioteca UDLA” permite a los usuarios acceder a la biblioteca de la Universidad de las Américas. Despliega un botón de acceso a la misma, Figura 122 – Figura 123.

Figura 122: Prueba: Función “Ir a Biblioteca UDLA”.

Figura 123: Prueba: Botón Función “Ir a Biblioteca UDLA”.

La función “Registro de Calificaciones” permite a los usuarios que son profesores acceder al registro de calificaciones de cada materia en curso. Despliega un botón de acceso al registro de calificaciones del aula virtual de la Universidad de las Américas, Figura 124.

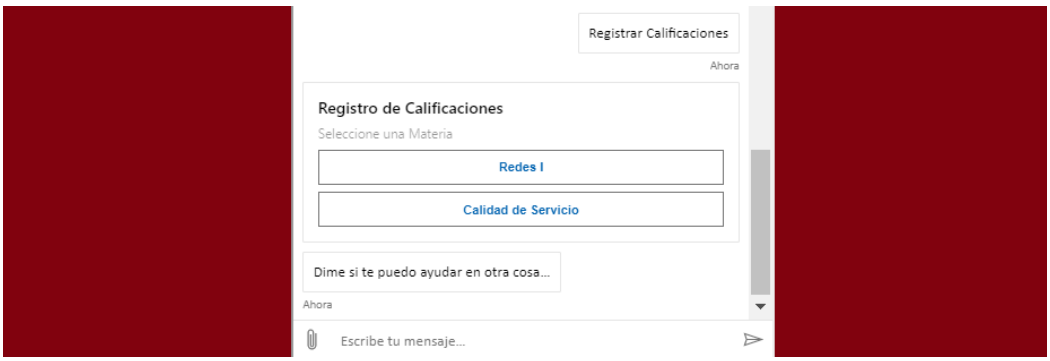


Figura 124: Prueba: Función “Registro de Calificaciones”.

La función “Ver Calificaciones” permite a los usuarios que son alumnos acceder la vista de las calificaciones de cada materia en curso. Despliega un botón de acceso a las calificaciones de cada materia del aula virtual de la Universidad de las Américas a las que está inscrito, Figura 125 – Figura 126.

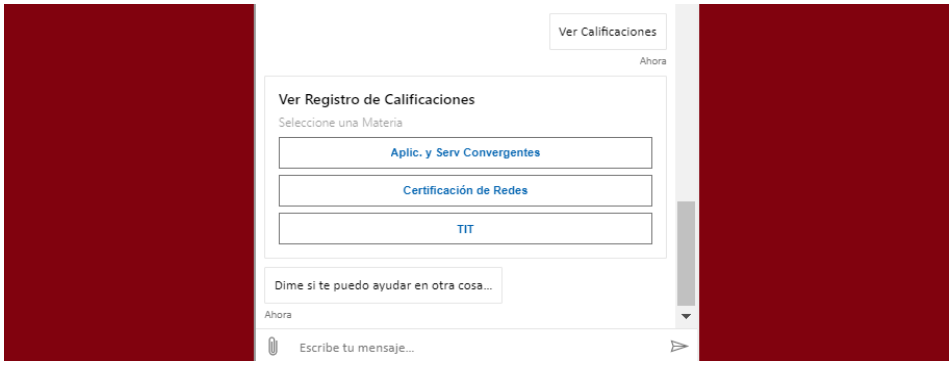


Figura 125: Prueba: Función “Ver Calificaciones”.

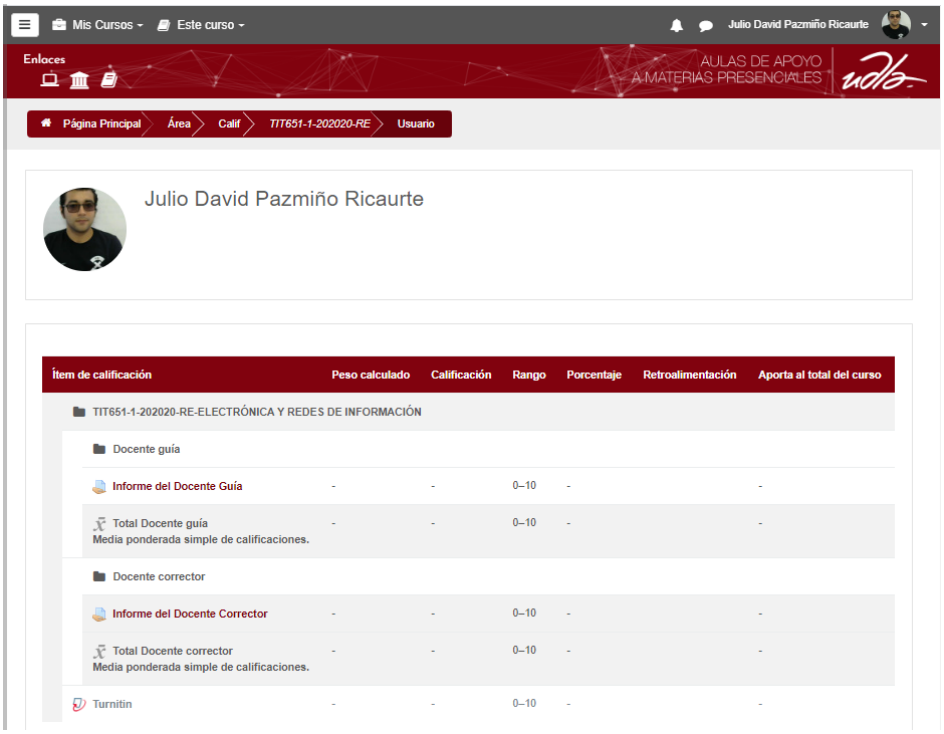


Figura 126: Prueba: Botón Función “Ver Calificaciones”.

La función “Realizar Búsqueda de Información” permite a los usuarios realizar una búsqueda web sobre cualquier tema. Al momento de seleccionar la opción en el menú, se despliega un mensaje que enseña al usuario como realizar una búsqueda, Figura 127.

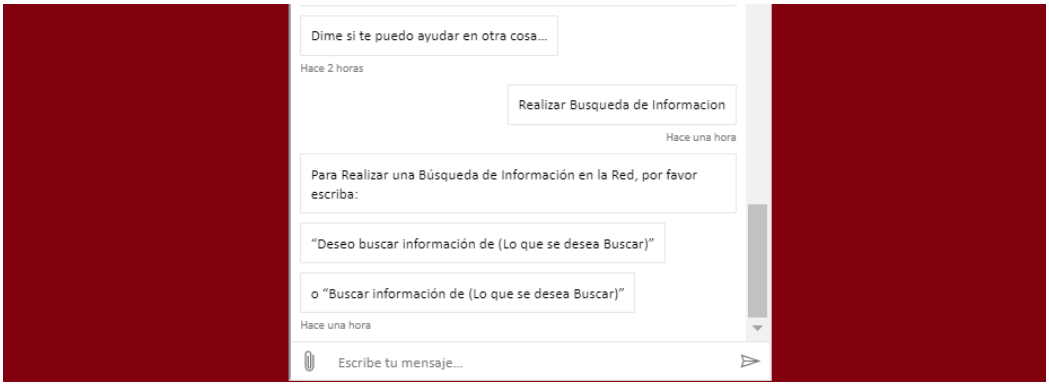


Figura 127: Prueba: Función "Realizar Búsqueda de Información".

El usuario pude realizar una búsqueda web en cualquier momento siguiendo las directrices indicadas, Figura 128.

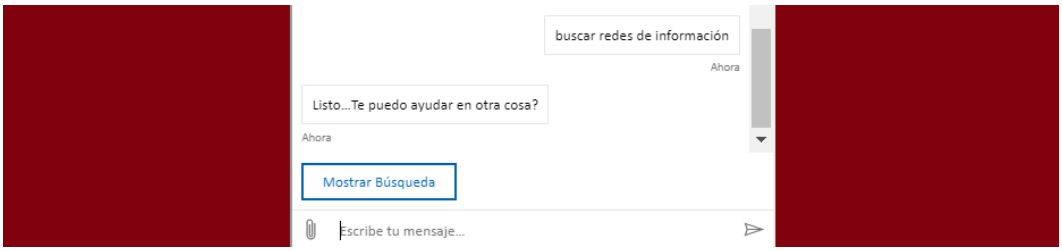


Figura 128: Prueba: Ejemplo de búsqueda de información.

La función "Calendario de Actividades" permite a los usuarios acceder al calendario de actividades en el aula virtual de la Universidad de las Américas. Despliega un botón de acceso a la misma, Figura 129.

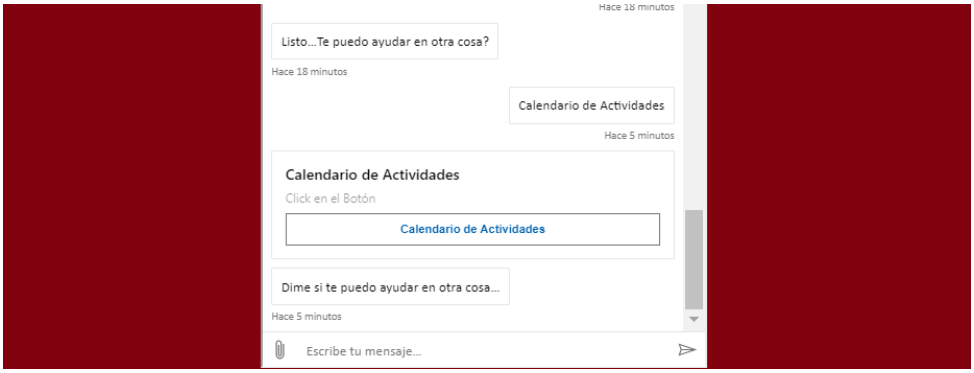


Figura 129: Prueba: Función “Calendario de Actividades”.

La función “Ir a Foro de Materia” permite a los usuarios acceder los foros de las materias en curso. Despliega un botón de acceso al foro de cada materia del aula virtual de la Universidad de las Américas a las que está inscrito, Figura 130 – Figura 131.

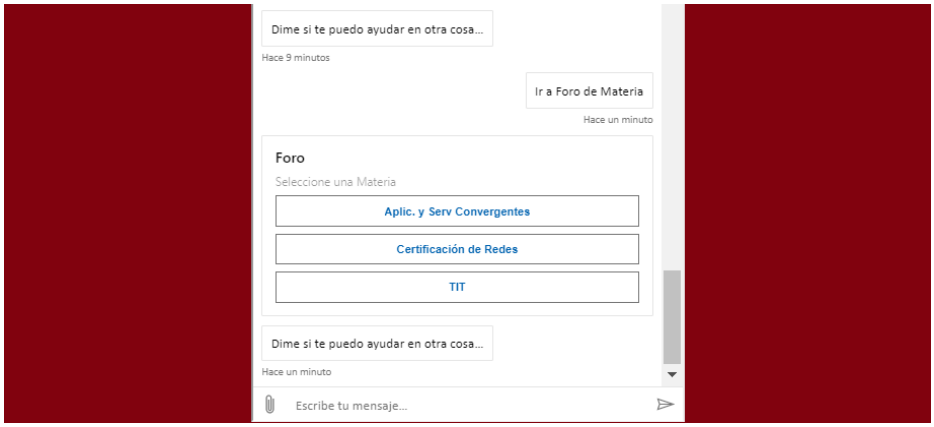


Figura 130: Prueba: Función “Ir a Foro de Materia”.

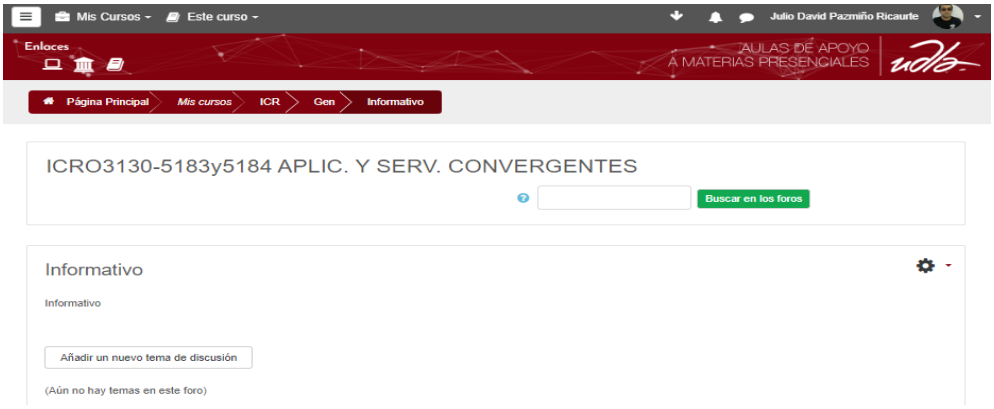


Figura 131: Prueba: Botón Función “Ir a Foro Materia”.

La función “Editar Información de Perfil” permite a los usuarios acceder a su perfil personal en el aula virtual de la Universidad de las Américas a fin de poder editarlo. Despliega un botón de acceso al mismo. Figura 132 – Figura 133.

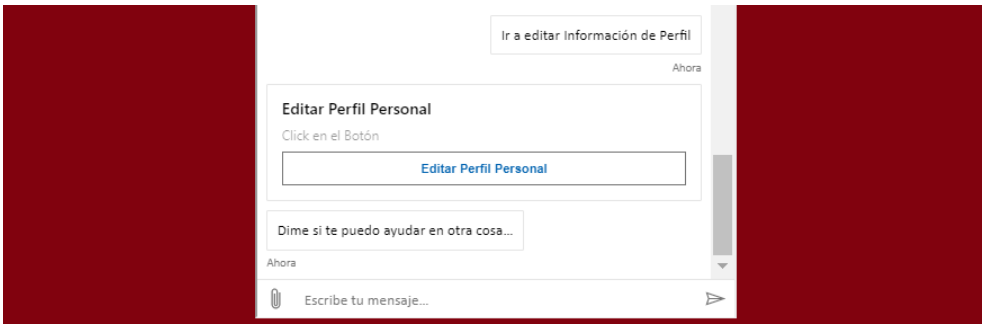


Figura 132: Prueba: Función “Editar Información de Perfil”.

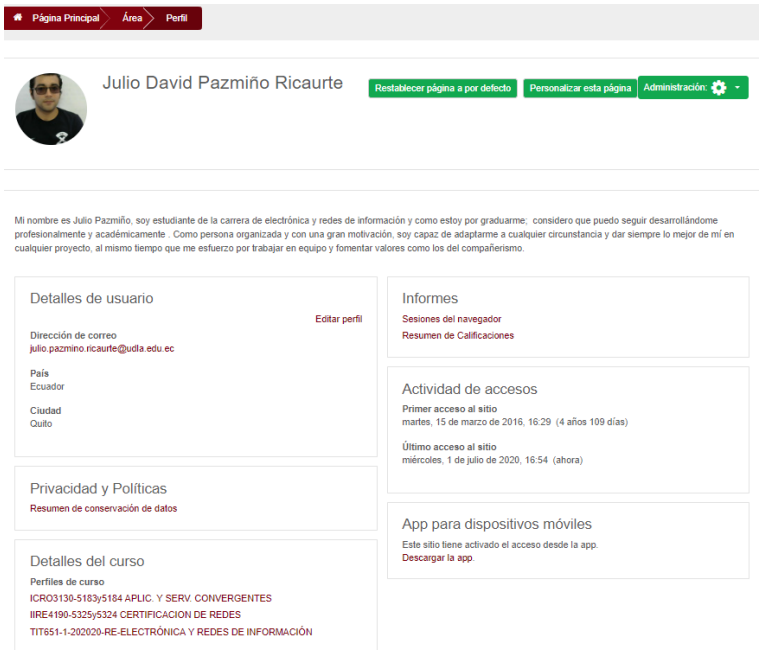


Figura 133: Prueba: Botón Función “Editar Información de Perfil”.

La función “Terminar la conversación” permite a los usuarios terminar la conversación actual que tiene con “CHAPIE”. Se despliega un mensaje de despedida y termina la interacción con el usuario hasta una nueva conversación, Figura 134.

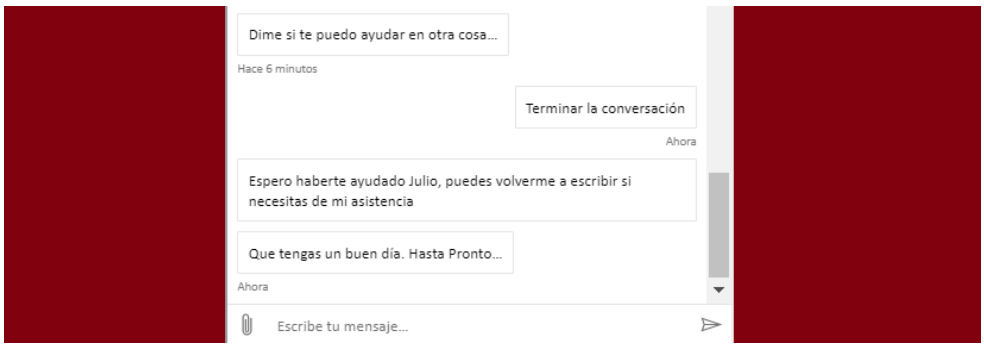


Figura 134: Prueba: Función “Terminar la conversación”.

4.2.2. APRENDIZAJE DE “CHAPIE”

Gracias al servicio cognitivo “LUIS”, se obtiene un “aprendizaje activo” para nuestro chatbot. Esto significa que “CHAPIE” podrá identificar enunciados sugeridos por parte de los usuarios que no estén dentro de los ejemplos que fueron agregados previamente a cada “intensión”.

Esta información se almacena en el programa creado en “LUIS” y puede ser revisado posteriormente por el administrador del sistema en la sección “Mejorar el rendimiento del chatbot”, específicamente en la pestaña “Revisar expresiones del punto de conexión”. En la Figura 135, se muestran ejemplos de expresiones que fueron utilizadas por los usuarios y no están dentro de los ejemplos previos de las “intenciones” de “LUIS”.

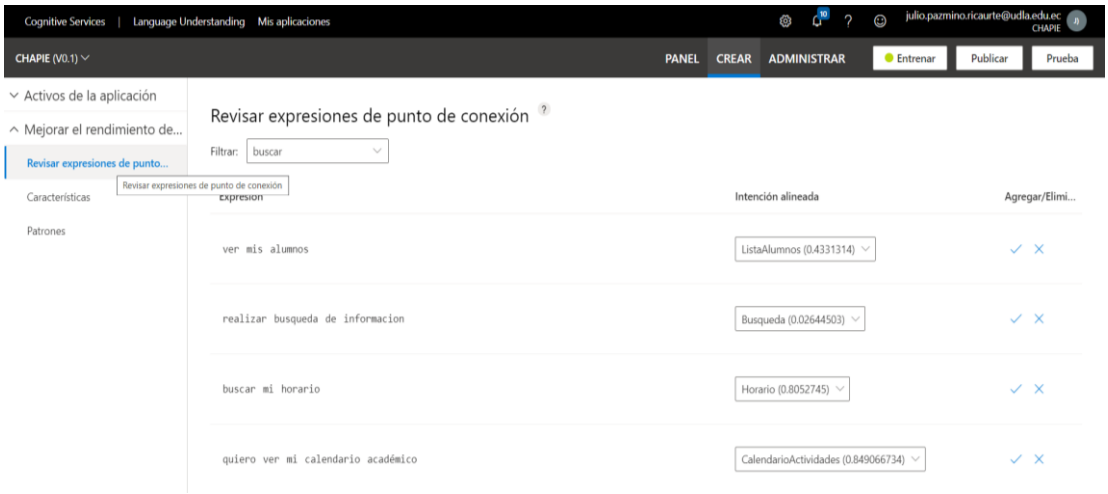


Figura 135: Expresiones nuevas de usuarios.

El programa permite revisar cada nueva expresión y asignarla a la “intención” con mayor afinidad o descartarla completamente. Una vez que se ha asignado la expresión a la “intención” correspondiente, se puede seleccionar la “entidad” contenida en cada una.

Una vez analizadas todas las expresiones necesarias, se procede a entrenar y publicar nuevamente el programa en “LUIS”. Esto ayudará a que “CHAPIE” detecte automáticamente los cambios e identifique estas expresiones como una petición de alguna de sus opciones.

Este tipo de aprendizaje supervisado tiene como objetivo la mejora en el rendimiento de “CHAPIE”. Ayudará a que el sistema comprenda de mejor manera la forma en la que se expresan normalmente las personas y pueda simular una conversación normal. También ayuda a comprender de diferentes maneras las expresiones en lenguaje natural y mejorar la fluidez de las conversaciones.

4.3. ANÁLISIS DE RESULTADOS

Este chatbot fue implementado en el aula virtual de la Universidad de las Américas con el propósito de facilitar la gestión académica a estudiantes y docentes. Una encuesta previa realizada a un grupo de estudiantes que se encuentran cursando alguna carrera de la universidad, nos muestra la necesidad de una herramienta de este tipo que facilite el manejo de los diferentes portales web que esta emplea. A continuación, se observan los resultados obtenidos en la encuesta realizada, los cuales validan la importancia y necesidad de la implementación de este tipo de tecnología.

Esta primera encuesta fue realizada a un grupo de 23 estudiantes (100%) que se desenvuelven dentro del entorno educativo de la universidad a fin de identificar la facilidad de manejo de los diferentes portales web de esta y la necesidad de una herramienta que agilite su uso.

La primera pregunta planteada hace referencia a la facilidad de manejo de las diferentes herramientas virtuales de la universidad, como se muestra en la Figura 136.

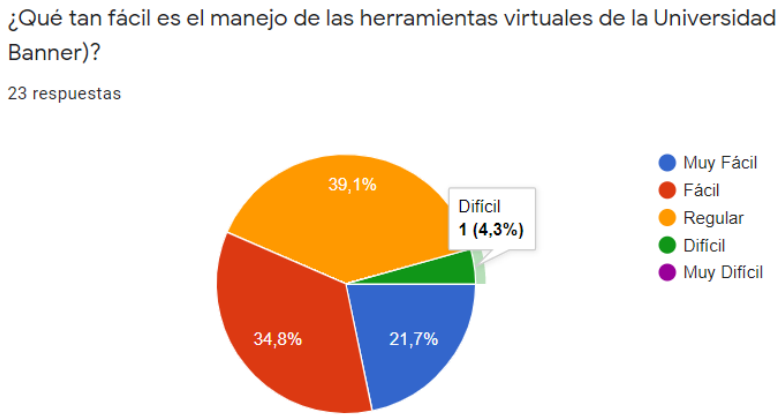


Figura 136: Encuesta 1: Pregunta 1.

El análisis de esta pregunta nos arroja que más del 40% del total de encuestados considera el manejo de las herramientas virtuales de la universidad entre regular y difícil.

La segunda pregunta realizada hace referencia a la facilidad que se tiene para acceder a los diferentes contenidos existentes en el aula virtual de la universidad, como se muestra en la Figura 137.

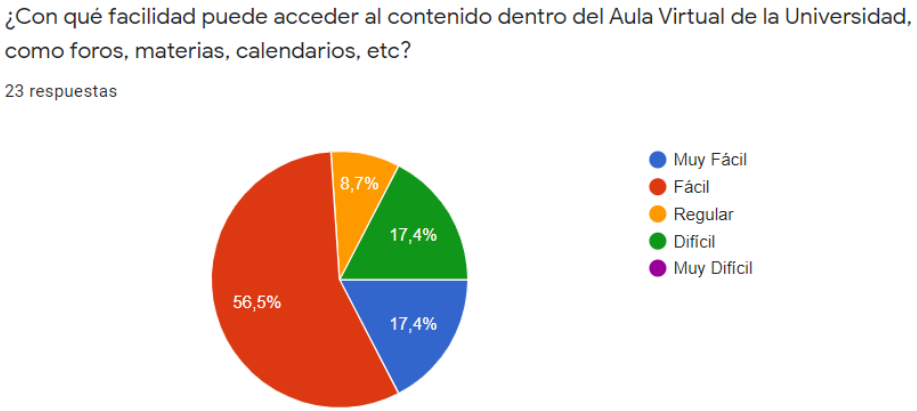


Figura 137: Encuesta 1: Pregunta 2.

Esta pregunta nos refleja que alrededor del 25% de los estudiantes encuestados se les dificulta encontrar y acceder al contenido existentes dentro del aula virtual de la universidad.

La tercera pregunta es similar a la anterior pero enfocada al contenido del servicio Banner de la universidad, como se observa en la Figura 138.

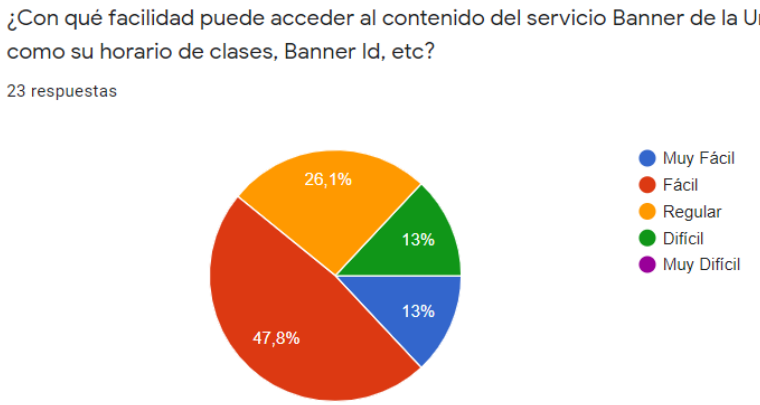


Figura 138: Encuesta 1: Pregunta 3.

Esta pregunta refleja que el 39% de los estudiantes encuestados tienen dificultades para acceder y encontrar el contenido existente dentro del servicio Banner de la universidad.

La cuarta pregunta tiene relación a la facilidad de manejo de estas herramientas y su contenido por separado, como se observa en la Figura 139.

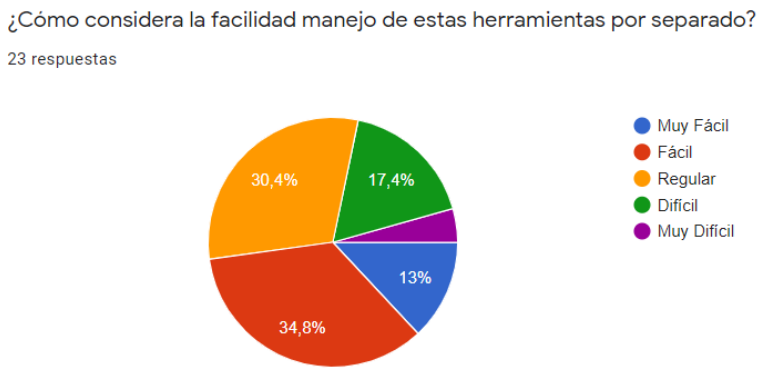


Figura 139: Encuesta 1: Pregunta 4.

Los resultados de esta pregunta nos dicen que más del 52 % de los estudiantes encuestados tienen dificultades para poder manejar los recursos existentes en estas herramientas por separado, entre regular y muy difícil; debido a que en la mayoría de los casos siempre necesitan estar en constante acceso a estos recursos.

La quinta pregunta se refiere a la posibilidad de acceder a los recursos de las herramientas existentes en la universidad dentro del mismo canal de comunicación y que esto ayude a mejorar la gestión educativa que se lleva a cabo dentro de ellas, como se muestra en la Figura 140.

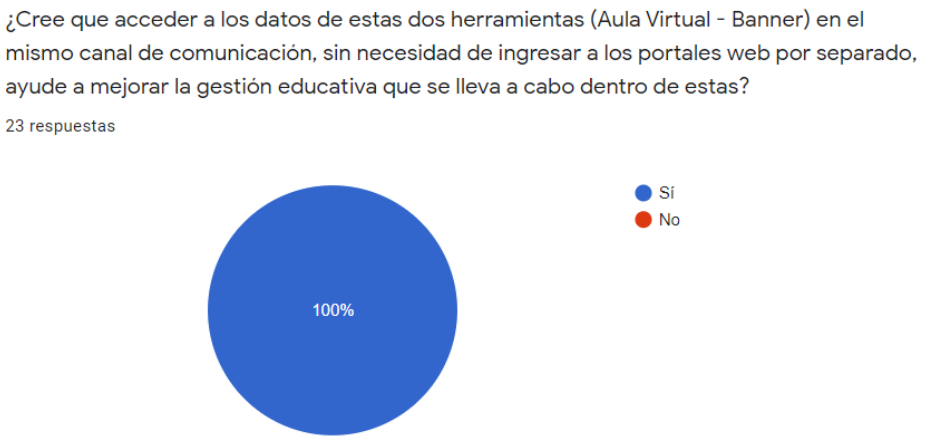


Figura 140: Encuesta 1: Pregunta 5.

El resultado de esta pregunta muestra que el 100% de los estudiantes encuestados considera que acceder a los recursos de estas herramientas dentro del mismo canal pude ayudar en la gestión que se lleva a cabo dentro de ellas.

En la sexta pregunta se hace referencia específica al uso de un asistente virtual conversacional que ayude al manejo de las herramientas de la universidad, como se muestra en la Figura 141.

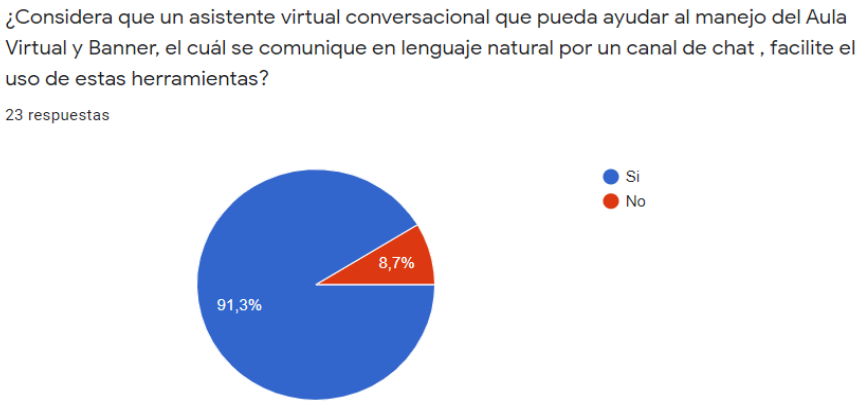


Figura 141: Encuesta 1: Pregunta 6.

El resultado de esta pregunta nos muestra que más del 91% de los estudiantes encuestados considera que un asistente conversacional que se comunique en lenguaje natural (chatbot), puede ayudar a facilitar el manejo de las herramientas virtuales de la universidad.

La séptima y última pregunta de esta encuesta hace referencia a la preferencia de los estudiantes por la ubicación de esta herramienta conversacional, como se muestra en la Figura 142.

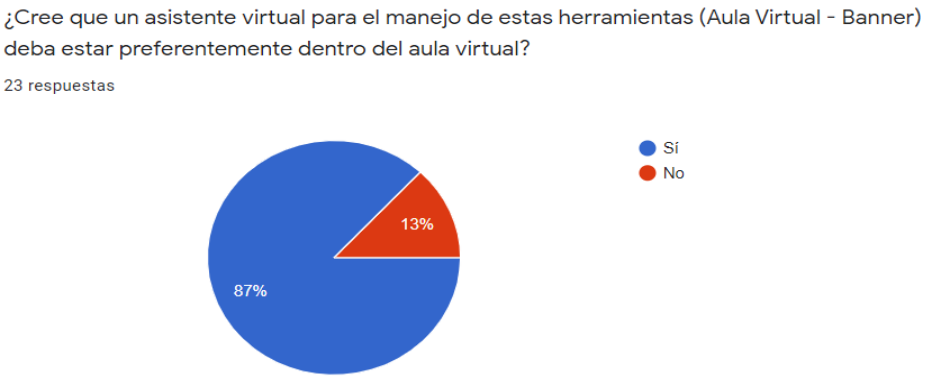


Figura 142: Encuesta 1: Pregunta 7.

El resultado de esta pregunta muestra que el 87% de los alumnos encuestados considera que este tipo de herramienta debe estar ubicada dentro del aula virtual de la universidad.

El análisis de esta encuesta indica la necesidad de un asistente virtual que pueda ayudar a la gestión que se lleva a cabo en las herramientas en línea de la universidad en el mismo canal de comunicación, además de que sea capaz de comunicarse en lenguaje natural con los usuarios y brinde un fácil acceso a los contenidos de estos recursos.

Una vez validada la necesidad de un asistente conversacional se procedió a probar la implementación de nuestro chatbot en un grupo de 12 estudiantes. Las pruebas de funcionalidad realizadas fueron ejecutadas dentro del aula virtual de la universidad para corroborar el funcionamiento adecuado del chatbot.

Para que “CHAPIE” despliegue todas sus funcionalidades el usuario debe tener una cuenta de acceso a los diferentes portales virtuales de la Universidad de las Américas (Quito-Ecuador).

“CHAPIE” puede entender diferentes expresiones en lenguaje natural, referentes a las opciones del menú que se despliega para cada usuario. En las pruebas realizadas, los usuarios ingresaron de diferentes maneras sus peticiones al sistema y fueron entendidas correctamente.

Culminadas las respectivas pruebas de funcionamiento se procedió a realizar una segunda encuesta al grupo de estudiantes que interactuaron con el chatbot “CHAPIE”. Esta encuesta tiene como fin verificar que este proyecto cumpla con los objetivos planteados.

Los resultados de los 12 estudiantes (100%) encuestados se muestran a continuación.

La pregunta 1 de esta encuesta se refiere a si los estudiantes que utilizaron esta herramienta consideran que la misma facilita la forma en la cual se accede a la información de las diferentes herramientas de la universidad, como se muestra en la Figura 143.

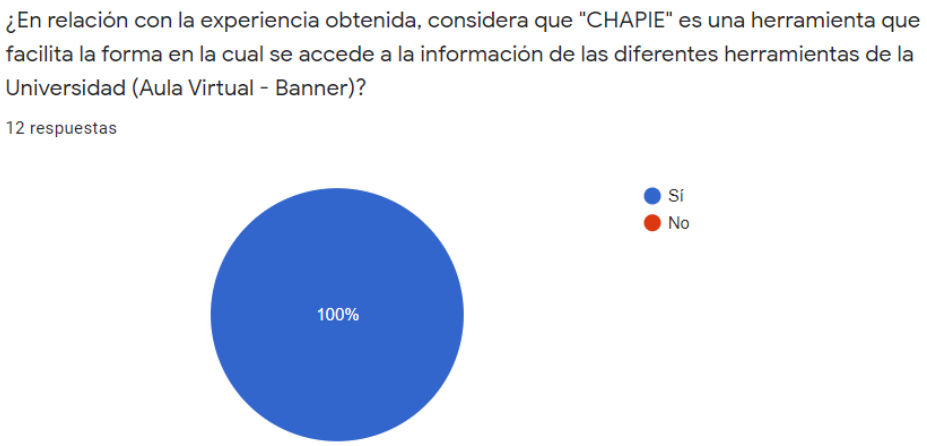


Figura 143: Encuesta 2: Pregunta 1.

El resultado de esta pregunta nos indica que el 100% de los estudiantes encuestados considera que “CHAPIE” facilita el acceso a la información de las diferentes herramientas virtuales de la universidad.

La pregunta 2 de esta encuesta hace referencia a la comunicación de “CHAPIE” con los usuarios en lenguaje natural, como se muestra en la Figura 144.

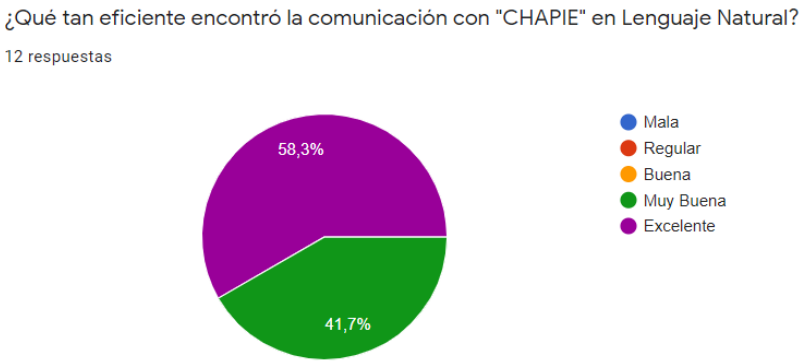


Figura 144: Encuesta 2: Pregunta 2.

El análisis de esta pregunta nos dice que todos los estudiantes encuestados consideran que la comunicación en Lenguaje Natural de “CHAPIE” oscila entre muy buena (47%) y excelente (58%).

La pregunta 3 de esta encuesta hace referencia a si “CHAPIE” facilita la gestión educativa que se lleva a cabo en las herramientas virtuales de la universidad, como se muestra en la Figura 145.

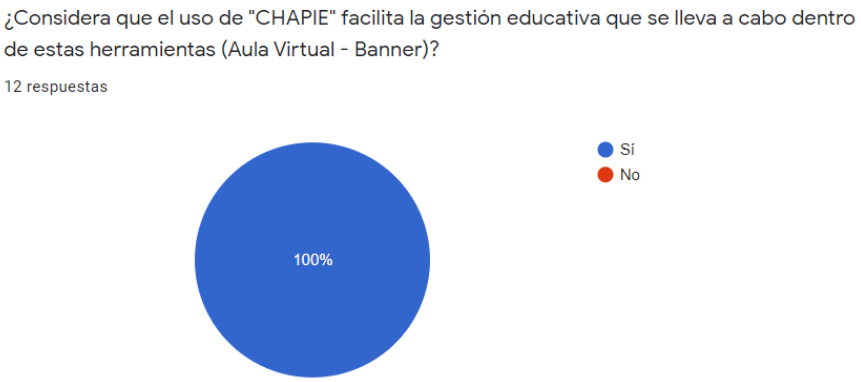


Figura 145: Encuesta 2: Pregunta 3.

El resultado de esta pregunta nos muestra que el 100% de los estudiantes encuestados considera que este chatbot ayuda en la gestión educativa que se lleva a cabo dentro de las herramientas virtuales de la universidad.

La pregunta 4 de esta encuesta hace referencia a si los usuarios consideran que los chatbots como “CHAPIE” pueden adaptarse para una atención más personalizada y enfocada en el aprendizaje, como se muestra en la Figura 146.

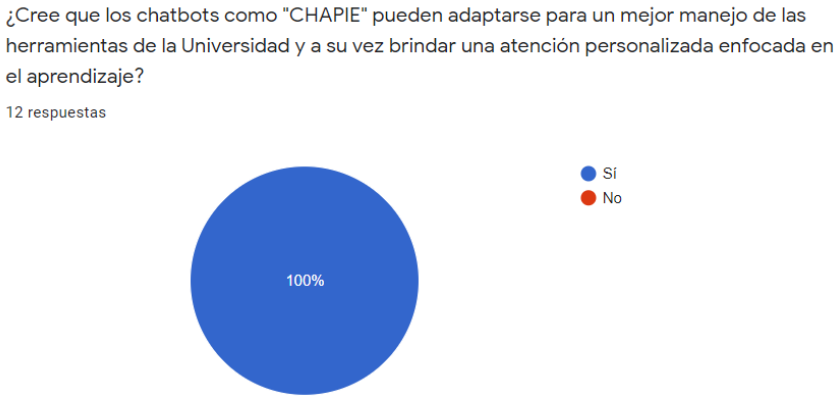


Figura 146: Encuesta 2: Pregunta 4.

El 100% de los estudiantes encuestados consideran que este tipo de tecnología puede mejorarse para un mejor desempeño, mucho más personalizado y enfocado en el aprendizaje.

El resultado de esta encuesta nos muestra el potencial de este proyecto para brindar ayuda en la gestión educativa que se lleva a cabo dentro de las herramientas virtuales de la universidad, además de, comunicarse con los usuarios en lenguaje natural de forma que ellos no tienen dificultad para utilizar este chatbot.

Este análisis refleja la importancia del desarrollo de este tipo de proyectos y a su vez, que estos pueden ayudar a agilitar los procesos y gestión que se lleva a cabo en las diferentes herramientas que la universidad posee. Igualmente, sugerir que las posibles futuras implementaciones de estas tecnologías puedan adaptarse para un mejor desempeño, mucho más personalizado y enfocado en la enseñanza.

5. CONCLUSIONES Y RECOMENDACIONES

CONCLUSIONES

Los chatbots, a medida que se mejoren y popularicen, son herramientas que habrá que considerar seriamente para posibles mejoras en temas educativos. Es una tecnología en alza, la cual tiene una mejora constante en el área de la inteligencia artificial, además de su uso progresivo en aplicaciones móviles; por lo que cada día será más común la interacción con los chatbots y que estos estén inmersos en el ámbito de la educación.

Este chatbot fue implementado en el aula virtual de la Universidad de las Américas, con la ayuda de la plataforma Microsoft Azure. La cual contiene en su nube pública la ubicación de los archivos de este proyecto y permite desplegarlo como un aplicativo web en el canal de comunicación deseado (aula virtual).

La importancia y necesidad para la implementación de proyectos similares a este trabajo de titulación se ve reflejado en una encuesta previa realizada, la cual demuestra que más del 40% de los estudiantes de la muestra tienen dificultades para el manejo de la información y contenido existente en las diferentes herramientas de la universidad.

A su vez, al consultar a los estudiantes sobre la implementación de un asistente conversacional para el manejo de estas herramientas en el mismo canal de comunicación, más del 91% está de acuerdo en que puede ayudar a facilitar la gestión que se realiza en las mismas.

Justificada la necesidad de un proyecto como “CHAPIE” y una vez probado en un grupo de estudiantes de la universidad. Por medio del análisis de una encuesta realizada acerca del uso de este chatbot en el ámbito educativo, podemos concluir que, el 100% de los estudiantes que utilizaron este chatbot considera que recibió una atención personalizada y directa; además de ayuda para el manejo y gestión educativa de los diferentes contenidos de las herramientas virtuales de la universidad.

Finalmente, esta muestra también refleja que la comprensión de “CHAPIE” y los usuarios en lenguaje natural oscila entre excelente (58%) y muy buna (47%). Por lo que su uso es bastante intuitivo.

RECOMENDACIONES

En base a la experiencia en el desarrollo de este chatbot, se recomienda para proyectos similares utilizar cualquier plataforma de despliegue que esté en la nube. Aunque tienen costo, mejoraría la disponibilidad del servicio, por ejemplo, en este caso Microsoft Azure.

Se podrían realizar algunos cambios para evitar el uso del teclado, ejecutando las interacciones con el usuario mediante la inclusión de audio. Esto podría optimizar la funcionalidad del chatbot.

Este proyecto tiene un amplio potencial de desarrollo debido a la facilidad de implementación y construcción de funcionalidades, por lo que podría mejorarse a futuro. Puede evolucionar a medida que se adquieran mayores conocimientos con respecto al tema y se pueda tener acceso a datos críticos de la universidad.

Se podrían aumentar funcionalidades específicas para cada usuario, como el asentamiento directo de notas, recordatorios, seguimientos personalizados, test, pruebas, etc. Sin embargo, para ello se requiere acceso a datos críticos de la universidad, los cuales son necesarios para la programación de estos requerimientos.

En el caso de que “CHAPIE” no pueda entender algunas expresiones en lenguaje natural referentes a sus funcionalidades, el administrador puede revisar estas expresiones y agregarlas en la “intención” correspondiente dentro del programa en “LUIS”. Así “CHAPIE” puede aprender estas nuevas expresiones y se podrán utilizar en la próxima interacción del usuario.

REFERENCIAS

Aron J. (2011). How innovative is Apple's new voice assistant, Siri? New Scientist, oct. 29, Recuperado de: https://doi.org/10.1016/S0262-4079(11)62647-X

Bernal R. D. (2020). Imagen de "Siri de Apple". Recuperado de: https://i.blogs.es/94ebf7/siri/1366_2000.jpg

Bii, P. (2013). "Chatbot technology: A possible means of unlocking student potential to learn how to learn". Educational Research, vol. 4, pp. 218-221.

Caravana C. (2017). Chatbot Elizza. Recuperado de: https://encrypted-tbn0.gstatic.com/images?q=tbn%3AANd9GcRJUBcv_hT6139lgQ54ECZ7YnekfVdqRi-bnktZoS_g9QuTg3hu&usqp=CAU

Cason H (2015). Imagen Chatbot "ALICE". Recuperado de: https://s1.dmcdn.net/v/AUIXU1Lc04KzyQ8CT/x1080

Cerdas, M. C. (2017). Historia de la Inteligencia artificial relacionada con los Chatbots. Sep. 1 Recuperado de: https://planetachatbot.com/historia-de-la-inteligencia-artificial-relacionada-con-los-chatbots-41a6cda22906

ChatCompose (2019). ChatBots para la educación: Aplicaciones y Beneficios. Recuperado de: https://www.chatcompose.com/chatbots-educacion.html

Delgado, J. (2017) Desarrollo de chatbot usando bot framework de Microsoft [En línea]. Recuperado de: http://revistaespirales.com/index.php/es/article/view/133

Ebling, M. R. (2016). Can Cognitive Assistants Disappear? IEEE Pervasive Computing, vol 15, pp. 4–6. Recuperado de:

https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=7508843

Farkash, Z. (2018). Education Chatbot: 4 ways chatbots are revolutionizing education. Chatbot Magazine., Mar. 6, Recuperado de: https://chatbotsmagazine.com/education-chatbot-4-ways-chatbots-arerevolutionizing-education-33f36627964c

García G., Fuertes M., Molas N. (2018). “Briefing paper: los chatbots en educación”. Universidad de Catalunya. Ryl. eLearn Center. Recuperado de: http://openaccess.uoc.edu/webapps/o2/bitstream/10609/85786/6/BRIEFING-PAPER-ES.pdf

Gartner (2016). Gartner Predicts a Virtual World of Exponential Change - Smarter With Gartner; Recuperado de: https://www.gartner.com/smarterwithgartner/gartner-predicts-a-virtual-world-of-exponentialchange/.

Glasgow, J., & Browse, R. (2002). Programming languages for artificial intelligence. Computers & Mathematics with Applications, vol 11, May. 20, pp. 431–448. Recuperado de: https://doi.org/10.1016/0898-1221(85)90049-5

Jia, J. (2003). The Study of the Application of a Keywords-based Chatbot System on the Teaching of Foreign Languages. Recuperado de: https://arxiv.org/abs/cs/0310018

Jo-Foley, M. (2013). ‘Cortana’: More on Microsoft’s next-generation personal assistant. ZDNet. Sep. 12. Recuperado de: http://www.zdnet.com/cortana-more-on-microsofts-next-generation-personal-assistant-7000020602/

Maruti Techlabs (2017). Why can chatbots replace Mobile Apps immediately? Maruti Techlabs, Recuperado de: https://www.marutitech.com/why-can-chatbots-replace-mobile-apps-immediately/.

McKinsey & Company (2016). Winning the expectations game in customer care. Recuperado de: https://www.mckinsey.com/business-functions/operations/our-insights/winning-theexpectations-game-in-customer-care.

Michael K. (2016), “Science Fiction Is Full of Bots That Hurt People: ... But these bots are here now.,” IEEE Consum. Electron. Mag., vol. 5, no. 4, pgs. 112–117.

Microsoft (2020). Visual Studio Enterprise. Recuperado de: https://visualstudio.microsoft.com/es/vs/enterprise/

Microsoft Azure (2019). Bot de conversación de nivel empresarial. Recuperado de: https://docs.microsoft.com/es-es/azure/architecture/reference-architectures/ai/conversational-bot

Microsoft Build (2020). ¿Qué es Language Understanding (LUIS)? Digital Event. May 19. Recuperado de: https://docs.microsoft.com/es-es/azure/cognitive-services/luis/what-is-luis

Microsoft, (2019). Welcome to the Visual Studio IDE. Mar 19. Recuperado de: https://docs.microsoft.com/en-us/visualstudio/get-started/visual-studio-ide?view=vs-2019

Microsoft. Bot Framework. (2017). Debug bots with the Bot Framework Emulator. [En línea] Recuperado de: https://www.theseus.fi/bitstream/handle/10024/142561/thesis.pdf

Núñez P. (2018). Imagen sobre “Alexa”. Recuperado de: http://www.uegmobile.com/wp-content/uploads/2019/01/Home-Amazon-Hero-1.png

Olson, C (2016). Just say it: The future of search is voice and personal digital assistants. Recuperado de https://www.campaignlive.co.uk/article/just-say-it-future-search-voice-personal-digitalassistants/1392459, 06 Apr 2020.

Palacios J. (2018). Imagen de “Clippy”. Recuperado de: https://i1.wp.com/www.sopitas.com/wp-content/uploads/2018/07/clipo.jpg

Pereira J., Medina H., Díaz Ó. (2016). “Uso de Chatbots en la Docencia Universitaria”, TICAI 2016: TICs para el Aprendizaje de la Ingeniería. pp. 97-103.

Redacción APD (2019). ¿Qué es Machine Learning y cómo funciona? Mar. 4., Recuperado de: https://www.apd.es/que-es-machine-learning/

Reddy, D. R. (1976). Speech recognition by machine: a review. Proceedings of the IEEE, vol 64, pp. 501–531. https://doi.org/10.1109/PROC.1976.10158

Reyna A. (2018). La inteligencia artificial acelera la evolución de los ‘chatbots’. BBVA Inteligencia Artificial. Recuperado de: https://www.bbva.com/es/inteligencia-artificial-acelera-evolucion-chatbots/

Rouse, M. (2020). Microsoft Azure. SearchCloudComputing. April. Recuperado de: https://searchcloudcomputing.techtarget.com/definition/Windows-Azure

Sabán A. (2019). Imagen sobre Microsoft “Cortana”. Recuperado de: https://i.blogs.es/1001cf/cortana/1366_2000.jpg

Sansonnet, J.-P., Leray, D., & Martin, J.-C. (2006). Architecture of a Framework for Generic Assisting Conversational Agents. Intelligent Virtual Agents Lecture Notes in Computer Science, pp 145–156. Recuperado de: http://doi.org/10.1007/11821830_12

Saygin, A., Cicekli, I., & Akman, V. (2000). Turing Test: 50 years later. MINDS AND MACHINES, vol 10, pp. 463–518. Recuperado de: https://planetachatbot.com/historia-de-la-inteligencia-artificial-relacionada-con-los-chatbots-41a6cda22906

Serban, I. V., Sankar, C., Germain, M., Zhang, S., Lin, Z., Subramanian, S., Kim, T., Pieper, M., Chandar, S., & Ke, N. R. (2017). A deep reinforcement learning chatbot. Nov 5. Recuperado de: https://arxiv.org/pdf/1709.02349.pdf

Sharoon E. (2019). La línea de tiempo de la historia de los Chatbots: antes, ahora y mañana. Planeta Chatbot. Aug 21. Recuperado de: https://planetachatbot.com/linea-tiempo-historia-de-chatbots-antes-ahora-y-manana-18a46fedc9cf

Smart Panel (2020). ¿Qué es el Deep Learning? Apr 10. Recuperado de: https://www.smartpanel.com/que-es-deep-learning/

Statista (2017). Chatbot Market size 2015-2024; Recuperado de: https://www.statista.com/statistics/656596/worldwidechatbot-market

Strauss, D. (2020). Conociendo Visual Studio 2019. En Comenzando con Visual Studio 2019 (pp. 1-60). Apress, Berkeley, CA. Disponible en: https://link.springer.com/chapter/10.1007/978-1-4842-5449-3_1

Tecuci G. (2012). Artificial Intelligence. WIREs Computational Statistics, Volume 4, Issue 2, March/April, pp 168-180.

Turing, A. M. (1950). Computing Machinery And Intelligence. Mind. Oxford Academic. pp. 433–460. Recuperado de: http://doi.org/10.1093/mind/lix.236.433

Turrado. J (2019). ¿Qué hay de nuevo en Visual Studio 2019? Recuperado de: https://www.campusmvp.es/recursos/post/que-hay-de-nuevo-en-visual-studio-2019.aspx

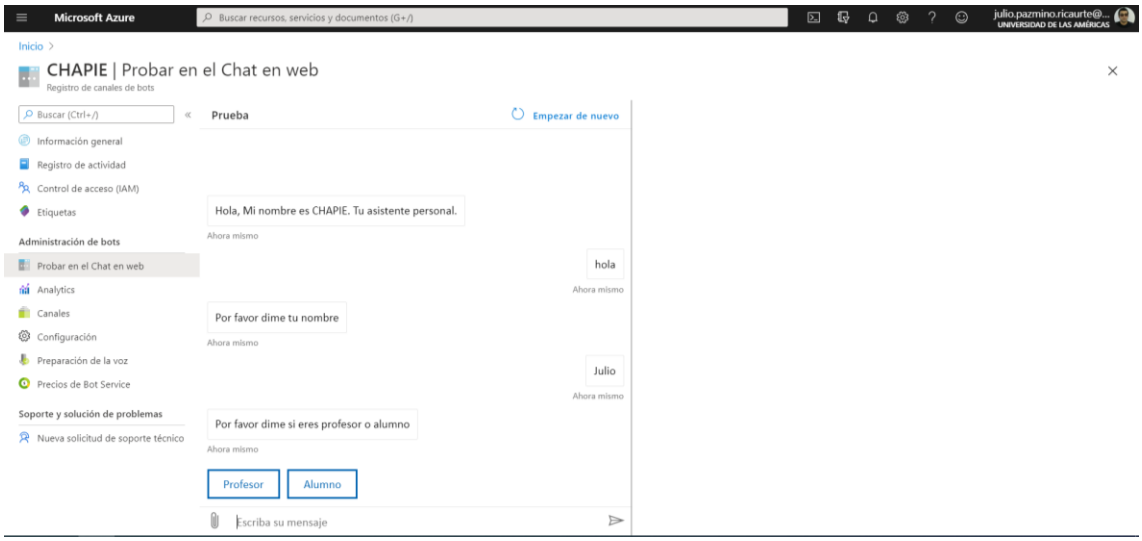
Wallace, R. S. The Anatomy of A.L.I.C.E. Recuperado de: http://www.alicebot.org/anatomy.html

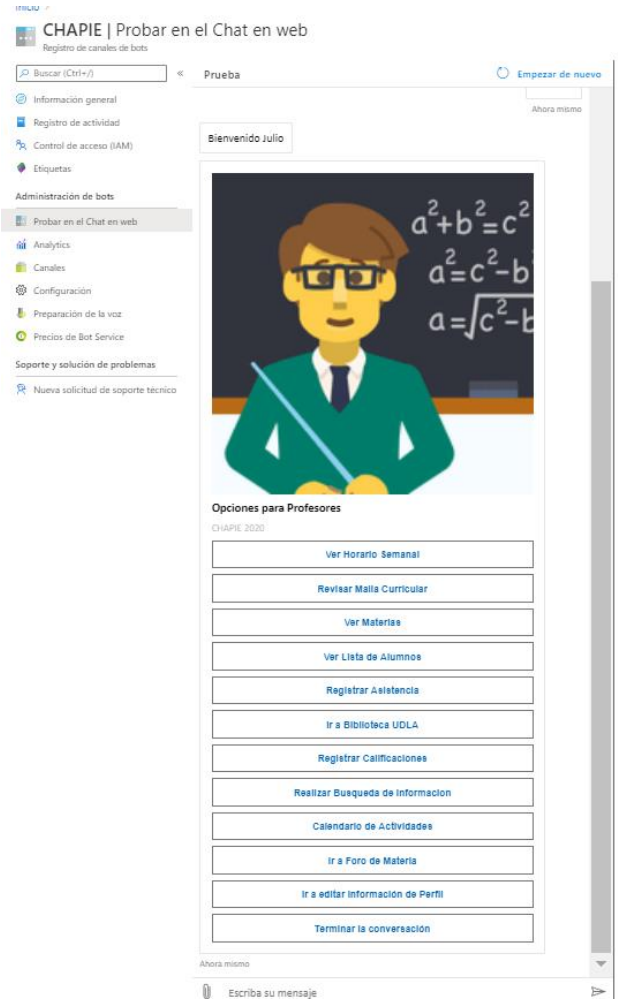
Weizenbaum, J. (1966). ELIZA---a computer program for the study of natural language communication between man and machine. Communications of the ACM, vol 9, Jan. pp. 36–45. http://doi.org/10.1145/365153.365168

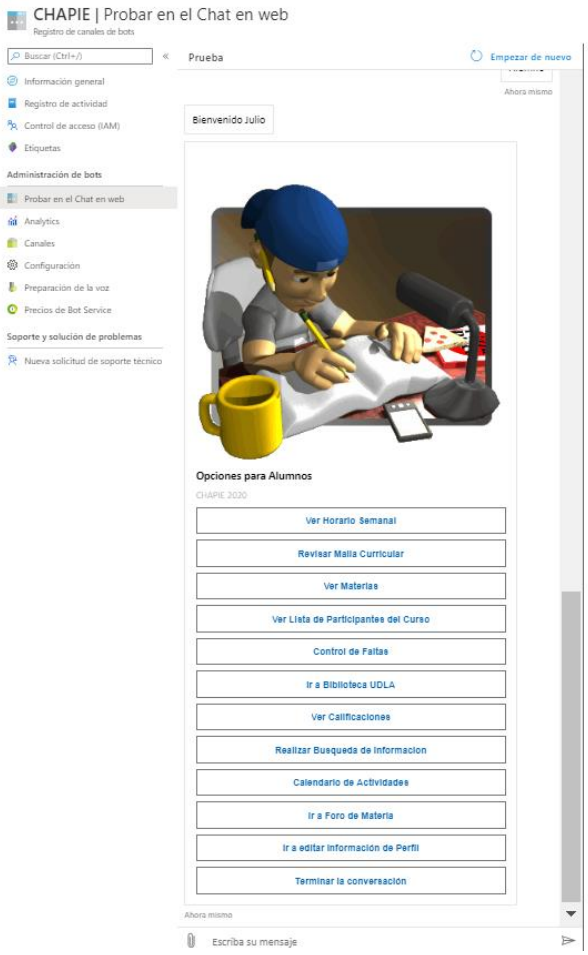
Wikipedia (2008), Modelo incremental. Recuperado de: https://es.wikipedia.org/wiki/Archivo:Modelo_Iterativo_Incremental.jpg

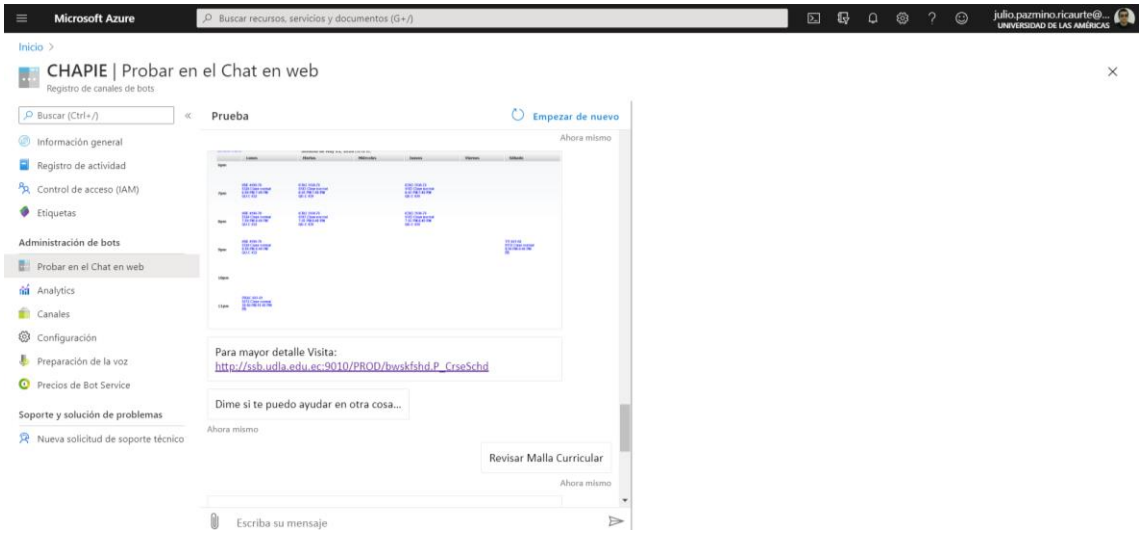
ANEXOS

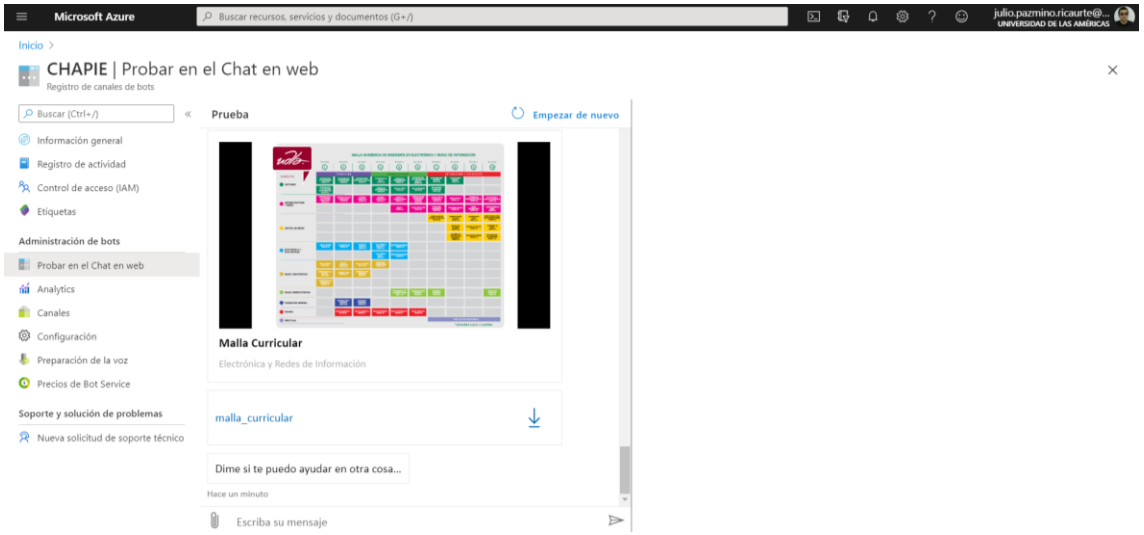
ANEXO 1. PRUEBAS DE “CHAPIE” EN LA WEB DE MICROSOFT AZURE

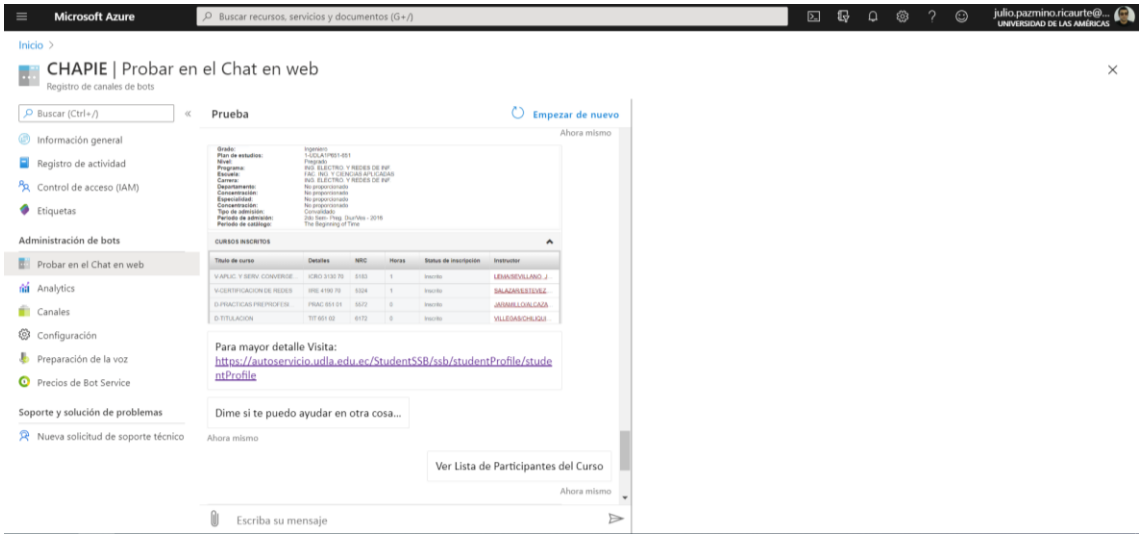


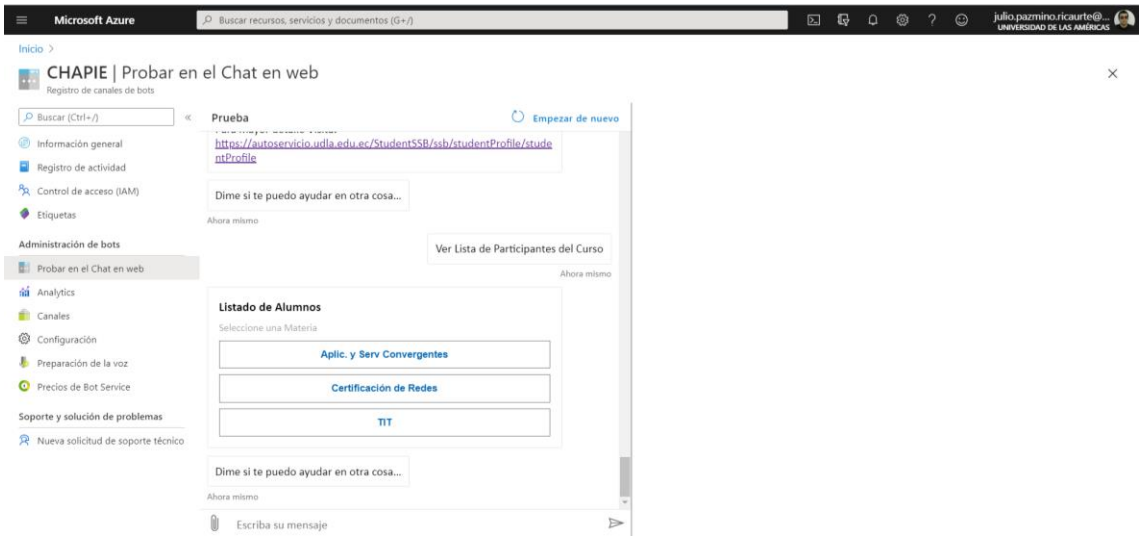


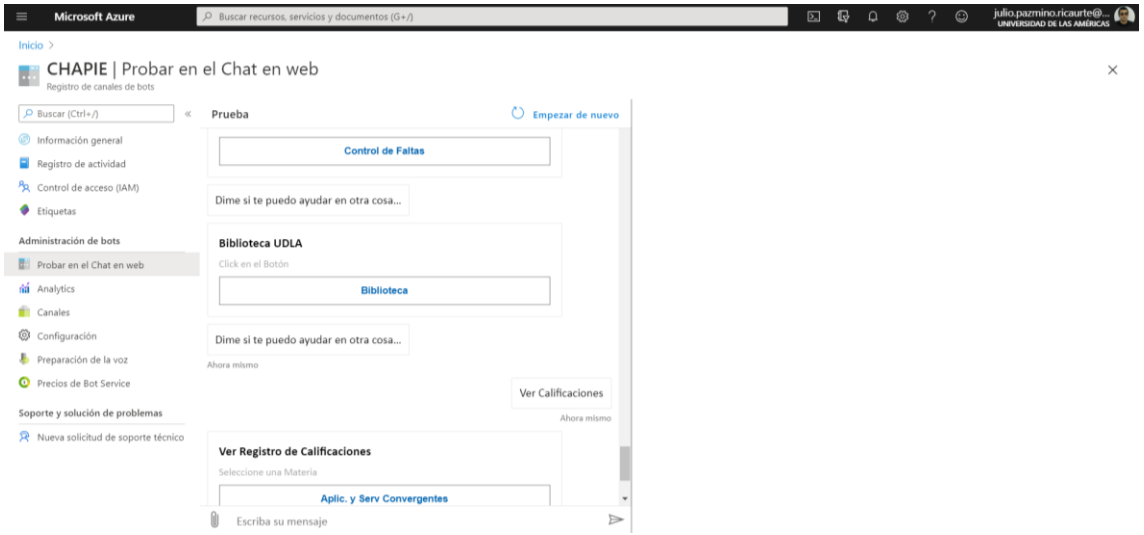

CHAPIE | Probar en el Chat en web
Registro de canales de bots
Buscar (Ctrl+/)
Información general
Registro de actividad
Control de acceso (IAM)
Etiquetas
Administración de bots
Probar en el Chat en web
Analytics
Canales
Configuración
Preparación de la voz
Precios de Bot Service
Soporte y solución de problemas
Nueva solicitud de soporte técnico
Prueba
Empezar de nuevo
Ahora mismo
Bienvenido Julio
Opciones para Alumnos
CHAPIE 2020
Ver Horario Semanal
Revisar Malla Curricular
Ver Materias
Ver Lista de Participantes del Curso
Control de Faltas
Ir a Biblioteca UDLA
Ver Calificaciones
Realizar Busqueda de Informacion
Calendario de Actividades
Ir a Foro de Materia
Ir a editar información de Perfil
Terminar la conversación
Ahora mismo
Escriba su mensaje

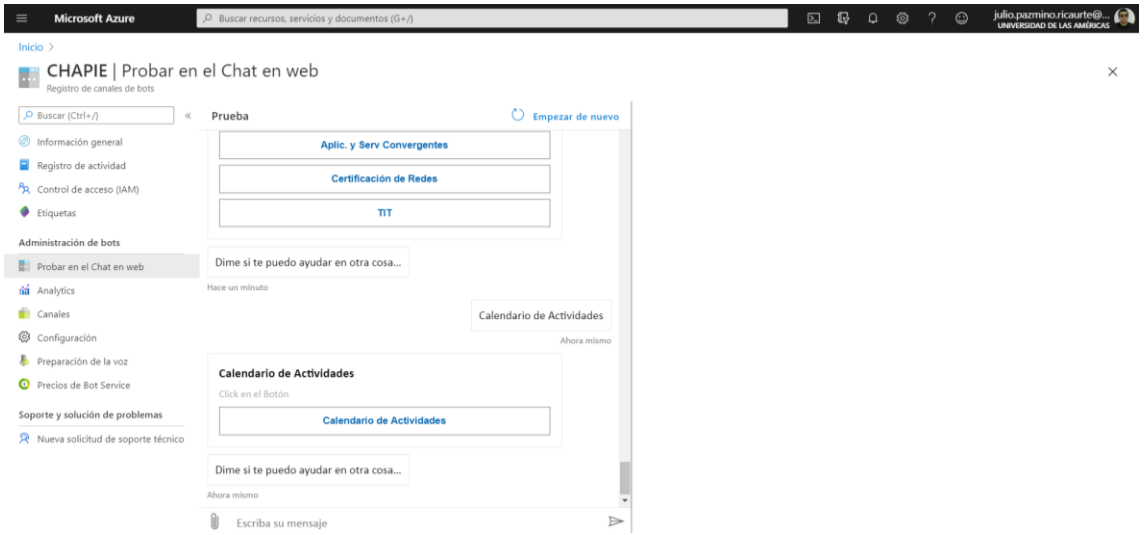

Microsoft Azure
Buscar recursos, servicios y documentos (G+/)
julio.pazmino.ricaurte@...
UNIVERSIDAD DE LAS AMÉRICAS
Inicio
CHAPIE | Probar en el Chat en web
Registro de canales de bots
Buscar (Ctrl+/)
Información general
Registro de actividad
Control de acceso (IAM)
Etiquetas
Administración de bots
Probar en el Chat en web
Analytics
Canales
Configuración
Preparación de la voz
Precios de Bot Service
Soporte y solución de problemas
Nueva solicitud de soporte técnico
Prueba
Empezar de nuevo
Ahora mismo
Para mayor detalle Visita:
http://ssb.udla.edu.ec:9010/PROD/bwskfshd.P_CrseSchd
Dime si te puedo ayudar en otra cosa...
Ahora mismo
Revisar Malla Curricular
Ahora mismo
Escriba su mensaje

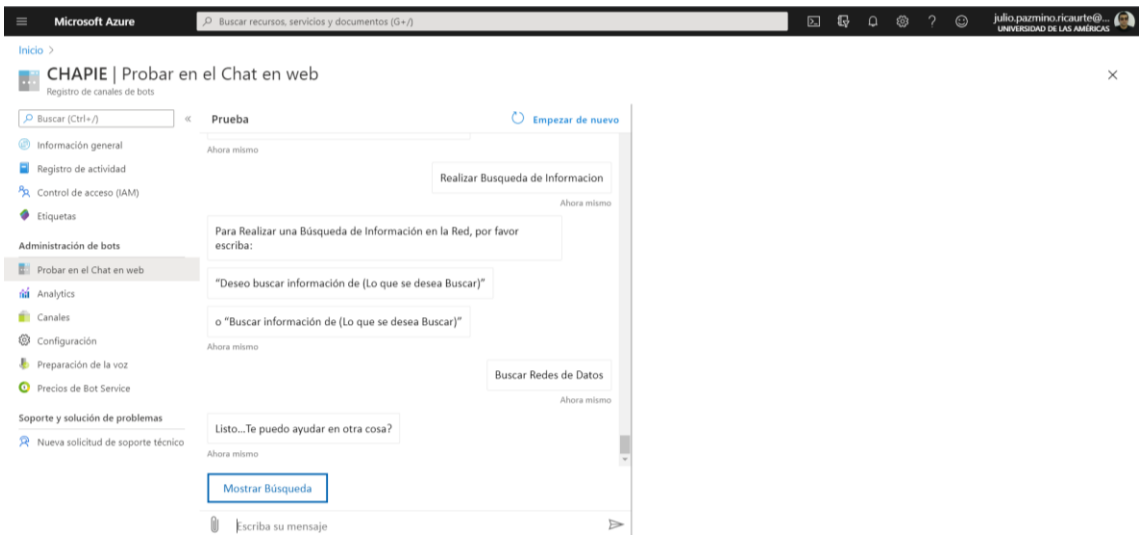

Microsoft Azure
Buscar recursos, servicios y documentos (G+/)
julio.pazmino.ricaurte@...
UNIVERSIDAD DE LAS AMÉRICAS
Inicio
CHAPIE | Probar en el Chat en web
Registro de canales de bots
Buscar (Ctrl+/)
Información general
Registro de actividad
Control de acceso (IAM)
Etiquetas
Administración de bots
Probar en el Chat en web
Analytics
Canales
Configuración
Preparación de la voz
Precios de Bot Service
Soporte y solución de problemas
Nueva solicitud de soporte técnico
Prueba
Empezar de nuevo
Malla Curricular
Electrónica y Redes de Información
malla_curricular
Dime si te puedo ayudar en otra cosa...
Hace un minuto
Escriba su mensaje

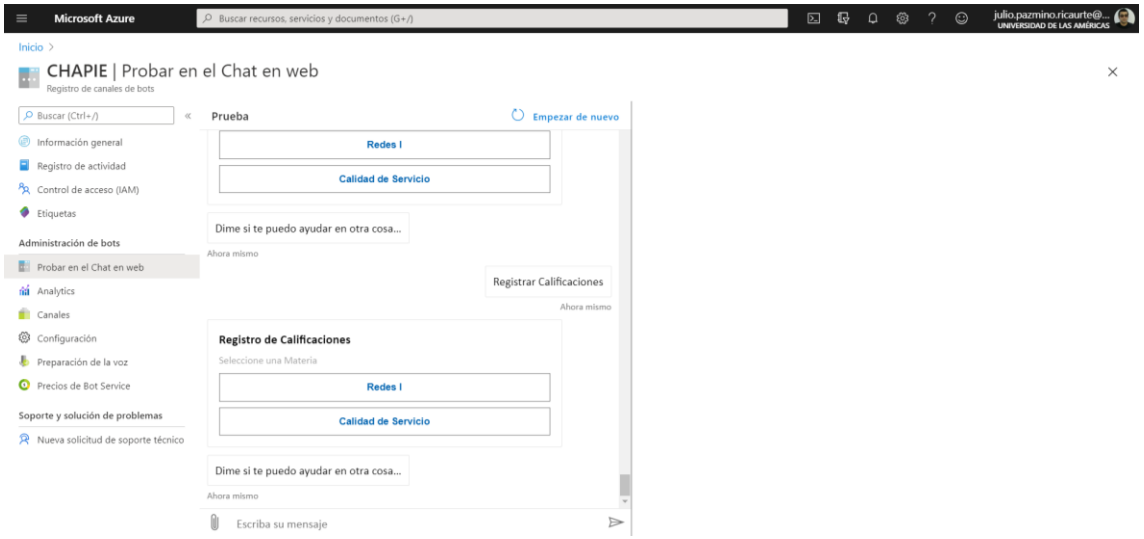

Microsoft Azure
Buscar recursos, servicios y documentos (G+/)
julio.pazmino.ricaurte@...
UNIVERSIDAD DE LAS AMÉRICAS
Inicio
CHAPIE | Probar en el Chat en web
Registro de canales de bots
Buscar (Ctrl+/)
Información general
Registro de actividad
Control de acceso (IAM)
Etiquetas
Administración de bots
Probar en el Chat en web
Analytics
Canales
Configuración
Preparación de la voz
Precios de Bot Service
Soporte y solución de problemas
Nueva solicitud de soporte técnico
Prueba
Empezar de nuevo
Ahora mismo
CURSOS INSCRITOS
Para mayor detalle Visita:
https://autoservicio.udla.edu.ec/StudentSSB/ssb/studentProfile/studentProfile
Dime si te puedo ayudar en otra cosa...
Ahora mismo
Ver Lista de Participantes del Curso
Ahora mismo
Escriba su mensaje

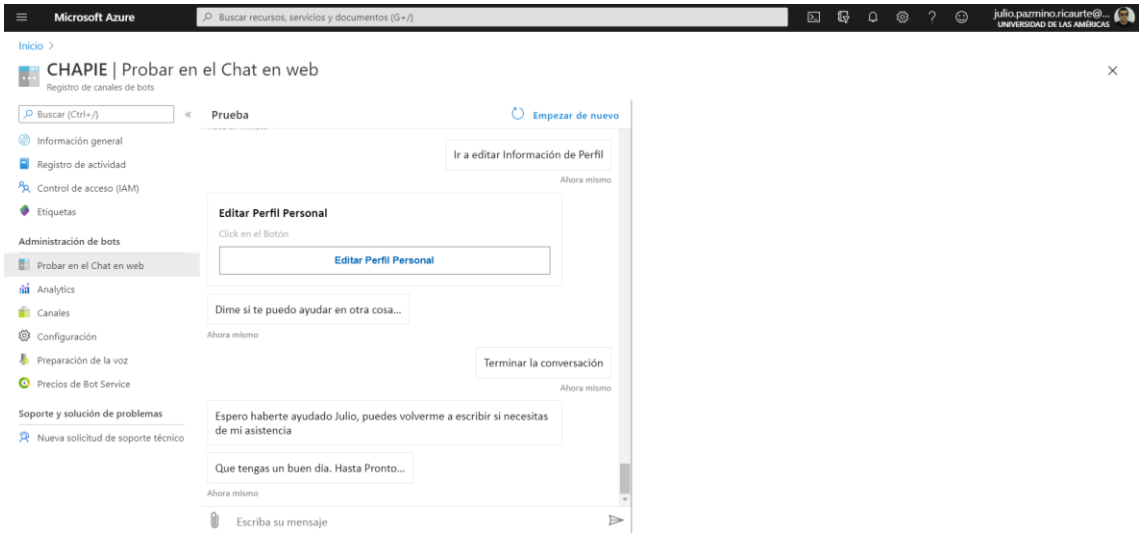

Microsoft Azure
Buscar recursos, servicios y documentos (G+/)
julio.pazmino.ricaurte@...
UNIVERSIDAD DE LAS AMÉRICAS
Inicio
CHAPIE | Probar en el Chat en web
Registro de canales de bots
Buscar (Ctrl+/)
Información general
Registro de actividad
Control de acceso (IAM)
Etiquetas
Administración de bots
Probar en el Chat en web
Analytics
Canales
Configuración
Preparación de la voz
Precios de Bot Service
Soporte y solución de problemas
Nueva solicitud de soporte técnico
Prueba
Empezar de nuevo
https://autoservicio.udla.edu.ec/StudentSSB/ssb/studentProfile/studentProfile
Dime si te puedo ayudar en otra cosa...
Ahora mismo
Ver Lista de Participantes del Curso
Ahora mismo
Listado de Alumnos
Seleccione una Materia
Aplic. y Serv Convergentes
Certificación de Redes
TIT
Dime si te puedo ayudar en otra cosa...
Ahora mismo
Escriba su mensaje


Microsoft Azure
Buscar recursos, servicios y documentos (G+/)
julio.pazmino.ricaurte@...
UNIVERSIDAD DE LAS AMÉRICAS
Inicio
CHAPIE | Probar en el Chat en web
Registro de canales de bots
Buscar (Ctrl+/)
Información general
Registro de actividad
Control de acceso (IAM)
Etiquetas
Administración de bots
Probar en el Chat en web
Analytics
Canales
Configuración
Preparación de la voz
Precios de Bot Service
Soporte y solución de problemas
Nueva solicitud de soporte técnico
Prueba
Empezar de nuevo
Control de Faltas
Dime si te puedo ayudar en otra cosa...
Biblioteca UDLA
Click en el Botón
Biblioteca
Dime si te puedo ayudar en otra cosa...
Ahora mismo
Ver Calificaciones
Ahora mismo
Ver Registro de Calificaciones
Seleccione una Materia
Aplic. y Serv Convergentes
Escriba su mensaje

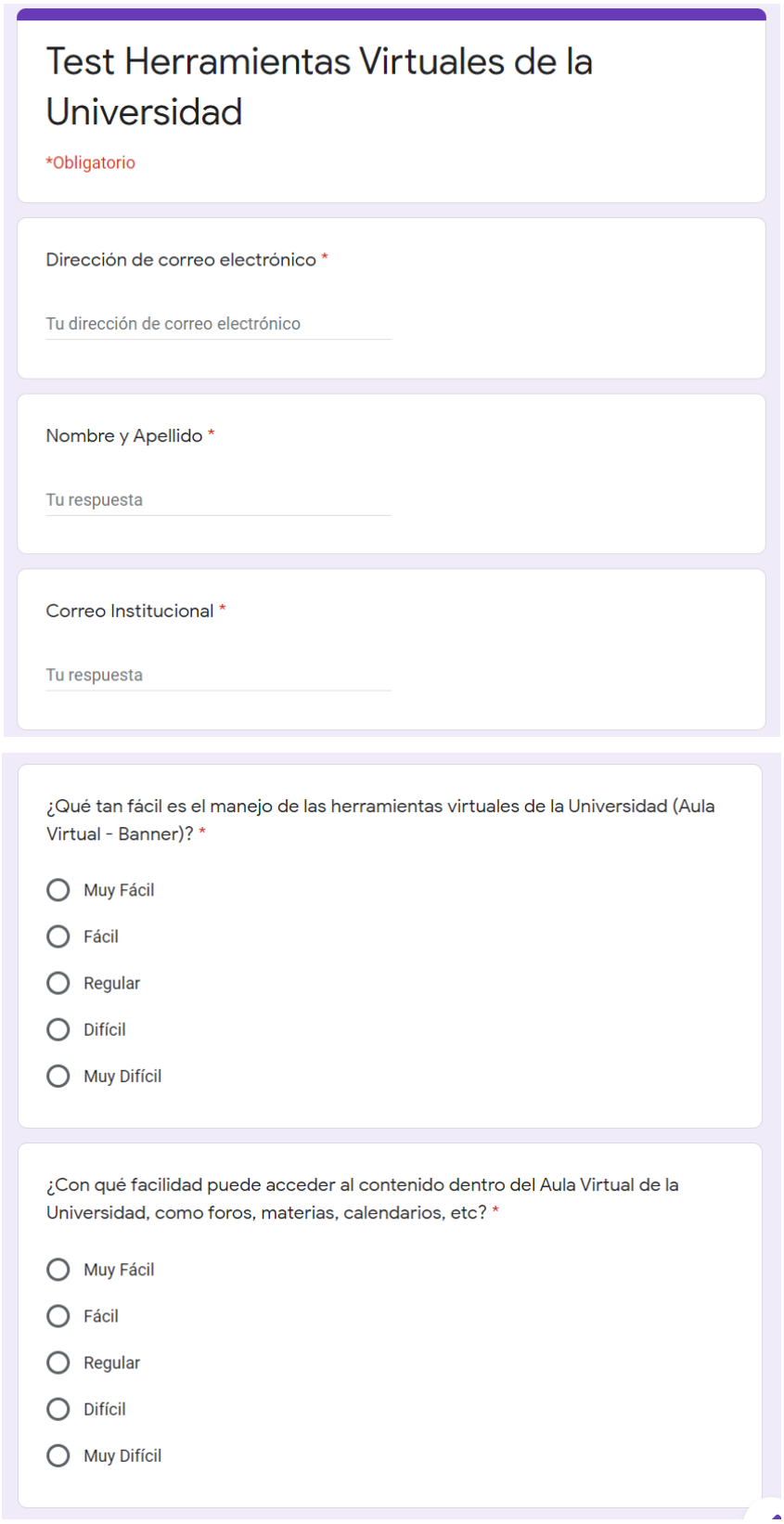

Microsoft Azure
Buscar recursos, servicios y documentos (G+/)
julio.pazmino.ricaurte@...
UNIVERSIDAD DE LAS AMÉRICAS
Inicio
CHAPIE | Probar en el Chat en web
Registro de canales de bots
Buscar (Ctrl+/)
Información general
Registro de actividad
Control de acceso (IAM)
Etiquetas
Administración de bots
Probar en el Chat en web
Analytics
Canales
Configuración
Preparación de la voz
Precios de Bot Service
Soporte y solución de problemas
Nueva solicitud de soporte técnico
Prueba
Empezar de nuevo
Aplic. y Serv Convergentes
Certificación de Redes
TIT
Dime si te puedo ayudar en otra cosa...
Hace un minuto
Calendario de Actividades
Ahora mismo
Calendario de Actividades
Click en el Botón
Calendario de Actividades
Dime si te puedo ayudar en otra cosa...
Ahora mismo
Escriba su mensaje


Microsoft Azure
Buscar recursos, servicios y documentos (G+/)
julio.pazmino.ricaurte@...
UNIVERSIDAD DE LAS AMÉRICAS
Inicio
CHAPIE | Probar en el Chat en web
Registro de canales de bots
Buscar (Ctrl+/)
Información general
Registro de actividad
Control de acceso (IAM)
Etiquetas
Administración de bots
Probar en el Chat en web
Analytics
Canales
Configuración
Preparación de la voz
Precios de Bot Service
Soporte y solución de problemas
Nueva solicitud de soporte técnico
Prueba
Empezar de nuevo
Ahora mismo
Realizar Busqueda de Informacion
Ahora mismo
Para Realizar una Búsqueda de Información en la Red, por favor escriba:
"Deseo buscar información de (Lo que se desea Buscar)"
o "Buscar información de (Lo que se desea Buscar)"
Ahora mismo
Buscar Redes de Datos
Ahora mismo
Listo...Te puedo ayudar en otra cosa?
Ahora mismo
Mostrar Búsqueda
Escriba su mensaje


Microsoft Azure
Buscar recursos, servicios y documentos (G+/)
julio.pazmino.ricaurte@...
UNIVERSIDAD DE LAS AMÉRICAS
Inicio
CHAPIE | Probar en el Chat en web
Registro de canales de bots
Buscar (Ctrl+/)
Información general
Registro de actividad
Control de acceso (IAM)
Etiquetas
Administración de bots
Probar en el Chat en web
Analytics
Canales
Configuración
Preparación de la voz
Precios de Bot Service
Soporte y solución de problemas
Nueva solicitud de soporte técnico
Prueba
Empezar de nuevo
Redes I
Calidad de Servicio
Dime si te puedo ayudar en otra cosa...
Ahora mismo
Registrar Calificaciones
Ahora mismo
Registro de Calificaciones
Seleccione una Materia
Redes I
Calidad de Servicio
Dime si te puedo ayudar en otra cosa...
Ahora mismo
Escriba su mensaje


Microsoft Azure
Buscar recursos, servicios y documentos (G+/)
julio.pazmino.ricaurte@...
UNIVERSIDAD DE LAS AMÉRICAS
Inicio
CHAPIE | Probar en el Chat en web
Registro de canales de bots
Buscar (Ctrl+/)
Información general
Registro de actividad
Control de acceso (IAM)
Etiquetas
Administración de bots
Probar en el Chat en web
Analytics
Canales
Configuración
Preparación de la voz
Precios de Bot Service
Soporte y solución de problemas
Nueva solicitud de soporte técnico
Prueba
Empezar de nuevo
Ir a editar Información de Perfil
Ahora mismo
Editar Perfil Personal
Click en el Botón
Editar Perfil Personal
Dime si te puedo ayudar en otra cosa...
Ahora mismo
Terminar la conversación
Ahora mismo
Espero haberte ayudado Julio, puedes volverme a escribir si necesitas de mi asistencia
Que tengas un buen día. Hasta Pronto...
Ahora mismo
Escriba su mensaje

ANEXO 2. ENCUESTA 1: TEST HERRAMIENTAS VIRTUALES DE LA UNIVERSIDAD



Test Herramientas Virtuales de la Universidad

*Obligatorio

Dirección de correo electrónico *

Tu dirección de correo electrónico

Nombre y Apellido *

Tu respuesta

Correo Institucional *

Tu respuesta

¿Qué tan fácil es el manejo de las herramientas virtuales de la Universidad (Aula Virtual - Banner)? *

- Muy Fácil
- Fácil
- Regular
- Difícil
- Muy Difícil

¿Con qué facilidad puede acceder al contenido dentro del Aula Virtual de la Universidad, como foros, materias, calendarios, etc? *

- Muy Fácil
- Fácil
- Regular
- Difícil
- Muy Difícil

¿Con qué facilidad puede acceder al contenido del servicio Banner de la Universidad, como su horario de clases, Banner Id, etc? *

- ○ Muy Fácil
- ○ Fácil
- ○ Regular
- ○ Difícil
- ○ Muy Difícil

¿Cómo considera la facilidad manejo de estas herramientas por separado? *

- ○ Muy Fácil
- ○ Fácil
- ○ Regular
- ○ Difícil
- ○ Muy Difícil

¿Cree que acceder a los datos de estas dos herramientas (Aula Virtual - Banner) en el mismo canal de comunicación, sin necesidad de ingresar a los portales web por separado, ayude a mejorar la gestión educativa que se lleva a cabo dentro de estas? *

- ○ Sí
- ○ No

¿Considera que un asistente virtual conversacional que pueda ayudar al manejo del Aula Virtual y Banner, el cuál se comunique en lenguaje natural por un canal de chat , facilite el uso de estas herramientas? *

- ○ Si
- ○ No

¿Cree que un asistente virtual para el manejo de estas herramientas (Aula Virtual - Banner) deba estar preferentemente dentro del aula virtual? *

- ○ Sí
- ○ No

ANEXO 3. RESULTADOS DE ENCUESTA 1

Test Herramientas Virtuales de la Universidad

23 respuestas

Publicar datos de análisis

Nombre y Apellido

23 respuestas

Gabriel Caiza

Alexander Caiza

Jimmy Masapanta

Kevin

Marco Sánchez

Luis Chacón

Marco Ricaurte

Francisco Balseca

Jose Guambo

Correo Institucional

23 respuestas

gabriel.caiza@udla.edu.ec

alexander.caiza@udla.edu.ec

jimmy.masapanta@udla.edu.ec

kevin.perez@udla.edu.ec

marco.sanchez.colon@udla.edu.ec

luis.chacon.alvarez@udla.edu.ec

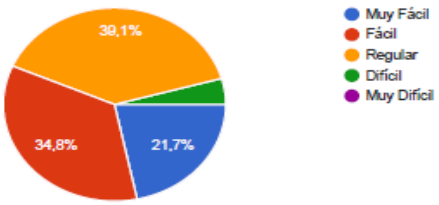
marco.ricaurte.lopez@udla.edu.ec

francisco.balseca@udla.edu.ec

jose.guambo@udla.edu.ec

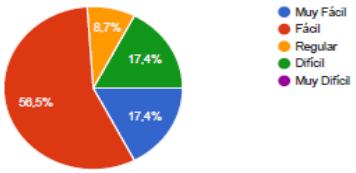
¿Qué tan fácil es el manejo de las herramientas virtuales de la Universidad (Aula Virtual - Banner)?

23 respuestas



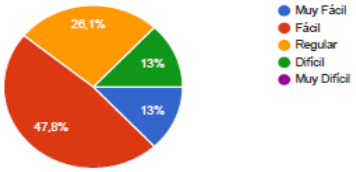
¿Con qué facilidad puede acceder al contenido dentro del Aula Virtual de la Universidad, como foros, materias, calendarios, etc?

23 respuestas



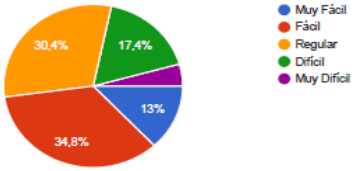
¿Con qué facilidad puede acceder al contenido del servicio Banner de la Universidad, como su horario de clases, Banner Id, etc?

23 respuestas



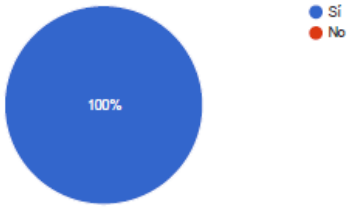
¿Cómo considera la facilidad manejo de estas herramientas por separado?

23 respuestas



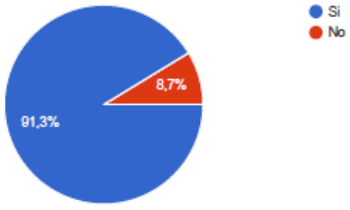
¿Cree que acceder a los datos de estas dos herramientas (Aula Virtual - Banner) en el mismo canal de comunicación, sin necesidad de ingresar a los portales web por separado, ayude a mejorar la gestión educativa que se lleva a cabo dentro de estas?

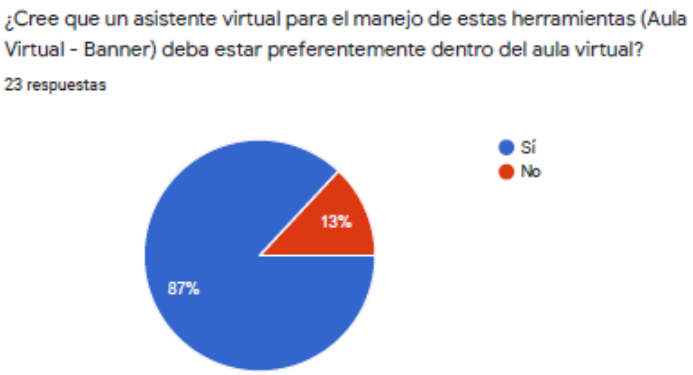
23 respuestas



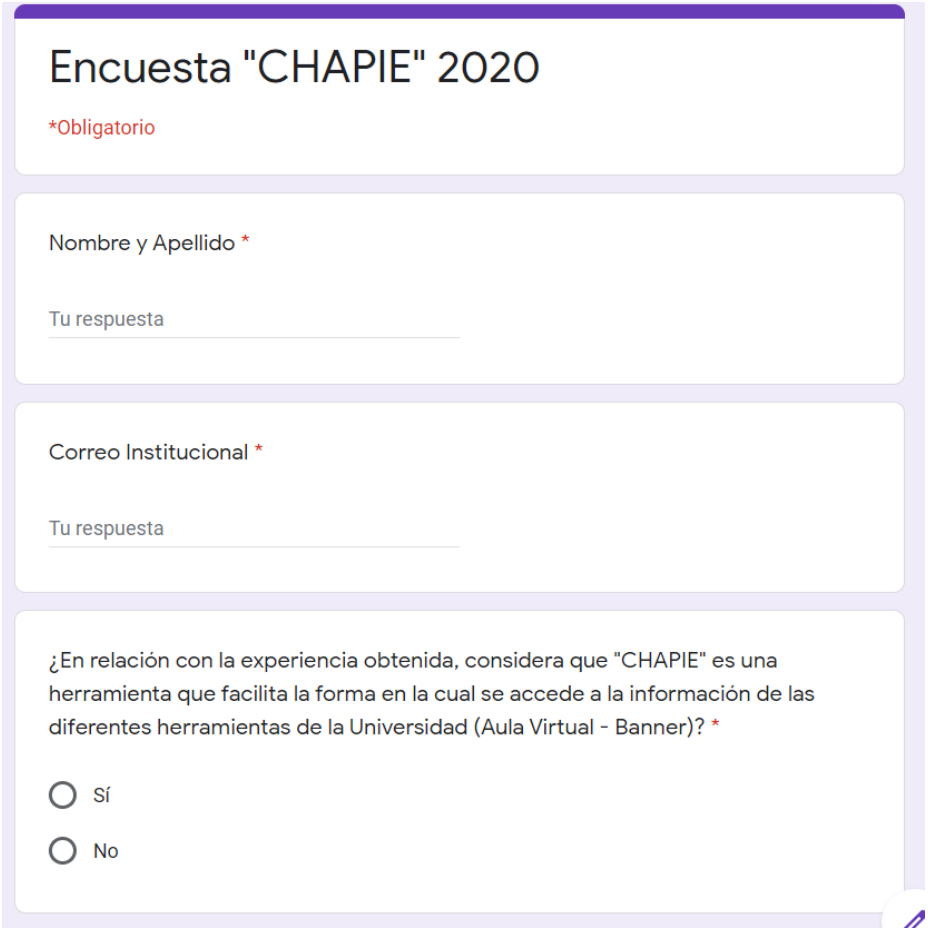
¿Considera que un asistente virtual conversacional que pueda ayudar al manejo del Aula Virtual y Banner, el cuál se comunique en lenguaje natural por un canal de chat , facilite el uso de estas herramientas?

23 respuestas





ANEXO 4. ENCUESTA 2: ENCUESTA "CHAPIE" 2020



Encuesta "CHAPIE" 2020

*Obligatorio

Nombre y Apellido *

Tu respuesta

Correo Institucional *

Tu respuesta

¿En relación con la experiencia obtenida, considera que "CHAPIE" es una herramienta que facilita la forma en la cual se accede a la información de las diferentes herramientas de la Universidad (Aula Virtual - Banner)? *

- Sí
- No

¿Qué tan eficiente encontró la comunicación con "CHAPIE" en Lenguaje Natural? *

- ◯ Mala
- ◯ Regular
- ◯ Buena
- ◯ Muy Buena
- ◯ Excelente

¿Considera que el uso de "CHAPIE" facilita la gestión educativa que se lleva a cabo dentro de estas herramientas (Aula Virtual - Banner)? *

- ◯ Sí
- ◯ No

¿Cree que los chatbots como "CHAPIE" pueden adaptarse para un mejor manejo de las herramientas de la Universidad y a su vez brindar una atención personalizada enfocada en el aprendizaje? *

- ◯ Sí
- ◯ No

Enviar

ANEXO 5. RESULTADOS DE ENCUESTA 2

Encuesta "CHAPIE" 2020

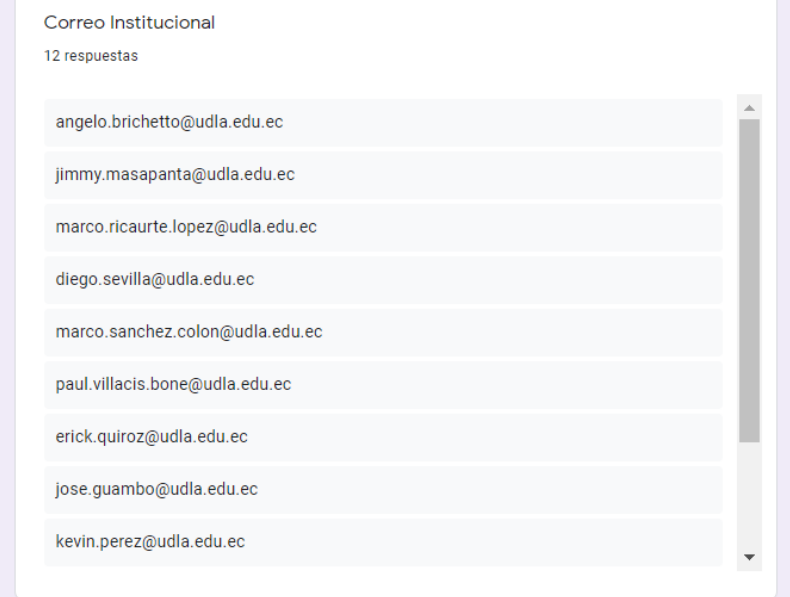
12 respuestas

Publicar datos de análisis

Nombre y Apellido

12 respuestas

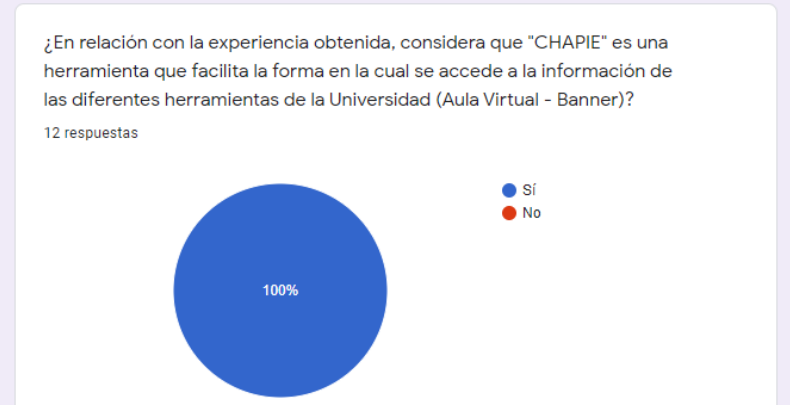
Angelo Brichetto
Jimmy Masapanta
Marco Ricaurte
Diego Sevilla
Marco Sánchez
Paul Villacis
Erick Quiroz
Jose Guambo
Kevin Pérez



Correo Institucional

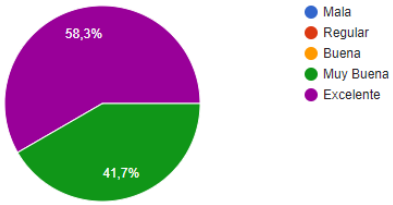
12 respuestas

angelo.brichetto@udla.edu.ec
jimmy.masapanta@udla.edu.ec
marco.ricaurte.lopez@udla.edu.ec
diego.sevilla@udla.edu.ec
marco.sanchez.colon@udla.edu.ec
paul.villacis.bone@udla.edu.ec
erick.quiroz@udla.edu.ec
jose.guambo@udla.edu.ec
kevin.perez@udla.edu.ec



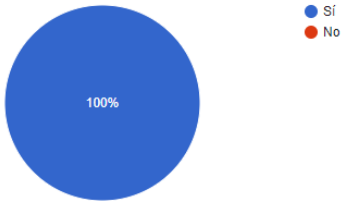
¿Qué tan eficiente encontró la comunicación con "CHAPIE" en Lenguaje Natural?

12 respuestas



¿Considera que el uso de "CHAPIE" facilita la gestión educativa que se lleva a cabo dentro de estas herramientas (Aula Virtual - Banner)?

12 respuestas



¿Cree que los chatbots como "CHAPIE" pueden adaptarse para un mejor manejo de las herramientas de la Universidad y a su vez brindar una atención personalizada enfocada en el aprendizaje?

12 respuestas