UNIVERSIDAD TÉCNICA DE AMBATO

FACULTAD DE TECNOLOGÍAS DE LA INFORMACIÓN, TELECOMUNICACIONES E INDUSTRIAL

CARRERA DE INGENIERÍA EN SISTEMAS COMPUTACIONALES E INFORMÁTICOS

TEMA:

PROTOTIPO DE UN CHATBOT PARA COMPRAS ONLINE UTILIZANDO BOT FRAMEWORK.

Trabajo de Graduación. Modalidad: Proyecto de Investigación, presentado previo la obtención del título de Ingeniero en Sistemas Computacionales e Informáticos

LÍNEA DE INVESTIGACIÓN:

Desarrollo de Software

AUTOR: Gamboa Teneta, Erick Daniel
TUTOR: Ing. Naranjo Ávalos, Hernán Fabricio Mg.

Ambato - Ecuador
Julio, 2019

APROBACIÓN DEL TUTOR

En mi calidad de Tutor del Trabajo de Investigación sobre el Tema:

"PROTOTIPO DE UN CHATBOT PARA COMPRAS ONLINE UTILIZANDO BOT FRAMEWORK", del señor Gamboa Teneta Erick Daniel, estudiante de la Carrera de Ingeniería en Sistemas Computacionales e Informáticos, de la Facultad de Tecnologías de la Información, Telecomunicaciones e Industrial, de la Universidad Técnica de Ambato, considero que el informe investigativo reúne los requisitos suficientes para que continúe con los trámites y consiguiente aprobación de conformidad con el numeral 7.2 de los Lineamientos Generales para la aplicación de Instructivos de las Modalidades de Titulación de las Facultades de la Universidad Técnica de Ambato.

Ambato, julio de 2019

EL TUTOR

Ing. Hernán Fabricio Naranjo Ávalos Mg.

AUTORÍA

El presente trabajo de investigación titulado: "PROTOTIPO DE UN CHATBOT PARA COMPRAS ONLINE UTILIZANDO BOT FRAMEWORK". Es absolutamente original, auténtico y personal, en tal virtud, el contenido, efectos legales y académicos que se desprenden del mismo son de exclusiva responsabilidad del autor.

Ambato, julio de 2019

Erick Daniel Gamboa Teneta

CC: 0503324360

DERECHOS DE AUTOR

Autorizo a la Universidad Técnica de Ambato, para que haga uso de este Trabajo de Titulación como un documento disponible para la lectura, consulta y procesos de investigación.

Cedo los derechos de mi Trabajo de Titulación, con fines de difusión pública. Además, autorizo su reproducción dentro de las regulaciones de la Universidad.

Ambato, julio de 2019

Erick Daniel Gamboa Teneta

CC: 0503324360

APROBACIÓN COMISIÓN CALIFICADORES

La Comisión Calificadora del presente trabajo conformada por los señores docentes Ing. Carlos Israel Núñez Miranda Mg. e Ing. Dennis Vinicio Chicaiza Castillo Mg., revisó y aprobó el Informe Final del trabajo de graduación titulado "PROTOTIPO DE UN CHATBOT PARA COMPRAS ONLINE UTILIZANDO BOT FRAMEWORK", presentado por el señor Gamboa Teneta Erick Daniel de acuerdo al numeral 9.1 de los Lineamientos Generales para la aplicación de Instructivos de las Modalidades de Titulación de las Facultades de la Universidad Técnica de Ambato.

Ing. Elsa Pilar Urrutia Urrutia Mg.

PRESIDENTA DEL TRIBUNAL

Ing. Carlos Israel Núñez Miranda Mg.

DOCENTE CALIFICADOR

Ing. Dennis Vinicio Chicaiza Castillo Mg.

DOCENTE CALIFICADOR

DEDICATORIA

Este trabajo está dedicado a Dios por regalarme la vida, guiarme en cada paso que doy y por permitirme llegar hasta este momento tan importante de mi formación, a mis padres Carlos y Elvia por brindarme su apoyo en momentos difíciles, su amor y fortaleza para cumplir mis metas.

A mis hermanos por brindarme siempre su cariño y apoyo incondicional, en especial a mis hermanas Irma y Tatiana por su ayuda brindada, paciencia y consejos oportunos.

A mi tutor Ing. Hernán Naranjo por su colaboración en la realización de este proyecto.

Erick Daniel Gamboa Teneta

AGRADECIMIENTO

El agradecimiento es un sentimiento de gratitud, por tal motivo mi eterno agradecimiento a Dios y a mis padres por haberme acompañado en mi trayecto de formación profesional y a lo largo de mi vida.

A mis maestros por haberme brindado sus conocimientos y encaminarme en mi vida profesional.

A mis compañeros de clase por compartir sus conocimientos, por compartir vivencias que me han dejado muchas experiencias y enseñanzas.

A Gissela Belén por la comprensión, amor y apoyo incondicional demostrado durante esta etapa de mi vida.

Al Ing. Hernán Naranjo por toda la paciencia que ha tenido en el tiempo de elaboración de la presente investigación.

Erick Daniel Gamboa Teneta

ÍNDICE

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

RESUMEN EJECUTIVO

En la actualidad, dentro del ámbito tecnológico han existido grandes avances. En el campo del desarrollo de software se ha producido una evolución de los sistemas convencionales. Esta evolución ha sido gracias al fortalecimiento de la inteligencia artificial, lo cual ha permitido dotar de conocimiento a varias tecnologías existentes.

Estos continuos avances, en el campo de la inteligencia artificial, han favorecido el aparecimiento y desarrollo de nuevas interfaces de usuario así como de nuevos contextos de aplicación, como es el caso de los chatbots.

El presente proyecto de investigación se desarrolló con el fin de presentar los contextos de aplicación de los agentes conversacionales. Además, mostrar las principales técnicas y herramientas para el desarrollo de agentes conversacionales, también conocidos como chatbots, para lo cual se ha desarrollado un prototipo.

Para el desarrollo del chatbot se utilizó Bot Framework, el cual está basado en el lenguaje de programación C#. Para dotar de inteligencia artificial se utilizó el procesamiento de lenguaje natural mediante el uso de DialogFlow. Además, para que el agente sea usado se demostrará la implementación en un entorno local.

ABSTRACT

Today, in the technological field there have been great advances. In the field of software development there has been an evolution of conventional systems. This evolution has been thanks to the strengthening of artificial intelligence, which has allowed to provide knowledge to several existing technologies.

These continuous advances, in the field of artificial intelligence, have favored the appearance and development of new user interfaces as well as new application contexts, as is the case of the chatbots.

The present research project was developed in order to present the contexts of application of the conversational agents. Also, show the main techniques and tools for the development of conversational agents, also known as chatbots, for this reason a prototype has been developed.

For the development of the chatbot, Bot Framework was used, which is based on the C # programming language. To provide artificial intelligence, natural language processing was used through the use of DialogFlow. Furthermore, for the agent to be used, the implementation in a local environment will be demonstrated.

INTRODUCCIÓN

El presente trabajo de investigación, que tiene como tema: "PROTOTIPO DE UN CHATBOT PARA COMPRAS ONLINE UTILIZANDO BOT FRAMEWORK", se encuentra dividido en capítulos, los cuales se detallan a continuación:

Capítulo I. "El Problema", en este capítulo se identifica y plantea el problema a investigar, la justificación de la investigación y los respectivos objetivos a llevar a cabo durante el desarrollo de la investigación.

Capítulo II. "Marco Teórico", en este capítulo se presenta los antecedentes investigativos referentes a la investigación, también contiene la recopilación de fundamentación teórica que sirve de soporte a la investigación y se plantea la propuesta de la investigación.

Capítulo III. "Metodología", en este capítulo se presenta los diferentes tipos de modalidades de investigación a utilizarse, especificando el método de recolección de información para el desarrollo de la investigación, por último, se presenta un listado de las diferentes actividades necesarias para cumplir con los objetivos planteados.

Capítulo IV. "Desarrollo de la propuesta", en este capítulo se detalla cada una de las actividades realizadas durante la investigación, detalladas en el capítulo anterior.

Capítulo V. "Conclusiones y Recomendaciones", en este capítulo se dan a conocer las conclusiones y recomendaciones que surgieron una vez concluido el desarrollo del proyecto de investigación.

CAPÍTULO 1

El problema

1.1. Tema de Investigación

"Prototipo de un chatbot para compras online utilizando Bot Framework."

1.2. Planteamiento del problema

El abrumador desarrollo que ha tenido en los últimos años la inteligencia artificial ha permitido un crecimiento de los sistemas de procesamiento del lenguaje natural. El crecimiento de estos sistemas ha impulsado el fortalecimiento y evolución de los llamados asistentes virtuales. Entre las evoluciones de los asistentes virtuales se pueden destacar los chatbots o agentes conversacionales.

La aparición de los chatbots en el contexto tecnológico actual se debe al mejoramiento que han tenido en los últimos años las aplicaciones de mensajería instantánea. Los continuos avances en este tipo de aplicaciones se han visto reflejados en el elevado número de usuarios que diariamente las usan. Se estima que existen alrededor de tres mil millones de usuarios que usan algún tipo de aplicación de mensajería instantánea.[1]

Los chatbots, en sus inicios, fueron diseñados para aplicaciones de mensajería. Sin embargo, en los últimos años estos han logrado trascender a otros campos de aplicación. La revolución en la comunicación de las personas y las necesidades surgidas en el ámbito tecnológico, han permitido que los chatbots se situaran desde simples agentes conversacionales hasta asesores virtuales. En la actualidad los chatbots no se encuentran atados únicamente a las aplicaciones de mensajería instantánea, sino que estos, han podido trascender a la web como servicios que brindan ayuda y soporte a los usuarios.

Varias empresas han tenido un especial interés en los chatbots, en especial en el campo de atención al cliente, debido a la alta disponibilidad y facilidad de uso que estos presentan. Campos como la psicología, la medicina, la educación, el comercio entre otros, también han sido beneficiados por el uso de esta tecnología.[2]

En el campo del comercio electrónico, no ha existido un especial interés en los chatbots, pese a esto, grandes empresas a nivel mundial han incorporado agentes conversacionales para este cometido. Las empresas que intentaron incursionar en el campo del comercio electrónico con chatbots, son empresas a escala mundial, por lo cual muy pocos usuarios han logrado experimentar un chatbot de este tipo.[3]

Teniendo en cuenta los grandes beneficios que ha demostrado tener un agente conversacional o chatbot, en varias partes del mundo han existido un especial interés en el desarrollo de este tipo de aplicaciones. En países como España, Italia, Noruega, Estados Unidos, China, Holanda, han demostrado la importancia y pertinencia del futuro de los chatbots, siendo precursores del desarrollo de esta tecnología. Las investigaciones que se han realizado en estos países exponen los principales paradigmas de desarrollo e implementación de chatbots.[4, 1, 5, 6, 7, 8, 9]

En el contexto nacional existe una tendencia de desarrollo de agentes conversacionales aplicados al campo de atención al cliente. En el caso puntual de la Escuela Politécnica Nacional, han planteado agentes conversacionales para ayudar a solventar las inquietudes de los clientes o usuarios de determinadas plataformas. Dichas investigaciones presentan la importancia de aplicar nuevas tecnologías para mejorar la experiencia de usuario.[3, 9]

Al hablar de experiencias de usuario, en el contexto tecnológico, se debe tener en cuenta la facilidad con la que los usuarios interactúan con un sistema o con una tecnología. Es así como el hecho de incorporar un chatbot a un sistema de compras online facilitará el proceso de adquisición un producto, haciéndolo más natural para los usuarios menos experimentados. Además, la razón de plantear un prototipo es validar la funcionalidad del aplicativo en entornos de negocio, y como este se puede replicar en diferentes ámbitos de aplicación.

1.3. Delimitación

1.3.1. De contenidos

Área Académica: Software

Línea de Investigación: Desarrollo de Software

Sublínea de Investigación: Intercambio de Información

1.3.2. Espacial

Universidad Técnica de Ambato (La investigación propuesta está delimitada a la universidad debido a que al tratarse de un prototipo podrá ser usado para ámbitos investigativos.)

1.3.3. Temporal

La presente investigación se desarrollará en el periodo académico Marzo 2019 – Agosto 2019.

1.4. Justificación

La importancia de esta investigación radica en analizar tecnologías relevantes para incursionar en la construcción y desarrollo de agentes conversacionales, es decir, presentar herramientas y componentes que intervienen en el proceso de desarrollo de un chatbot. La investigación pretende exponer el concepto de agente conversacional, arquitectura y paradigmas de desarrollo.

Al existir diversidad de campos de aplicación, se hace necesario contar con una guía de desarrollo de agentes conversacionales. Además, por el hecho de tratarse de un prototipo, la investigación pretende cimentar los conocimientos en el desarrollo de chatbots.

Dentro de la factibilidad, se cuenta con los conocimientos, herramientas y recursos necesarios para la realización del proyecto.

Al final la investigación pretende aportar de forma significativa a futuras aplicaciones que involucren agentes conversacionales dentro de la comunidad

universitaria. Apoyada en la investigación presentada se pueden desarrollar aplicaciones que fortalezcan el aprendizaje.

1.5. Objetivos

1.5.1. General

Desarrollar un prototipo de chatbot para compras utilizando Bot Framework.

1.5.2. Específicos

- Analizar la arquitectura y el ámbito de aplicación de un chatbot para demostrar las mejores técnicas de desarrollo de un agente conversacional.
- Describir el flujo de trabajo de un chatbot acorde al contexto de la aplicación a desarrollar.
- Sintetizar el uso de Bot Framework para el desarrollo de chatbots.
- Construir un prototipo de chatbot de compras online para exponer los beneficios de incorporar nuevas interfaces de usuario a sistemas convencionales.

CAPÍTULO 2

Marco Teórico

2.1. Antecedentes Investigativos

Dentro de los diferentes trabajos realizados a nivel país y a nivel internacional en el campo de agentes conversacionales, se pueden mencionar los siguientes trabajos:

En la investigación realizada por Asbjørn Følstad, Cecilie Bertinussen Nordheim y Cato Alexander Bjørkli titulado: "What Makes Users Trust a Chatbot for Customer Service? An Exploratory Interview Study" de 2018, detalla un estudio realizado a usuarios de chatbots de servicio al cliente. Dentro del estudio se exponen las diferentes experiencias que han tenido los usuarios, los inconvenientes que los usuarios han percibido y el nivel de confianza que un usuario percibe al momento de usar este tipo de tecnología. Además, se resumen los beneficios, retos y factores que se deben tomar en cuenta al momento de desarrollar un chatbot para así conseguir una mejor experiencia de usuario.[6]

Agnese Augello, Manuel Gentile, Lucas Weideveld y Frank Dignum en su trabajo titulado: "A Model of a Social Chatbot" de 2016, plantea un modelo de chatbot basado en el contexto social. La investigación explica principalmente un modelo de arquitectura para un chatbot de tipo social. La arquitectura presentada se basa en una base de conocimiento descrita mediante un lenguaje de marcado de inteligencia artificial, pero dicho lenguaje de marcado es usado bajo un modelo de práctica social. Además, la investigación incluye las deficiencias existentes en la comunicación de un chatbot con una persona dentro de un contexto social.[4]

AM Rahman, Abdullah Al Mamun y Alma Islam en su investigación titulada: "Programming challenges of chatbot: Current and future prospective" de 2017, presenta la situación actual del desarrollo de chatbots, las herramientas que se usan y los principales retos que existen al desarrollar un chatbot. En la investigación se

pretende aclarar el panorama de desarrollo de los chatbots, incluyendo una revisión básica de la constitución de un chatbot.[8]

Javier Medina, Eduardo M. Eisman y Juan Luis Castro en su trabajo titulado: "Asistentes virtuales en plataformas 3.0" de 2013, plantean la opción de desplegar los agentes conversacionales en varias plataformas. El artículo define una arquitectura de chatbot basado en técnicas de procesamiento del lenguaje natural. Se analiza la forma de desplegar un agente conversacional multiplataforma. Además, plantea varias consideraciones que se deben tener al momento de desplegar un agente conversacional en diferentes plataformas.[9]

Omar Zarabia en su trabajo titulado: "Implementación de un chatbot con botframework: caso de estudio, servicios a clientes del área de finanzas de seguros Equinoccial" de 2018, describe el desarrollo de un chatbot transaccional enfocado en atención al cliente. El trabajo expone los beneficios de incorporar un chatbot. Otro aporte relevante del estudio se centra en la importancia de combinar técnicas de inteligencia artificial para mejorar el procesamiento del lenguaje natural.[10]

Katherine Hurtado y Jonathan Zúñiga en su trabajo titulado: "Desarrollo de un asistente virtual web para la EPN y un asistente dirigido por voz en los kioscos digitales de la DGIP" de 2019, presentan la construcción de un agente conversacional enfocado en la atención de dudas de los estudiantes. El chatbot fue concebido con la intención de ser desplegado en diferentes plataformas. Además, se analiza los principales beneficios de incluir agentes conversacionales en canales electrónicos de información.[3]

2.2. Fundamentación teórica

2.2.1. Chatbots o Agentes Conversacionales

Un chatbot o agente conversacional es un sistema de software que puede interactuar con un usuario mediante el uso del lenguaje natural.[7] Un chatbot no se encuentra únicamente ligado a mensajes de texto, si no a una serie de contenido multimedia, lo cual permite una mejor interacción con el usuario. Para una mejor interpretación del lenguaje natural un chatbot, en la actualidad, utiliza técnicas de procesamiento del lenguaje natural.[2]

Mediante el uso de computación cognitiva, a través del uso de inteligencia artificial,

un chatbot entiende lo que el usuario está intentando decir y responde con un mensaje coherente, relevante y directo relacionado con la tarea o petición que el usuario está solicitando.[5]

Un chatbot por la forma de interacción con el usuario viene a ser un nuevo tipo de interfaz de usuario que utiliza diálogos para establecer una comunicación. Un agente conversacional deja de lado los componentes tradicionales que se usan para la comunicación con el usuario.[11]

Además, los agentes para mejorar las respuestas al usuario se pueden conectar con aplicaciones externas que responden a las peticiones que el bot haga.

Las características propias que contiene un chatbot lo convierten en una especie de sistema experto, que basado en el conocimiento que contiene simula un diálogo inteligente con el usuario.[11]

2.2.1.1. Reseña Histórica

Desde que se empezó a utilizar el término de inteligencia artificial en 1950, ha existido un especial interés en probar la inteligencia de las máquinas. Uno de los precursores de la informática moderna, Alan Turing, desarrolló un método de evaluación de la inteligencia artificial (IA) llamado el Test de Turing. El método propuesto por Turing pretendía catalogar el nivel de inteligencia que podía tener un equipo computacional. Desde ese entonces ha existido varios avances en el campo de la IA, el procesamiento del lenguaje natural es uno de los campos que ha presentado un mayor desarrollo.[10, 2]

Este desarrollo en el procesamiento del lenguaje natural se puede ver plasmado en la evolución que han tenido los agentes conversacionales.

El programa denominado “ELIZA” puede ser considerado como el primer agente conversacional, fue desarrollado por el profesor Joseph Weizenbaum en 1966. Este agente fue concebido como una ayuda para el análisis del lenguaje. Aunque era muy sencillo se aproximó a resultados muy convincentes dependiendo de las frases que se introducían. Eliza utilizaba un modelo de patrones que simulan el comportamiento humano, por este modelo es considerado el punto de partida de los chatbots actuales.[3, 10, 2]

El agente conversacional “Parry”, era un programa que intentaba simular el comportamiento de un paciente paranoico. El principal objetivo de Parry era evaluar el comportamiento de pacientes que tenían paranoia, posteriormente replicaba estos comportamientos frente a un médico para el estudio de esta enfermedad.[3, 2]

Debido a la evolución que han tenido los chatbots, Hugh Loebner conjuntamente con el Centro de Estudios del Comportamiento de Cambridge, en 1990, crean el concurso denominado Loebner. El concurso está basado en las premisas del Test de Turing, entrega $100.000 y una medalla de oro para el ordenador que logré pasar el test. Se realiza cada año con la intención de valorar los avances en agentes conversacionales. Además, este concurso ha impulsado el desarrollo de tecnologías y agentes conversacionales sofisticados.[10, 2]

En 1995, el Dr. Richard Wallace desarrolla A.L.I.C.E., este agente usaba un lenguaje basado en lógica matemática denominado SETL, el cual no tuvo mucho éxito. Debido al fracaso de SETL, surge una de las tecnologías más complejas para el desarrollo de chatbots, AIML. [2]

Dicha tecnología fue usada conjuntamente con Java para el desarrollo de la segunda versión de A.L.I.C.E. Para la tercera versión de este agente los desarrolladores deciden tomar como base AIML, pero aplicado como XML, lo cual les permitió ganar el premio Loebner del año 2000.[2]

A partir de este hito que consiguió A.L.I.C.E., se han desarrollado dos versiones más. Además, en el año 2001 se funda “The A.L.I.C.E. AI Foundation”, organización sin ánimo de lucro que fomenta el desarrollo y mantenimiento de Alicebot con fines de investigación y formación.[2]

En la última década, ha existido grandes avances en el tema de desarrollo de agentes conversacionales. Uno de los grandes hitos que se ha alcanzado en el campo de desarrollo de bots que integran IA se produjo en 2014, cuando un chatbot llamado Eugene Goostman logra pasar el test de Turing por primera vez en la historia. Este suceso puso en duda la eficacia del test de Turing, ya que el chatbot logró convencer al 33 % de los jueces que era un ser humano.[10]

En 2016, Microsoft logra vencer la barrera de las aplicaciones desarrolladas hasta ese entonces, con un sistema basado en el reconocimiento de voz. Ese mismo año,

Facebook anuncia el lanzamiento de su plataforma para el desarrollo de chatbots denominada Messenger Platform.[10]

En la actualidad empresas como Google, Amazon, Microsoft, IBM, entre otras, brindan servicios para el procesamiento del lenguaje natural y desarrollo de chatbots, lo cual ha posibilitado la difusión de este tipo de tecnologías.

2.2.1.2. Características de los chatbots

Cada agente conversacional o chatbot tiene características diferentes dependiendo del campo de aplicación, entre las características más relevantes y que permiten diferenciar a los agentes se encuentran:

- **Autonomía:** es la capacidad que tiene un agente para actuar de forma independiente, únicamente basándose en la experiencia adquirida. Esta característica está relacionada con la capacidad de adaptación que puede tener un agente.[2]
- **Sociabilidad:** capacidad de comunicarse con otros agentes o entidades.[2]
- **Racionalidad:** capacidad de generar respuestas apropiadas acordes al contexto y datos ingresados.[2]
- **Reactividad:** capacidad de emitir respuestas enriquecidas, es decir, no se debe limitar a respuestas con texto.[2]
- **Proactividad:** capacidad de tomar la iniciativa en una conversación, es decir, es la forma de como un agente dirige una conversación.[2]
- **Adaptabilidad:** capacidad de aprender y usar lo aprendido.[2]
- **Veracidad:** capacidad de presentar información fiable.
- **Personalidad:** el agente es único y posee una cantidad de características propias que el programador le brindó, puede mostrar emociones, interpretar sentimiento o tener un comportamiento no verbal.[2]

2.2.1.3. Tipos de chatbots

No existe una categorización formal de los chatbots, sin embargo, se los ha agrupado dependiendo de características comunes y utilidad presentada. A continuación, se muestra en formal general los tipos de chatbots existentes.

Chatbots según el tipo de servicio

- **Operativos o empresariales:** son bots que facilitan los servicios ofrecidos por una organización. Por lo general son usados para mejorar los tiempos en los procesos de una empresa.[10]
- **Informativos:** son bots que cumplen tareas muy sencillas, por lo general son usados para sistemas de preguntas y respuestas. También conocidos como FAQ bot o bots de preguntas y respuestas frecuentes.[12]
- **De e-commerce:** conocidos como asistentes virtuales comerciales, son bots que facilitan el proceso de compra de algún producto.[10]

Chatbots según el diseño de la interfaz

- **Con interfaz sólo de texto:** son bots que realizan el intercambio de información con el usuario mediante mensajes de texto.[12]
- **Con interfaz combinada entre texto, imágenes y botones:** son bots que para el intercambio de información con el usuario utilizan diálogos enriquecidos, es decir utilizan imágenes, audio, texto y botones.[12]

Chatbots según la tecnología usada

- **Simples:** son bots sencillos, cuyo funcionamiento se basa en coincidencias de patrones básicos, es decir muestran una repuesta muy ambigua.[12]
- **Complejos:** son bots sofisticados que utilizan técnicas de IA para el procesamiento de la información, lo cual les permite tener una conversación coherente con los usuarios. Además, permiten la integración con servicios externos para mejorar el nivel de coherencia.[12]

Chatbots según el campo de aplicación

- **De utilidades:** son bots que cumplen funciones específicas, se encuentran enfocados en un objetivo específico y su éxito es medido mediante el nivel de cumplimiento del objetivo.[13]
- **Sociales:** son bots que tienen una personalidad influyente, tratan de mantener un buen nivel de conversación. Este tipo de bots tienen un mayor nivel de flexibilidad en la interacción con los usuarios, por lo general son implementados en servicios de atención al público y en redes sociales. El éxito es medido en

relación con la satisfacción que tuvo un usuario con la conversación o el tiempo de duración de una conversación.[13]

- **Asistentes:** son bots que actúan como facilitadores al momento de realizar alguna tarea, generalmente están incluidos dentro de los sistemas operativos, como Cortana o Google Assistant. Al igual que los bots sociales tienen un nivel de personalidad influyente, pero los bots asistentes tienen la posibilidad de ejecutar alguna tarea determinada.[13]

2.2.2. Procesamiento de Lenguaje Natural (PLN)

Se entiende como procesamiento del lenguaje natural a la habilidad que tiene una máquina para procesar la información recibida, ya sea letras o sonidos. El PLN se basa en la extracción e identificación de información esencial a partir del lenguaje natural que utiliza un usuario.[10]

2.2.2.1. Arquitectura de un sistema PLN

La arquitectura de un sistema PLN se sustenta en una definición del lenguaje natural por niveles[14]:

- **Nivel Fonológico:** trata de cómo las palabras se relacionan con los sonidos que representan.
- **Nivel Morfológico:** trata de cómo las palabras se construyen a partir de unidades de significado más pequeñas conocidas como morfemas.
- **Nivel Sintáctico:** trata de cómo las palabras se unen para formar oraciones, definiendo el orden de cada palabra dentro de una oración.
- **Nivel Semántico:** trata del significado de las palabras, y cómo el significado de cada palabra se une para dar coherencia a una oración.
- **Nivel Pragmático:** trata de cómo las oraciones se usan en distintas situaciones y de cómo el significado de una oración se ve afectado por otras oraciones anteriores.

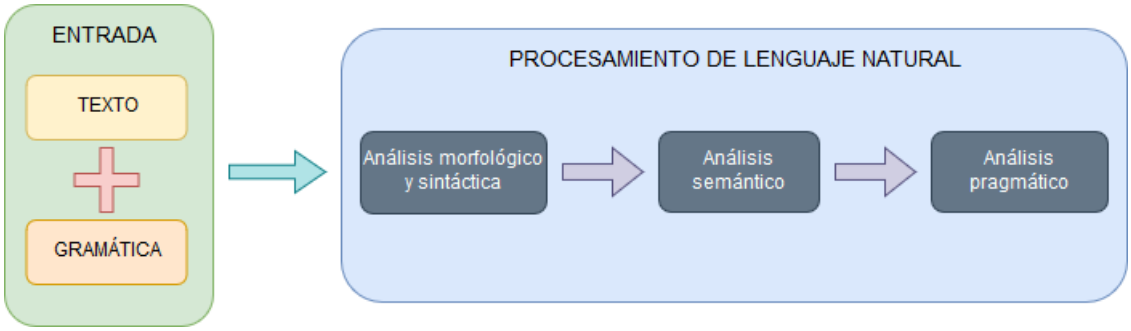

Figura 2.1: Arquitectura de un sistema de procesamiento de lenguaje natural. Fuente: [10, 14]

En la figura 2.1 se muestra la arquitectura de un sistema de procesamiento del lenguaje natural. En el sistema la entrada de información es una expresión en lenguaje natural, la cual será sometida a un proceso de análisis e interpretación.[10]

El funcionamiento del sistema es sencillo. Una vez obtenida la información, el sistema realiza un análisis de la información en sentido morfológico y sintáctico, es decir, analiza las palabras y la estructura de las oraciones. En esta etapa de análisis el sistema utiliza un analizador lexicográfico conocido como scanner y un analizador sintáctico conocido como parser. El scanner identifica los componentes léxicos definidos a priori y el parser verifica el orden gramatical entre los elementos identificados por el scanner.[14]

A continuación, el sistema realiza un análisis semántico de las oraciones para saber el significado de cada oración a la vez que le asigna una expresión lógica (true o false) a cada significado.[14]

Para finalizar, el sistema realiza un análisis pragmático de la información, es decir, toma todas las oraciones analizadas previamente y las analiza conjuntamente tomando en cuenta la situación de cada oración. De este análisis obtiene una expresión resultante, la cual será enviada al usuario.[14]

2.2.2.2. Aplicaciones del PLN

Las aplicaciones que puede tener un sistema de PLN son muy variadas. Dependen del campo de aplicación y de la combinación con otras tecnologías, algunas de las aplicaciones son[10, 14]:

- Traducción automática.
- Extracción de información.
- Tutores inteligentes.
- Agentes conversacionales (chatbots).
- Respuestas automáticas.
- Reconocimiento de voz.
- Análisis de sentimientos.

2.2.3. Prototipo

Un prototipo es una representación limitada de un producto, permite a los involucrados probarlo en situaciones reales, explorar su uso y mejorarlo. Un prototipo puede ser desde un simple trozo de papel hasta un software complejo.[15]

En el ámbito del desarrollo de software un prototipo es definido como un sistema funcional a pequeña escala que permite descubrir las necesidades de los usuarios. Entre las principales ventajas están: el rápido desarrollo y el bajo costo económico. El desarrollo de un prototipo se realiza de forma iterativa, una vez realizada la primera versión, los usuarios y analistas mejoran el prototipo hasta llegar al sistema deseado.[16]

Por lo general los prototipos sólo tienen implementado la parte funcional (que afecta al usuario), mientras que la parte de seguridad y errores informáticos se resuelven posteriormente.[16]

Los prototipos son útiles en la evaluación de productos ya que permiten clarificar los requisitos de los usuarios. Además, permiten discutir y definir ideas entre diseñadores y partes responsables. Sirven de mucha utilidad en las fases iniciales de desarrollo y durante la fase de diseño.[15]

2.2.3.1. Tipos de prototipos

Existen varios tipos de prototipos dependiendo el uso que se les dé, entre los principales se pueden encontrar[15]:

- **Prototipo rápido:** utiliza una metodología de diseño en la cual se desarrollan nuevos diseños, se evalúan y se actualiza cuando el nuevo diseño está listo.
- **Prototipo reutilizable:** también conocido como prototipo evolutivo o evolutionary prototyping, las partes usadas en la construcción del prototipo son utilizadas para la construcción de producto final. Este tipo de prototipo por lo general es usado en el desarrollo de software.
- **Prototipo modular:** también conocido como prototipo incremental o incremental prototyping, este prototipo permite añadir nuevos elementos a medida que el diseño avanza.
- **Prototipo de baja fidelidad:** el prototipo se implementa usando papel y lápiz, simula la funcionalidad del producto final sin mostrar el aspecto real del producto, es útil para realizar test rápidos.
- **Prototipo de alta fidelidad:** el prototipo se diseña como una aproximación real del producto final, es decir, tiene en cuenta el diseño, interacción con el usuario y tiempo.

2.2.4. Metodologías de desarrollo de software

En el ámbito del desarrollo de software, una metodología es una serie de actividades relacionadas que conducen a la elaboración de un producto de software. El desarrollo de software es un proceso complejo por lo cual es necesario el uso de metodologías. Las metodologías brindan un marco de referencia para hacer que el proceso de desarrollo de software sea eficaz y eficiente.[17, 18]

Pese a que existen diferentes metodologías para el desarrollo de software, todas deben incluir cuatro actividades que son fundamentales para la construcción de software[18]:

- **Especificación del software:** tienen que definirse tanto la funcionalidad del software como las restricciones de su operación.
- **Diseño e implementación del software:** debe desarrollarse el software para cumplir con las especificaciones.
- **Validación del software:** se debe validar el funcionamiento del software para asegurarse que satisfaga las necesidades del cliente.
- **Evolución del software:** el software evoluciona para satisfacer las necesidades cambiantes del cliente.

2.2.4.1. Componentes de una metodología

En la figura 2.2 se muestran los elementos básicos que componen una metodología.

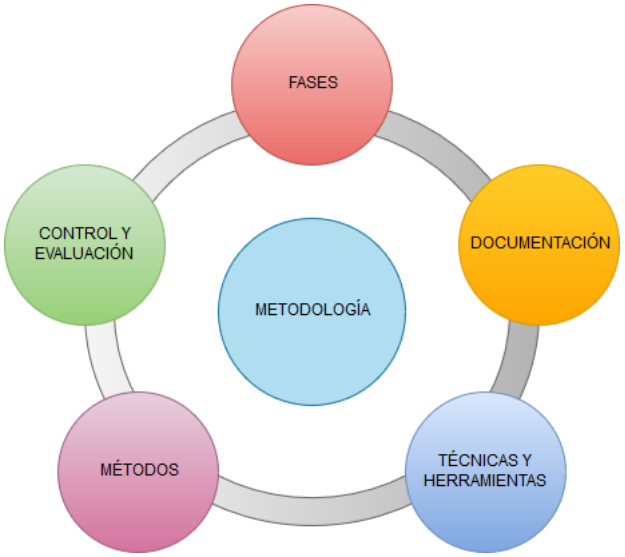

Figura 2.2: Elementos de una metodología.
Fuente: [19]

1. **Las fases:** definen las actividades que se van a realizar en cada fase.

2. **Los métodos:** definen el proceso y el modo que se debe seguir durante el desarrollo para alcanzar el producto final. Este componente plantea la descomposición de los procesos en tareas más pequeñas y la asignación de cada tarea a una fase o fases. Además, define el entregable que se generará en cada fase y la técnica que se debe usar para realizar la tarea.[19]

3. **Técnicas y herramientas:** definen la forma de resolver cada tarea y las posibles herramientas a usar. Existen diferentes tipos de técnicas, algunas de ellas son[19]:

   a) De recopilación de datos: uso de entrevistas, formularios, entre otras.

   b) Técnicas gráficas: diagramas, organigramas, diagramas de matrices, entre otras.

   c) Técnicas de modelado: desarrollos estructurados y orientado a objetos.

4. **Documentación:** define la documentación que se debe entregar en cada fase. La documentación se debe realizar de forma completa tomando en cuenta los

valores que se van generando, lo cual sirve para recoger resultados y tomar de decisiones.[19]

5. **Control y evaluación:** consiste en comprobar y aceptar/negar los resultados que se van obteniendo. Este componente permite, en caso de ser necesario, replantear la planificación de las tareas. El control y evaluación se debe realizar durante todo el ciclo de vida del proyecto, se suelen usar técnicas como PERT o diagramas de Gannt.[19]

2.2.4.2. Clasificación de las metodologías

Las metodologías se pueden clasificar de varias maneras dependiendo el enfoque. Una de las clasificaciones más aceptadas es por la filosofía de desarrollo.

Según la filosofía de desarrollo las metodologías se clasifican en dos grupos. Las metodologías tradicionales, que se basan en una fuerte planificación durante todo el desarrollo, y las metodologías ágiles, en las que el desarrollo de software es incremental, cooperativo, sencillo y adaptado.[20]

Metodologías tradicionales

Las metodologías tradicionales, también conocidas como metodologías pesadas, son el primer tipo de metodología que surge como guía para desarrollar software basado en la calidad. Se centran en la documentación exhaustiva y en cumplir con un plan de proyecto. Además, define un conjunto de fases secuenciales en las que se indican las tareas a realizar, tiempo que van a llevar y cuál va a ser su coste.[20, 19]

Para alcanzar un producto de software de calidad, esta metodología se basa en un riguroso proceso de planificación en todas las etapas del ciclo de vida del producto. Debido a que únicamente cuando se haya planificado todo el proyecto se inicia el desarrollo, la incorporación de algún cambio o requerimiento nuevo durante el desarrollo presenta un alto coste.[17, 20, 21]

Entre las metodologías tradicionales se pueden citar[17]:

- RUP (Relational Unified Process)
- MSF (Microsoft Solution Framework)
- Win-Win Spiral Model
- Iconix

Metodologías ágiles

Las metodologías ágiles nacen como una respuesta a los problemas que pueden ocasionar las metodologías tradicionales y se basan en la adaptabilidad de los procesos de desarrollo de software. Este tipo de metodología se basa en los 12 principios de desarrollo ágil que se encuentran plasmados en un documento conocido como "Manifiesto Ágil".[20, 22]

Este tipo de metodología permite un desarrollo de software incremental, cooperativo, sencillo y adaptable a la realidad de cada equipo de trabajo. Permiten que el proyecto se subdivida en proyectos más pequeños desarrollados en cortos periodos de tiempo con entregas constantes al cliente, esto permite un mejoramiento continuo y asegura la calidad del software.[17, 21]

Entre las metodologías ágiles más destacadas se encuentran[22]:

- XP (Extreme Programming)
- Scrum
- Crystal Clear
- DSDM (Dynamic Systems Development Method)
- FDD (Feature Driven Development)

Comparación entre metodologías ágiles y tradicionales

En la tabla 2.1 se definen las principales deferencias de las metodologías ágiles con respecto a las metodologías tradicionales. Estas diferencias afectan tanto al proceso como a la organización de los equipos de desarrollo.[23]

Metodologías Ágiles	Metodologías Tradicionales
Basadas en heurísticas provenientes de prácticas de producción de código.	Basados en normas provenientes de estándares seguidos por el entorno de desarrollo.
Especialmente preparados para cambios durante el proyecto.	Cierta resistencia a los cambios.
Impuestas internamente (por el equipo).	Impuestas externamente.
Proceso menos controlado, con pocos principios.	Proceso mucho más controlado, con numerosas políticas/normas.
No existe contrato tradicional o al menos es bastante flexible.	Existe un contrato prefijado.
El cliente es parte del equipo de desarrollo.	El cliente interactúa con el equipo de desarrollo mediante reuniones.
Grupos pequeños (< 10 integrantes) y trabajando en el mismo sitio.	Grupos grandes y posiblemente distribuidos.
Pocos artefactos.	Más artefactos.
Pocos roles.	Más roles.
Menos énfasis en la arquitectura del software.	La arquitectura del software es esencial y se expresa mediante modelos.

Tabla 2.1: Diferencias entre metodologías ágiles y tradicionales.

Fuente: [23]

2.2.5. Arquitectura de software

La arquitectura de software presenta una vista general del sistema, en el cual se incluye los componentes, la relación entre ellos, el ambiente y los principios que orientan su diseño y evolución. Es decir, abarca todo lo relativo a la estructura de alto nivel de los sistemas, es decir, su organización en subsistemas y la relación entre ellos.[24]

La arquitectura de software está relacionada con el diseño de un sistema, es por eso que se manifiesta en las fases iniciales del proceso de desarrollo de software. La arquitectura a diferencia del diseño se ocupa de los componentes del sistema y no de los procedimientos; de las interacciones entre los componentes y no de las interfaces; de las restricciones a ejercer sobre los componentes y no de los algoritmos.[24]

La arquitectura de software debe representar los distintos aspectos del software. Para describir cada uno de estos aspectos se utilizan modelos o vistas. Las vistas

de una arquitectura pueden formularse por medio de uno o varios lenguajes, por ejemplo, usando: el lenguaje natural, diagramas de estado, diagramas de flujo, entre otros.[24]

Para el diseño de la arquitectura se usan modelos estructurales, los cuales representan a la arquitectura como una colección organizada de componentes de programa. Estos modelos aumentan en nivel de abstracción de diseño lo cual permite identificar estructuras de diseño arquitectónico repetibles (patrones) que se encuentran en aplicaciones similares.[24]

2.2.5.1. Patrones

Los patrones son bloques de construcción mental útiles para proceder con aspectos de diseño limitados y específicos en el momento de desarrollar un software. Se basan en la reutilización y ayudan a promover buenas prácticas de diseño.[24]

Los patrones son considerados una disciplina de resolución de problemas en la ingeniería de software, que han tenido mayor impacto en la programación orientada a objetos. Los patrones pueden ser usados en cualquier ámbito de la informática y de las ciencias en general.[24]

Como se muestra en la figura 2.3, un patrón se representa como un esquema de tres partes: contexto-problema-solución.

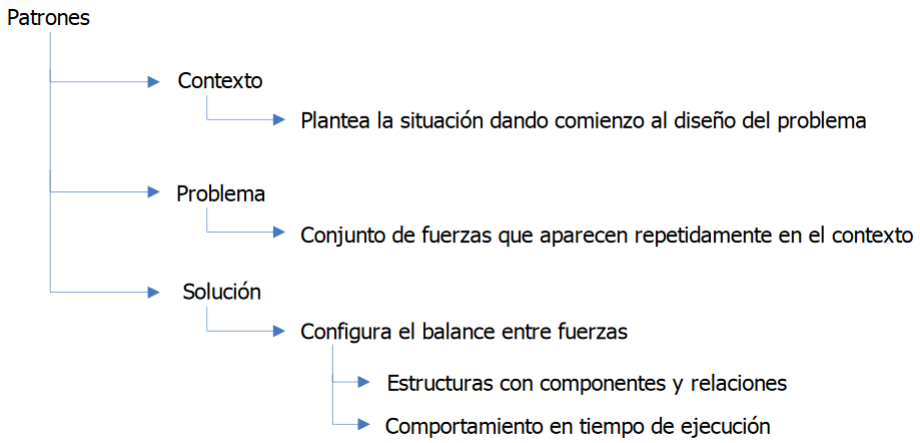

Figura 2.3: Esquema de un patrón.
Fuente: [24]

El esquema denota una regla que establece una relación entre un contexto dado, un cierto problema que tiene lugar en ese contexto y una solución apropiada al problema.[24]

- **Contexto:** describe la situación actual en la cual ocurre un problema. Es difícil especificar el contexto de un patrón lo cual implica que es casi imposible determinar todas las situaciones en las cuales se deben aplicar. Un acercamiento pragmático establece una guía para aplicar esta práctica de diseño.[24]
- **Problema:** establece una situación problemática que se repite en un contexto dado. Realiza una especificación en general para determinar el problema y dar una posible solución.[24]
- **Solución:** muestra cómo resolver un problema recurrente o como balancear las fuerzas que actúan sobre el problema.[24]

Categorías de patrones

Los patrones se agrupan en función de un rango similar de abstracción en tres categorías: patrones arquitectónicos, patrones de diseño e idioms.[24]

Patrones arquitectónicos

Son plantillas que describen los principios estructurales globales que construyen las distintas arquitecturas de software viables. Plantean una organización fundamental para un sistema de software, expresando un conjunto de subsistemas predefinidos, especificando responsabilidades y organizando las relaciones entre ellos.[24]

Este tipo de patrones representan el nivel más alto en el diseño de software, por lo cual ayudan a desarrollar una funcionalidad específica de un software.[24]

Los patrones arquitectónicos se clasifican en las siguientes categorías[24]:

- **Del fango a la estructura:** tratan de evitar el exceso de componentes u objetos. Ayudan a descomponer una tarea del sistema global en subtareas. En esta categoría se encuentran los patrones: Layers, Pipes and Filters y Blackboard.
- **Sistemas distribuidos:** definen los patrones que se deben usar en el desarrollo de sistemas distribuidos. En esta categoría se encuentran los patrones: Broker,

Microkernel y Pipes and Filters, estos dos últimos patrones también se encuentran en otras categorías. El patrón broker define una infraestructura completa para aplicaciones distribuidas, los otros dos patrones consideran a la distribución como un concepto secundario.

- **Sistemas interactivos:** definen los patrones que ayudan a la estructuración de sistemas de software que ofrecen una interacción entre el usuario y la computadora. En esta categoría se encuentran los patrones: Model-View-Controller (MVC) y Presentation-Abstraction-Control (PAC).
- **Sistemas adaptables:** definen patrones que permiten la escalabilidad de los sistemas, es decir, son adaptables a cambios en los requisitos funcionales y a las nuevas tecnologías. En esta categoría se encuentran los patrones: Reflection y Microkernel.

Patrones de diseño

Provee un esquema para refinar componentes de un sistema de software y la forma como se relacionan entre sí. Define la estructura de comunicación de los componentes en un contexto particular. Tienen un nivel de abstracción menor en comparación con los patrones arquitectónicos, lo que los hace independientes de lenguaje o paradigma de programación.[24]

Los patrones de diseño son soluciones bien documentadas por lo cual son usados para dar solución a nuevos problemas similares.Principalmente son usados para compartir y reutilizar conocimiento en el desarrollo de software.[24]

Idioms

Es un patrón de bajo nivel específico para un lenguaje de programación, se relacionan con implementación de diseño de problemas particulares. Definen como implementar aspectos particulares de un componente o las relaciones entre ellos dependiendo del lenguaje de programación usado.[24]

2.2.5.2. Patrón MVC

El patrón MVC es usado por Microsoft para la creación de aplicaciones web y APIs. Bot Framework fue concebido como API de mensajería para el desarrollo de bots, por lo cual es necesario conocer el funcionamiento de este patrón.

Este patrón logra separar el software en función del tipo de trabajo que realiza, es decir, separa la lógica de negocios, de la lógica de la interfaz y de la infraestructura.[25]

El patrón de arquitectura Model-View-Controller separa una aplicación en tres tipos de componentes: modelos, vistas y controladores.[25]

- **Modelo:** contiene la lógica de negocios y la lógica que permite conservar el estado de una aplicación.
- **Vista:** se encarga de presentar el contenido a través de la interfaz de usuario.
- **Controlador:** controla la interacción con el usuario, trabaja con el modelo y en ocasiones selecciona la vista para presentarla al usuario.

En la figura 2.4 se muestra el funcionamiento del patrón MVC.

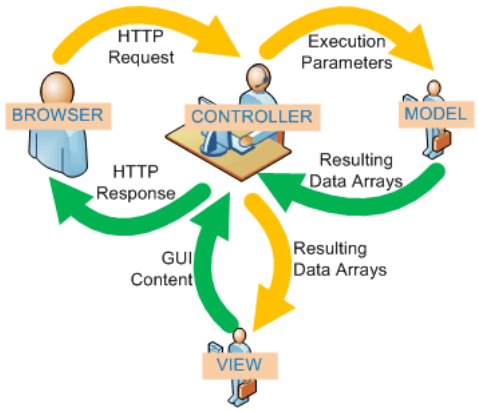

Figura 2.4: Funcionamiento patrón MVC.
Fuente: [25]

En MVC, las solicitudes de usuario se enrutan a un controlador que se encarga de trabajar con un modelo para realizar las acciones que el usuario solicita o para recuperar los resultados de consultas. El controlador elige la vista para mostrar al usuario y proporciona cualquier dato del modelo que sea necesario.[25]

2.2.6. Framework

Un framework es un diseño reutilizable de un sistema completo o de alguna de sus partes y se expresa mediante un conjunto de clases abstractas y la forma de interactuar de sus instancias.[24]

Un framework involucra muchos varios patrones de diseño, lo cual indica que los patrones son más pequeños que los framework y con un menor nivel de abstracción. Los patrones son elementos microarquitectónicos de los frameworks.[24]

2.3. Propuesta de Solución

La presente investigación propone el desarrollo de un prototipo de un chatbot orientado a compras online, usando Bot Framework. Con la investigación y desarrollo del chatbot se pretende dar pautas para el desarrollo de agentes conversacionales. Además, se expondrá los beneficios de utilizar nuevas tecnologías en sistemas convencionales.

CAPÍTULO 3

Metodología

3.1. Modalidad de la investigación

La investigación será de tipo bibliográfica porque utilizará fuentes como libros, documentos y artículos de carácter científico, revistas, etc. Para la construcción del marco teórico y posteriormente para la contextualización del prototipo propuesto.

La investigación tendrá una modalidad aplicada porque para la realización del prototipo de chatbot para compras se aplicará los conocimientos adquiridos durante la carrera.

3.2. Recolección de información

La información que se necesitará para la realización de este proyecto se obtendrá mediante las bibliotecas y repositorios virtuales de la Universidad Técnica de Ambato, además, se necesitará datos sobre la arquitectura y estado actual de los chatbots, dicha información será recabada de repositorios virtuales de organizaciones internacionales que avalen la información existente en dichos repositorios.

3.3. Procesamiento y análisis de datos

Procesamiento de la Información

1. Revisión crítica de la información recogida, es decir; filtrado de la información defectuosa o no procedente para la investigación.
2. Revisión analítica de la información recolectada, es decir, determinación de modelos idóneos que soporten y ayuden al modelo planteado.
3. Tabulación de características de cada modelo.
4. Estudio de clasificadores que mejoren el rendimiento del modelo planteado.

5. Estudio estadístico de datos para la presentación de resultados.

Análisis de Resultados

1. Análisis de resultados estadísticos, destacando el nivel de aprendizaje supervisado obtenido.
2. Interpretación de resultados, con apoyo del marco teórico, en el aspecto pertinente.
3. Establecimiento de conclusiones y recomendaciones.

3.4. Desarrollo del Proyecto

- Descripción de la arquitectura y tecnologías usadas para el desarrollo de chatbots.
- Descripción del ámbito de aplicación de los chatbots.
- Descripción del flujo de trabajo de un chatbot.
- Descripción de Bot Framework para el desarrollo de chatbots.
- Planteamiento del prototipo propuesto.
- Descripción del prototipo planteado.
- Desarrollo del prototipo propuesto.
- Evaluación del funcionamiento de desempeño.

CAPÍTULO 4

Desarrollo de la propuesta

4.1. Descripción de la arquitectura y tecnologías usadas para el desarrollo de chatbots

4.1.1. Arquitectura de un chatbot

En forma genérica un chatbot está constituido por tres componentes: interfaz de usuario, motor de inferencia y base de conocimiento, como se muestra en la figura 4.1.

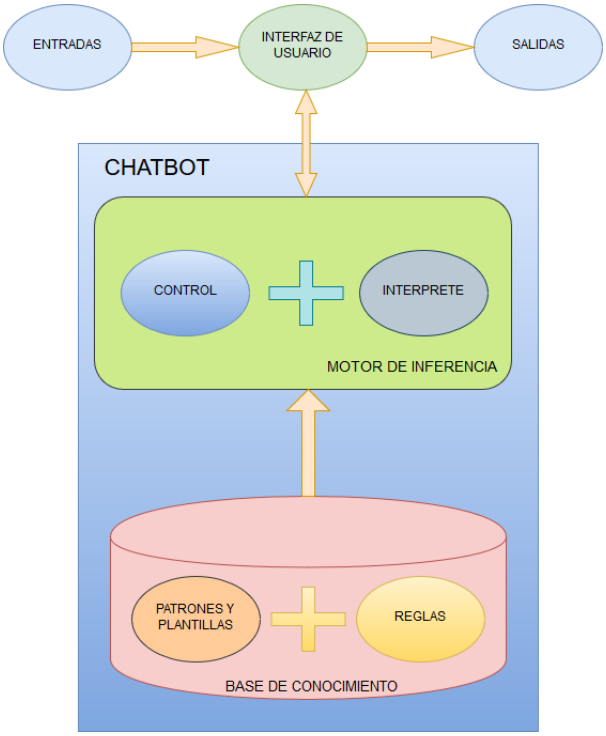

Figura 4.1: Arquitectura genérica de un chatbot.
Fuente: [10, 2, 17]

1. **Interfaz de Usuario:** es el medio de comunicación entre el usuario y el chatbot. Permite que un usuario ingrese consultas o introduzca entradas de información y presenta el resultado de las consultas o muestra salidas de información. A su vez la interfaz recibe y envía información al motor de inferencia.

2. **Motor de Inferencia:** analiza y procesa la información recibida para obtener una respuesta de acuerdo a la base de conocimientos, dicha respuesta es enviada de vuelta a la interfaz de usuario.[2] Es decir, realiza el procesamiento del lenguaje natural y como resultado devuelve una respuesta acorde a la información contenida en la base de conocimiento. Un motor de inferencia utiliza dos tipos de elementos: los datos (hechos y evidencias) y el conocimiento (conjunto de reglas almacenado en la base de conocimiento) para obtener nuevas conclusiones o hechos.[26]

3. **Base de Conocimiento:** contiene todo el conocimiento del experto humano, dicho conocimiento es introducido en la base de conocimiento en base a plantillas, patrones y reglas.[2]

Para una mejor comprensión del funcionamiento básico de un chatbot en la figura 4.2 se detalla en forma genérica el proceso que realiza un agente conversacional.

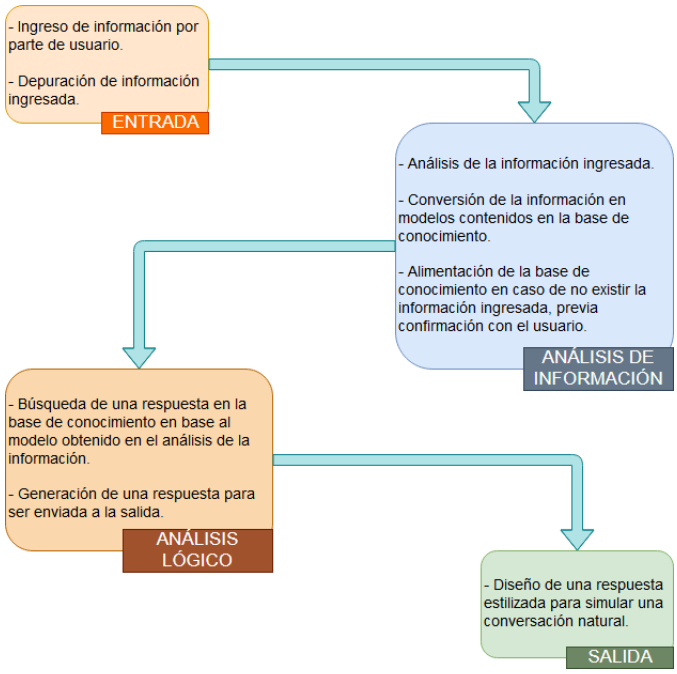

Figura 4.2: Funcionamiento genérico de un chatbot.
Fuente: [10, 2]

4.1.2. Tecnologías usadas para el desarrollo de chatbots

En la actualidad, las grandes empresas de tecnología han reconocido el futuro prometedor de los agentes conversacionales como una nueva forma de interacción con los usuarios. Es por ello que ofrecen soluciones con infraestructura que facilita el desarrollo de aplicaciones basadas en IA. [27]

Dentro del contexto de desarrollo de agentes conversacionales existen varias herramientas. La mayoría de estas herramientas son de pago, cobran un valor dependiendo el uso que se dé. Entre las herramientas más conocidas se encuentran:

- **Microsoft Bot Framework:** es una plataforma que proporciona todos los elementos necesarios para construir y conectar bots inteligentes que interactúen de forma natural con los usuarios. Microsoft Bot Framework funciona como una interface que permite conectar la lógica de negocio con el usuario donde sea que él se encuentre. A través del Bot Framework se puede utilizar recursos de Inteligencia Artificial, recursos de Machine Learning, realizar una comunicación con datos REST, reconocimiento del lenguaje natural, procesamiento de imágenes, entre otros.[13]

- **IBM Watson Conversation:** es una API, creada por IBM, para el desarrollo de bots o aplicaciones conversacionales. Para la creación de aplicaciones, utiliza un entorno gráfico que permite el desarrollo y entrenamiento de diálogos. [17, 28] Permite el despliegue de los agentes conversacionales en varios canales (móvil, web, mensajería, robots, entre otros). Además, Watson Conversation cuenta con un kit de desarrollo basado en los servicios cognitivos de Watson e IBM Cloud.[10, 28]

- **Wit.ai:** es un servicio web, adquirido por Facebook, que permite crear bots para diferentes plataformas (slack, messenger, telegram, entre otras). Wit.ai utiliza historias y flujos de diálogo para la creación del agente conversacional. Uno de los puntos a favor de Wit.ai es su interpretación del lenguaje natural.[8, 10, 29]

- **Amazon Lex:** es un servicio de Amazon Web Services que permite la creación de interfaces de conversación. Presenta un motor de conversación muy sofisticado, lo cual se ve reflejado en el procesamiento del lenguaje natural y el reconocimiento de voz. Para la creación de bots, utiliza flujos

de conversación básicos. Permite el despliegue del agente conversacional en dispositivos móviles, aplicaciones web y plataformas de chat.[30]

- **Bot Libre:** es una plataforma de código abierto, basada en un motor de inteligencia artificial desarrollado en Java. Permite la creación de bots para la web, dispositivos móviles, redes sociales y el internet de las cosas. Además, permite conectar el bot con Twitter, Facebook, Telegram, Skype, Kik, Slack, WeChat, entre otros. Bot Libre cuenta con su propio kit de desarrollo que posibilita el acceso a su API web desde JavaScript, Android, Java, IOS y Objetive C, permite la descarga del código fuente y librerías JAR desde GitHub y Sourceforge.[3, 31, 32]

- **DialogFlow:** es un servicio de Google, que permite crear interfaces conversacionales y chatbots de voz y texto basados en inteligencia artificial. Para la creación de bots conversacionales, cuenta con un entorno gráfico fácil de utilizar. Permite el despliegue de los agentes conversacionales en páginas web, aplicaciones móviles, Google Assistant, Amazon Alexa, Facebook Messenger, entre otras plataformas y dispositivos móviles.[33] Permite la creación de flujos de diálogo basado en palabras clave, además, permite la incorporación de llamadas a API's externas para mejorar el procesamiento de información.[3, 17]

4.1.2.1. Comparativa de tecnologías de desarrollo de chatbots

En la tabla 4.1 se muestra un comparativa con los aspectos más relevantes, de las tecnologías de desarrollo de chatbots.

	Microsoft Bot Framework	IBM Watson Conversation	Wit.ai	Amazon Lex	Bot Libre	DialogFlow
Plataformas de desarrollo	Escritorio / Cloud	Cloud	Cloud	Cloud	Cloud	Cloud
Tipo de herramienta	Plataforma	API	Servicio Web	Servicio Web	Plataforma	Servicio Web
Lenguaje de programación	C#	-	-	-	Java	JavaScript
Procesamiento del lenguaje natural	No	Si	Si	Si	Si	Si
Presenta SDK	Si	Si	Si	Si	Si	Si
Compatibilidad con apps externas	Si	No	No	Si, sólo servicios de AWS	No	Si
Despliegue chatbots	Local / Multicanal mediante Azure	Multicanal	Redes sociales	Multiplataforma	Multicanal	Multicanal / Multiplataforma
Licenciamiento	Gratis / Azure es pagado	Gratis / Pagado	Gratis / Pagado	Gratis / Pagado	Gratis / Pagado	Gratis / Pagado
Soporta varios idiomas	Si	Si	Si	Si	Si	Si

Tabla 4.1: Comparativa de tecnologías de desarrollo de chatbots.

Fuente: Elaborado por el investigador y basado en [8, 3, 10, 17, 13, 28, 29, 30, 31, 32, 33]

Una vez analizado las principales tecnologías y herramientas usadas para el desarrollo de chatbots y de acuerdo a la tabla anterior se decide utilizar Bot Framework para el desarrollo del prototipo. Este framework de desarrollo permite la creación y despliegue de agentes conversacionales tanto en la web como local. Pese a no tener un procesador de lenguaje natural, permite la integración de aplicaciones externas para este cometido.

Para el despliegue de los agentes permite la integración con Azure Bot Service o con cualquier otro servicio de publicación de bots. Además, el desarrollo e implementación local no tiene costo. Sólo si se utiliza los servicios de Azure para publicación o desarrollo se debe pagar por el uso.

4.2. Descripción del ámbito de aplicación de los chatbots

La facilidad de integración en diversas plataformas, aplicaciones y dispositivos ha permitido que los chatbot o agentes conversacionales puedan ser incluidos en varios campos de aplicación. Además, la alta disponibilidad y usabilidad que presentan ha sido un factor determinante para la expansión y difusión de chatbots.[3]

Dentro de las principales áreas en las que un agente conversacional puede ser usado están: Industria, Comercio, Educación, Medicina, entre otras. [3]

A pesar de que los agentes se encuentren presentes en varios campos y ámbitos de aplicación, dentro del campo del comercio ha existido una mayor proliferación de esta tecnología.

Los principales campos de aplicación son:

- **Gestión de información:** dentro de este campo los agentes son usados para procesar información o noticias dentro de la web. Además, ayudan con el almacenamiento, aprendizaje y manejo de preferencias de usuario. En este campo de aplicación se encuentran englobados los agentes o asistentes personales.[3]
- **E-Commerce:** dentro de este campo los agentes son usados para la compra, venta, búsqueda de productos, verificación de precios, entre otros. Facilitan el comercio electrónico dentro de distintas plataformas.

- **Monitorización:** dentro de este campo los agentes son usados para mantener informados a los usuarios cuando existe alguna variación en la situación de interés del usuario. Principalmente son usados para informar de variaciones en la bolsa de valores, clima, estado de los sistemas, entre otros.[3]
- **Mediador entre distintas fuentes de información:** dentro de este campo los agentes son usados para diferenciar entre el contexto de datos y el entorno en el cual se desarrollan los datos.[3]

Dentro del área médica, los agentes han sido usados para la automatización de ciertas tareas, ya sea dentro o fuera de una casa asistencial de salud como apoyo al personal o como asistentes para diagnóstico y monitoreo de pacientes.[3]

En el campo del entretenimiento digital, específicamente en el campo de los videojuegos han permitido la creación de diálogos y conversaciones más naturales, mejorando la experiencia de usuario.[3]

En la figura 4.3 se muestra un resumen del uso de agentes conversacionales en diferentes campos de la industria. En la figura 4.4 se muestra algunos ejemplos de los campos de aplicación en los cuales se puede usar chatbots.

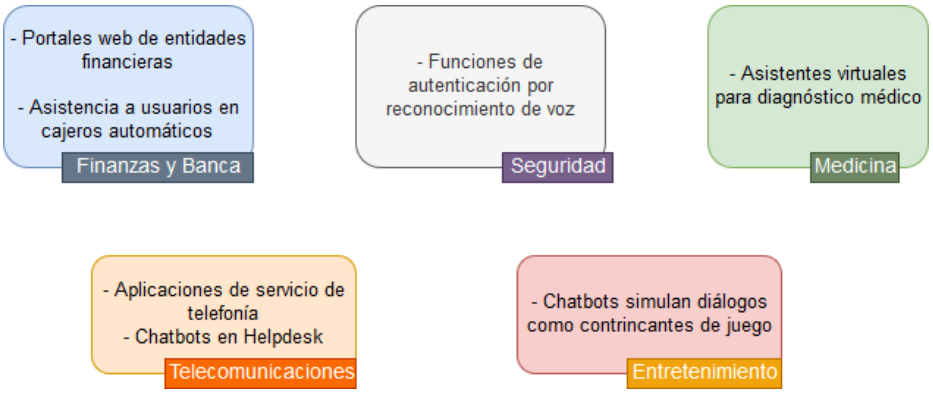

Figura 4.3: Uso de agentes conversacionales en la industria.
Fuente: [3]

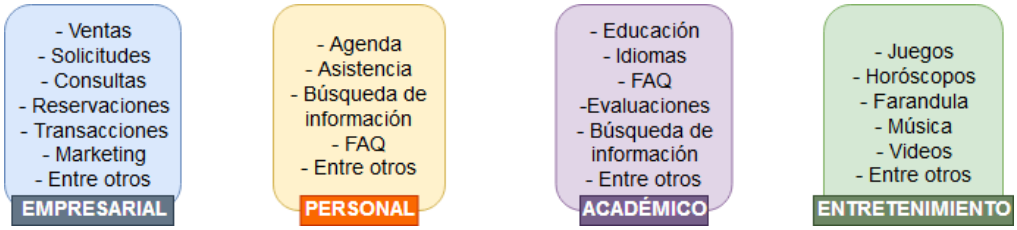

Figura 4.4: Campos de aplicación de los chatbots.
Fuente: [10]

4.3. Descripción del flujo de trabajo de un chatbot

Un chatbot no debe ser comparado con una aplicación tradicional ya que el flujo de trabajo es diferente. En una aplicación tradicional, la interfaz de usuario consta de una serie de pantallas para el intercambio de información con el usuario. En la mayoría de las aplicaciones web o de escritorio existe una pantalla principal que proporciona características de navegación para que los usuarios puedan realizar diversas actividades. [12]

Por otro lado, los chatbots al igual que las aplicaciones tradicionales presentan interfaces de usuario, pero estas interfaces se encuentran compuestas por diálogos. Los diálogos son de gran utilidad para administrar un flujo de conversación y ayudan a comprender las actividades que los usuarios desean realizar.[12]

Los diálogos permiten a los desarrolladores de agentes conversacionales la separación lógica de las funcionalidades de un bot y guiar el flujo de conversación. Los diálogos pueden tener interfaces gráficas como botones, textos y otros elementos basados en una conversación. Además, contienen acciones que permiten invocar otros diálogos o procesar información suministrada por el usuario.[12]

En la figura 4.5 se muestra una comparación entre el flujo de trabajo de una aplicación tradicional y el flujo de trabajo de un bot.

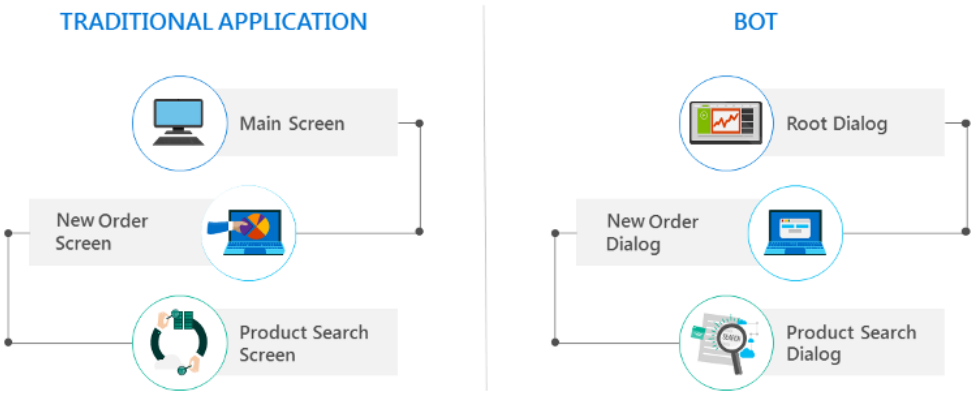

Figura 4.5: Comparación de flujos de trabajo de una aplicación tradicional y un bot. Fuente: [12]

En una aplicación tradicional, todo comienza con una pantalla principal, esta abre una nueva pantalla para realizar un nuevo pedido. La pantalla de nuevo pedido permanece hasta que se cierra o invoca otras pantallas. Si la pantalla de nuevo pedido se cierra, se devuelve al usuario a la pantalla principal.[12]

Por otro lado, en un bot, todo comienza con un diálogo raíz, este invoca a un diálogo de nuevo pedido. En ese momento, el diálogo de nuevo pedido toma el control de la conversación y permanece como principal hasta que se cierra o invoca otros diálogos. Si el diálogo de nuevo pedido se cierra, se devuelve el control de la conversación al diálogo raíz.[12]

4.3.1. Flujo de conversación

La interacción con un bot, por lo general, se centra en una tarea específica que el bot intenta lograr. El proceso ordenado de recopilación de información que el bot realiza para completar la tarea se conoce como flujo de procedimientos. El flujo de trabajo de un agente conversacional está definido por un flujo de conversación basado en procedimiento también conocido como flujo de conversación de procedimientos.[12]

Un flujo de conversación de procedimientos es un conjunto de diálogos con un propósito en común, en el cual cada diálogo busca completar una tarea específica para cumplir con el objetivo en común. En el flujo de conversación de procedimientos el desarrollador define un orden de los diálogos y el bot se encargará de mantener una conversación en el orden que los diálogos fueron definidos.[12]

En la figura 4.6, se describe, mediante un flujograma, un ejemplo de un flujo de conversación de procedimientos, en el cual cada hilo de conversación se transforma en un diálogo.

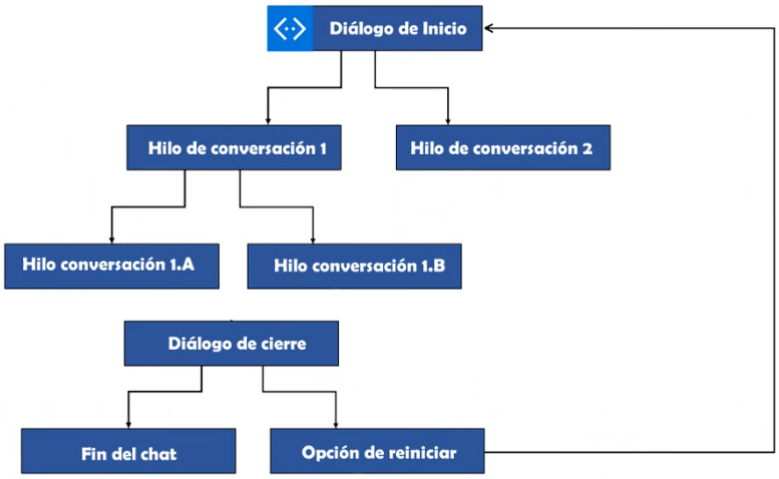

Figura 4.6: Ejemplo de flujo de conversación de procedimientos.
Fuente: [13]

En el flujo de conversación cuando un diálogo invoca a otro, el generador del Bot agrega el nuevo diálogo a la parte superior de una pila de diálogos. El último diálogo ingresado en la pila es el que tiene el control de la conversación y se encarga de procesar cada nuevo mensaje que el usuario envíe hasta que este se cierre o se agregue un nuevo diálogo. Una vez que un diálogo es eliminado de la pila, el dialogo anterior en la pila asume el control de la conversación. El correcto diseño de un flujo de conversación de un bot se basa en comprender el uso de la pila de diálogos.[12]

Para el correcto diseño del flujo de procedimientos, se debe tener en la secuencia de pasos que un usuario debe seguir para completar una tarea. Pero, también se debe tener en cuenta que la secuencia de pasos no siempre es lineal u ordenada, por lo general, los usuarios tienden a cambiar de opinión. Es por eso que, aunque el bot se base en procedimientos, se debe contemplar las interrupciones que pueden ocurrir durante la conversación.[12]

Para el manejo de interrupciones se debe considerar los siguientes puntos, los cuales se basan en la pregunta: ¿Cómo debe responder el bot en caso de una interrupción?:

- Insistir en que el usuario responda a la pregunta formulada por el bot.
- Pasar por alto todo el flujo de conversación que el bot ha tenido con el usuario hasta el momento y reiniciar la pila de diálogos, es decir, empezar desde cero e intentar procesar la información que el usuario ingresó.
- Intentar procesar la información que el usuario ingresó e intentar retomar el hilo de la conversación para terminar la tarea del diálogo.

Cualquiera de los puntos anteriores puede ser el correcto, todo depende del escenario en el que se despliegue el bot.[12]

4.3.2. Diseño de navegación

Para el diseño de navegación de un bot se debe tener en cuenta que un bot no es una aplicación tradicional, es decir, no cuenta con botones de navegación (adelante, atrás) ni rutas de navegación. Esto dificulta la aplicación alguna técnica de navegación estandarizada dentro de un bot. Debido a que el diseño de navegación va ligado con el manejo de interrupciones se debe considerar un correcto manejo de interrupciones para brindar una buena experiencia de usuario.[12]

Los detalles de la navegación dentro de un bot dependerán en gran medida de las características, funcionalidades y campo de aplicación de un bot. En general, independientemente del bot que se esté desarrollando se debe evitar los riesgos comunes que denotan interfaces conversacionales mal diseñadas. Estos riesgos se describen en cinco personalidades del bot: el "bot terco", el "bot despistado", el "bot misterioso", el "bot capitán obviedad" y el "bot que no puede olvidar". [12]

A continuación, se detalla cada personalidad:

El “bot terco”: en este tipo de personalidad el bot insiste en mantener el curso actual de la conversación, incluso cuando el usuario intenta dirigir las cosas en otra dirección. Este tipo de personalidad se da debido a que los usuarios cambian de parecer, deciden cancelar o a veces desean volver empezar por completo. Para evitar este tipo de personalidad, al diseñar el bot se debe tener en cuenta que un usuario

puede intentar cambiar el curso de la conversación en cualquier momento, no se debe ignorar la entrada del usuario y seguir repitiendo la misma pregunta en un bucle interminable.

Una manera de mitigar este comportamiento es especificar un número máximo de reintentos para cada entrada, a pesar de que el bot no haga nada para comprender al usuario y responder de manera apropiada, al menos evitará hacer la misma pregunta en un bucle infinito.[12]

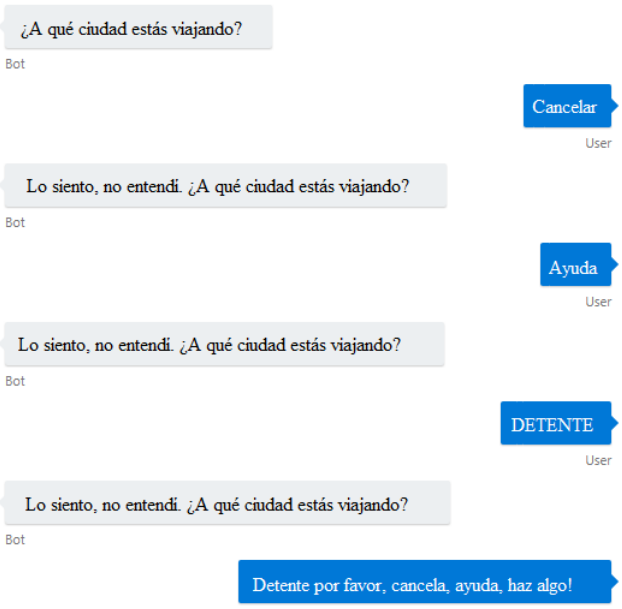

Figura 4.7: Ejemplo personalidad “bot terco”.
Fuente: [12]

El “bot despistado”: en este tipo de personalidad el bot responde de manera absurda cuando no entiende la entrada de un usuario. A pesar de que el usuario intente retomar la conversación con palabras clave, como "ayuda" o "cancelar" el bot no responderá de manera apropiada. Para evitar este tipo de personalidad, se debe implementar clases intermedias que procesen palabras claves e incluso las pasen por alto si es necesario, no se debe diseñar diálogos exclusivos para el tratamiento de palabras clave.[12]

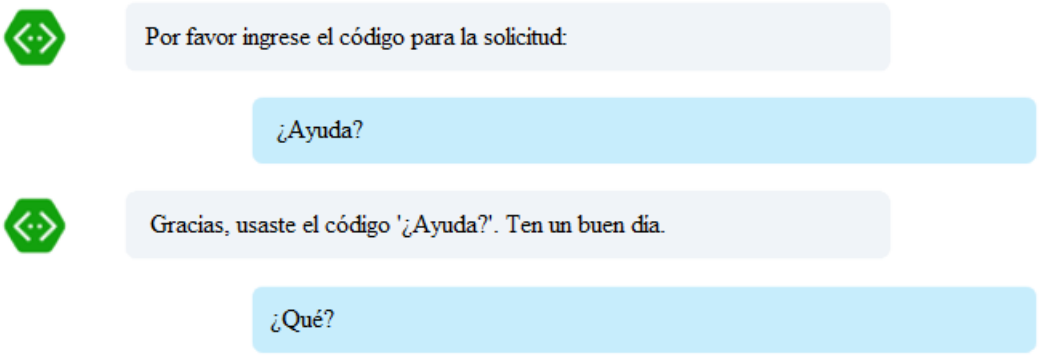

Figura 4.8: Ejemplo personalidad “bot despistado”.
Fuente: [12]

El “bot misterioso”: en este tipo de personalidad el bot no reconoce de inmediato la entrada del usuario de ninguna manera. Esta situación podría ser un indicador de que el bot tiene una interrupción del servicio o simplemente que el bot está ocupado procesando la entrada del usuario y aún no ha terminado de compilar la respuesta. Para evitar este tipo de personalidad, se debe diseñar el bot para reconocer de inmediato la entrada del usuario, eliminando así cualquier posibilidad de confusión en cuanto al estado del bot.

Si la respuesta tarda mucho tiempo en compilarse, se debe considerar la posibilidad de enviar un mensaje de "escritura" para indicar que el bot está trabajando. No se debe posponer la confirmación de la entrada del usuario hasta que el bot termine de procesar una respuesta.[12]

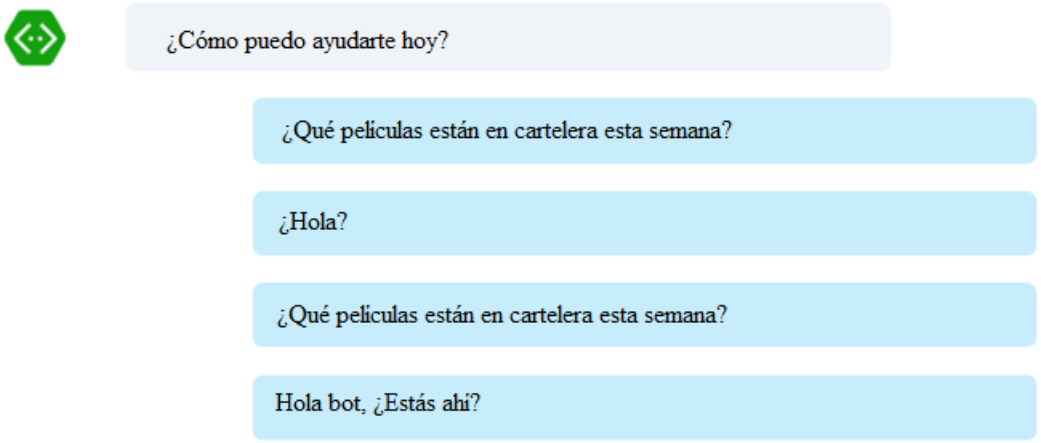

Figura 4.9: Ejemplo personalidad “bot misterioso”.
Fuente: [12]

El “bot capitán obviedad”: en este tipo de personalidad el bot proporciona información que el usuario no solicitó y que es totalmente obvia y, por tanto, inútil. Para evitar este tipo de personalidad, se debe diseñar el bot para proporcionar información útil, lo cual aumentará la posibilidad de interacción con el bot.[12]

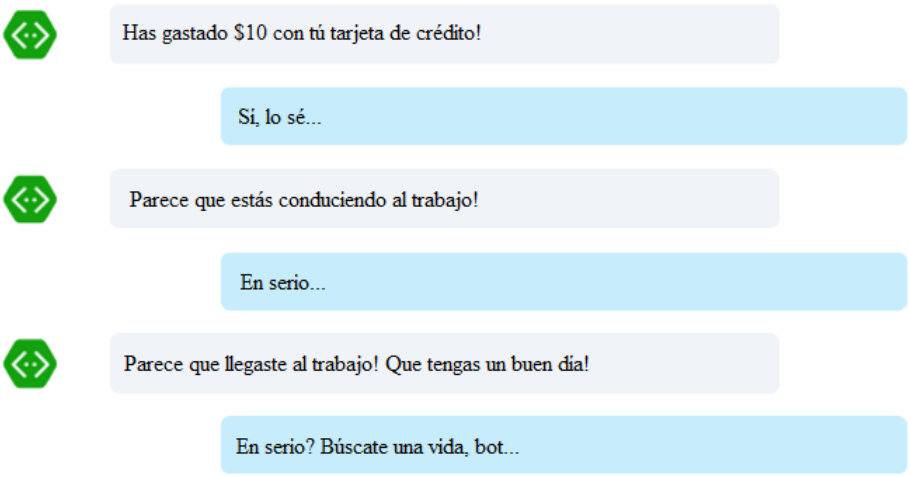

Figura 4.10: Ejemplo personalidad “bot capitán obviedad”.
Fuente: [12]

El “bot que no puede olvidar”: en este tipo de personalidad el bot integra de manera incorrecta información de conversaciones anteriores en la conversación

actual. Para evitar este tipo de personalidad, se debe diseñar el bot para que se mantenga en la conversación actual, el bot debe retomar conversaciones anteriores sólo si el usuario así lo manifiesta. Mantener el hilo de la conversación reduce la posibilidad de confusión, frustración y mejora la experiencia de usuario, lo cual aumenta la probabilidad de que el usuario continúe interactuando con el bot.[12]

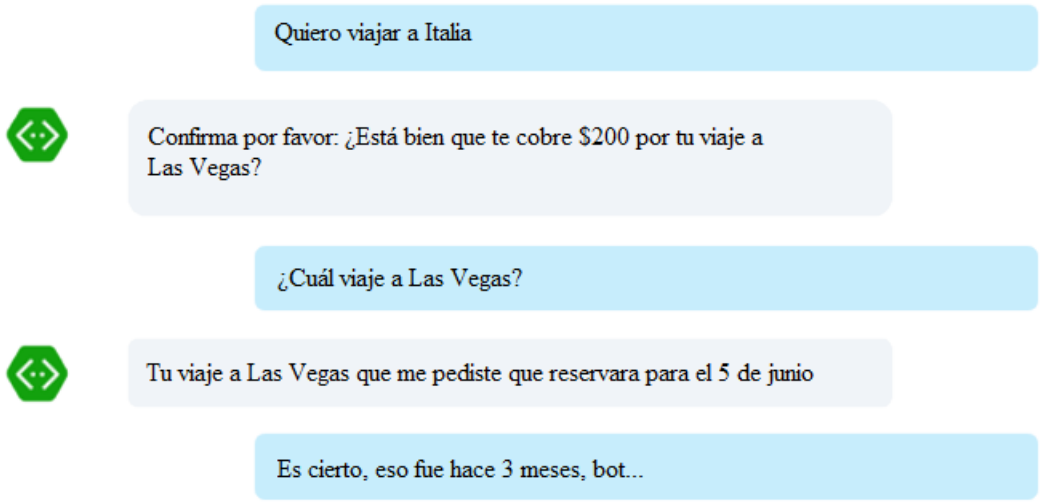

Figura 4.11: Ejemplo personalidad "bot que no puede olvidar".
Fuente: [12]

4.3.3. Diseño de flujos de conversación

Algo muy común al momento de desarrollar un bot es el cuestionamiento del diseño, ya que se desea un bot de calidad y que pueda proporcionar una buena experiencia de usuario.

Todas las buenas prácticas de desarrollo de aplicaciones deben ser migradas al desarrollo de bots. Esto implica desarrollar de forma simple y objetiva todas las funciones de un bot, es decir, se debe desarrollar diálogos que guíen a los usuarios a realizar actividades de manera correcta. Un excelente bot se caracteriza por realizar de forma correcta las actividades para las que fue desarrollado.[13]

Para facilitar el diseño de flujos de conversación Microsoft plantea el uso de tablas de flujos de comunicación, las cuales contienen información referente a una conversación, lo cual permite sintetizar y clarificar el desarrollo de diálogos.[13]

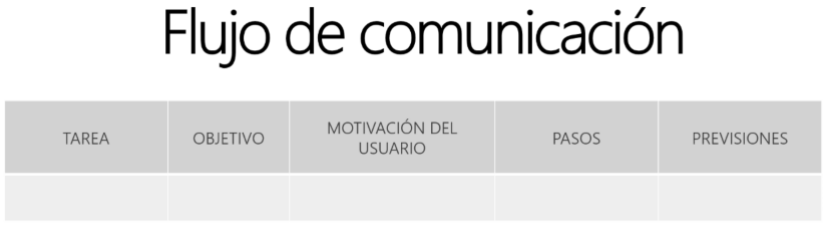

Figura 4.12: Tabla de flujo de comunicación.
Fuente: [13]

Según el planteamiento de Microsoft, en esta tabla se debe incluir las tareas principales de un flujo de conversación, así como los objetivos de cada tarea.

Es importante aclarar cuál es la motivación de un usuario, que utiliza el bot, para ejecutar una determinada tarea. Además, se debe determinar los pasos necesarios para cumplir con el objetivo de la conversación. Para determinar los pasos se debe tener en cuenta las etapas de esa funcionalidad en otras plataformas.[13]

En la determinación de pasos, se debe tomar en cuenta la navegación, las interrupciones, validaciones de información y otros eventos que pueden afectar el flujo de conversación.

Es decir, en esta herramienta de diseño de flujos de conversación se debe simplificar los procesos que se encuentran envueltos dentro de las diferentes funcionalidades que compondrán el bot.

4.4. Descripción de Bot Framework para el desarrollo de chatbots

Bot Framework es una herramienta open source, creada por Microsoft, para facilitar el desarrollo de bots. Bot Framework fue concebido con la idea de permitir el desarrollo de código único y que este pueda ser desplegado en varias plataformas de conversación.[13]

Como ya se describió anteriormente, Bot Framework proporciona los elementos necesarios para construir bots inteligentes y conectarlos con diversas plataformas. Permite incorporar la lógica de negocio dependiendo del ámbito de aplicación. Además, permite la utilización de aplicaciones externas para mejorar el funcionamiento del bot.

4.4.1. Funcionamiento de Bot Framework

En la figura 4.13 se muestra la forma de comunicación de Bot Framework:

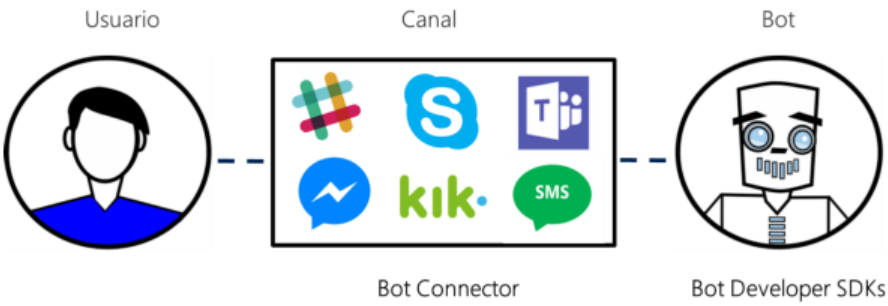

Figura 4.13: Forma de comunicación de Bot Framework.
Fuente: [13]

Entre los elementos que constituyen el diagrama de comunicación, se encuentran el usuario, el canal y el bot.

- **Usuario:** es el cliente que va a estar interactuando o conversando con el bot mediante mensajes de texto, gráficos o voz.
- **Canal:** es la aplicación o plataforma donde se va a desplegar el bot.
- **Bot:** es la aplicación que contiene varias funcionalidades o servicios, los cuales son expuestos a los usuarios en forma de una conversación.

El puente de enlace entre el bot y los canales es un servicio online conocido como Bot Connector.[13]

4.4.2. Arquitectura de Bot Framework

Bot Framework está constituido por dos componentes el Bot Builder SDK y el Bot Connector. A pesar de que los canales de conversación, en los cuales se despliega el bot, son parte del ecosistema de funcionamiento de Bot Framework no se los considera como parte de la arquitectura, esto debido a que los canales son independientes de Bot Framework.

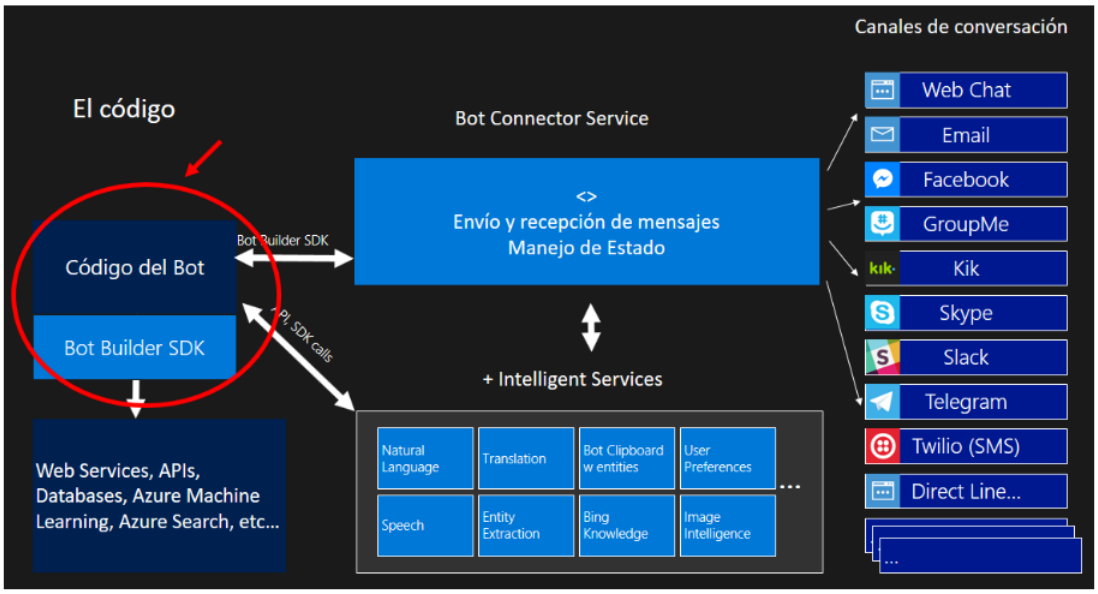

Figura 4.14: Arquitectura de Bot Framework.
Fuente: [13]

Bot Builder SDK

Bot Builder SDK es una librería en C# y Node.js que se encarga de gestionar la conversación, al mismo tiempo que se conecta con el código del bot. El código del bot, es donde se encuentra la lógica y funcionalidad del bot, a su vez es en dónde se utilizan las aplicaciones externas. [13]

El SDK proporciona bibliotecas, ejemplos y herramientas para la creación de bots. Además, contiene flujos de diálogo integrados que permiten manejar las interacciones con el usuario.[12]

Bot Connector

Bot Connector es un servicio online que permite conectar el bot con uno o más canales, para esto utiliza una API Rest que se encuentra implementada en el código del bot. Además de manejar el envío y recepción de mensajes, Bot Connector puede conectarse con servicios de IA para: guardar el estado de una conversación, permitir la comunicación con usuarios que no tengan el mismo idioma y recopilar información del funcionamiento del bot.[13]

Este servicio permite al bot comunicarse a través de muchos canales. La comunicación se realiza mediante una comunicación REST, el intercambio de información es en formato json. Este tipo de funcionamiento permite que la comunicación sea universal, independientemente del canal o plataforma.[13]

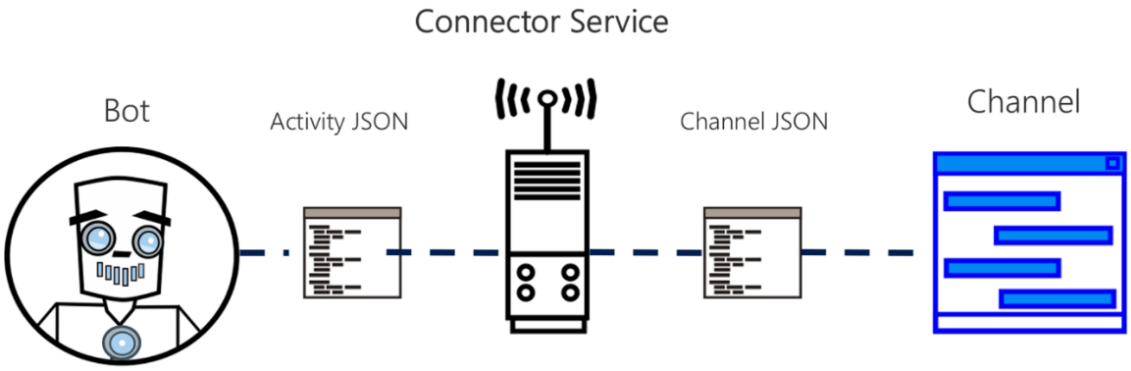

Figura 4.15: Funcionamiento de Bot Connector.
Fuente: [13]

Cabe recalcar que la publicación del bot en distintos canales se la hace mediante Azure Bot Service, el cual contiene Bot Connector. A pesar de esto, se puede realizar el despliegue en cualquier otro servicio o en un entorno local.

4.4.3. Seguridad con Bot Framework

Para que Bot Connector pueda acceder al bot desde distintos servicios o canales, se utiliza una configuración basada en https a través de un canal seguro y mediante autenticación. Esta autenticación se la realiza al momento de registrar un bot en la plataforma de Bot Framework. Bot Connector especifica un token de acceso en la cabecera de cada solicitud, lo cual asegura que la comunicación con el bot se produzca únicamente con el token de acceso.[12]

Para la conexión entre el bot y Bot Connector, Bot Framework utiliza cuatro tecnologías de autenticación, las mismas que se describen en la siguiente tabla.

Tecnología	Descripción
SSL / TLS	SSL / TLS se usa para todas las conexiones de servicio a servicio. Los certificados se usan para establecer la identidad de todos los servicios HTTPS.
Oauth 2.0	El inicio de sesión de Oauth 2.0 en el servicio de inicio de sesión de cuenta de Microsoft (MSA) / ADD v2 se utiliza para generar un token seguro que un bot puede usar para enviar mensajes.
Token web JSON (JWT)	Los tokens web JSON se utilizan para codificar tokens que se envían desde y hacia el bot.
Metadatos OpenID	El servicio Bot Connector publica una lista de tokens válidos que utiliza para firmar sus propios tokens JWT a los metadatos OpenID en un punto final estático bien conocido.

Tabla 4.2: Tecnologías de autenticación que usa Bot Framework.

Fuente: [10, 12]

4.5. Planteamiento del prototipo propuesto

Para el desarrollo del agente conversacional se debe tener en cuenta la arquitectura que se utilizará. Bot Framework utiliza la arquitectura MVC (Model-View-Controller) para el desarrollo de chatbots, bajo esta premisa el bot tendrá una arquitectura MVC. La arquitectura deberá ser ajustada tanto al contexto de aplicación como al framework usado.

Dentro la arquitectura se debe contemplar el uso de un procesador de lenguaje natural ya que Bot Framework no contiene un procesador de lenguaje propio.

Una consideración que se debe tener en cuenta es que el chatbot servirá como una nueva interfaz de usuario de un sistema existente. Es por eso que dentro de la arquitectura se incorporará procesos de un sistema de compras online existente. El sistema existente expondrá mediante servicios web sólo los servicios esenciales que necesite el chatbot.

Debido a que la investigación se centra en presentar las mejores técnicas de

desarrollo de agentes conversacionales las funcionalidades que contendrá el bot serán limitadas. Pese a que el bot contendrá pocas funciones, este será desarrollado de manera genérica permitiendo la escalabilidad y reutilización del mismo.

Además, se debe considerar el o los canales y plataformas en los cuales se va a desplegar. En este caso por tratarse de un prototipo se realizará un despliegue local, sin embargo, se explicará la manera de realizar un despliegue en otros canales mediante el uso de servicios existentes en la web.

4.5.1. Requerimientos del chatbot

El chatbot propuesto permitirá realizar las siguientes acciones:

- Ingresar en el sistema de compras online.
- Consultar productos.
- Agregar productos a la orden de compra.
- Guardar la orden de compra.
- Visualizar la orden de compra realizada.

Cada una de las acciones detallas se convertirán en flujos de diálogo.

4.6. Descripción del prototipo planteado

4.6.1. Arquitectura para el desarrollo del chatbot

Como ya se planteó en el punto anterior para el desarrollo del bot se usará la arquitectura MVC, la cual está definida en el marco de trabajo de Bot Builder SDK.

En el contexto de desarrollo de chatbots no existe una vista como tal, sino más bien existen diálogos que trabajan como una especie de vista. Es por eso que las vistas, en el desarrollo de bots, se las conoce como diálogos.

En la figura 4.16 se muestra la interacción de los componentes de la arquitectura MVC orientados al prototipo propuesto.

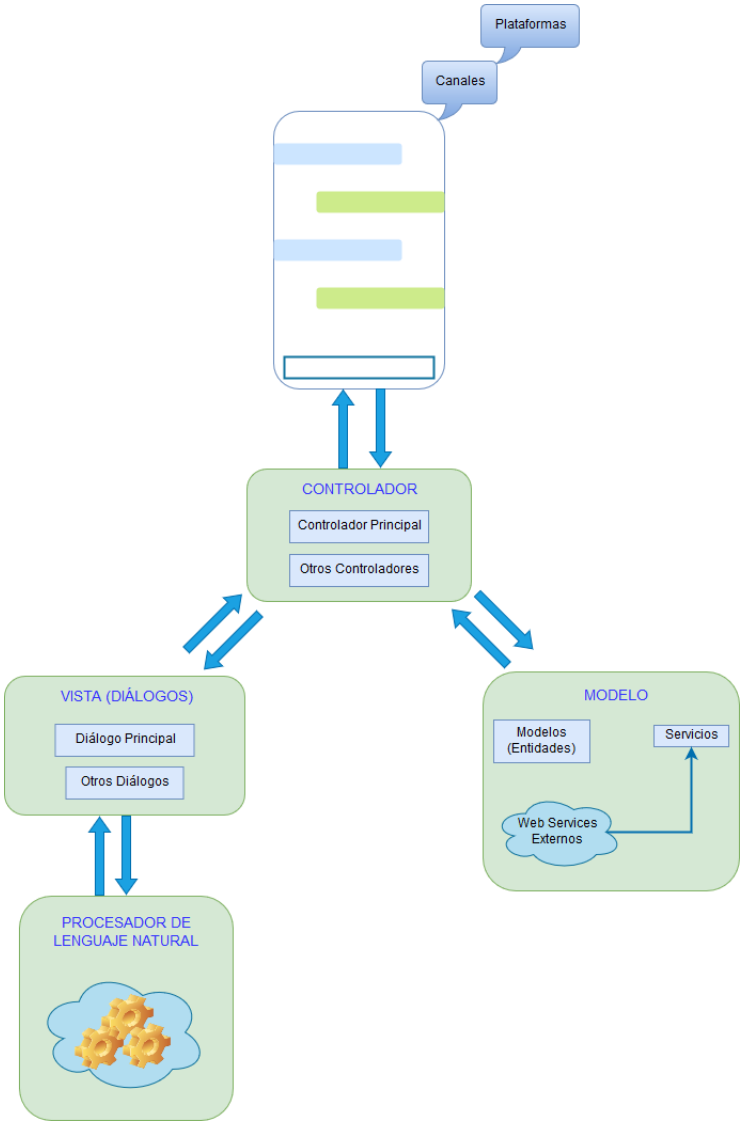

Figura 4.16: Arquitectura MVC para el desarrollo del chatbot.
Fuente: Elaborado por el investigador y basado en [10]

La arquitectura contendrá tres componentes modelo, diálogo y controlador.

- **Modelo:** contendrá las entidades, modelos u objetos de negocio que se usarán para la comunicación del bot con el controlador. Además, las clases que consumirán los servicios del sistema de compras.
- **Diálogo:** contendrá el diálogo principal del flujo de conversación, el cual manejará la pila de diálogos. También, contendrá los diálogos secundarios que realizarán una función específica dentro del flujo de diálogo.

- **Controlador:** contendrá el controlador principal, el cual se encargará del intercambio de información entre el canal o plataforma y el bot. Además, contendrá los controladores secundarios que se encargarán de la comunicación entre los modelos y los diálogos.

4.6.2. Arquitectura del prototipo

Para comprender lo que el usuario está intentando realizar, el bot será basado en intenciones. El reconocimiento de intenciones lo realizará el procesador de lenguaje natural. En la figura 4.17 se muestra la arquitectura a nivel general que tendrá el prototipo propuesto.

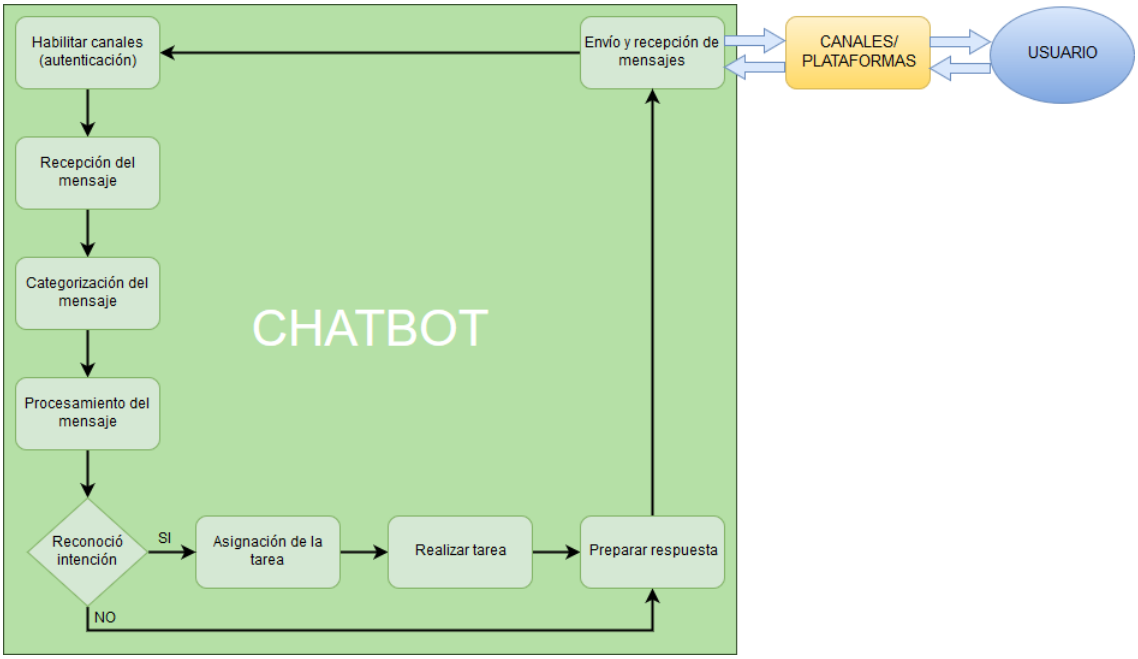

Figura 4.17: Arquitectura del prototipo propuesto.
Fuente: Elaborado por el investigador y basado en [10]

El prototipo planteado se compone de tres elementos chatbot, canales o plataformas y usuario.

- **Chatbot:** es una aplicación que se encarga de procesar la información recibida y genera una respuesta para el usuario. Realiza la comunicación con los canales o plataformas mediante protocolos HTTP.
- **Canales o Plataformas:** son los medios en los cuales se desplegará el agente conversacional, estos dependen del servicio que se use para desplegar el bot. Además, se encargan de conectar el bot con el usuario.

- **Usuario:** es el cliente que interactúa con el agente conversacional. Para la comunicación con el bot el usuario puede utilizar texto, imágenes o mensajes de voz. En este caso el bot únicamente reconoce texto.

4.7. Desarrollo del prototipo propuesto

4.7.1. Procesador del lenguaje natural usado en el chatbot

En el análisis de las tecnologías usadas para el desarrollo de chatbots, se realizó un estudio de los procesadores de lenguaje natural. Esto debido a que los principales procesadores de lenguaje son parte de las herramientas que permiten la creación de agentes conversacionales. Una vez realizado una comparativa basada en el estudio se decidió utilizar DialogFlow de Google. Las principales características por las cuales fue elegido están:

- Simplicidad para entrenar un agente.
- Licenciamiento gratuito de la mayor parte de sus funcionalidades.
- Facilidad para consumir su API.

4.7.2. Metodología para el desarrollo del chatbot

Como paso previo al desarrollo del chatbot, se debe seleccionar una metodología que se ajuste a los objetivos planteados. Una vez analizadas las diferencias entre las principales metodologías de desarrollo de software, las mismas que se encuentran en la tabla 2.1, se decide usar una metodología ágil que es la que más se ajusta a la realización del prototipo.

Antes de elegir la metodología ágil que se va a utilizar para el desarrollo del prototipo se realizará una descripción de las principales metodologías ágiles para finalmente poder compararlas y realizar una elección.

4.7.2.1. Metodología XP (Extreme Programming)

Es una metodología ágil centrada en potenciar las relaciones interpersonales como clave para el éxito en el desarrollo de software. Esta metodología promueve el trabajo en equipo, se preocupa por el aprendizaje de los desarrolladores y trata de crear un buen clima de trabajo.[34, 22]

Se basa en un proceso de comunicación continuo entre el cliente y el equipo de desarrollo, simplifica el desarrollo de las tareas y permite afrontar los cambios de manera muy simple. XP es usada en proyectos con requisitos imprecisos y cambiantes, es decir, en proyectos en los cuales existe un alto riesgo técnico.[34, 22]

A continuación, se explican varios conceptos que son usados dentro de XP.

Historias de usuario

El tratamiento de las historias de usuario es muy dinámico y flexible, en cualquier momento las historias de usuario pueden romperse o reemplazarse por historias más específicas o más generales. Cada historia de usuario es comprensible y se encuentra bien delimitada para que sea implementada un corto periodo de tiempo.[23, 22]

Por lo general las historias de usuario deben ser independientes de otras. El hecho de que exista alguna dependencia entre las historias crea dificultades en la planificación y ejecución de las mismas. Para evitar este inconveniente se puede combinar las historias o se debe plantear las historias de forma diferente.[23, 22]

Una buena historia de usuario no debe exceder el esfuerzo de 2-3 personas por cada semana de trabajo.[23]

Planeación del lanzamiento

Utilizando historias de usuario, el cliente define los requerimientos y precisa la importancia de cada uno de ellos. Con base en las historias el equipo de desarrolladores estima el tiempo y esfuerzo que se necesitará para terminar cada historia.[34]

Iteraciones

Son ciclos cortos de desarrollo, las cuales permiten mostrar funcionalidades terminadas al cliente y recibir una retroalimentación para mejorar la calidad del software. El término de funcionalidades terminadas se relaciona con las pruebas de aceptación. Las iteraciones permiten presentar diseños simples sin tener en cuenta el cambio que puedan presentar las funcionalidades en un futuro.[34]

Velocidad del proyecto

Es una medida de la capacidad que tiene el equipo de desarrollo para terminar las historias de usuario en una iteración. Esta medida se calcula contabilizando el número de historias de usuario que se han terminado en una iteración.[35]

Programación en pareja

Esta metodología promueve el trabajo en parejas, es decir, que el código sea escrito en parejas usando un mismo ordenador. Para XP la programación en parejas incrementa la calidad del código sin tener impacto alguno en la fecha de entrega.[22]

Reuniones diarias

El objetivo de tener reuniones diarias es mantener la comunicación entre el equipo de desarrollo para compartir problemas y soluciones. En este tipo de reuniones, la mayoría de los participantes escuchan lo que evita perder tiempo, para estas reuniones plantea la posibilidad de que sean en círculo y de pie.[22]

Simplicidad

Un diseño simple se implementa más rápido que uno complejo. Por eso XP propone implementar diseños simples pero funcionales. Además, sugiere no implementar funcionalidades que no correspondan a la iteración en la que se esté trabajando. El código debe ser: testeable, legible, comprensible y explicable.[22]

Metáforas

Una metáfora es algo que todos entienden sin necesidad de explicaciones complejas. La metodología XP sugiere utilizar este concepto como una manera fácil de explicar el propósito del proyecto, así como para guiar la estructura y arquitectura del mismo.[22]

Solución "spike"

Consiste en plantear soluciones muy simples a problemas, lo cual permite abordar el problema en concreto aislándolo de otro tipo de preocupaciones.[22]

Refactorización

La refactorización o recodificación consiste en volver a escribir parte del código sin afectar la funcionalidad del mismo, esto se realiza para optimizar el código.[22]

Glosario de términos

El uso de un glosario de términos y el correcto nombramiento de métodos y clases ayuda a comprender el diseño. Además, facilita la escalabilidad de la aplicación y la reutilización del código.[36]

Riesgos

Para tratar de reducir la frecuencia en que puedan surgir problemas potenciales durante el desarrollo, esta metodología plantea el trabajo en parejas.[36]

Funcionalidad extra

No se deben añadir funcionalidades extras al software ya que esto implica desperdicio de tiempo y recursos. Aunque se piense que serán utilizadas, sólo el 10 % son de utilidad.[36]

Codificación

Una vez que se han definido las historias de usuario, se procede a transformar estas historias en código funcional. En este punto la participación del cliente es de vital importancia, ya que él debe especificar detalladamente cada historia de usuario y debe verificar el correcto funcionamiento de cada historia que haya sido implementada.[36]

Pruebas

Uno de los pilares de XP es el uso de pruebas comprobar el correcto funcionamiento del código. XP define varios puntos que se deben tomar en cuenta para realizar pruebas de funcionalidad.[36]

Debido a que una aplicación se encuentra compuesta por varias funcionalidades no tan extensas, las pruebas pretenden evaluar las funcionalidades en general teniendo en cuenta los requerimientos del software.[36]

4.7.2.2. Metodología Scrum

Define un marco para la administración de proyectos, esta metodología ha sido ampliamente usada y difundida durante los 10 últimos años. Está diseñada para proyectos, que, por su naturaleza, tienen cambios en los requerimientos iniciales.[34]

Sus características se pueden definir en dos grupos: las iteraciones, conocidos como sprints, que tienen una duración de 30 días y generan un producto o entregable que se muestra al cliente; y las reuniones que se producen a lo largo del proyecto, estás tienen una duración de 15 minutos y se producen diariamente, esto permite una mejor coordinación.[34]

4.7.2.3. Crystal Methodologies

Se tratan de un conjunto de metodologías para el desarrollo de software, que se caracterizan por estar centradas en las personas que componen el equipo y la reducción del número de entregables producidos. Esta metodología define el desarrollo de software como un juego cooperativo de invención y comunicación que se limita por los recursos que se usen.[34]

El elemento principal es el equipo de desarrollo por lo cual se invierten esfuerzos en mejorar sus habilidades y destrezas. Se definen políticas de trabajo en equipo, estas políticas dependen del tamaño del equipo. Según el tamaño del equipo de trabajo se ha establecido una clasificación por colores, por ejemplo: Crystal Clear (3 a 8 miembros) y Crystal Orange (25 a 50 miembros).[34]

4.7.2.4. DSDM (Dynamic Systems Development Method)

Definen un marco para el desarrollo de software, nace con el objetivo de crear una metodología unificada para el desarrollo rápido de aplicaciones.[34]

Sus principales características son: es un proceso iterativo e incremental y el equipo de desarrollo y el usuario trabajan juntos. Propone cinco fases: estudio de viabilidad, estudio del negocio, modelo funcional, diseño y construcción, y finalmente la implementación. Las tres últimas fases son iterativas, además, en todas las fases existe un proceso de retroalimentación.[34]

4.7.2.5. FDD (Feature Driven Development)

Define un proceso iterativo que consta de cinco pasos. Las iteraciones son cortas, con una duración de hasta dos semanas. Se centra en las fases de diseño e implementación del sistema partiendo de una lista de características que debe reunir el software.[34]

4.7.2.6. Comparativa de metodologías ágiles

En la tabla 4.3 se presenta una comparación de las principales metodologías ágiles para el desarrollo de software, en la cual se evalúan tres parámetros con una valoración de uno a cinco.[37]

Los parámetros que se evalúan son[37]:

- **Vista del sistema como algo cambiante:** se puntúa teniendo en cuenta la metodología de desarrollo que más se adapta a cada etapa del desarrollo del proyecto.
- **Colaboración:** se puntúa tomando como base la colaboración de cada uno de los miembros del equipo de desarrollo.
- **Características específicas de la metodología:** se puntúa teniendo en cuenta la simplicidad, excelencia técnica, resultados, adaptabilidad, entre otros.

	Crystal	DSDM	FDD	Scrum	XP
Sistema como algo cambiante	4	3	3	5	5
Colaboración	5	4	4	5	5
Características de la metodología (CM)					
- Resultados	5	4	4	5	5
- Simplicidad	4	3	5	5	5
- Adaptabilidad	5	3	3	4	3
- Excelencia técnica	3	4	4	3	4
- Prácticas de colaboración	5	4	3	4	5
Media CM	**4.4**	**3.6**	**3.8**	**4.2**	**4.4**
Media Total	**4.5**	**3.6**	**3.6**	**4.7**	**4.8**

Tabla 4.3: Comparativa metodologías ágiles.

Fuente: [37]

Según la valoración descrita en la tabla anterior, XP tiene la puntuación más alta con 4.8, seguida por Scrum con 4.7. Para la elección de la metodología se debe tener en cuenta la que más se ajuste a la naturaleza del proyecto. Por tal razón se decide utilizar la metodología XP (eXtreme Programming) para el desarrollo del prototipo de chatbot. Esta metodología permite obtener productos tangibles en cortos periodos de tiempo e incluir funcionalidades no contempladas en las etapas de definición del proyecto.

4.7.2.7. Metodología a aplicar

En el presente proyecto se aplicó la metodología XP, la cual permite un análisis, diseño, desarrollo y pruebas de manera rápida, además utilizar iteraciones para la creación de productos incrementales. Esta metodología consta de las siguientes fases, las cuales serán implementadas en el desarrollo del prototipo[36]:

- **Fase: Planificación del proyecto**
- **Fase: Diseño**
- **Fase: Codificación**
- **Fase: Pruebas**

4.7.3. Aplicación de la metodología

4.7.3.1. Fase: Planificación del proyecto

Para la planificación del proyecto se usan historias de usuario, las cuales permiten definir los requerimientos que debe tener el agente conversacional. En esta parte se tuvo en cuenta que es un prototipo, por lo cual las historias de usuario se plantearon desde el punto de vista de un usuario y basándose en la experiencia del investigador.

En la tabla 4.4 se definen los roles de los involucrados en el proyecto, lo cual permite definir de mejor manera las historias de usuario.

Rol	Descripción
Desarrollador	Autor del trabajo del presente trabajo de investigación.
Usuario Final	Personas que usarán el chatbot y realizarán pruebas de las funcionalidades.

Tabla 4.4: Descripción de roles.

Fuente: Elaborado por el investigador.

Una vez definidas las funcionalidades del agente conversacional, se generaron las Historias de Usuario acorde a los requerimientos planteados. La cuales se muestran en las siguientes tablas:

	Historia de Usuario
Código: H1	**Usuario:** Usuario final
Nombre historia: Diálogo de inicio	
Prioridad en negocio: Media	**Riesgo en desarrollo:** Medio
Puntos estimados: 1	**Iteración asignada:** 1
Programador responsable: Erick Gamboa	
Descripción: Una vez que se inicie la conversación el chatbot debe saludar y presentarse, aclarando que es un chatbot.	
Observación: El saludo estará definido en el procesador del lenguaje natural. Además, si el servicio no está disponible se presentará un saludo indicando que no se encuentra disponible.	

Tabla 4.5: Historia de Usuario - Diálogo de inicio.

Fuente: Elaborado por el investigador.

	Historia de Usuario
Código: H2	**Usuario:** Usuario final
Nombre historia: Diálogo de saludo	
Prioridad en negocio: Media	**Riesgo en desarrollo:** Medio
Puntos estimados: 1	**Iteración asignada:** 1
Programador responsable: Erick Gamboa	
Descripción: El chatbot debe reconocer las palabras que el usuario use para saludar y responder el saludo. Además, debe presentar las opciones que el usuario puede realizar.	
Observación: Las opciones que presente el chatbot están basadas en las funcionalidades que puede realizar.	

Tabla 4.6: Historia de Usuario - Diálogo de saludo.

Fuente: Elaborado por el investigador.

	Historia de Usuario
Código: H3	**Usuario:** Usuario final
Nombre historia: Diálogo de ingreso al sistema	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Alto
Puntos estimados: 3	**Iteración asignada:** 2
Programador responsable: Erick Gamboa	
Descripción: El chatbot debe presentar la posibilidad de ingresar al sistema, para lo cual debe solicitar usuario y contraseña.	
Observación: Sólo los usuarios que estén registrados en el sistema tendrán acceso a las funcionalidades que necesiten que el usuario esté autenticado.	

Tabla 4.7: Historia de Usuario - Diálogo de ingreso al sistema.

Fuente: Elaborado por el investigador.

	Historia de Usuario
Código: H4	**Usuario:** Usuario final
Nombre historia: Diálogo para consultar productos	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Alto
Puntos estimados: 3	**Iteración asignada:** 3
Programador responsable: Erick Gamboa	
Descripción: El chatbot debe presentar los productos que se encuentren en el sistema.	
Observación: Dependiendo de la cantidad de productos se debe definir algún filtro para presentar sólo unos pocos productos, tomando en cuenta que se trata de un diálogo. Esta funcionalidad no necesita que el usuario se haya autenticado.	

Tabla 4.8: Historia de Usuario - Diálogo para consultar productos.

Fuente: Elaborado por el investigador.

	Historia de Usuario
Código: H5	**Usuario:** Usuario final
Nombre historia: Diálogo para agregar productos a la orden de compra.	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Alto
Puntos estimados: 3	**Iteración asignada:** 4
Programador responsable: Erick Gamboa	
Descripción: El chatbot debe preguntar si desea añadir un producto seleccionado y la cantidad del producto para añadirlo a la compra.	
Observación: Se debe tener en cuenta el stock existente de un producto, para realizar esta actividad el usuario debe encontrarse autenticado caso contrario el chatbot debe permitirle autenticarse.	

Tabla 4.9: Historia de Usuario - Diálogo para agregar productos a la orden de compra.

Fuente: Elaborado por el investigador.

	Historia de Usuario
Código: H6	**Usuario:** Usuario final
Nombre historia: Diálogo para guardar la orden de compra.	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Alto
Puntos estimados: 3	**Iteración asignada:** 5
Programador responsable: Erick Gamboa	
Descripción: El chatbot debe preguntar si desea terminar de hacer el pedido y guardar la orden de compra.	
Observación: Para realizar esta actividad el usuario debe encontrarse autenticado caso contrario el chatbot debe permitirle autenticarse.	

Tabla 4.10: Historia de Usuario - Diálogo para guardar la orden de compra.

Fuente: Elaborado por el investigador.

	Historia de Usuario
Código: H7	**Usuario:** Usuario final
Nombre historia: Diálogo para visualizar la orden de compra.	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Alto
Puntos estimados: 2	**Iteración asignada:** 6
Programador responsable: Erick Gamboa	
Descripción: El chatbot debe permitir visualizar una orden de compra realizada por el usuario.	
Observación: Para visualizar la orden de compra el chatbot debe solicitar el número de orden que desea visualizar. Para realizar esta actividad el usuario debe encontrarse autenticado caso contrario el chatbot debe permitirle autenticarse.	

Tabla 4.11: Historia de Usuario - Diálogo para visualizar la orden de compra.

Fuente: Elaborado por el investigador.

	Historia de Usuario
Código: H8	**Usuario:** Usuario final
Nombre historia: Diálogo de fin	
Prioridad en negocio: Media	**Riesgo en desarrollo:** Medio
Puntos estimados: 1	**Iteración asignada:** 1
Programador responsable: Erick Gamboa	
Descripción: Una vez que termine la conversación el chatbot presentará un mensaje de despedida.	
Observación: El mensaje de despedida estará definido en el procesador del lenguaje natural.	

Tabla 4.12: Historia de Usuario - Diálogo de fin.

Fuente: Elaborado por el investigador.

	Historia de Usuario
Código: H9	**Usuario:** Desarrollador
Nombre historia: Entrenamiento del procesador del lenguaje natural	
Prioridad en negocio: Muy alta	**Riesgo en desarrollo:** Alto
Puntos estimados: 4	**Iteración asignada:** 7
Programador responsable: Erick Gamboa	
Descripción: El procesador del lenguaje natural deberá ser entrenado con posibles frases que los usuarios usarían para comunicarse con el chatbot.	
Observación: El procesador será entrenado con pocas frases de prueba.	

Tabla 4.13: Historia de Usuario - Entrenamiento del procesador del lenguaje natural.

Fuente: Elaborado por el investigador.

En base a las Historias de Usuario que se han definido, se generaron las siguientes actividades, las mismas que se muestran en las siguientes tablas:

Historia: Diálogo de inicio

	Tarea
Código: T1	**Código de historia:** H1
Nombre tarea: Diseñar el diálogo de inicio	
Tipo de tarea: Desarrollo	**Puntos estimados:** 0.3
Tiempo: 1 día	
Programador responsable: Erick Gamboa	
Descripción: Definir el diálogo que dará inicio a la conversación con el chatbot.	

Tabla 4.14: Actividad 1 - Historia 1 - Diseñar el diálogo de inicio.

Fuente: Elaborado por el investigador.

	Tarea
Código: T2	**Código de historia:** H1
Nombre tarea: Diseñar el diálogo de fuera de línea	
Tipo de tarea: Desarrollo	**Puntos estimados:** 0.3
Tiempo: 1 día	
Programador responsable: Erick Gamboa	
Descripción: Definir el diálogo que dará el chatbot cuando el sistema no se encuentre disponible.	

Tabla 4.15: Actividad 2 - Historia 1 - Diseñar el diálogo de fuera de línea.

Fuente: Elaborado por el investigador.

	Tarea
Código: T3	**Código de historia:** H1
Nombre tarea: Implementar diálogo de inicio	
Tipo de tarea: Desarrollo	**Puntos estimados:** 0.4
Tiempo: 3 días	
Programador responsable: Erick Gamboa	
Descripción: Implementar los diálogos de inicio tanto en el chatbot como en el procesador de lenguaje natural.	

Tabla 4.16: Actividad 3 - Historia 1 - Implementar diálogo de inicio.

Fuente: Elaborado por el investigador.

Historia: Diálogo de saludo

	Tarea
Código: T1	**Código de historia:** H2
Nombre tarea: Diseñar el diálogo de saludo	
Tipo de tarea: Desarrollo	**Puntos estimados:** 0.3
Tiempo: 1 día	
Programador responsable: Erick Gamboa	
Descripción: Definir el diálogo que el chatbot presentará cuando una persona use palabras de saludo.	

Tabla 4.17: Actividad 1 - Historia 2 - Diseñar el diálogo de saludo.

Fuente: Elaborado por el investigador.

	Tarea
Código: T2	**Código de historia:** H2
Nombre tarea: Diseñar el diálogo para presentar funcionalidades	
Tipo de tarea: Desarrollo	**Puntos estimados:** 0.3
Tiempo: 1 día	
Programador responsable: Erick Gamboa	
Descripción: Definir el diálogo que presentará las funcionalidades del sistema.	

Tabla 4.18: Actividad 2 - Historia 2 - Diseñar el diálogo para presentar funcionalidades.

Fuente: Elaborado por el investigador.

	Tarea
Código: T3	**Código de historia:** H2
Nombre tarea: Implementar diálogo de saludo	
Tipo de tarea: Desarrollo	**Puntos estimados:** 0.4
Tiempo: 3 días	
Programador responsable: Erick Gamboa	
Descripción: Implementar los diálogos de saludo tanto en el chatbot como en el procesador de lenguaje natural.	

Tabla 4.19: Actividad 3 - Historia 2 - Implementar diálogo de saludo.

Fuente: Elaborado por el investigador.

Historia: Diálogo de ingreso al sistema

	Tarea
Código: T1	**Código de historia:** H3
Nombre tarea: Diseñar el diálogo de ingreso al sistema	
Tipo de tarea: Desarrollo	**Puntos estimados:** 0.4
Tiempo: 1 día	
Programador responsable: Erick Gamboa	
Descripción: Definir el diálogo que permita ingresar el nombre de usuario y contraseña.	

Tabla 4.20: Actividad 1 - Historia 3 - Diseñar el diálogo de ingreso al sistema.

Fuente: Elaborado por el investigador.

	Tarea
Código: T2	**Código de historia:** H3
Nombre tarea: Validar el ingreso al sistema	
Tipo de tarea: Desarrollo	**Puntos estimados:** 1.6
Tiempo: 2 días	
Programador responsable: Erick Gamboa	
Descripción: Se debe realizar la conexión con el sistema de compras para validar el usuario.	

Tabla 4.21: Actividad 2 - Historia 3 - Validar el ingreso al sistema.

Fuente: Elaborado por el investigador.

	Tarea
Código: T3	**Código de historia:** H3
Nombre tarea: Implementar diálogo de ingreso al sistema	
Tipo de tarea: Desarrollo	**Puntos estimados:** 1.0
Tiempo: 2 días	
Programador responsable: Erick Gamboa	
Descripción: Implementar el dialogo que permitirá la autenticación del usuario.	

Tabla 4.22: Actividad 3 - Historia 3 - Implementar diálogo de ingreso al sistema.

Fuente: Elaborado por el investigador.

Historia: Diálogo para consultar productos

	Tarea
Código: T1	**Código de historia:** H4
Nombre tarea: Diseñar el diálogo para presentar los productos	
Tipo de tarea: Desarrollo	**Puntos estimados:** 0.3
Tiempo: 1 día	
Programador responsable: Erick Gamboa	
Descripción: Definir el diálogo que permita mostrar los productos al usuario.	

Tabla 4.23: Actividad 1 - Historia 4 - Diseñar el diálogo para presentar los productos.

Fuente: Elaborado por el investigador.

	Tarea
Código: T2	**Código de historia:** H4
Nombre tarea: Diseñar el diálogo para filtrar los productos	
Tipo de tarea: Desarrollo	**Puntos estimados:** 0.3
Tiempo: 1 día	
Programador responsable: Erick Gamboa	
Descripción: Definir el diálogo para filtrar los productos a presentar al usuario.	

Tabla 4.24: Actividad 2 - Historia 4 - Diseñar el diálogo para filtrar los productos.

Fuente: Elaborado por el investigador.

	Tarea
Código: T3	**Código de historia:** H4
Nombre tarea: Consultar productos	
Tipo de tarea: Desarrollo	**Puntos estimados:** 1.6
Tiempo: 3 días	
Programador responsable: Erick Gamboa	
Descripción: Se debe realizar la conexión con el sistema de compras para consultar los productos existentes.	

Tabla 4.25: Actividad 3 - Historia 4 - Consultar productos.

Fuente: Elaborado por el investigador.

	Tarea
Código: T4	**Código de historia:** H4
Nombre tarea: Implementar diálogo para consultar productos	
Tipo de tarea: Desarrollo	**Puntos estimados:** 0.8
Tiempo: 3 días	
Programador responsable: Erick Gamboa	
Descripción: Implementar el diálogo que permitirá consultar productos en el sistema de compras.	

Tabla 4.26: Actividad 3 - Historia 4 - Implementar diálogo para consultar productos.

Fuente: Elaborado por el investigador.

Historia: Diálogo para agregar productos a la orden de compra

	Tarea
Código: T1	**Código de historia:** H5
Nombre tarea: Diseñar el diálogo para ingresar la cantidad de un producto	
Tipo de tarea: Desarrollo	**Puntos estimados:** 0.3
Tiempo: 1 día	
Programador responsable: Erick Gamboa	
Descripción: Definir el diálogo que permitirá ingresar la cantidad un producto que se desea añadir a la orden de compra.	

Tabla 4.27: Actividad 1 - Historia 5 - Diseñar el diálogo para ingresar la cantidad de un producto.

Fuente: Elaborado por el investigador.

	Tarea
Código: T2	**Código de historia:** H5
Nombre tarea: Diseñar el diálogo de confirmación al añadir un producto	
Tipo de tarea: Desarrollo	**Puntos estimados:** 0.3
Tiempo: 1 día	
Programador responsable: Erick Gamboa	
Descripción: Definir el diálogo que confirmará que el usuario quiere añadir un producto a la orden de compra.	

Tabla 4.28: Actividad 2 - Historia 5 - Diseñar el diálogo de confirmación al añadir un producto.

Fuente: Elaborado por el investigador.

	Tarea
Código: T3	**Código de historia:** H5
Nombre tarea: Verificación de stock disponible	
Tipo de tarea: Desarrollo	**Puntos estimados:** 1.4
Tiempo: 3 días	
Programador responsable: Erick Gamboa	
Descripción: Se debe realizar la conexión con el sistema de compras para verificar el stock de un producto antes añadirlo a la orden de compra.	

Tabla 4.29: Actividad 3 - Historia 5 - Verificación de stock disponible.

Fuente: Elaborado por el investigador.

	Tarea
Código: T4	**Código de historia:** H5
Nombre tarea: Implementar diálogo para agregar productos a la orden de compra	
Tipo de tarea: Desarrollo	**Puntos estimados:** 1.0
Tiempo: 3 días	
Programador responsable: Erick Gamboa	
Descripción: Implementar el diálogo que permitirá agregar productos a la orden de compra.	

Tabla 4.30: Actividad 4 - Historia 5 - Implementar diálogo para agregar productos a la orden de compra.

Fuente: Elaborado por el investigador.

Historia: Diálogo para guardar la orden de compra

	Tarea
Código: T1	**Código de historia:** H6
Nombre tarea: Diseñar el diálogo para confirmar la terminación de la compra	
Tipo de tarea: Desarrollo	**Puntos estimados:** 0.3
Tiempo: 1 día	
Programador responsable: Erick Gamboa	
Descripción: Definir el diálogo que confirmará que el usuario desea terminar la compra.	

Tabla 4.31: Actividad 1 - Historia 6 - Diseñar el diálogo para confirmar la terminación de la compra.

Fuente: Elaborado por el investigador.

	Tarea
Código: T2	**Código de historia:** H6
Nombre tarea: Diseñar el diálogo para el resumen de la compra	
Tipo de tarea: Desarrollo	**Puntos estimados:** 0.3
Tiempo: 1 día	
Programador responsable: Erick Gamboa	
Descripción: Definir el diálogo que mostrará un resumen de la orden de compra.	

Tabla 4.32: Actividad 2 - Historia 6 - Diseñar el diálogo para el resumen de la compra.

Fuente: Elaborado por el investigador.

	Tarea
Código: T3	**Código de historia:** H6
Nombre tarea: Diseñar el diálogo para confirmar datos de la compra	
Tipo de tarea: Desarrollo	**Puntos estimados:** 0.3
Tiempo: 1 día	
Programador responsable: Erick Gamboa	
Descripción: Definir el diálogo que confirmará que los datos de la orden de compra son correctos.	

Tabla 4.33: Actividad 3 - Historia 6 - Diseñar el diálogo para confirmar datos de la compra.

Fuente: Elaborado por el investigador.

	Tarea
Código: T4	**Código de historia:** H6
Nombre tarea: Guardar orden de compra	
Tipo de tarea: Desarrollo	**Puntos estimados:** 1.1
Tiempo: 3 días	
Programador responsable: Erick Gamboa	
Descripción: Se debe realizar la conexión con el sistema de compras para guardar los datos de la orden de compra.	

Tabla 4.34: Actividad 3 - Historia 5 - Guardar orden de compra.

Fuente: Elaborado por el investigador.

	Tarea
Código: T5	**Código de historia:** H6
Nombre tarea: Implementar diálogo para guardar la orden de compra	
Tipo de tarea: Desarrollo	**Puntos estimados:** 1.0
Tiempo: 3 días	
Programador responsable: Erick Gamboa	
Descripción: Implementar el diálogo que permitirá guardar la orden de compra.	

Tabla 4.35: Actividad 5 - Historia 6 - Implementar diálogo para guardar la orden de compra.

Fuente: Elaborado por el investigador.

Historia: Diálogo para visualizar la orden de compra

	Tarea
Código: T1	**Código de historia:** H7
Nombre tarea: Diseñar el diálogo para visualizar la orden de compra	
Tipo de tarea: Desarrollo	**Puntos estimados:** 0.3
Tiempo: 1 día	
Programador responsable: Erick Gamboa	
Descripción: Definir el diálogo que permitirá visualizar el detalle de la orden de compra.	

Tabla 4.36: Actividad 1 - Historia 7 - Diseñar el diálogo para visualizar la orden de compra.

Fuente: Elaborado por el investigador.

	Tarea
Código: T2	**Código de historia:** H7
Nombre tarea: Diseñar el diálogo para ingresar el número de orden	
Tipo de tarea: Desarrollo	**Puntos estimados:** 0.3
Tiempo: 1 día	
Programador responsable: Erick Gamboa	
Descripción: Definir el diálogo que permitirá ingresar el número de orden a consultar.	

Tabla 4.37: Actividad 2 - Historia 7 - Diseñar el diálogo para ingresar el número de orden.

Fuente: Elaborado por el investigador.

	Tarea
Código: T3	**Código de historia:** H7
Nombre tarea: Consultar orden de compra	
Tipo de tarea: Desarrollo	**Puntos estimados:** 1.2
Tiempo: 3 días	
Programador responsable: Erick Gamboa	
Descripción: Se debe realizar la conexión con el sistema de compras para consultar la orden de compra en base a número de orden.	

Tabla 4.38: Actividad 3 - Historia 7 - Consultar orden de compra.

Fuente: Elaborado por el investigador.

	Tarea
Código: T4	**Código de historia:** H7
Nombre tarea: Implementar diálogo para visualizar la orden de compra	
Tipo de tarea: Desarrollo	**Puntos estimados:** 1.2
Tiempo: 3 días	
Programador responsable: Erick Gamboa	
Descripción: Implementar el diálogo que permita visualizar el detalle de la orden de compra.	

Tabla 4.39: Actividad 4 - Historia 7 - Implementar diálogo para visualizar la orden de compra.

Fuente: Elaborado por el investigador.

Historia: Diálogo de fin

	Tarea
Código: T1	**Código de historia:** H8
Nombre tarea: Diseñar el diálogo de fin	
Tipo de tarea: Desarrollo	**Puntos estimados:** 0.4
Tiempo: 1 día	
Programador responsable: Erick Gamboa	
Descripción: Definir el diálogo que dará fin a la conversación con el chatbot.	

Tabla 4.40: Actividad 1 - Historia 8 - Diseñar el diálogo de fin.

Fuente: Elaborado por el investigador.

	Tarea
Código: T2	**Código de historia:** H8
Nombre tarea: Implementar diálogo de finalización	
Tipo de tarea: Desarrollo	**Puntos estimados:** 0.6
Tiempo: 4 días	
Programador responsable: Erick Gamboa	
Descripción: Implementar el diálogo de finalización tanto en el chatbot como en el procesador de lenguaje natural.	

Tabla 4.41: Actividad 2 - Historia 8 - Implementar diálogo de finalización.

Fuente: Elaborado por el investigador.

Historia: Entrenamiento del procesador del lenguaje natural

	Tarea
Código: T1	**Código de historia:** H9
Nombre tarea: Definir las frases de entrenamiento del procesador del lenguaje natural	
Tipo de tarea: Desarrollo	**Puntos estimados:** 1.4
Tiempo: 3 días	
Programador responsable: Erick Gamboa	
Descripción: Definir posibles frases que los usuarios utilizarían para comunicarse con el chatbot.	

Tabla 4.42: Actividad 1 - Historia 9 - Definir las frases de entrenamiento del procesador del lenguaje natural.

Fuente: Elaborado por el investigador.

	Tarea
Código: T2	**Código de historia:** H9
Nombre tarea: Entrenar el procesador de lenguaje natural	
Tipo de tarea: Desarrollo	**Puntos estimados:** 2.6
Tiempo: 4 días	
Programador responsable: Erick Gamboa	
Descripción: Ingresar y entrenar el procesador de lenguaje natural con las frases definidas anteriormente.	

Tabla 4.43: Actividad 2 - Historia 9 - Entrenar el procesador de lenguaje natural.

Fuente: Elaborado por el investigador.

Una vez realizada la definición de tareas se pudo realizar una valoración de cada historia de usuario y estimar el tiempo que se necesitará para que cada una sea terminada. Cada historia está definida en semanas de 5 días.

Para el desarrollo del prototipo, se realizó una estimación del esfuerzo de cada historia de usuario, las cuales se agruparon en tres módulos.

Módulo de Presentación

Nº	Historia de Usuario	Tiempo estimado		
		Semanas	Días	Horas
1	Diálogo de inicio	1	5	20
2	Diálogo de saludo	1	5	20
8	Diálogo de fin	1	5	20
Tiempo estimado total		3	15	60

Tabla 4.44: Estimación del módulo de presentación.

Fuente: Elaborado por el investigador a partir de historias de usuario y tareas.

Módulo de Acceso

Nº	Historia de Usuario	Tiempo estimado		
		Semanas	Días	Horas
3	Diálogo de ingreso al sistema	1	5	20
Tiempo estimado total		1	5	20

Tabla 4.45: Estimación del módulo de acceso.

Fuente: Elaborado por el investigador a partir de historias de usuario y tareas.

Módulo de Integración

Nº	Historia de Usuario	Tiempo estimado		
		Semanas	Días	Horas
4	Diálogo para consultar productos	2	8	32
5	Diálogo para agregar productos a la orden de compra	2	8	32
6	Diálogo para guardar la orden de compra	2	9	36
7	Diálogo para visualizar la orden de compra	2	8	32
9	Entrenamiento del procesador del lenguaje natural	2	7	28
Tiempo estimado total		10	40	160

Tabla 4.46: Estimación del módulo de integración.

Fuente: Elaborado por el investigador a partir de historias de usuario y tareas.

Una vez que se han establecido las historias de usuario y tareas se realiza la planificación de entregables, como resultado se establece el siguiente plan de entregas.

Módulo	Historia de Usuario	Tiempo estimado		
		Semanas	Días	Horas
Presentación	Diálogo de inicio	1	5	20
	Diálogo de saludo	1	5	20
	Diálogo de fin	1	5	20
Acceso	Diálogo de ingreso al sistema	1	5	20
Integración	Diálogo para consultar productos	2	8	32
	Diálogo para agregar productos a la orden de compra	2	8	32
	Diálogo para guardar la orden de compra	2	9	36
	Diálogo para visualizar la orden de compra	2	8	32
	Entrenamiento del procesador del lenguaje natural	2	7	28
Tiempo estimado total		14	60	240

Tabla 4.47: Resumen estimación de módulos.

Fuente: Elaborado por el investigador a partir de estimaciones de cada módulo.

Módulo	Historia de Usuario	Iteración Asignada							Entrega Asignada						
		1	2	3	4	5	6	7	1	2	3	4	5	6	7
Presentación	Diálogo de inicio	X							X						
	Diálogo de saludo	X							X						
	Diálogo de fin	X							X						
Acceso	Diálogo de ingreso al sistema		X							X					
Integración	Diálogo para consultar productos			X							X				
	Diálogo para agregar productos a la orden de compra				X							X			
	Diálogo para guardar la orden de compra					X							X		
	Diálogo para visualizar la orden de compra						X							X	
	Entrenamiento del procesador del lenguaje natural							X							X

Tabla 4.48: Plan de entrega e iteraciones.

Fuente: Elaborado por el investigador a partir de estimaciones de cada módulo.

4.7.3.2. Fase: Diseño

Para realizar el diseño del software la metodología XP recomienda hacer uso de las tarjetas CRC (Clase-Responsabilidad-Colaboración). Las tarjetas CRC permiten realizar un diseño orientado a objetos, ya que son una representación directa de cada funcionalidad.

Por cada historia de usuario se debe realizar una tarjeta CRC, la cual contiene: una clase que puede ser una persona, concepto, evento pantalla o reporte; las responsabilidades que son las acciones que son realizadas por los atributos y métodos; y los colaboradores que son otras clases que ayudan a llevar a cabo las responsabilidades.

Diálogo de inicio

Diálogo de inicio	
Responsabilidad	**Colaboradores**
Presentar mensaje de texto	Procesador de lenguaje natural. Métodos de integración con el PLN.
Observaciones:	

Tabla 4.49: Tarjeta CRC - Diálogo de inicio.

Fuente: Elaborado por el investigador a partir de historias de usuario.

Diálogo de saludo

Diálogo de saludo	
Responsabilidad	**Colaboradores**
Presentar mensaje de texto	Procesador de lenguaje natural. Métodos de integración con el PLN.
Observaciones:	

Tabla 4.50: Tarjeta CRC - Diálogo de saludo.

Fuente: Elaborado por el investigador a partir de historias de usuario.

Diálogo de ingreso al sistema

Diálogo de ingreso al sistema	
Responsabilidad	**Colaboradores**
Validar usuario y contraseña	Métodos de integración con el sistema de compras
Observaciones:	

Tabla 4.51: Tarjeta CRC - Diálogo de ingreso al sistema.

Fuente: Elaborado por el investigador a partir de historias de usuario.

Diálogo para consultar productos

Diálogo para consultar productos	
Responsabilidad	**Colaboradores**
Obtener parámetros Mostrar productos	Métodos de integración con el sistema de compras
Observaciones:	

Tabla 4.52: Tarjeta CRC - Diálogo para consultar productos.

Fuente: Elaborado por el investigador a partir de historias de usuario.

Diálogo para agregar productos a la orden de compra

Diálogo para agregar productos a la orden de compra	
Responsabilidad	**Colaboradores**
Obtener parámetros Verificar stock disponible	Métodos de integración con el sistema de compras
Observaciones:	

Tabla 4.53: Tarjeta CRC - Diálogo para agregar productos a la orden de compra.

Fuente: Elaborado por el investigador a partir de historias de usuario.

Diálogo para guardar la orden de compra

Diálogo para guardar la orden de compra	
Responsabilidad	**Colaboradores**
Obtener parámetros Guardar orden de compra	Métodos de integración con el sistema de compras
Observaciones:	

Tabla 4.54: Tarjeta CRC - Diálogo para guardar la orden de compra.

Fuente: Elaborado por el investigador a partir de historias de usuario.

Diálogo para visualizar la orden de compra

Diálogo para visualizar la orden de compra	
Responsabilidad	**Colaboradores**
Obtener parámetros Mostrar orden de compra	Métodos de integración con el sistema de compras
Observaciones:	

Tabla 4.55: Tarjeta CRC - Diálogo para visualizar la orden de compra.

Fuente: Elaborado por el investigador a partir de historias de usuario.

Diálogo de fin

Diálogo de fin	
Responsabilidad	**Colaboradores**
Presentar mensaje de texto	Procesador de lenguaje natural. Métodos de integración con el PLN.
Observaciones:	

Tabla 4.56: Tarjeta CRC - Diálogo de fin.

Fuente: Elaborado por el investigador a partir de historias de usuario.

Entrenamiento del procesador del lenguaje natural

Procesador del lenguaje natural	
Responsabilidad	**Colaboradores**
Obtener texto en lenguaje natural Retornar intenciones encontradas durante el procesamiento del lenguaje	Procesador de lenguaje natural.
Observaciones:	

Tabla 4.57: Tarjeta CRC - Procesador del lenguaje natural.

Fuente: Elaborado por el investigador a partir de historias de usuario.

Además, en la fase de diseño se deben generar los flujos de comunicación o conversación, los cuales son definidos a partir de las historias de usuario. Sólo las historias necesarias tendrán un flujo de conversación.

Diálogo de inicio

Flujo de comunicación				
Tarea	**Objetivo**	**Motivación del usuario**	**Pasos**	**Previsiones**
Presentar mensaje de bienvenida	Informar al usuario que ha iniciado la conversación	Iniciar una conversación con el bot	- Iniciar la conversación - Presentar el mensaje	El servicio puede estar fuera de línea

Tabla 4.58: Flujo de conversación - Diálogo de inicio.

Fuente: Elaborado por el investigador a partir de historias de usuario.

Diálogo de saludo

Flujo de comunicación				
Tarea	**Objetivo**	**Motivación del usuario**	**Pasos**	**Previsiones**
Presentar mensaje de saludo	Saludar al usuario y presentar las funcionalidades del bot	Conversar con el bot	- Procesar el texto de saludo ingresado por el usuario - Presentar un saludo - Presentar las funcionalidades del bot	El usuario puede usar palabras o frases que no son consideradas como un saludo

Tabla 4.59: Flujo de conversación - Diálogo de saludo.

Fuente: Elaborado por el investigador a partir de historias de usuario.

Diálogo de ingreso al sistema

Flujo de comunicación				
Tarea	**Objetivo**	**Motivación del usuario**	**Pasos**	**Previsiones**
Ingresar al sistema	Permitir la autenticación del usuario	Poder utilizar el sistema	- Presentar el formulario de ingreso - Ingresar información de usuario - Validar la información - Presentar mensaje dependiendo si se pudo o no autenticar el usuario	El usuario puede ingresar alguna frase fuera del ámbito de autenticación

Tabla 4.60: Flujo de conversación - Diálogo de ingreso al sistema.

Fuente: Elaborado por el investigador a partir de historias de usuario.

Diálogo para consultar productos

Flujo de comunicación				
Tarea	**Objetivo**	**Motivación del usuario**	**Pasos**	**Previsiones**
Consultar productos	Mostrar los productos existentes en el sistema de compras	Ver los productos existentes	- Presentar filtros para consultar productos - Seleccionar el filtro - Consultar productos - Mostrar productos	El usuario puede ingresar alguna frase fuera del ámbito de consulta

Tabla 4.61: Flujo de conversación - Diálogo para consultar productos.

Fuente: Elaborado por el investigador a partir de historias de usuario.

Diálogo para agregar productos a la orden de compra

Flujo de comunicación				
Tarea	**Objetivo**	**Motivación del usuario**	**Pasos**	**Previsiones**
Agregar productos a la orden de compra	Seleccionar un producto y agregarlo a la orden de compra	Realizar la compra de un producto	- Confirmar la selección del producto - Preguntar la cantidad del producto seleccionado - Ingresar la cantidad - Verificar el stock del producto seleccionado - Indicar si se añadió o no el producto	- El usuario puede ingresar alguna frase fuera del ámbito de compra - No existe la cantidad solicitada

Tabla 4.62: Flujo de conversación - Diálogo para agregar productos a la orden de compra.

Fuente: Elaborado por el investigador a partir de historias de usuario.

Diálogo para guardar la orden de compra

Flujo de comunicación				
Tarea	**Objetivo**	**Motivación del usuario**	**Pasos**	**Previsiones**
Terminar de realizar la compra	Guardar la orden de compra	Terminar con el proceso de compra	- Confirmar la orden de compra - Guardar los productos seleccionados - Indicar el estado del proceso de guardado - Visualizar un resumen de productos comprados	El usuario puede ingresar alguna frase fuera del ámbito de compra

Tabla 4.63: Flujo de conversación - Diálogo para guardar la orden de compra.

Fuente: Elaborado por el investigador a partir de historias de usuario.

Diálogo para visualizar la orden de compra

Flujo de comunicación				
Tarea	**Objetivo**	**Motivación del usuario**	**Pasos**	**Previsiones**
Mostrar los productos de una orden de compra	Visualizar un resumen de una compra	Ver los productos que compró	- Solicitar el número de orden - Ingresar el número de orden - Verificar si la orden existe - Visualizar los productos	El usuario puede ingresar alguna frase fuera del ámbito de visualización

Tabla 4.64: Flujo de conversación - Diálogo para visualizar la orden de compra.

Fuente: Elaborado por el investigador a partir de historias de usuario.

Diálogo de fin

Flujo de comunicación				
Tarea	**Objetivo**	**Motivación del usuario**	**Pasos**	**Previsiones**
Presentar mensaje de fin	Informar al usuario que ha terminado la conversación	Terminar una conversación con el bot	- Preguntar si el bot puede ayudar en algo más - Terminar la conversación - Presentar un mensaje	- El servicio puede estar fuera de línea - El usuario se salga del contexto que el bot le pregunta

Tabla 4.65: Flujo de conversación - Diálogo de fin.

Fuente: Elaborado por el investigador a partir de historias de usuario.

4.7.3.3. Fase: Codificación

Para una mejor comprensión de los diálogos en el Anexo A se presentan los diagramas de flujo del proceso de cada diálogo. Además, como parte del proceso de codificación se realizaron prototipos de las interfaces de cada diálogo, las cuales se encuentran en el Anexo B.

Para el proceso de codificación se utilizó C# como lenguaje de desarrollo, para el desarrollo de cada diálogo se utilizó el patrón MVC. Además, para la integración tanto del procesador como del sistema de compras se realizaron los modelos y los controladores.

En el Anexo C se presentan varias imágenes de la codificación del chatbot.

4.7.3.4. Fase: Pruebas

Las pruebas son la parte de la metodología que permite comprobar el funcionamiento de cada funcionalidad del sistema definida en las historias de usuario. Además, permite verificar si lo que se implementó es lo que se deseaba. En este proceso se ejecutan las pruebas realizadas en cada historia de usuario.

	Prueba de aceptación
Código: P1	**Código de historia:** H1, Diálogo de inicio
Descripción: El bot presentará un texto a manera de saludo, en el cual se indica que es un chatbot. Si el servicio no está disponible el presentará un texto de saluda e indicará que no se encuentra disponible.	
Condición de ejecución: El usuario deberá haber iniciado el chatbot.	
Entrada: - Abrir el chat con el bot	
Resultado: Texto a manera de mensaje.	
Evaluación de prueba: La prueba se realizó de manera satisfactoria.	

Tabla 4.66: Prueba 1 - Diálogo de inicio.

Fuente: Elaborado por el investigador.

	Prueba de aceptación
Código: P2	**Código de historia:** H2, Diálogo de saludo
Descripción: El bot reconocerá las palabras de saludo, responderá con un saludo y presentará las funcionalidades que puede realizar.	
Condición de ejecución: El usuario deberá haber ingresado un texto a modo de saludo.	
Entrada: - Ingresar un mensaje de saludo - Presionar la tecla enter	
Resultado: Texto a manera de mensaje con el saludo y las funcionalidades.	
Evaluación de prueba: La prueba se realizó de manera satisfactoria.	

Tabla 4.67: Prueba 2 - Diálogo de saludo.

Fuente: Elaborado por el investigador.

	Prueba de aceptación
Código: P3	**Código de historia:** H3, Diálogo de ingreso al sistema
Descripción: El bot permitirá la autenticación del usuario para lo cual mostrará un formulario de ingreso.	
Condición de ejecución: El usuario deberá estar registrado en el sistema.	
Entrada: - Ingresar un usuario y contraseña - Presionar la tecla enter	
Resultado: Texto indicando si el usuario pudo o no autenticarse en el sistema.	
Evaluación de prueba: La prueba se realizó de manera satisfactoria.	

Tabla 4.68: Prueba 3 - Diálogo de ingreso al sistema.

Fuente: Elaborado por el investigador.

	Prueba de aceptación
Código: P4	**Código de historia:** H4, Diálogo para consultar productos
Descripción: El bot permitirá consultar los productos que se encuentren en el sistema, para lo cual presentará varios filtros.	
Condición de ejecución: El usuario deberá elegir un filtro, dependiendo de eso el sistema mostrará los productos.	
Entrada: - Clic en el filtro deseado	
Resultado: Diálogo con los productos.	
Evaluación de prueba: La prueba se realizó de manera satisfactoria.	

Tabla 4.69: Prueba 4 - Diálogo para consultar productos.

Fuente: Elaborado por el investigador.

	Prueba de aceptación
Código: P5	**Código de historia:** H5, Diálogo para agregar productos a la orden de compra
Descripción: El bot permitirá agregar un determinado producto a la orden de compra.	
Condición de ejecución: Verificar si el usuario quiere añadir un producto seleccionado e ingresar la cantidad. Además, el sistema comprobará la existencia en el stock.	
Entrada: - Clic sobre el producto deseado - Ingresar la cantidad	
Resultado: Texto indicando si se pudo agregar el producto a la orden de compra.	
Evaluación de prueba: La prueba se realizó de manera satisfactoria.	

Tabla 4.70: Prueba 5 - Diálogo para agregar productos a la orden de compra.

Fuente: Elaborado por el investigador.

	Prueba de aceptación
Código: P6	**Código de historia:** H6, Diálogo para guardar la orden de compra
Descripción: El bot permitirá terminar de realizar el pedido y guardar la orden de compra.	
Condición de ejecución: Preguntar si el usuario quiere finalizar la compra.	
Entrada: - Clic sobre la opción de si o no	
Resultado: Texto indicando que se guardó la orden de compra.	
Evaluación de prueba: La prueba se realizó de manera satisfactoria.	

Tabla 4.71: Prueba 6 - Diálogo para guardar la orden de compra.

Fuente: Elaborado por el investigador.

	Prueba de aceptación
Código: P7	**Código de historia:** H7, Diálogo para visualizar la orden de compra
Descripción: El bot permitirá visualizar la orden de compra.	
Condición de ejecución: El usuario deberá ingresar el número de orden.	
Entrada: - Ingresar número de orden - Presionar tecla enter	
Resultado: Diálogo mostrando detalle de la orden de compra.	
Evaluación de prueba: La prueba se realizó de manera satisfactoria.	

Tabla 4.72: Prueba 7 - Diálogo para visualizar la orden de compra.

Fuente: Elaborado por el investigador.

	Prueba de aceptación
Código: P8	**Código de historia:** H8, Diálogo de fin
Descripción: El bot presentará un texto a manera de despedida.	
Condición de ejecución: El usuario deberá haber terminado la conversación con el chatbot.	
Entrada: - Mensaje indicando que ya no quiere realizar algo más.	
Resultado: Texto a manera de mensaje.	
Evaluación de prueba: La prueba se realizó de manera satisfactoria.	

Tabla 4.73: Prueba 8 - Diálogo de fin.

Fuente: Elaborado por el investigador.

	Prueba de aceptación
Código: P9	**Código de historia:** H9, Entrenamiento del procesador de lenguaje natural
Descripción: El procesador del lenguaje natural será entrenado con frases de prueba.	
Condición de ejecución: El programador responsable deberá definir las frases que se adapten a las funcionalidades del sistema.	
Entrada: - Frases comunes definidas.	
Resultado: Correcto funcionamiento del bot.	
Evaluación de prueba: La prueba se realizó de manera satisfactoria.	

Tabla 4.74: Prueba 9 - Entrenamiento del procesador de lenguaje natural.

Fuente: Elaborado por el investigador.

4.7.4. Implementación del chatbot

4.7.4.1. Implementación en entorno local

El despliegue del chatbot se realizó en un entorno local, para lo cual se realizó una implementación en un entorno de pruebas local. En el entorno de pruebas se realizó la validación de las funcionalidades.

Para la implementación del bot en un entorno local primero se debe habilitar el IIS(Internet Information Services), caso de no tenerlo habilitado. Para lo cual se debe dirigir al panel de control, en la pantalla de programas existe una opción que se llama “Activar o desactivar las características de Windows”, como se muestra en la figura 4.18, clic sobre esta opción, aparecerá una interfaz como se muestra en la figura 4.19.

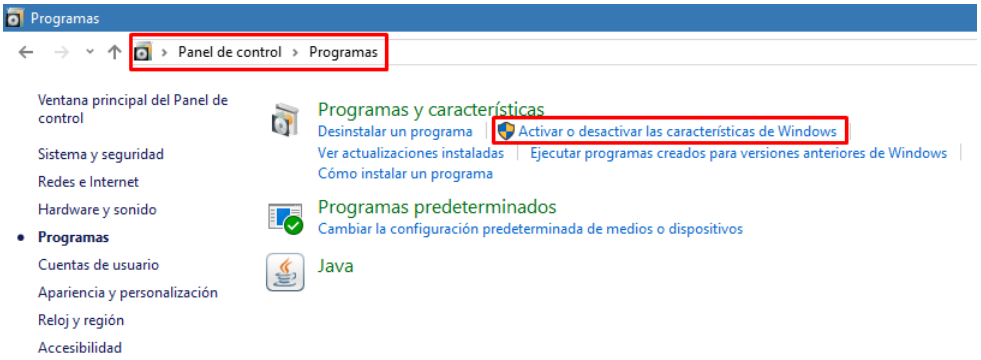

Figura 4.18: Panel de control.
Fuente: Elaborado por el investigador.

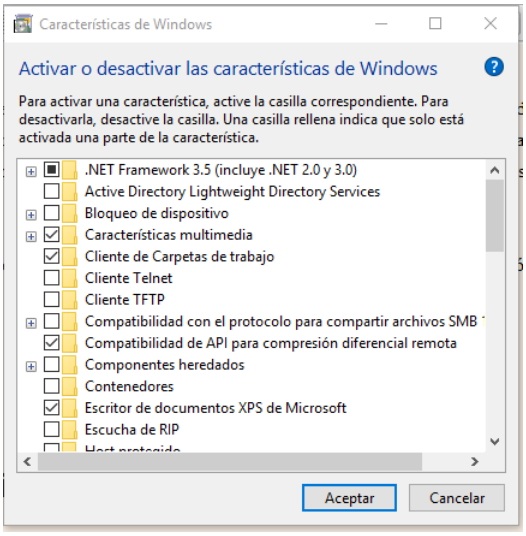

Figura 4.19: Activar o desactivar características de Windows.
Fuente: Elaborado por el investigador.

Una vez que se encuentre en esta interfaz, se debe habilitar el IIS con la configuración por defecto, para lo cual se debe seleccionar la opción Internet Information Services, como se muestra en la figura 4.20, clic en aceptar y se debe esperar hasta que se habiliten los servicios.

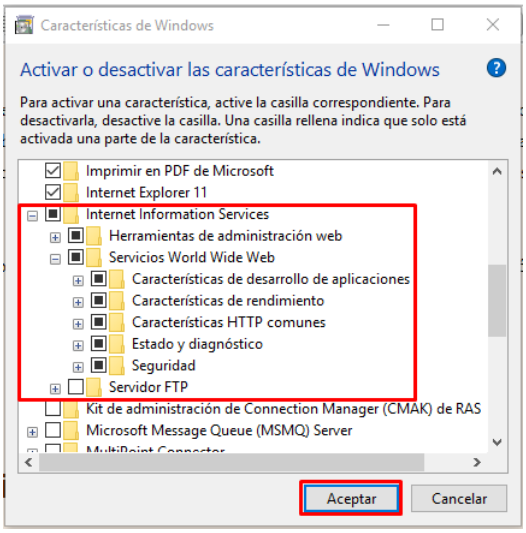

Figura 4.20: Habilitar Internet Information Services (IIS).
Fuente: Elaborado por el investigador.

Una vez que se haya habilitado el IIS, se debe publicar la aplicación en un directorio local, para lo cual se debe dar clic derecho sobre la aplicación y clic en la opción "Publicar...". A continuación, se debe crear un perfil para publicar como se muestra en la figura 4.21 y clic en publicar.

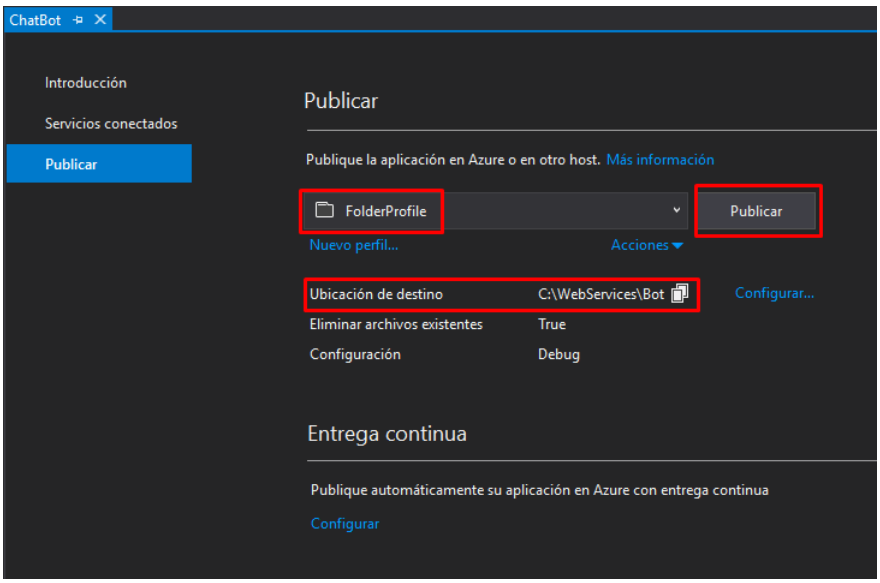

Figura 4.21: Publicar aplicación en directorio local.
Fuente: Elaborado por el investigador.

Cuando ya se encuentre publicada la aplicación, se debe configurar la misma en el IIS, para esto abrir el Administrador de Internet Information Services, clic derecho sobre sitios y clic en la opción “Agregar sitio web...”, en la interfaz que aparece se debe configurar el sitio web que contendrá la aplicación, como se muestra en la figura 4.22.

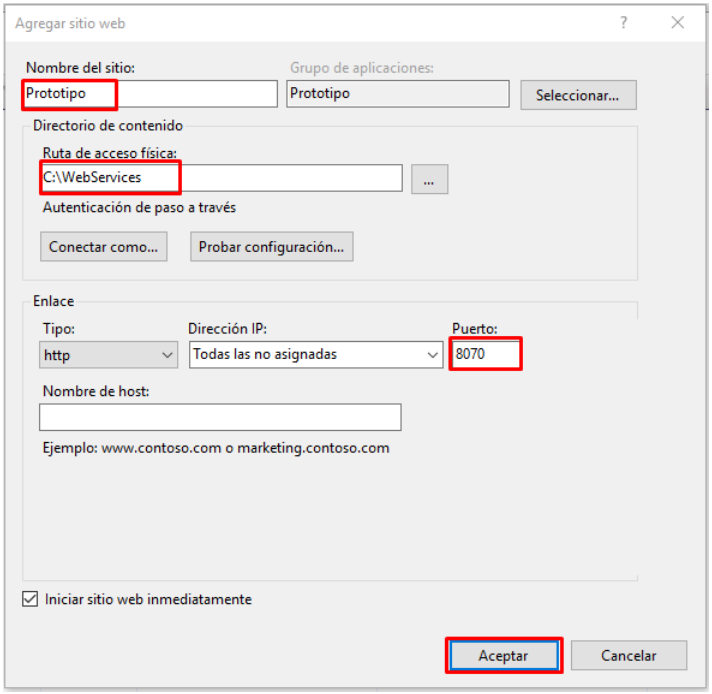

Figura 4.22: Configuración nuevo sitio web en IIS.
Fuente: Elaborado por el investigador.

Por defecto reconocerá todas las carpetas que están dentro del directorio que se especificó para el sitio web, en la carpeta que se publicó la aplicación clic derecho y clic en la opción “Convertir en aplicación”, como se muestra en la figura 4.23.

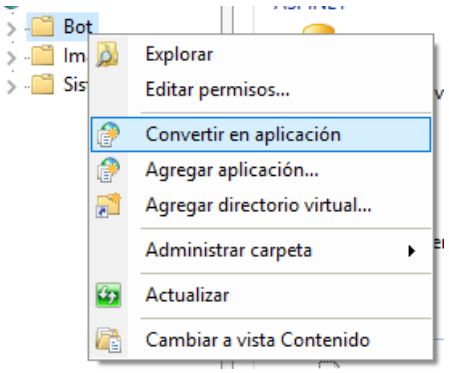

Figura 4.23: Convertir en aplicación.
Fuente: Elaborado por el investigador.

Una vez que se haya publicado la aplicación estará lista para hacer pruebas desde el emulador de bot framework, para lo cual se debe abrir el emulador y escribir la dirección url donde se encuentra publicado el chatbot y clic en connect, como se muestra en la figura 4.24. Listo, ya se tendrá implementado el chatbot en un entorno local, como se muestra en la figura 4.25.

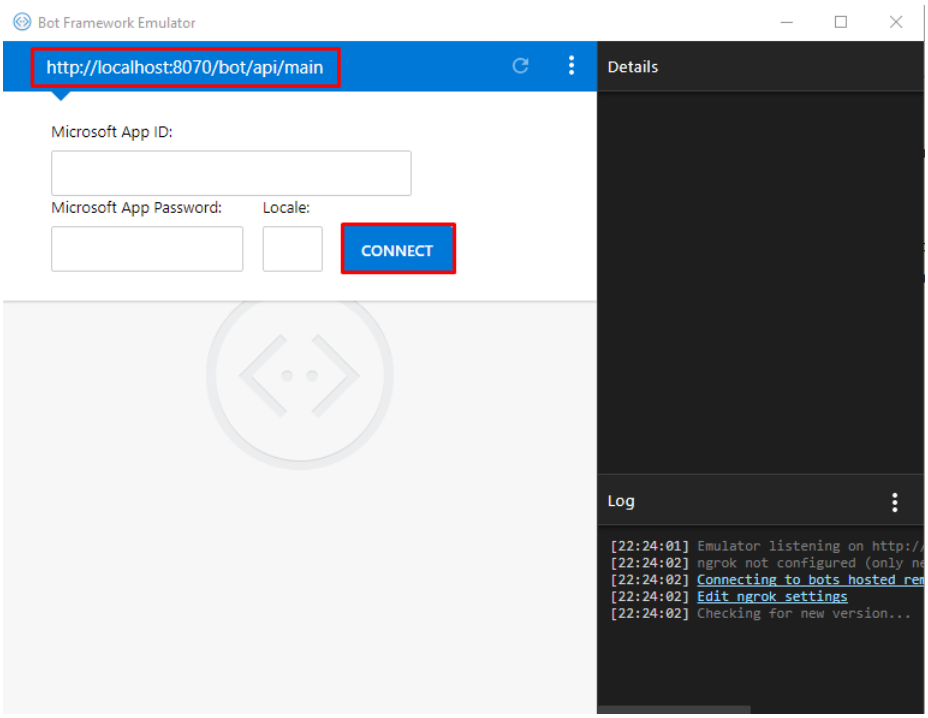

Figura 4.24: Emulador de Bot Framework.
Fuente: Elaborado por el investigador.

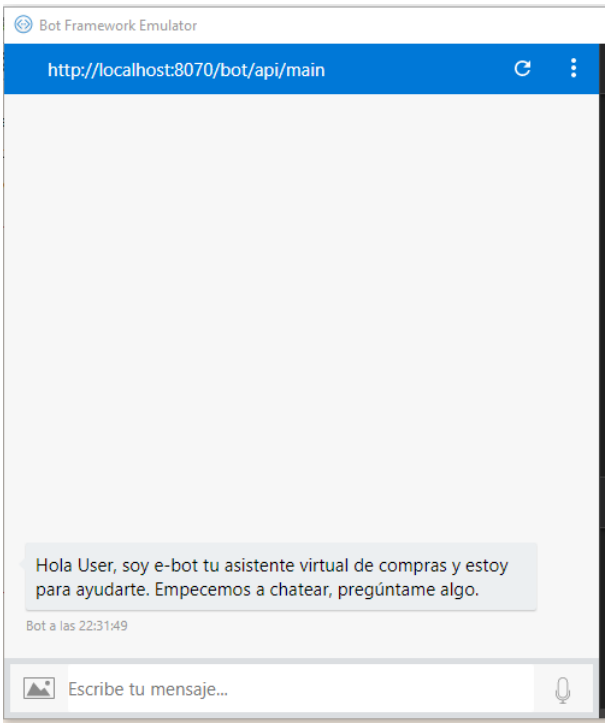

Figura 4.25: Chatbot funcionando en entorno local.
Fuente: Elaborado por el investigador.

4.7.4.2. Implementación en la web

Si el bot se va a desplegar en la web se debe tener la aplicación publicada en algún servicio de hospedaje o tener un servidor de aplicaciones configurado con salida hacia la web. Para utilizar el chatbot en la web se debe registrarlo mediante el portal de Microsoft Bot Framework en la dirección https://dev.botframework.com/.

En este portal se debe ingresar usando una cuenta de Microsoft, una vez logueado se debe ir a la dirección https://dev.botframework.com/bots/new, aquí se debe registrar el chatbot como se muestra en la figura 4.26. Para publicar el chatbot se debe ingresar la url donde está publicado el chatbot, se debe aceptar los términos del servicio y clic en register, como se muestra en la figura 4.27.

Además, solicitará un id de aplicación de microsoft para crear este id clic en create microsoft app id and password, se abrirá el portal de azure ahí se debe crear una aplicación y de esa aplicación se debe generar un password.

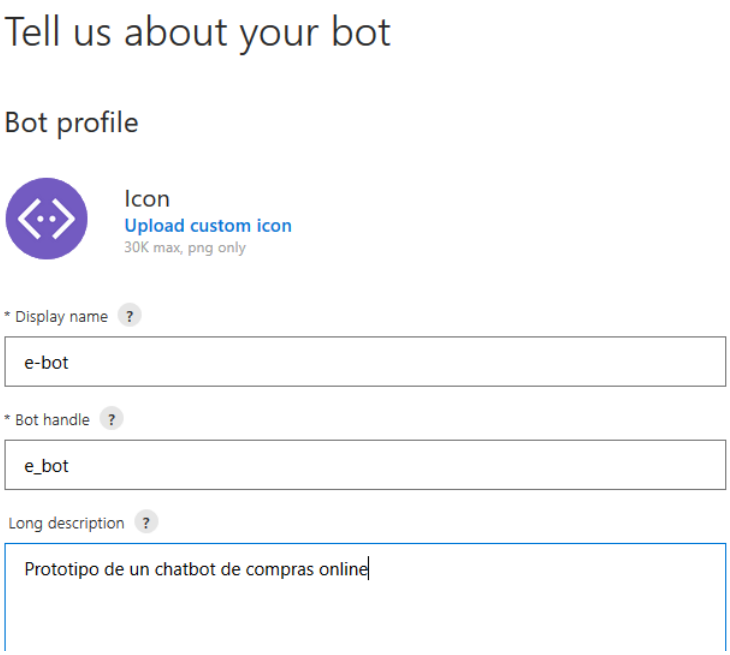

Figura 4.26: Registro chatbot.
Fuente: Elaborado por el investigador.

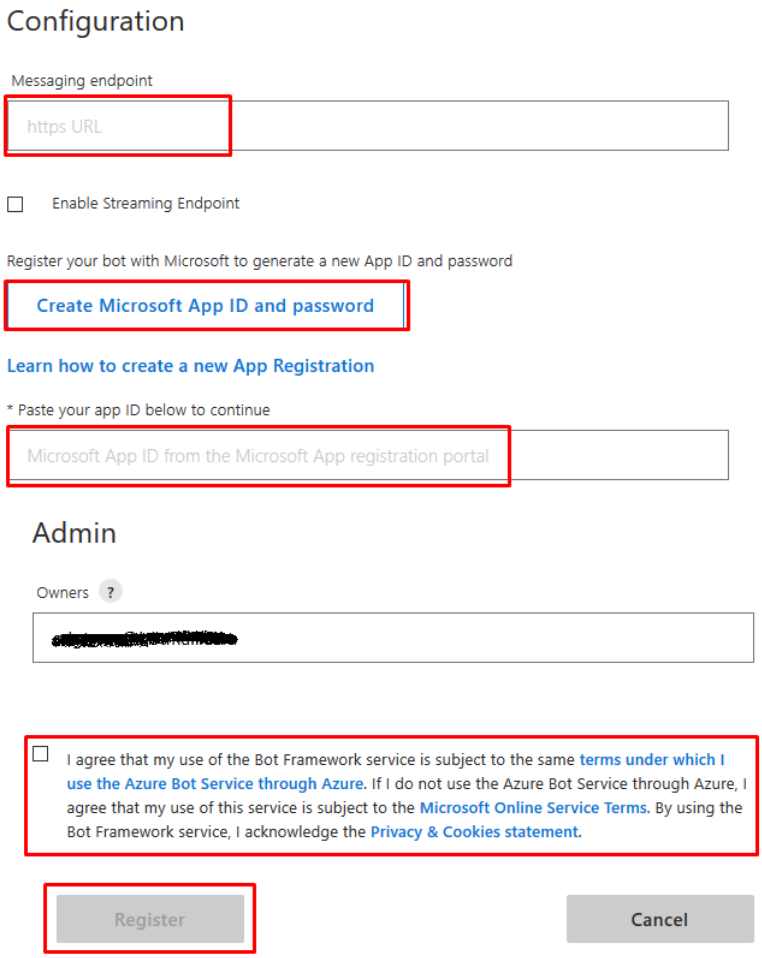

Figura 4.27: Datos para registrar chatbot.
Fuente: Elaborado por el investigador.

Una vez se haya publicado el bot se tendrá la opción de probarlo en un chat embebido en la página, como se muestra en la figura 4.28. Además, se puede obtener el código para embeberlo en una página web, como se muestra en la figura 4.29.

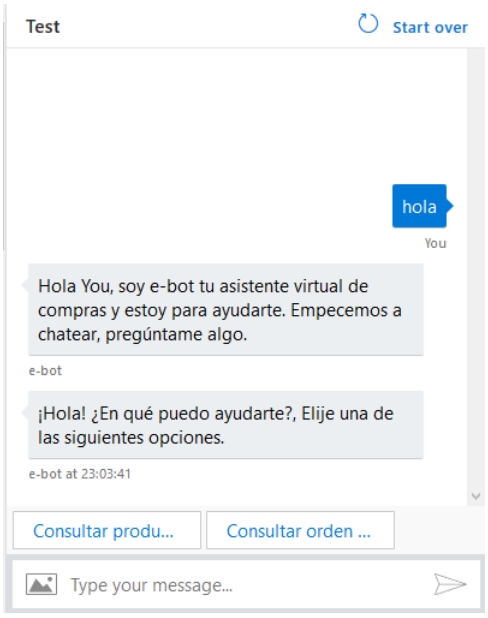

Figura 4.28: Chatbot en la web.
Fuente: Elaborado por el investigador.

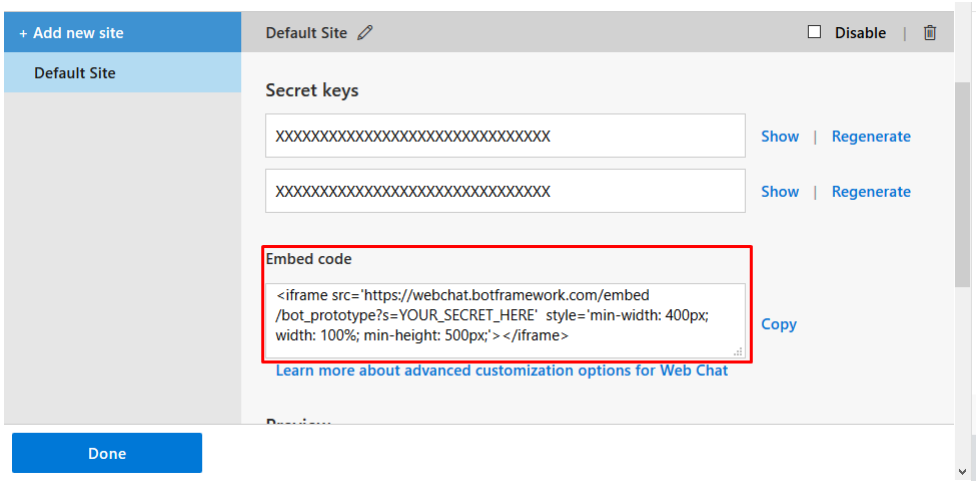

Figura 4.29: Código para embeber chatbot en una página web.
Fuente: Elaborado por el investigador.

En la figura 4.30 se muestra el chatbot embebido en una página web.

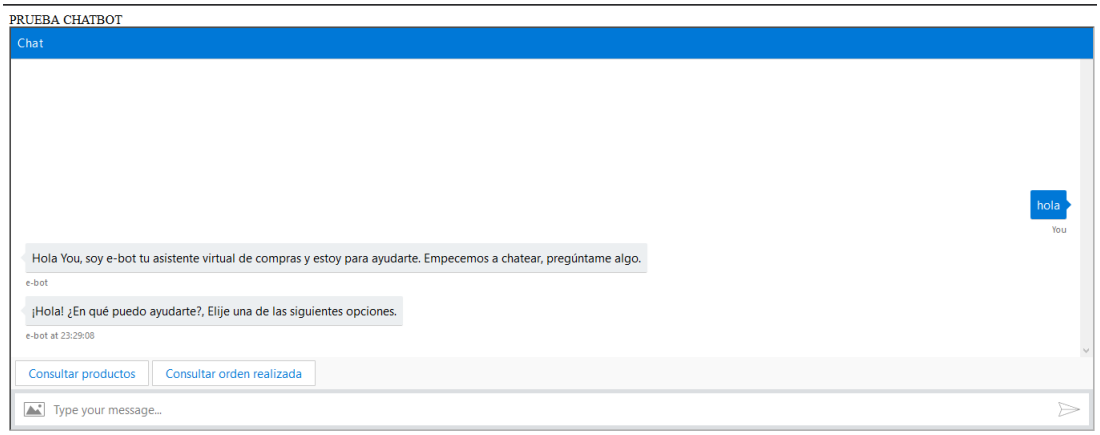

Figura 4.30: Chatbot embebido en una página web.
Fuente: Elaborado por el investigador.

4.7.4.3. Pruebas de funcionamiento

Para garantizar el funcionamiento del chatbot al momento de la implementación tanto en local como en la web se realizaron pruebas de funcionamiento. Con estas pruebas se pudieron identificar errores, cada prueba se la aplicó en base a cada flujo de comunicación, verificando que cada funcionalidad cumpla con su objetivo. Los resultados obtenidos se muestran en la tabla 4.75.

Nº	Flujo de comunicación	Evaluación de la prueba
1	Diálogo de inicio	Prueba satisfactoria
2	Diálogo de saludo	Prueba satisfactoria
3	Diálogo de fin	Prueba satisfactoria
4	Diálogo de ingreso al sistema	Prueba satisfactoria
5	Diálogo para consultar productos	Prueba satisfactoria
6	Diálogo para agregar productos a la orden de compra	Prueba satisfactoria
7	Diálogo para guardar la orden de compra	Prueba satisfactoria
8	Diálogo para visualizar la orden de compra	Prueba satisfactoria

Tabla 4.75: Resultados pruebas de funcionamiento.

Fuente: Elaborado por el investigador a partir del funcionamiento del chatbot.

CAPÍTULO 5

Conclusiones y Recomendaciones

5.1. Conclusiones

- El desarrollo de chatbots o agentes conversacionales no está establecido en un marco de trabajo, por lo cual existen varias técnicas que apoyan la creación de este tipo de aplicaciones. La creación de flujos de conversación es una técnica que, a pesar de no ser muy conocida, apoya a la definición de tareas en un chatbot permitiendo la delimitación de las funcionalidades de un agente conversacional.
- Los chatbots no deben ser tomados como una interfaz de usuario cualquiera, ya que por el ámbito en el cual se encuentran trabajando, van a tener limitaciones en comparación con otro tipo de interfaces.
- Se utilizó el framework de desarrollo "Bot Framework" el cual permitió la combinación varias tecnologías, logrando una independencia al momento del desarrollo. Este beneficio se suscita debido a que este framework está basado en un lenguaje de programación y no en una plataforma en concreto.
- El hecho de implementar nuevas interfaces de usuario más amigables mejora la experiencia de usuario. Los chatbots simplifican la realización de una tarea, esto lo hacen a través de tareas guiadas.

5.2. Recomendaciones

- Se recomienda tener en cuenta el tema de seguridad antes de implementar una tarea en un chatbot, para ello es recomendable no implementar mecanismos tradicionales de autenticación.
- Se recomienda considerar el espacio de trabajo de un chatbot antes de implementar una funcionalidad en específico.

- Al momento de implementar un chatbot en una determinada plataforma o canal se debe analizar a profundidad las posibles limitaciones que estas posean.
- Es necesario analizar los flujos de trabajo de un determinado sistema antes replicar las funcionalidades en un chatbot, ya que no todos los procesos pueden ser implementados.
- Se debe considerar que no todo lo que se desarrolla y funciona en pruebas va a funcionar en un ambiente de producción, esto debido a que en medianos plazos la tecnología es cambiante.

Bibliografia

[1] V. Hristidis, "Chatbot technologies and challenges," in *Proc. First Int. Conf. Artificial Intelligence for Industries (AI4I)*, p. 126, Sept. 2018.

[2] J. C. Cobos Torres, "Integración de un chatbot como habilidad de un robot social con gestor de diálogos," mathesis, Universidad Carlos III de Madrid, Oct. 2013.

[3] K. L. Hurtado Moína and J. I. Zúñiga Loaiza, "Desarrollo de un asistente virtual web para la epn y un asistente dirigido por voz en los kioscos digitales de la dgip," Master's thesis, Escuela Politécnica Nacional, Jan. 2019.

[4] A. Augello, M. Gentile, L. Weideveld, and F. Dignum, "A model of a social chatbot," in *Intelligent Interactive Multimedia Systems and Services 2016*, Springer, Jan. 2016.

[5] A. Bozzon, "Enterprise crowd computing for human aided chatbots," in *Proceedings of the 1st International Workshop on Software Engineering for Cognitive Services*, SE4COG '18, (New York, NY, USA), pp. 29–30, ACM, 2018.

[6] A. Folstad, C. B. Nordheim, and C. A. Bjorkli, "What makes users trust a chatbot for customer service? an exploratory interview study," *Internet Science*, Jan. 2018.

[7] H. N. Io and C. B. Lee, "Chatbots and conversational agents: A bibliometric analysis," in *Proc. IEEE Int. Conf. Industrial Engineering and Engineering Management (IEEM)*, pp. 215–219, Dec. 2017.

[8] A. M. Rahman, A. A. Mamun, and A. Islam, "Programming challenges of chatbot: Current and future prospective," in *Proc. IEEE Region 10 Humanitarian Technology Conf. (R10-HTC)*, pp. 75–78, Dec. 2017.

[9] J. Medina, E. M. Eisman, and J. L. Castro, "Asistentes virtuales en plataformas 3.0," *Revista Iberoamericana de Informática Educativa*, no. 18, pp. 41–49, 2013.

[10] O. H. Zarabia Zuñiga, "Implementación de un chatbot con botframework: caso de estudio, servicios a clientes del área de fianzas de seguros equinoccial.," Master's thesis, Escuela Politécnica Nacional, Aug. 2018.

[11] J. M. Rodríguez, H. Merlino, and E. Fernández, "Comportamiento adaptable de chatbots dependiente del contexto," *Revista Latinoamericana de Ingeniería de Software*, vol. 2, no. 2, pp. 115–136, 2014.

[12] M. Azure, *Guía para desarrolladores, Azure Bot Service.* Microsoft, 2018.

[13] Microsoft, *Material Técnico de Diplomado de Bots.* Microsoft, 2018 ed., Jan. 2018.

[14] A. Cortez Vásquez, J. Pariona Quispe, and H. Vega Huerta, "Procesamiento de lenguaje natural," *Revista de Ingeniería de Sistemas e Informática*, vol. 6, no. 2, pp. 45–54, 2009.

[15] C. P. Caicedo Barreno, "Prototipo enfocado al aprendizaje de lógica de programación en niños de edades comprendidas entre 4 a 10 años," mathesis, Universidad Técnica de Ambato, June 2018.

[16] V. Fernández Alarcón, *Desarrollo de sistemas de información: una metodología basada en el modelado*, vol. 120 of *Aula Politècnica.* Universitat Politècnica de Catalunya, primera ed., June 2006.

[17] A. F. Toledo Cambizaca, "Desarrollo de un chatbot que ayude a responder a preguntas frecuentes referentes a becas en la universidad técnica particular de loja," B.S. thesis, Universidad Técnica Particular de Loja, Feb. 2018.

[18] I. Sommerville, *Ingeniería de software (Spanish Edition).* Pearson Educación (México), novena ed., 2011.

[19] M. Trigás Gallego, "Metodología scrum," *Repositorio institucional de la Universitat Oberta de Catalunya*, 2012.

[20] Inteco, "Ingeniería del software: Metodologías y ciclos de vida," *Laboratorio Nacional de Calidad Del Software*, vol. 83, Mar. 2009.

[21] D. I. Paucar Quile, "Sistema informático para emprendimientos en la facultad de ciencia e ingeniería en alimentos de la universidad técnica de ambato y comunidad," mathesis, Universidad Técnica de Ambato, Apr. 2019.

[22] A. D. Moyano González, "Sistema de transferencia de datos de origen múltiple a oracle para la empresa solinfo," mathesis, Universidad Técnica de Ambato, Apr. 2019.

[23] J. H. Canós, P. Letelier, and C. Penadés, "Metodologías ágiles en el desarrollo de software," *Repositorio institucional de la Universidad de Las Tunas*, Mar. 2012.

[24] A. Almeira and V. Perez Cavenago, "Arquitectura de software: Estilos y patrones," *Repositorio de la Universidad Nacional de la Patagonia San Juan Bosco*, vol. 28, Mar. 2007.

[25] S. Smith, "Información general de asp.net core mvc." Online, Url: https://docs.microsoft.com/es-es/aspnet/core/mvc/overview?view=aspnetcore-2.2, Jan. 2018. Accedido 09-06-2019.

[26] J. M. Gutiérrez, "Sistemas expertos basados en reglas," *Recurso personal. Departamento de Matemática Aplicada. Universidad de Cantabria*, pp. 1–12, 2008.

[27] Futurizable, "Estado del arte en el desarrollo de chatbots a nivel mundial." Online, Url: https://futurizable.com/chatbot/, Sept. 2017. Accedido 21-05-2019.

[28] M. E. Carrillo Calderón, J. Gallego Gómez, and E. Pulido, "Agentes virtuales con capacidades cognitivas utilizando ibm watson," B.S. thesis, UAM. Departamento de Ingeniería Informática, June 2017.

[29] P. Hernández Delgado, "Programación de un [ro] bot para una consulta interactiva de la información de un proyecto," *UPCommons. Portal de acceso abierto al conocimiento de la UPC*, 2016.

[30] AWS, *Guía para desarrolladores, Amazon Lex.* Amazon Web Services Inc., 2019.

[31] BotLibre, "Acerca de bot libre." Online, Url: http://www.botlibre.org/, 2016. Accedido 24-05-2019.

[32] F. J. Hidalgo, "Bot libre, plataforma de código abierto para la creación de bots." Online, Url: https://wwwhatsnew.com/2016/01/29/bot-libre-plataforma-de-codigo-abierto-para-la-creacion-de-bots/, Jan. 2016. Accedido 24-05-2019.

[33] DialogFlow, *Guía para desarrolladores, DialogFlow.* Google, May 2018.

[34] A. Navarro Cadavid, J. D. Fernández Martínez, and J. Morales Vélez, "Revisión de metodologías ágiles para el desarrollo de software," *Prospectiva*, vol. 11, no. 2, pp. 30–39, 2013.

[35] L. M. Echeverry Tobón and L. E. Carmona Delgado, "Caso práctico de la metodología ágil xp al desarrollo de software," mathesis, Universidad Tecnológica de Pereira, Oct. 2007.

[36] D. Bustamante and J. Rodríguez, "Metodología actual metodología xp," *Informe. Barinas: Universidad Nacional Experimental de los Llanos occidentales Ezequiel Zamora, Facultad de informática*, 2014.

[37] P. Letelier and C. Penadés, "Metodologías ágiles para el desarrollo de software: extreme programming (xp)," *Repositorio institucional de la Universidad de Las Tunas*, 2006.

Anexos y Apéndices

Anexo A

Diagramas de flujo del proceso de cada diálogo

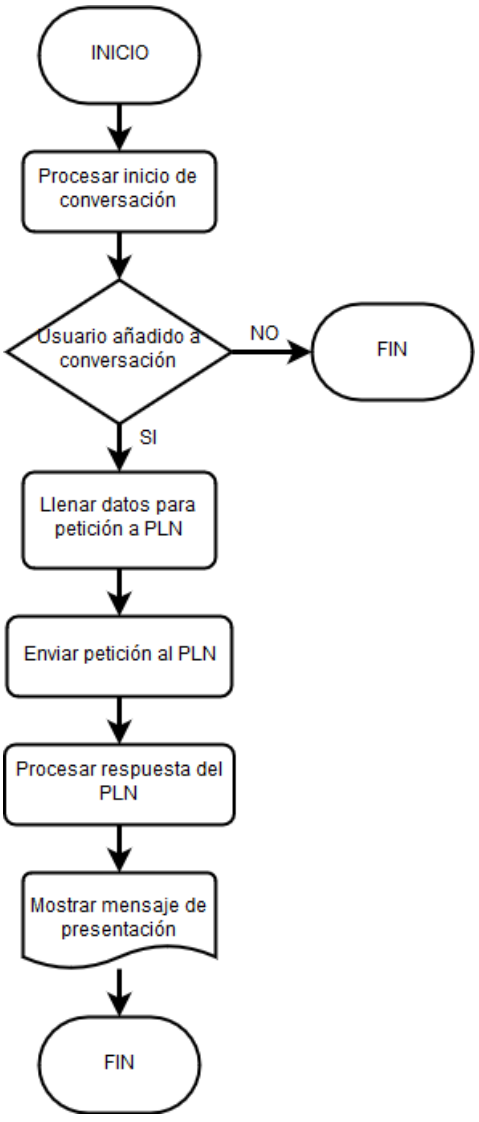

Figura A.1: Diagrama de flujo del diálogo de inicio.
Fuente: Elaborado por el investigador.

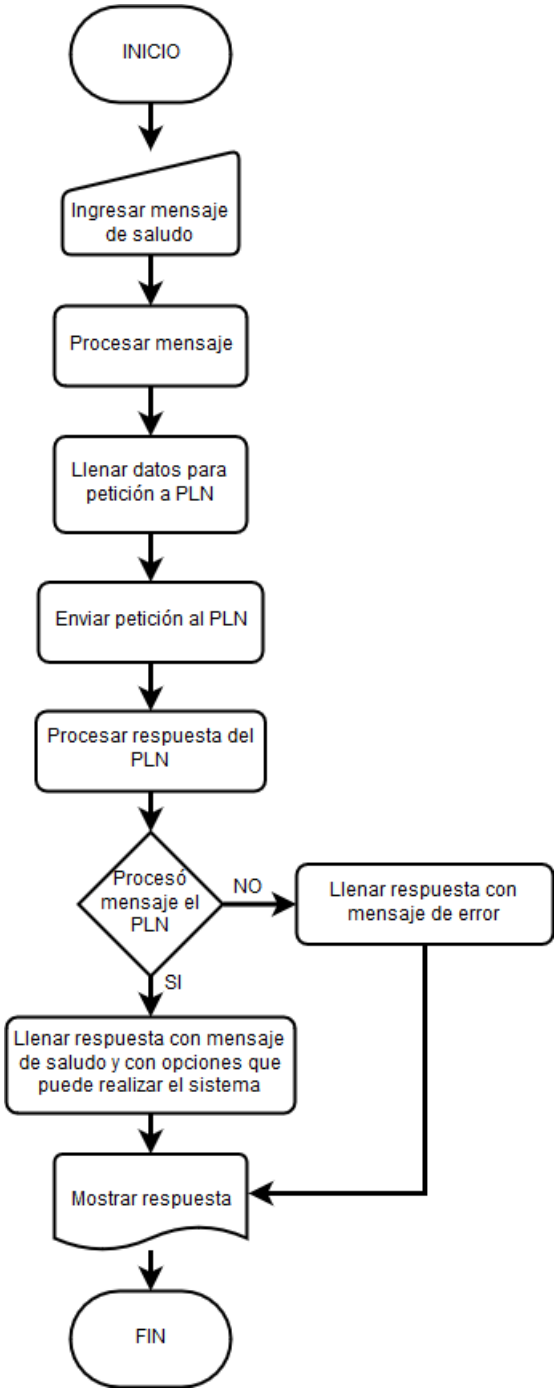

Figura A.2: Diagrama de flujo del diálogo de saludo.
Fuente: Elaborado por el investigador.

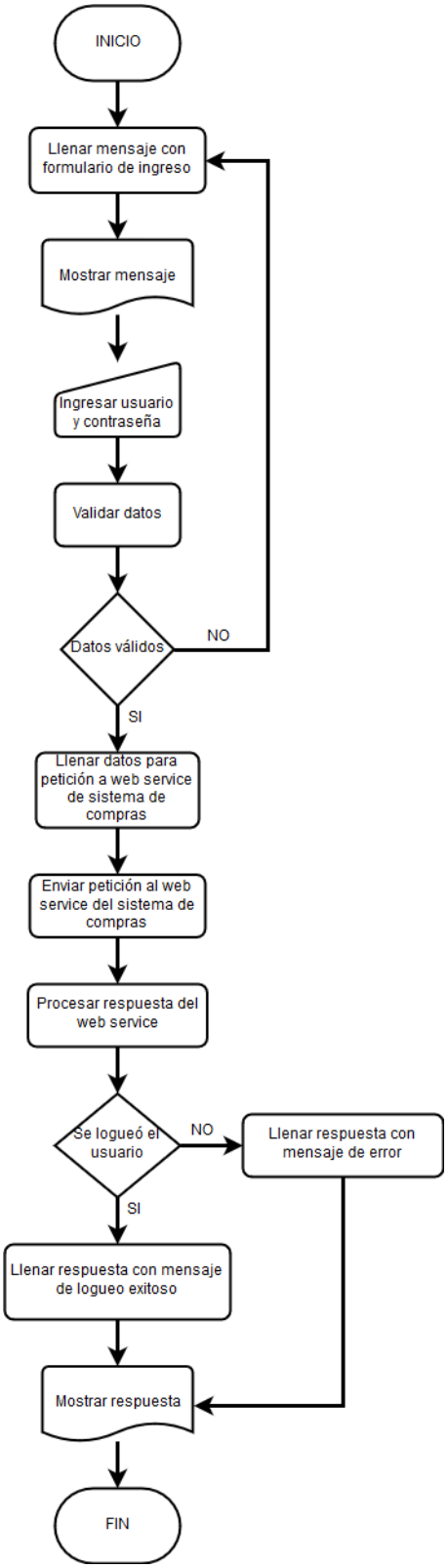

Figura A.3: Diagrama de flujo del diálogo de ingreso al sistema.
Fuente: Elaborado por el investigador.

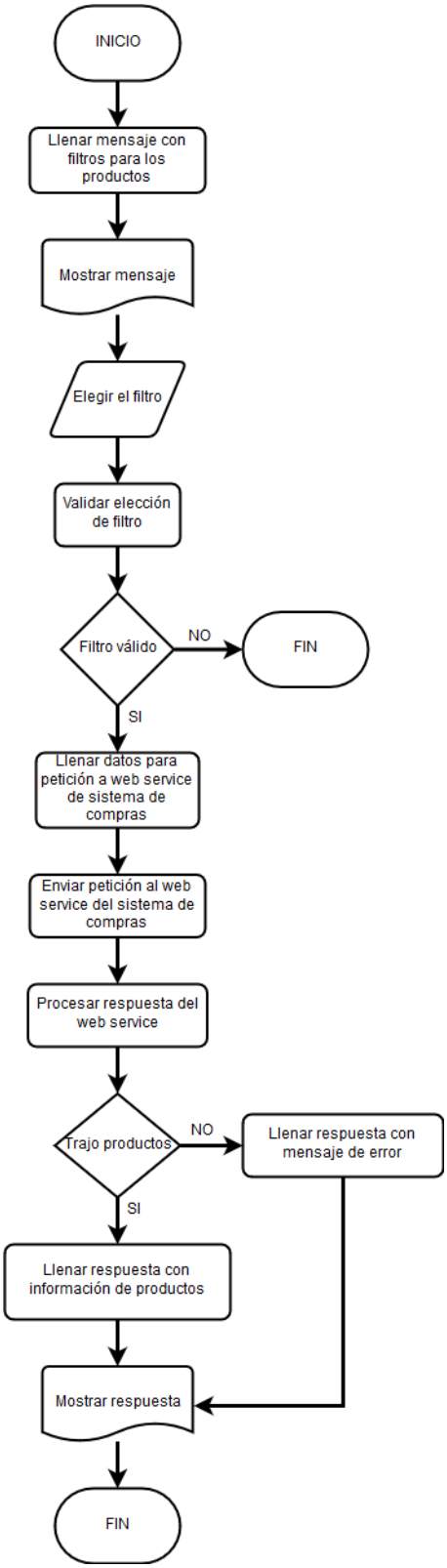

Figura A.4: Diagrama de flujo del diálogo para consultar productos.
Fuente: Elaborado por el investigador.

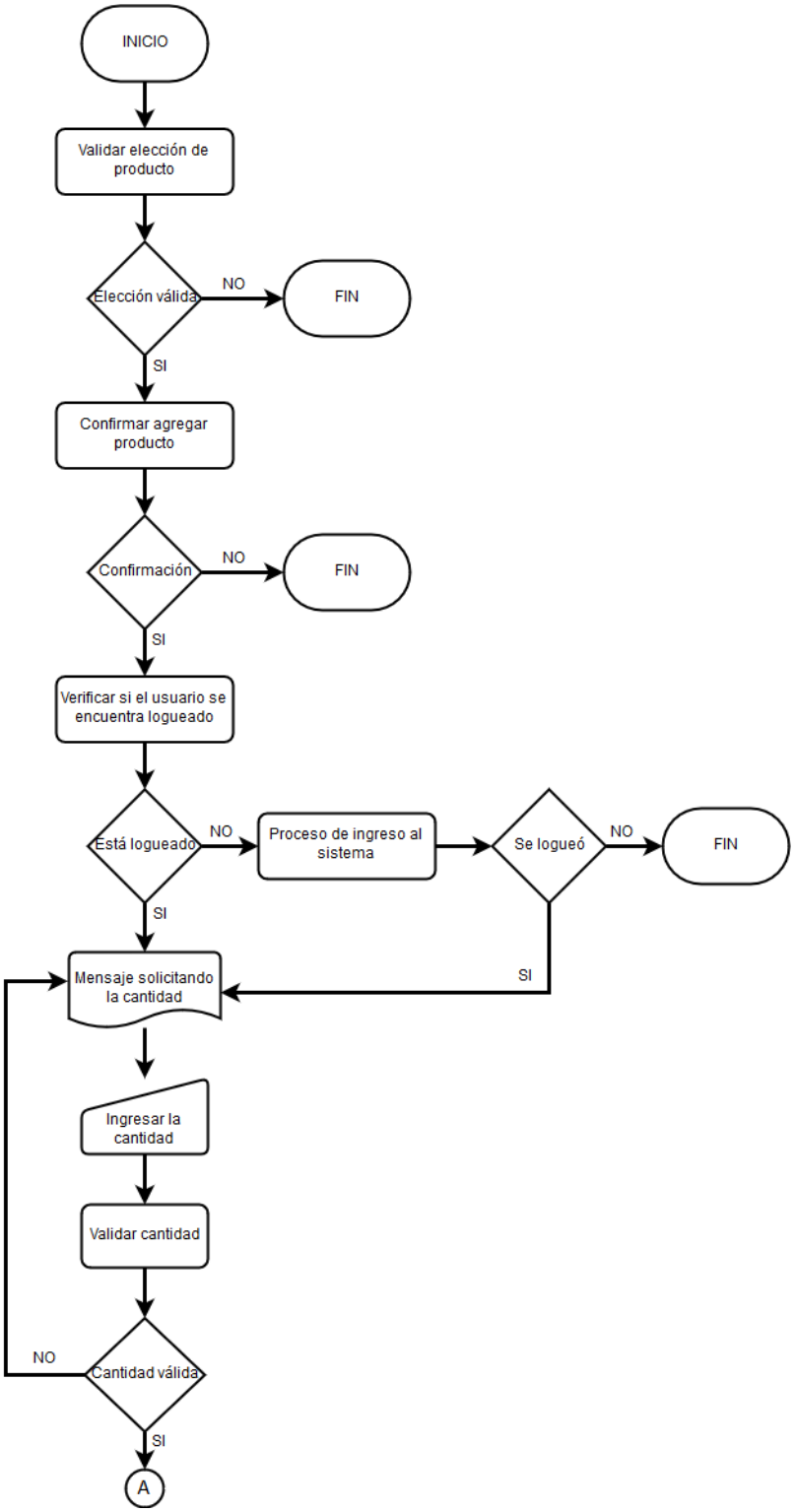

Figura A.5: Diagrama de flujo del diálogo para agregar productos a la orden de compra.

Fuente: Elaborado por el investigador.

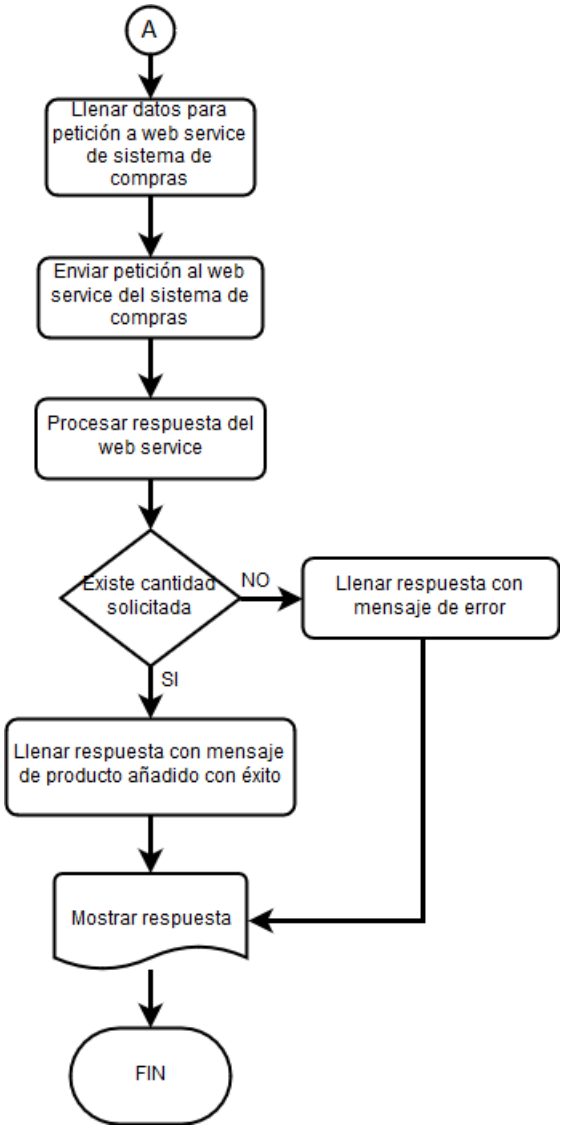

Figura A.6: Continuación diagrama de flujo de la figura A.5.
Fuente: Elaborado por el investigador.

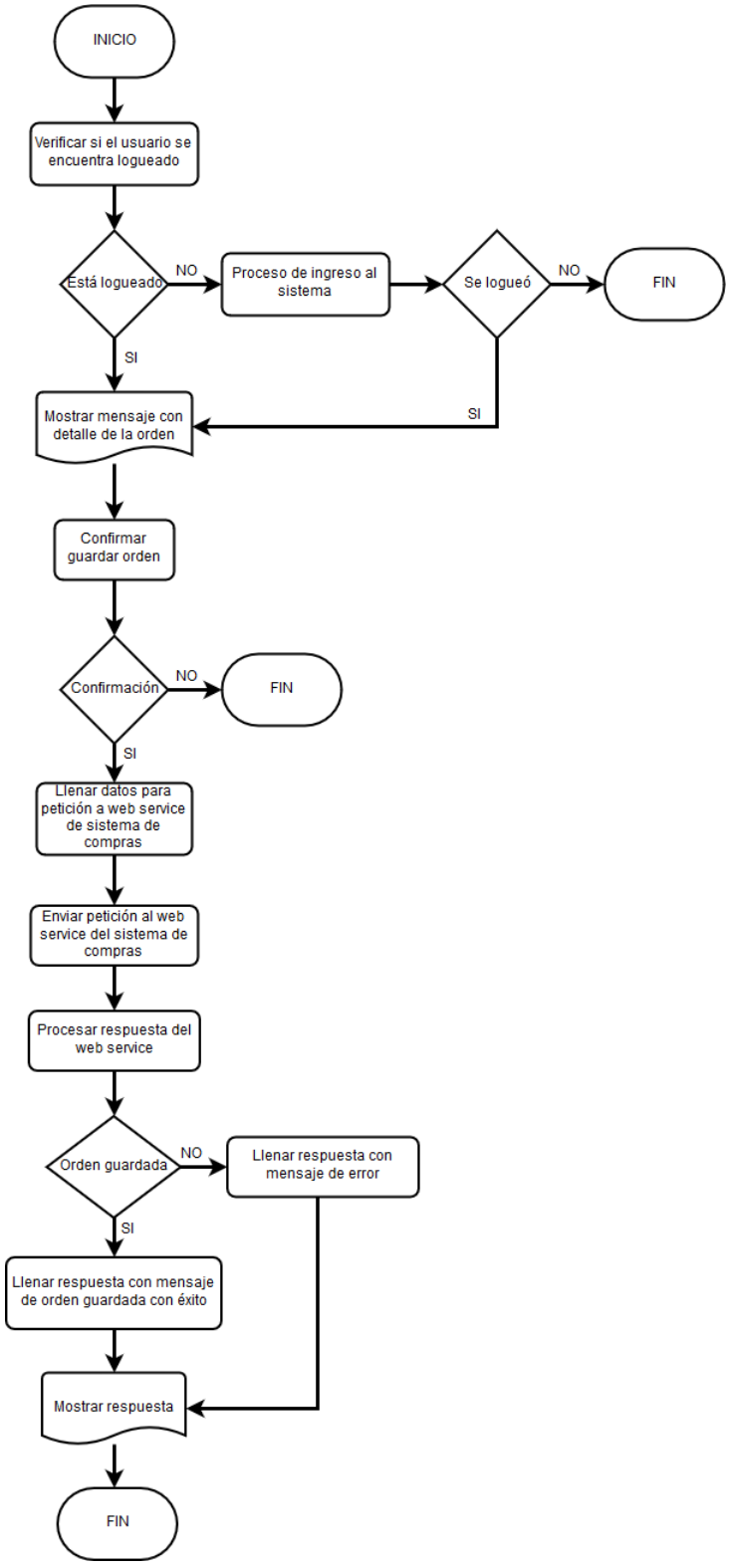

Figura A.7: Diagrama de flujo del diálogo para guardar la orden de compra.
Fuente: Elaborado por el investigador.

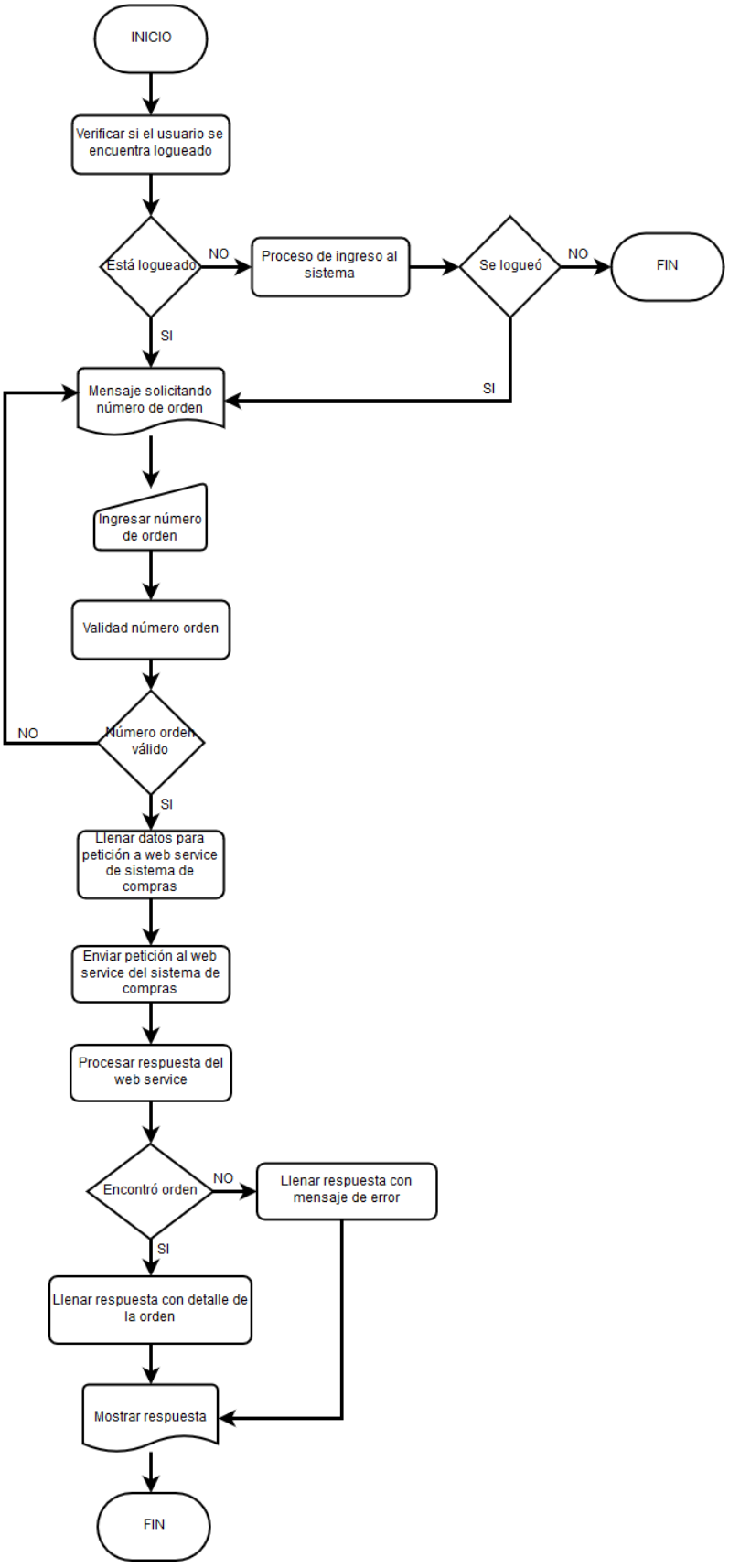

Figura A.8: Diagrama de flujo para visualizar la orden de compra.
Fuente: Elaborado por el investigador.

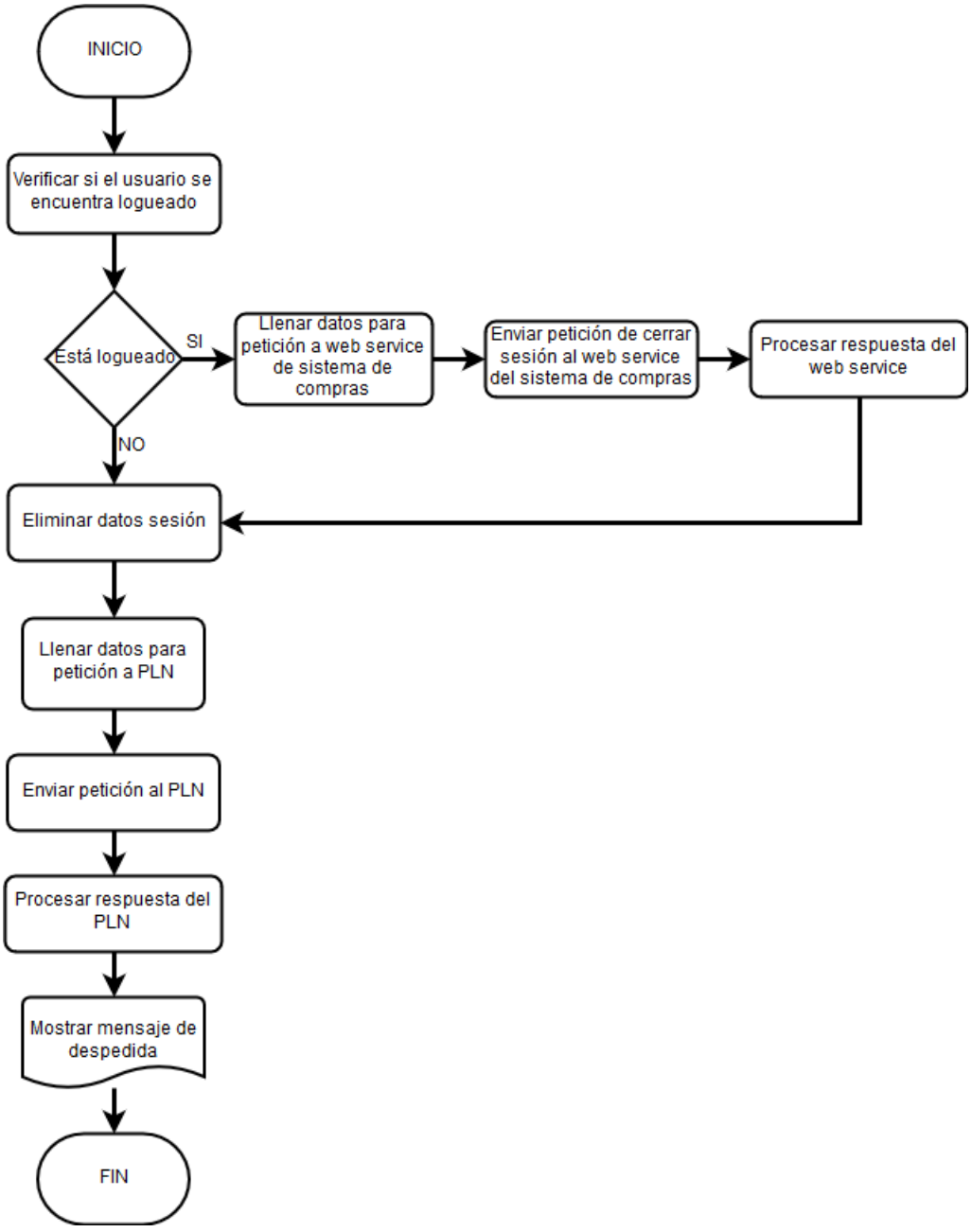

Figura A.9: Diagrama de flujo del diálogo de fin.
Fuente: Elaborado por el investigador.

Anexo B

Prototipos de interfaces de diálogos

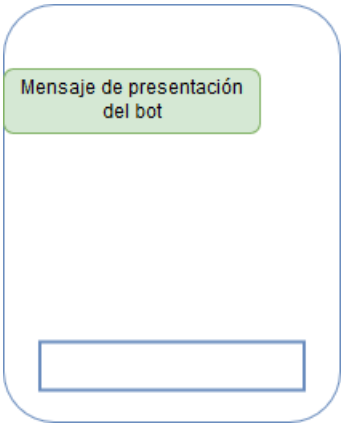

Figura B.1: Prototipo del diálogo de presentación.
Fuente: Elaborado por el investigador.

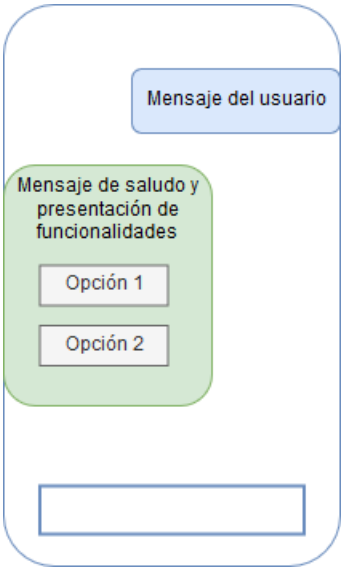

Figura B.2: Prototipo del diálogo de saludo.
Fuente: Elaborado por el investigador.

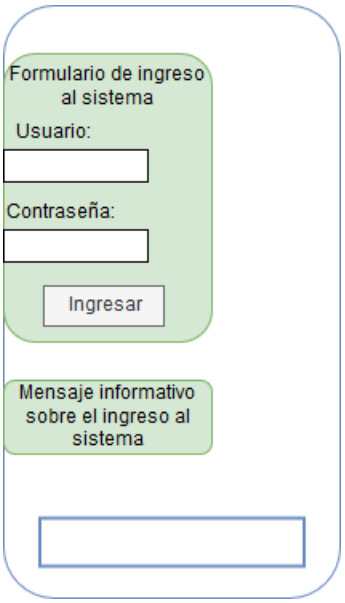

Figura B.3: Prototipo del diálogo de ingreso.
Fuente: Elaborado por el investigador.

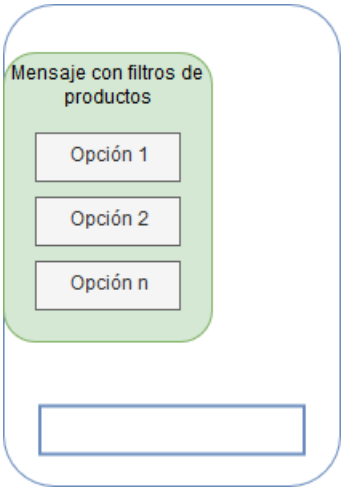

Figura B.4: Prototipo del diálogo para filtrar productos.
Fuente: Elaborado por el investigador.

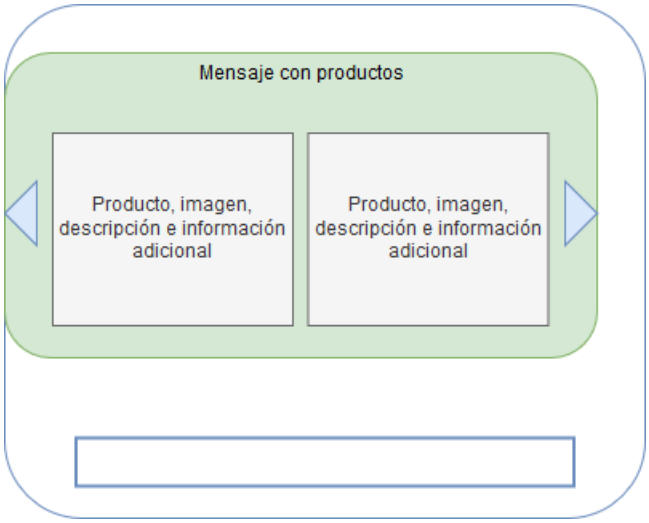

Figura B.5: Prototipo de diálogo para consultar productos.
Fuente: Elaborado por el investigador.

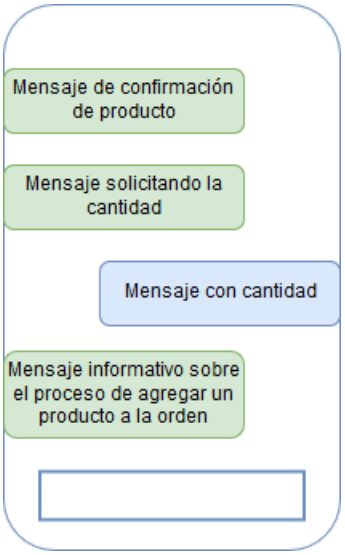

Figura B.6: Prototipo de diálogo para agregar productos a la orden de compra.
Fuente: Elaborado por el investigador.

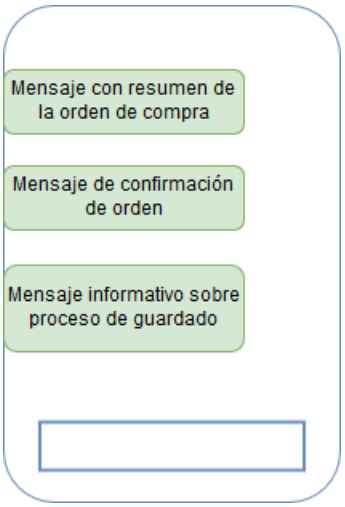

Figura B.7: Prototipo de diálogo para guardar orden.
Fuente: Elaborado por el investigador.

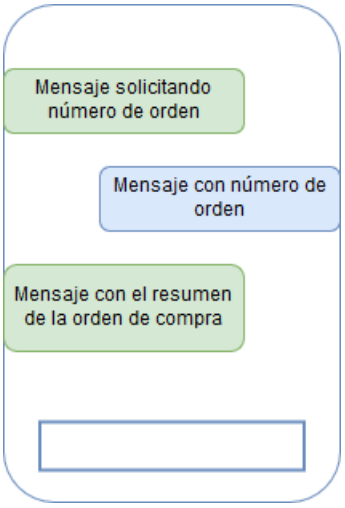

Figura B.8: Prototipo de diálogo para visualizar la orden de compra.
Fuente: Elaborado por el investigador.

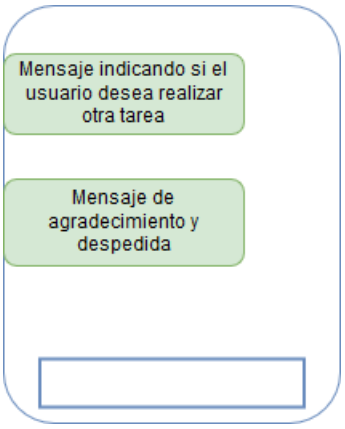

Figura B.9: Prototipo de diálogo de fin.
Fuente: Elaborado por el investigador.

Anexo C

Imágenes del código del chatbot

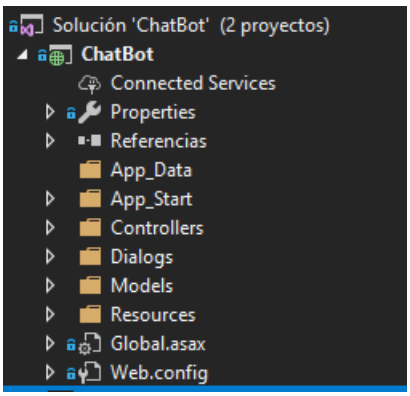

Figura C.1: Estructura del proyecto del chatbot.
Fuente: Elaborado por el investigador.

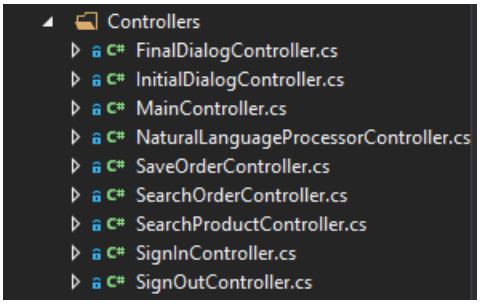

Figura C.2: Controladores usados en el chatbot.
Fuente: Elaborado por el investigador.

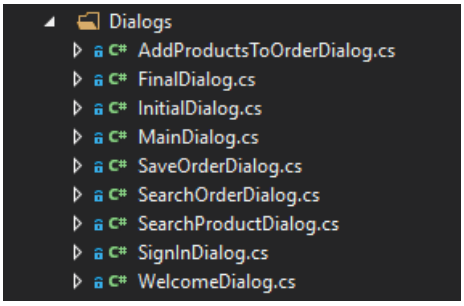

Figura C.3: Diálogos usados en el chatbot.
Fuente: Elaborado por el investigador.

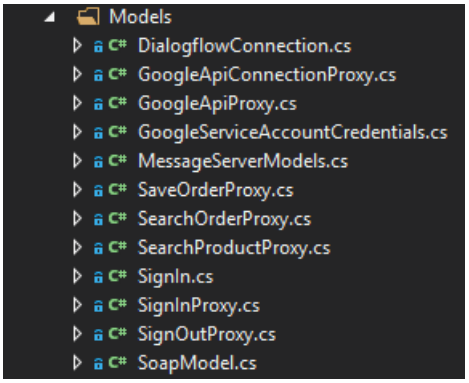

Figura C.4: Modelos usados en el chatbot.
Fuente: Elaborado por el investigador.

```
MainController.cs
ChatBot                                   ChatBot.Controllers.MainController
 9  namespace ChatBot.Controllers
10  {
11      //[BotAuthentication]
        0 referencias | Erick Gamboa, Hace 18 horas | 1 autor, 4 cambios
12      public class MainController : ApiController
13      {
            0 referencias | Erick Gamboa, Hace 9 días | 1 autor, 1 cambio
14          public async Task<HttpResponseMessage> Post([FromBody] Activity activity)
15          {
16              await this.HandleSystemMessage(activity);
17
18              var response = Request.CreateResponse(HttpStatusCode.OK);
19              return response;
20          }
21
            1 referencia | Erick Gamboa, Hace 1 día | 1 autor, 3 cambios
22          private async Task HandleSystemMessage(Activity message)
23          {
24              switch (message.Type)
25              {
26                  case ActivityTypes.Message:
27                      await Conversation.SendAsync(message, () => new Dialogs.MainDialog());
28                      break;
29                  case ActivityTypes.ConversationUpdate:
30                      await SendTypingMessage(message);
31                      await new Dialogs.InitialDialog().OnMembersAddedAsync(message);
32                      break;
33                  case ActivityTypes.EndOfConversation:
34                      await new Dialogs.FinalDialog().OnEndConversation(message);
35                      break;
36                  default:
37                      break;
38              }
39          }
40
```

Figura C.5: Controlador principal del chatbot.
Fuente: Elaborado por el investigador.

```
SignInController.cs
ChatBot                                   ChatBot.Controllers.SignInController   ProcessRequest(stri
 1  using ChatBot.Models;
 2  using System;
 3
 4  namespace ChatBot.Controllers
 5  {
        1 referencia | Erick Gamboa, Hace 2 días | 1 autor, 1 cambio
 6      internal class SignInController
 7      {
            1 referencia | Erick Gamboa, Hace 2 días | 1 autor, 1 cambio
 8          internal static AutenticarResponse ProcessRequest(string userName, string password)
 9          {
10              AutenticarResponse processOutput;
11              try
12              {
13                  processOutput = SignInProxy.Authentication(new AutenticarRequest(userName, password));
14              }
15              catch (Exception ex)
16              {
17                  processOutput = new AutenticarResponse();
18                  processOutput.success = false;
19                  processOutput.data = ex.Message;
20              }
21              return processOutput;
22          }
23      }
24  }
```

Figura C.6: Controlador de ingreso al sistema.
Fuente: Elaborado por el investigador.

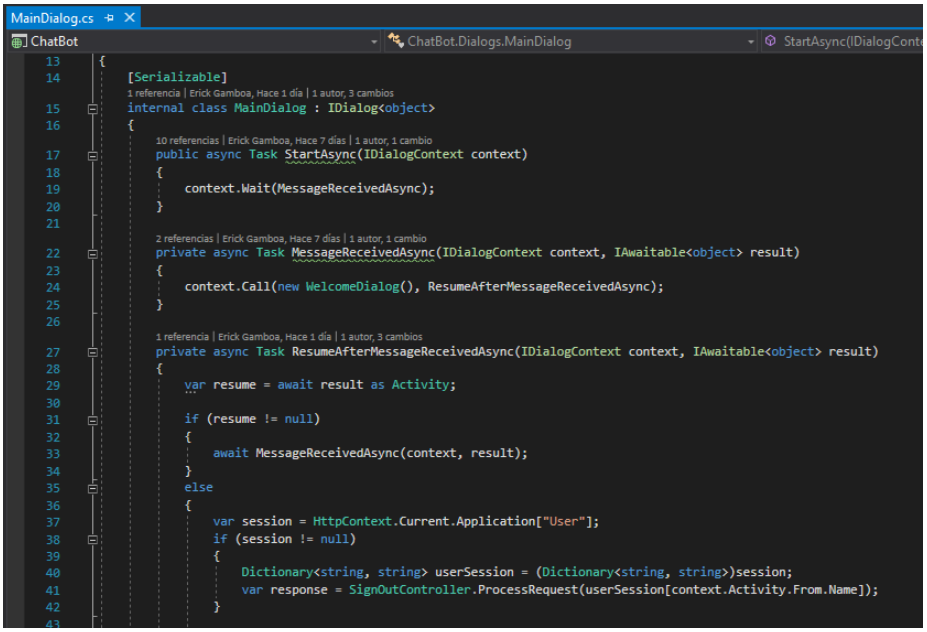

Figura C.7: Diálogo principal del chatbot.
Fuente: Elaborado por el investigador.

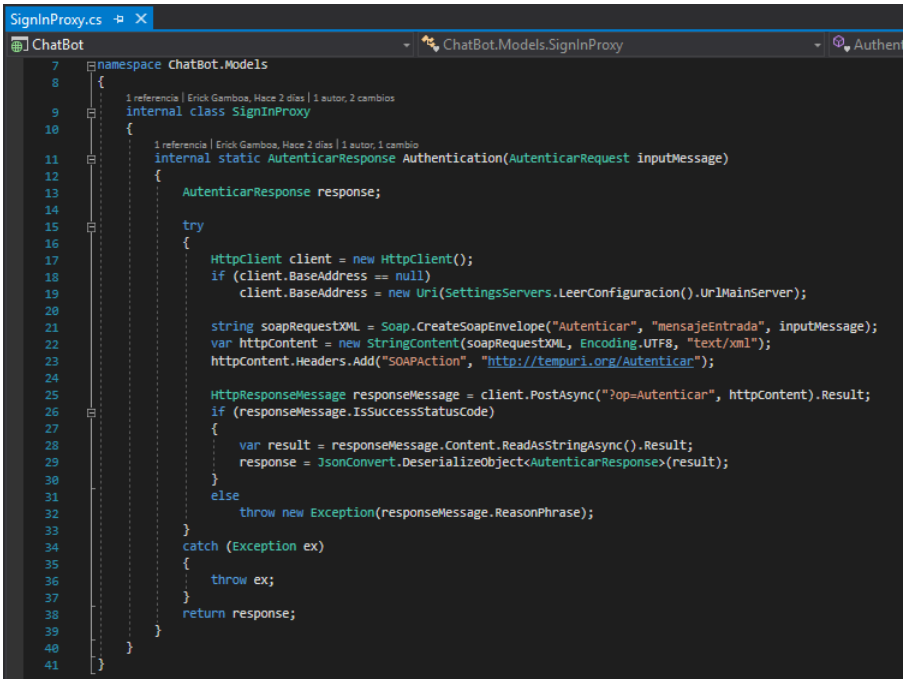

Figura C.8: Método que realiza comunicación con el sistema de compras para autenticar un usuario.
Fuente: Elaborado por el investigador.

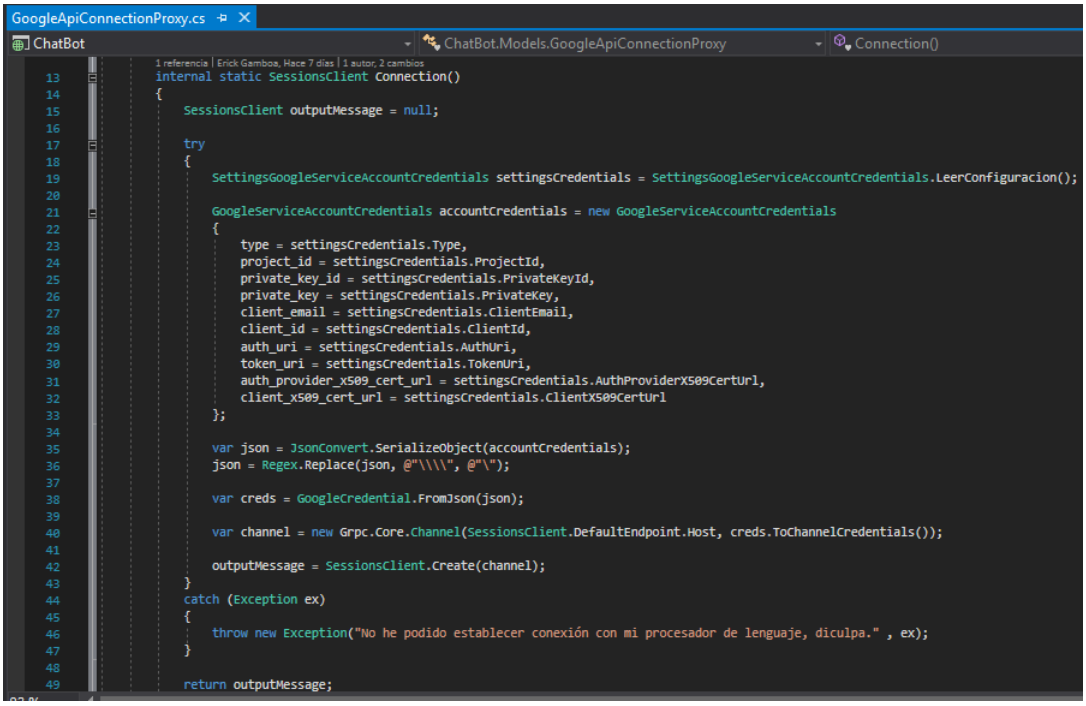

Figura C.9: Método que establece la comunicación con el procesador del lenguaje natural.

Fuente: Elaborado por el investigador.

Figura C.10: Método que realiza una petición al procesador del lenguaje natural.
Fuente: Elaborado por el investigador.