Universidad de las Ciencias Informáticas

Facultad 2

Desarrollo de una entidad conversacional artificial en la Mensajería Colaborativa de Cuba

Trabajo de Diploma para optar por el título de Ingeniero en Ciencias Informáticas.

Autor: Yosvani Vázquez Cruz

Tutores: Ms. C. Allan Pierra Fuentes

Ing. Haniel Cáceres Navarro

Ing. Damián Ilizasteguí Arriba

La Habana

2018

DECLARACIÓN DE AUTORÍA

Declaro ser autor de la presente tesis que tiene por título: Desarrollo de una entidad conversacional artificial para la Mensajería Colaborativa de Cuba; y reconozco a la Universidad de Ciencias Informáticas los derechos patrimoniales de la misma con carácter exclusivo. Para que así conste firmo la presente a los ___ días del mes de _____ del año _____.

Yosvani Vázquez Cruz

Firma del Autor

Ms. C. Allan Pierra Fuentes

Firma del Tutor

Ing. Haniel Cáceres Navarro

Firma del Cotutor

Ing. Damián Ilizasteguí Arriba

Firma del Cotutor

DATOS DE CONTACTO

Autor:

Yosvani Vázquez Cruz

yvcruz@estudiantes.uci.cu

Tutores:

Ms. C. Allan Pierra Fuentes
Director del Centro de Soporte Tecnológico
apierra@uci.cu

Ing. Haniel Cáceres Navarro
Especialista Centro de Software Libre.
hcaceres@uci.cu

Ing. Damián Ilizasteguí Arriba
Especialista Centro de Telemática.
dilizastegui@uci.cu

AGRADECIMIENTOS

A mis padres, por ser incondicionales en cada momento de la vida, por el apoyo siempre ofrecido; por ser mis maestros, mis guías y los motores de mi motivación por ser ingeniero.

A mi hermana preferida, por ser única, original, especial y en grandes rasgos mi modelo a seguir.

A Yaíma por acompañarme, aconsejarme y fundamentalmente por estar para mí, en cada momento desde que nos conocimos.

A Santino, Haniel y los Eduardos por ser más que mis amigos: mis hermanos.

A los profesores que me transmitieron su conocimiento y las enseñanzas que me guiaron hasta este momento.

A todas mis amistades por permitirme estar entre ellos.

DEDICATORIA

A mi papá:

“Quien me enseñó a pensar con lógica.”

A mi abuelo S. Silvano:

“Quien me enseñó a pensar como ingeniero.”

RESUMEN

El auge y expansión de las redes sociales en línea dio paso a la evolución de las aplicaciones de mensajería instantánea y al enriquecimiento de los servicios complementarios que estas ofrecen. El incremento del tráfico e intercambio de datos entre los usuarios de estas aplicaciones las convierten en un objetivo fundamental para la recolección y análisis de la información generada. Uno de los servicios ofrecidos por estos sistemas de mensajería son las entidades conversacionales o bots de charla; capaces de ofrecer respuestas automáticas a determinadas instrucciones realizando consultas a fuentes de datos en internet y siguiendo una estructura lógica en sus respuestas. La implementación e integración de estos bots de charla incrementa el valor funcional de las aplicaciones permitiendo la inserción de nuevos recursos y servicios que aseguran la fidelización de los clientes, y su posicionamiento entre las similares existentes. Tomando la situación descrita como punto de partida, la presente investigación define como objetivo: diseñar e implementar una entidad conversacional artificial, o bot de charla, como servicio complementario a los sistemas de mensajería instantánea. Este servicio permitirá realizar consultas a fuentes de información en internet; ofrecer respuestas automáticas y lógicamente estructuradas; e incrementar directamente el alcance y usabilidad de las aplicaciones que lo consuman, en la mayoría de los sectores de la sociedad a los que se destine. Lo que logra aumentar el número de usuarios activos en estas aplicaciones, y les permite enriquecer el entorno de servicios, con nuevos elementos disponibles que adquieren un mayor dinamismo y aceptación en ellas.

Palabras clave: mensajería instantánea, entidad conversacional, bot de charla.

ABSTRACT

The rise and expansion of online social networks gave way to the evolution of instant messaging applications and the enrichment of the complementary services they offer. The increase in traffic and data exchange among the users of these applications make them a fundamental objective for the collection and analysis of the information generated. One of the services offered by these messaging systems are the conversational entities or chatbots. The development and integration of these chatbots increase the functional value of the applications, allowing the insertion of new resources and services that ensure customer loyalty, and their positioning among existing similar ones. Taking the described situation as starting point, the present research defines as objective: to design an artificial conversational entity, or chatbot, as a complementary service to instant messaging systems. This service will allow consulting information sources on internet; offer automatic and logically structured responses; and directly increase the scope and usability of the applications that consume it, in most of the sectors of society to which it is destined. Achieving increase the number of active users in these applications. Since it allows them to enrich the service environment, with new element that acquire a greater dynamism and acceptance in them.

Keywords: instant messaging, conversational entity, chatbot

ÍNDICE

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

Introducción

"El hombre es un ser social por naturaleza" es una frase del filósofo Aristóteles (384 a. C.-322 a. C.) para constatar que los humanos nacen con la característica de ser social. Siendo objeto de desarrollo a lo largo de la vida, ya que experimentan esta necesidad para sobrevivir. (Significados.com, 2016)

El proceso de sociabilización es el conjunto de aprendizajes que el hombre necesita para relacionarse con autonomía dentro de una sociedad. Por ejemplo, la incorporación de normas de conductas en el lenguaje y la cultura. En suma, representan elementos para mejorar la capacidad de comunicación y la de relación en comunidad. (Dewey, 2014)

Debido a que un hombre aislado no puede desarrollarse como persona, se evidencia la tendencia a agruparse en vez de aislarse. Estos continúan con la necesidad de comunicarse, aun cuando los avances científicos y tecnológicos han permitido que la interrelación y la comunicación entre estos sean menos indispensable. Definiéndose como comunicación: el acto por el cual un individuo establece con otro un contacto que le permite transmitir una información. Este proceso es posible debido a la adopción de las formas de comunicación verbal y no verbal. (Maslow, 2016)

La comunicación verbal es la que se realiza a través de signos orales y palabras habladas o escritas. Permitiéndose por medio de la representación gráfica de signos; por ejemplo: gritos, silbidos, llantos o risas que pueden expresar diferentes situaciones anímicas. Esta forma representa la estructura más primaria de la comunicación.

La comunicación no verbal se realiza a través de multitud de signos de gran variedad: Imágenes sensoriales, sonidos, gestos, movimientos corporales, etc. Convirtiéndola en una forma impersonal de comunicación debido a que intervienen diversos elementos que pueden facilitar o dificultar el proceso de comunicación. Es por ello que se han desarrollado nuevas formas de comunicación y convivencia en sociedad; ejemplo de ello es el nacimiento de las redes sociales y su rápida expansión y adaptación en la comunidad. (Monar y Vicente, 2017)

Como redes sociales se denominan las estructuras que representan a un conjunto de individuos que se encuentran interrelacionados entre sí. Son una especie de mapa que muestra los lazos que vinculan a un grupo de personas. Estas se adaptan a muchas aplicaciones en la vida cotidiana de las personas. Pueden emplearse fundamentalmente para socializar, encontrar viejos amigos y entablar nuevas amistades; pero también para hacer contactos profesionales y buscar trabajo. Existen redes sociales temáticas, que permiten que personas con las mismas inquietudes y afinidades puedan ponerse contacto y compartir intereses. Este concepto se introdujo a la red de redes a partir del año 1995 por el estadounidense Randy Conrads cuando

creó el sitio classmates.com. Este sitio tenía el propósito que los usuarios pudieran recuperar o mantener el contacto con antiguos compañeros del colegio, instituto o universidad. Permitiendo que la Internet empezara a ser una herramienta masificadora a través de las redes sociales en línea. (Rosas, 2015)

Las redes sociales en línea son una combinación de las formas verbales y no verbales de interacción social. Están definidas como el intercambio dinámico entre personas, grupos e instituciones en contextos de complejidad. Un sistema abierto y en construcción que involucra a conjuntos de individuos que se identifican en las mismas necesidades. Estas permiten la interrelación entre personas conocidas o desconocidas, creándose grupos de comunicación con cada miembro que se integra a la red social. Las redes sociales en línea tienen como requisito en la mayoría de los casos, la creación de un contexto histórico-social en cada perfil del usuario.

Las redes sociales en línea emplean protocolos de redes que permiten la comunicación entre los usuarios como el SMTP (Simple Mail Transference Protocol)[1]. Estas fomentan la conjugación entre la popularidad y la comunidad, así como la actualización automática de la libreta de direcciones y la capacidad de crear nuevos enlaces mediante servicios de presentación. Destacándose el empleo de estos servicios por la gran cantidad de usuarios suscritos a estas: Facebook, Twitter y LinkedIn. (Ponce, 2016)

El avance de la tecnología móvil y la integración de esta a la sociedad, marcó un punto de inflexión en la forma de comunicación entre las personas. Específicamente en las redes sociales en línea que dieron paso a la mensajería instantánea. Estas cuentan con características similares a sus antecesoras y destacan la posibilidad de comunicación en tiempo real. Basando su funcionamiento en el envío de texto entre dos o más individuos a través de dispositivos conectados a una red, como Internet. (Liu, Navarrete y Wivagg, 2014)

La mensajería instantánea se basa en el uso de programas que brindan este servicio, conocidos como clientes de mensajería instantánea, que se instalan en una computadora o dispositivo móvil. La comunicación entre clientes está regulada por la conexión entre estos y un servidor en Internet que emplee algún protocolo de comunicación como XMPP (Extensible Messaging and Presence Protocol, en inglés). Siendo posible el envío mutuo de mensajes en formato de texto, imágenes, audios y videos simultáneamente.

El desarrollo de las aplicaciones móviles que permiten la IM está influenciado fundamentalmente por el incremento de la preferencia de los usuarios a utilizarlas por encima de las tradicionales redes sociales.

[1] El Simple Mail Transfer Protocol (SMTP) o “protocolo para transferencia simple de correo”, es un protocolo de red utilizado para el intercambio de mensajes de correo electrónico entre computadoras u otros dispositivos.

Esto se debe a la tendencia de que es más cómodo establecer una conversación directa en tiempo real, en vez de crear un entorno histórico-social o temático cómo el que tiende a requerir el empleo redes sociales en línea. De ahí que las compañías propietarias de estas redes sociales, implementasen entre ellas sistemas de mensajería instantánea que se ajustaran a las actuales exigencias de los clientes, como es el caso de Facebook Messenger. (Oghuma et al., 2016)

Otras compañías emergieron en el mercado de las aplicaciones móviles desarrollando servicios de IM, siendo este el caso de WhatsApp. En su lanzamiento oficial en 2009 por el ucraniano-estadounidense Jan Koum, WhatsApp, brindaba a sus usuarios la posibilidad de establecer una comunicación en tiempo real a través de internet. Más tarde fue complementándose con otros servicios que impulsaron el auge de esta aplicación. Reflejado en el registro de 1300 millones de usuarios a finales del 2014; cifra que la convirtió en una de las más populares a nivel mundial. (Montag et al., 2015)

Para esta misma fecha se evidenció un incremento en el tráfico de los usuarios en la red de redes, debido a la competencia existente en el mercado de aplicaciones entre las potencias de mensajería instantánea. Registrándose en estas grandes cantidades de individuos como el caso de Facebook Messenger con un total de 1200 millones de usuarios y Telegram 180 millones de usuarios. (Greenwood, Perrin y Duggan, 2016)

Las aplicaciones de este tipo con mayor éxito en la actualidad, ofrecen disímiles servicios a los usuarios que complementan la función básica de estas (comunicación par a par en tiempo real). Por ejemplo: conversaciones entre grupos, personalización del perfil de usuario, video llamadas, entidades conversacionales entre otros. Estos servicios varían según el enfoque comercial de las aplicaciones y las compañías que los implementan; y siempre están orientados a lograr la afinación y fidelización de los usuarios. Teniéndose como propósito, la creación de un entorno personalizado, cómodo y familiar, para estos.

Uno de los servicios complementarios de mayor consumo en las aplicaciones de mensajería instantánea, son las entidades conversacionales o bots de charla. Estos representan en algunos casos, el elemento clave sobre la elección de cual aplicación o servicio de mensajería instantánea emplear. Definidos como programas que simulan mantener una conversación con una persona o usuario, al proveer respuestas automáticas mediante texto, los bots son empleados con variados fines. Los bot de consulta a fuentes de información, se encuentran entre los de mayor utilización, ya que permite la capacidad de realizar búsquedas en tiempo real sobre cualquier temática accediendo a fuentes de información confiable, y brindando un servicio automatizado de atención al cliente. (Beilby, Zakos y McLaughlin, 2014)

Nuestro país se encuentra inmerso en un conjunto de cambios que inciden directa o indirectamente a la mayor parte de los sectores en los que se fomenta la sociedad. Especialmente sobre la tecnología se han realizado acciones a favor de alcanzar la soberanía tecnológica. El proceso de la informatización de la sociedad cubana ha dado paso a la adopción de soluciones informáticas desarrolladas por cubanos y para los cubanos, como es el caso de los sistemas operativos Nova y Novadroid.

Específicamente en el campo de la tecnología móvil y la mensajería instantánea, se han realizado notables avances en pos de mejorar los servicios brindados a empresas, organizaciones y el pueblo en general. Permitiendo emplear mecanismos y herramientas que incrementan la efectividad y comunicación entre estos, a través de los medios que posibilitan la ejecución de los servicios de mensajería instantánea en la Mensajería Colaborativa de Cuba.

Las aplicaciones que brindan servicios de mensajería instantánea están en constante evolución y desarrollo. La integración de bot de charla a estos como servicios complementarios, les permitirían estar al nivel de las exigencias de los usuarios y competir con los similares existentes en el mercado internacional. Por otra parte la utilización de bots conversacionales en estas aplicaciones permitirá la inserción de un conjunto de nuevos recursos y servicios que permitirán la afinación y fidelización de los usuarios con las aplicaciones nacionales. Teniendo en cuenta la situación problemática antes descrita; se propone como **problema de la investigación**: ¿Cómo proveer a los servicios de mensajería instantánea de una entidad conversacional artificial?

Definiéndose como **objeto de estudio** de la investigación: el desarrollo de entidades conversacionales artificiales.

Tomando el punto de partida antes expuesto la presente investigación define como **objetivo general** desarrollar una entidad conversacional artificial para los servicios de mensajería instantánea, enmarcándose en el **campo de acción**: la integración de entidades conversacionales en aplicaciones de mensajerías instantánea.

Para darle solución al mismo se definen como **objetivos específicos**:

1. Elaborar el marco teórico de la investigación sobre el uso de bots conversacionales en servicios de mensajería instantánea.
2. Diseñar e implementar una propuesta de solución que integre bots conversacionales en servicios de mensajería instantánea.
3. Valorar el correcto funcionamiento de la integración de bots conversacionales en servicios de mensajería instantánea.

Para darle cumplimiento a los objetivos específicos se plantean las siguientes **tareas de la investigación**:

1. Elaboración del marco teórico a partir del estado del arte de los bots en los sistemas de mensajería instantánea, que permita analizar los conceptos relacionados a estos y su integración en los servicios de mensajería instantánea.

2. Selección del modelo de bot conversacional según su propósito, para la integración del mismo en servicios de mensajería instantánea.

3. Elección de la metodología, las herramientas de desarrollo, lenguajes a utilizar en la implementación del módulo de bots conversacionales para la integración de este en servicios de mensajería instantánea.

4. Elaboración del análisis de bots conversacionales para la integración en servicios de mensajería instantánea.

5. Elaboración y ejecución de una estrategia de pruebas, que permita valorar la calidad de bots conversacionales y la correcta integración en servicios de mensajería instantánea.

Para un mejor desarrollo de la investigación se utilizaron los siguientes **métodos científicos**:

- Del nivel teórico:
    - Método histórico: El cual permitió consultar la bibliografía referente a los bots conversacionales, la trayectoria, evolución y comportamiento en los sistemas de mensajería instantánea.
    - Método Lógico: Condujo a estructurar la documentación investigada de una manera organizada y cronológica, permitiendo un mejor entendimiento de la información consultada en la misma.
    - Método de la Modelación: Facilitó la creación de modelos, representando de manera gráfica parte del contenido de la presente investigación.
    - Método Sistémico: La utilización de este método permitió el estudio de la integración de las herramientas utilizadas, mediante la determinación de sus componentes y la relación entre ellos. Determinando la estructura y la jerarquía de cada componente y su dinamismo en diferentes etapas de empleo; siendo también la expresión del comportamiento del sistema como totalidad, en que un componente depende de otro u otros.

- Del nivel empírico
    - Método de la observación: Se empleó con el fin de concebir de forma consiente la planificación de la investigación, orientada al desarrollo de una entidad conversacional artificial para servicios de mensajería instantánea.
    - Método experimental: El continuo diseño de experimentos permitió estudiar paulatinamente el desarrollo de entidades conversacionales artificiales.
- Se evidenció un apoyo sobre los procesos de:
    - Análisis: Permitió establecer empíricamente, las relaciones y componentes de los bots conversacionales para una mejor comprensión.
    - Síntesis: Posibilitó establecer mentalmente la unión entre los componentes de los bots conversacionales resaltando sus principales características, conceptos y relaciones.
    - Documental: Definió los estándares y normas para consultar en fuentes de carácter documental tales como libros, artículos de revistas, ensayos y sitios en línea.

El presente trabajo de diploma está compuesto por un resumen, introducción, tres capítulos, conclusiones y recomendaciones y las referencias bibliográficas. Cumpliendo los objetivos planteados en el trabajo, a continuación se describe los principales aspectos abordados en cada uno de los capítulos.

Capítulo 1: Fundamentación teórica de los bot de charla, donde se expondrán los principales conceptos relacionados con el estado del arte de los bots conversacionales en las aplicaciones de mensajería instantánea y los modelos más empleados para la implementación de estos. Además de una breve descripción de los de mayor uso en las aplicaciones de mensajería instantánea.

Capítulo 2: Propuesta de solución, se realiza la fundamentación de la propuesta, se describe el flujo de los procesos involucrados en la solución a modo de comprenderlos totalmente. Se plantea la elaboración del modelo de dominio, la captura de requisitos funcionales y no funcionales de la propuesta. Se exponen a través de un conjunto de artefactos, la solución que se le dará al problema en cuestión.

Capítulo 3: Implementación y realización de pruebas, se describe el estándar de codificación empleado en el desarrollo de la propuesta de solución, asimismo se describen las pruebas realizadas con el fin de validar el desempeño y funcionalidad de la misma.

Capítulo 1 : Fundamentación teórica

En el presente capítulo se exponen los elementos teóricos que sustentan el proceso de investigación y desarrollo del tema propuesto. A través del estudio y análisis de soluciones existentes se analizan los conceptos y definiciones fundamentales relacionadas con los bots conversacionales. Así como los estándares para su implementación e integración. Se describen las herramientas y tecnologías utilizadas para el análisis, diseño e implementación de bots conversacionales sobre las que se complementa la propuesta de solución.

1.1. Bots conversacionales

Según plantean algunos autores, se define como bot de charla o bot conversacional, un programa de computadora diseñado para simular una conversación de relativa inteligencia con uno o más humanos (o con otros bots conversacionales) por medio de texto y/o audio. Al proveerle respuestas automáticas a las entradas realizadas por un usuario, a través de servicios de mensajería instantánea como: Facebook, WhatsApp, WeChat[2] y Telegram[3]. (Rodríguez Roa, 2016)

La palabra “bot” hace referencia a robot, debido a la naturaleza del funcionamiento automatizado que las entidades conversacionales proveen. Desde el entorno que se ejecutan, los bots pueden ofrecer diferentes resultados y realizar cualquier tarea previamente definida en su interacción con los humanos.

Los bots de charla tienen disímiles empleos según el propósito por el que fueron diseñados. Fundamentalmente se aplican en la automatización de respuestas en las aplicaciones de mensajería instantánea, a partir del análisis de varias consultas realizadas simultáneamente, con el propósito de optimizar del tiempo de los clientes. Ya que estos pueden realizar búsquedas específicas de contenido sin tener que cambiar de aplicación en el momento de realizar dicha búsqueda. (Gonzales y González, 2017)

1.1.1. Usos de los bots conversacionales

Los servicios de mensajería instantánea son aplicaciones ejecutadas desde un dispositivo conectado a una red, como internet o datos móviles y se ejecutan a través de un servidor en línea. Actualmente este tipo de servicio se encuentra en constante desarrollo y evolución, y surge en respuesta a la tendencia de comunicación en tiempo real; que sociedad actual requiere. Debido a estas características los usuarios los emplean como una herramienta útil que les permite establecer la comunicación entre estos sin la necesidad

[2] **WeChat** es un servicio de mensajería de texto móvil y servicio de comunicación de mensajes de voz creado por la fábrica Tencent en China, en enero del 2011.

[3] **Telegram Messenger** es un servicio de mensajería por Internet desarrollado desde el año 2013 por los hermanos Nikolai y Pavel Durov.

de contar con agentes externos. En este tipo de aplicación o servicio, se establece la comunicación entre dos o más usuarios inscritos en grupos de charla a través de mensajes en formato de texto o audio. También son posibles los envíos de multimedia a través los canales de conexión que esta realiza. (Maxinez, Rangel y Garrido, 2010)

El constante acceso a toda la infraestructura de servidores anfitriones de los servicios de mensajería instantánea, permite que sean posibles los procesos de actualización y mantenimiento de los servicios. Asimismo es posible incluirle nuevas instancias que enriquezcan la calidad del software, como es el caso de los bots de charla. La integración de estos a los servicios de mensajería instantánea permite que los usuarios exploten y consuman los servicios que estas ofrecen. Por otra parte el empleo de los bots incrementa la utilidad y eficacia de las aplicaciones, al brindarlos como servicios complementarios que realizan consultas a fuentes de información especializadas (diccionarios, meteorología, deportes, etc.) y de asistencia automatizada de atención al cliente. (Kar y Haldar, 2016)

La integración de bots de charla en las aplicaciones de mensajería instantánea se define en varios niveles de complejidad, en consecuencia de la interacción en el dispositivo donde se ejecutan, sin la necesidad de interactuar con otras aplicaciones. Los bots pueden ser implementados en dependencia de un fin u objetivo específico; aunque se establece que se introducen en estas aplicaciones para automatizar los procesos de búsquedas de información y atención al cliente.

A continuación se describen los principales objetivos a los que son orientados los bot de charlas en las aplicaciones que brindan servicios de mensajería instantánea:

Ubicación: Los usuarios pueden charlar por un canal de mensajería y a través de una solicitud a un bot, este les proporcione la ubicación más cercana de algún negocio o lugar de interés.

Atención al cliente: Enfocado en servicios técnicos o soporte al usuario. Los usuarios pueden realizar consultas técnicas, consultas de estado de cuentas de banco y transferir dinero a través de bots. Ahorrándole recursos de infraestructuras a empresas con grandes masas de clientes.

Promoción de Ventas: Algunas aplicaciones de mensajería permiten a través de chatbots, realizar compras y pedidos entre sus clientes; brindando cómodos servicios de compra desde una misma aplicación en el dispositivo.

Difusión: Los usuarios pueden recibir noticias de relevancia, difundidas por bots, que una vez definidos los parámetros, pudieran ser enfocadas específicamente a algún interés.

El constante e incrementado empleo de los bots en las aplicaciones de IM han permitido el almacenamiento de grandes volúmenes de información generadas por las consultas realizadas. Permitiéndose el análisis de

estos datos, las aplicaciones pueden mejorar la experiencia del usuario, brindándole sugerencias de temas y bots afines a estos.

1.1.2. Tipos de bots

Los bots conversacionales de mayor éxito entre los usuarios son aquellos con los este interactúa frecuentemente, por lo que se crea una estrecha relación entre el cliente y la aplicación. Lo que se debe a que las aplicaciones de mensajería instantánea, implementan los bots con el propósito de brindar una amplia gama de estos y puedan ser utilizados según los intereses de los individuos. A continuación se detallan los tipos de bot de charla de mayor utilización.

- **Búsquedas y consultas**: Son los más utilizados, permiten realizar consultas a fuentes de información sobre un tema específico desde la aplicación. Realizan búsquedas a través de los parámetros definidos por el usuario según su necesidad. A partir de varios posibles resultados afines: noticias actuales, marcadores deportivos, metadatos de multimedia, etc. Pueden ser especializados según las preferencias y enfocarse a determinados sectores (salud, economía, sociedad, etc.). Pueden ser genéricos; en ambos casos, consultan grandes bases de datos en línea de artículos y sitios web hasta hallar un posible resultado.
- **Archivos**: Almacenan de forma segura notas y archivos, permiten compartir los archivos con otros usuarios y dispositivos. Generan mecanismos de transferencia (sitios web) a usuarios que no comparten la aplicación. Brindan oportunidad de conversión de archivos de texto, búsqueda de multimedia en fuentes específicas: YouTube[4], SoundCloud[5] e Instagram[6]. Además complementan servicios de antivirus, detectando posibles amenazas en estos.
- **Idiomas**: Traducen en tiempo real la conversación en forma de texto a varios idiomas y permiten establecer una comunicación entre usuarios de otra lengua. Reproducen a través de audios la pronunciación de palabras y frases en varios idiomas. Ofrecen mecanismos de estudios de una amplia gama de vocablos.
- **Juegos**: Los bots de juegos, explotan el área del entretenimiento a través de la interacción de usuarios aleatorios o conocidos, logrando la interrelación entre las personas. Pueden ser de varios tipos: estrategia, juegos al azar, puzles, trivias, etc.

[4] **YouTube** es un sitio web dedicado a compartir vídeos. Aloja una variedad de clips de películas, programas de televisión y vídeos musicales, así como contenidos aficionados como videoblogs y YouTube Gaming.

[5] **SoundCloud** es una plataforma de distribución de audio en línea en la que sus usuarios pueden colaborar, promocionar y distribuir sus proyectos musicales.

[6] **Instagram** es una red social y aplicación para subir fotos y videos.

- **Redes sociales**: Permiten gestionar el contenido de las redes sociales, así como participar en discusiones en foros y comunicarse con los usuarios de la red social desde la aplicación de IM. Además de la posibilidad de vincular todas las cuentas del cliente.

1.2. Gestión de datos en los bots

La interacción con los bots de charla, permite que este conozca y reconozca a los usuarios, a partir de la simplificación de la experiencia en el consumo de sus servicios. Esto está dado a que su naturaleza se basa en la recopilación de información para realizar su desempeño. Como consecuencia, las bases de datos de asociadas a bots conversacionales han crecido con la explotación exponencial de estos, almacenándose grandes volúmenes de información llamados Big Data. (Muñoz, 2017)

1.2.1. Los bots en la extracción de datos

Antes de la llegada de los bots, los sistemas de Big Data obtenían información sobre los comportamientos de los usuarios, a partir de los clicks, taps y los movimientos del ratón capturados desde las aplicaciones; elementos que contribuían a su base de datos. Sin embargo, con la integración de los bots es posible ir más adelante, en consecuencia de que las conversaciones entre un usuario y un bot son más íntimas y personales, este puede aprender mucho sobre la persona que lo emplea:

- **identidad** (nombre, apellidos, edad, fecha de nacimiento, género…)
- **privacidad** (correo electrónico, ubicación, dirección, número de teléfono…)
- **información administrativa** (cuentas bancarias, registros de usuario…)
- **datos sensibles** (religión, política, estado psicológico, vida amorosa)

Sin duda, la gestión de estos datos permite que se establezca una comunicación más humana.

Toda esta información en principio pertenece a las empresas y aplicaciones que implementan los bots; son estas quienes deciden como gestionarla. A partir de las regulaciones definidas por los acuerdos de confidencialidad de sus clientes y el Reglamento General de Protección de Datos, establecido por la Unión Europea en abril del 2016. La entrada en vigor en el presente año de esta regulación, permite la integración de nuevos cambios y la adaptación de los bots de charla para la extracción de Big Data. (Tóala et al., 2017)

1.2.2. Big Data en las empresas

El Big Data que las empresas obtienen a través de bots de charla, les permite el análisis de estos como herramienta estratégica para apoyar la toma de decisiones empresariales. Permite predecir futuros comportamientos o tendencias en las que pudieran prevenir situaciones negativas para las organizaciones. Casi el 80% de las empresas internacionales utiliza BD con esta finalidad. (John Walker, 2014)

Aplicando estas nuevas herramientas tecnológicas se consigue agilizar los sistemas de información, lo que permite a las empresas ser más ágiles en la toma de decisiones y prever la demanda de manera más ajustada, contando con una mayor capacidad de reacción, utilizándose en disímiles fines:

- **Conocimiento de los clientes.** Los datos se utilizan para estudiar el comportamiento y preferencia de sus clientes, enfocándose en el comportamiento de estos con el fin de predecir sus futuras acciones y a partir de estas ofrecerles posibles soluciones.
- **Recursos Humanos.** Se trazan estrategias que permiten la gestión del talento basado en métricas, a través de un enfoque analítico de estadísticas de desempeño. Además posibilitan que esta sea más objetiva y rigurosa, permitiendo que la toma de decisiones sea más acertada en base a los resultados de los análisis con una mayor alineación con los objetivos del negocio.
- **Operaciones Administrativas.** Destaca el empleo de BD en esta área en la elección de posibles proveedores, realizar consultas y registros sobre el valor del inventario e incrementar la calidad de las entregas.

1.2.3. Big Data en los gobiernos

El empleo del bots de charla para la extracción de BD no está limitado al sector empresarial. Los gobiernos han encontrado vías en las que puede emplear estos bots y enfocarlos para la recopilación de datos, y al igual que a las empresas, servirles como apoyo en la toma de decisiones. La recopilación de datos históricos de sucesos que afectarían la nación, puede emplearse en la predicción de desastres naturales, anticipación de la tasa de desempleo, alertas tempranas sobre focos epidémicos, prevención de delitos e influenciar directamente en la medicina.

Big Data en la política

Actualmente existen gobiernos que, por políticas y regulaciones internas tienen acceso a la gestión, análisis y manipulación de los datos almacenados mediante la interacción de los usuarios con bots conversacionales. Estos gobiernos pueden impedir a otros estados el acceso a los datos. Afectando indirectamente el sector económico y financiero, bloqueando el análisis y técnicas de estudio de BD. Que permitan reconocer a sus clientes y sus comportamientos en el mercado. Asimismo emplean estos datos para influenciar interna y externamente la política en otras naciones, evidenciándose en procesos electorales y subversión de la población. (CRAWFORD, 2014)

Un ejemplo de esto se evidencia en cómo en el gobierno estadounidense ha empleado en repetidas ocasiones la interacción de los usuarios cubanos en aplicaciones de mensajería. Con el propósito de atraerlos a grupos sensacionalistas y bombardearlos con falsas propagandas. Teniendo como objetivo

generar descontento e incertidumbre en las masas populares de un país, que se le imposibilita el acceso a los volúmenes de datos generados por sus propios usuarios.

1.3. Infraestructura de los bots de charla

La ejecución y funcionamiento de los chatbots están dados según el tipo de bot y canal de mensajería en el que se emplearía respectivamente, los que pueden ser modelos de simples diálogos en los que el usuario/cliente interactúa con el bot. O en otros casos, complejos sistemas que integran tecnologías de inteligencia artificial (AI, por sus siglas en inglés, *Artificial Intelligence*) capaces de elaborar complejas respuestas a las preguntas de los usuarios.

1.3.1. Modelo en base de diálogos

Es un mecanismo simple en el que el chatbot ofrece al usuario un menú de opciones, por las que puede interactuar con el programa. Por lo general está diseñado con el fin de realizar consultas predefinidas por los desarrolladores del bot a bases de datos o sitios en línea de noticias. Usualmente los tipos de bots que emplean este modelo son los enfocados al entretenimiento como juegos y preguntas al azar. Ya que este modelo permite establecer un sistema de preguntas y respuestas para cada interacción, abstrayéndose de búsquedas complejas. (Wen et al., 2016)

Los bots desarrollados bajo este modelo carecen de complejidad durante las etapas de diseño e implementación de los mismos, requiriendo un menor entrenamiento. Análogamente son incapaces de procesar diálogos confusos y elaborar respuestas haciendo uso de un lenguaje natural. Esto se debe a la carencia de técnicas de AI durante su elaboración. Actualmente estos son los de mayor existencia y uso en los canales de mensajería instantánea, cubriendo cada sector disponible al ofrecer servicios complementarios a los usuarios.

1.3.2. Modelo en base de inteligencia artificial

El proceso de elaboración de bots conversacionales definido bajo este modelo es complejo y requiere un mayor empleo de tiempo y recursos. Esto se debe a que en las etapas de diseño e implementación; se deben definir el alcance y límite del bot. Los bots que se desarrollan empleando técnicas de AI son capaces de analizar consultas complejas y al unísono arrojar respuestas que siguen una lógica y lenguaje natural de una conversación tradicional entre personas. (Hatwar, Patil y Gondane, 2016)

Durante las primeras etapas de construcción de bots con AI se analizan varios escenarios que dan forma a una conversación. Lo que permite elaborar un flujo de diálogo interactivo entre el programa y el usuario. Genéricamente estos bots se emplean para realizar búsquedas complejas desde el campo de entrada de

texto de la aplicación de mensajería. Las que pudieran estar asociadas sobre un negocio específico como reservas y ubicaciones o sobre búsquedas genéricas de contenidos e información.

El chatbot examina cada elemento de la petición realizada en búsqueda de intenciones y entidades en el cuerpo de la misma; permitiéndole establecer diálogos con la persona, personas u otros bots.

Intenciones: El sistema de inteligencia artificial, en sus primeros momentos, pretende identificar la "intención" de lo que el usuario ha querido decir. Por ejemplo, si se tratara de un chatbot destinado para realizar búsquedas de personalidades: los desarrolladores deben definir intenciones propias del negocio como #BuscarPersona.

Entidades: Son las entradas del usuario (palabras, categorías y frases) que determinarán la respuesta del chatbot, debido a que son fundamentales para ejecutar las búsquedas. Estas van asociadas a las intenciones. Por ejemplo, si se tratara de un bot que brinda servicios de búsqueda de información de personalidades, entonces para la intención #BuscarPersona, se podría definir la entidad @TipoBusqueda con valores (*"resumen", "detallado")*.

Diálogos: Este es la propia estructura de la conversación, se definen las posibles respuestas del chatbot cuando este identifica que es lo que el usuario ha querido decir (intención). De este modo y de forma visual se genera una conversación en modo de árbol, en el que se indica que respuesta debe arrojar el bot una vez que definida la intención. Es decir, si el usuario realiza una entrada *"quién fue Carlos J. Finlay"*, el motor de AI identificará la intención #BuscarPersona. En este punto, una vez configurado diálogo este debe responder *"¿Qué prefieres: resumen o detallado?"*; siguiendo el mismo algoritmo hasta concluir el objetivo.

Flujo de diálogo o *bot flow*

Se define como flujo de diálogo o *bot flow* a la secuencia lógica y finita de pasos que establecen las acciones de un bot conversacional con AI, en la que se analiza la entrada de un usuario como intención. Especifica la entidad dentro de los límites del programa y establece un diálogo en leguaje natural con el usuario. Para el diseño del flujo de diálogo se analiza la información que se pretende ofrecer. Son establecidos los límites, el comportamiento entre el usuario y el bot y que rol ocupará este, a partir de una estrecha relación determinada entre estos elementos. (Robalino y Javier 2017)

En dependencia del diseño del flujo de diálogo elaborado por los desarrolladores, y las respuestas arrojadas por el motor de inteligencia artificial. El bot podría realizar análisis en profundidad, a partir de la posible identidad a adoptada. Genera respuestas concretas y coherentes con un enfoque atractivo e ingenioso. Lo que logra crear una experiencia positiva y crearle al usuario la sensación de tener una conversación con un humano.

1.3.3. Motor de inteligencia artificial

Dentro de la inteligencia artificial, un motor de AI o *machine learning*[7], es la tecnología enfocada al desarrollo de algoritmos informáticos que aprenden de las experiencias de los usuarios; y posibilitan el desarrollo de programas informáticos que analizan reglas explícitas. Debido a esta tecnología de aprendizaje automático, las empresas son capaces de realizar actividades de segundo plano sin la necesidad de contratar personal dedicado al desempeño de estas. (Robert, 2014)

Redes neuronales

Los motores de AI están compuestos en esencia, por redes neuronales: una serie de algoritmos inspirados en el funcionamiento y organización del cerebro humano, las que necesitan ser entrenadas según su objetivo. El entrenamiento exitoso de las redes neuronales permite incrementar la percepción de estas y posibilitar el reconocimiento de una posible solución sin la interacción de una persona. (Lanzarini et al., 2015)

Actualmente existen modelos y mecanismos que vinculan las redes neuronales en motores de AI; por lo genera siguen el propósito de crear sistemas cognitivos capaces de reconocer el lenguaje natural. Una vez analizada e interpretada la petición del usuario, sustrae los datos e intenciones para la generación de una puntuación o *score*. Si el resultado de este umbral no es el deseado, el motor de AI ofrece la posibilidad de entrenarlo. El proceso de entrenamiento consiste en añadir y probar un conjunto de acciones bien definidas que apoyarán en la comprensión del usuario.

En una etapa siguiente, el motor de AI comprueba la entrada del usuario con cada intención definida durante el flujo y obtiene la de mayor *score.* Si se obtiene un *score* ganador significa que el motor de AI ha entendido la petición del usuario. Por otra parte, si se manifiesta el otro caso, el motor lo representa fuera del rango de las intenciones y pide al usuario a reformular su petición. Luego comienza nuevamente la espera de entrada del usuario, durante este proceso se pueden generar bucles infinitos. Como estrategia, en un segundo caso de *score* con valores inferiores, el bot conversacional, indicará al usuario sugerencias de intenciones, en las que dependiendo de la entidad seleccionada por este, el bot le redirigirá hacia una intención u otra.

[7] El **aprendizaje automático** o **aprendizaje de máquinas** (del inglés, "Machine Learning") es el campo de las ciencias de la computación y una rama de la inteligencia artificial cuyo objetivo es desarrollar técnicas que permitan a las computadoras *aprender*.

Para el desarrollo de la solución que se propone, se establece la implementación de una entidad conversacional basada en componentes de inteligencia artificial; que permita el cumplimiento exitoso de los estándares de calidad y empleo para las aplicaciones que brindan servicios de mensajería instantánea.

1.3.4. Estado de los bots a nivel internacional

En la actualidad se ha evidenciado el exponencial crecimiento del empleo de los bots conversacionales en redes sociales, empresas, organizaciones y estados gubernamentales; siempre enfocados a complementar los principales servicios que ofrecen. Los bots de charla permiten la recopilación de datos de los usuarios, reducir gastos de recursos y materiales. Por las características de estos pueden ser orientados a cubrir las áreas de intereses de los usuarios en la red de redes.

Basados en diálogos

La gran mayoría de aplicaciones de mensajería instantánea de mayor número de usuarios adoptan el uso de bots conversacionales basados en diálogos prediseñados. Es por esta razón que los servicios complementarios que ofrecen, no siempre resultan ser eficientes. Por lo general este modelo se orienta a soluciones simples para determinadas situaciones donde no es requerido un análisis complejo de las entradas del usuario. Por ejemplo Telegram y Facebook Messenger ofrecen varios bots basados en este modelo, destinados a la recopilación de noticias temáticas en línea de fuentes definidas por las aplicaciones. Lo que posibilita el acceso a la información estructurada desde la misma aplicación, a partir del empleo de bots que les permiten realizar estas acciones. Otro caso en el que este modelo es empleado, es en la selección de otros bots dentro de las aplicaciones, ya que puede generar un listado de opciones relacionadas a los intereses del usuario. (Muñiz Pasarín, 2018)

También destacan en dichos canales, siguiendo el modelo en base de diálogos, los bots de consultas del estado meteorológico, resultados de eventos deportivos y encuestas, preguntas al azar. Por lo general estas áreas de interés representan la mayor parte del objetivo al que son orientados. La consecuencia de esto se debe a la poca complejidad de las etapas de diseño, implementación e integración en los servicios de mensajería instantánea.

En términos de la experiencia de usuario UX del inglés *user experience*, los bots basados en diálogos, destacan entre los usuarios, debido a que no requieren complejos mecanismos de empleo. Ofrecen menús de navegación, que posibilitan que su desempeño sea protagonizado por las personas que los emplean. Como inconveniencia de esto se refleja que la calidad de los resultados no son siempre los esperados y el diálogo establecido entre el bot y el usuario resulta impersonal y limitado a las opciones predefinidas. En suma, estos efectos entorpecen la recopilación de información e impiden un mayor entendimiento de los clientes que emplean la aplicación. (Duan, 2014)

Basados en inteligencia artificial

En el presente año se ha reflejado un aumento de los bots de charla basados en AI, fundamentalmente en las empresas de mayor éxito y número de clientes, aplicaciones de mensajería y gobiernos. A pesar de la complejidad de las fases de diseño, implementación e integración que estos proponen; se han popularizado en las aplicaciones de mensajería instantáneas, ya que posibilitan a los usuarios la realización de búsquedas complejas con resultados de gran calidad. Como es el caso de CleverBot, el que emplea un motor de inteligencia artificial y procesamiento del lenguaje natural, que le permite establecer una comunicación fluida con humanos. Al mismo tiempo ofrece informaciones de búsquedas realizadas en segundo plano y se complementa como servicios en las aplicaciones de mensajería instantánea. (Ly Pichponreay et al., 2016)

El empleo de estos bots en las empresas está orientado esencialmente al subsistema de información. Permiten atender grandes masas de usuarios y al mismo tiempo crear perfiles informativos de estos, lo que permite que este proceso sea eficiente y productivo, debido que requiere menor personal y equipamiento. Un ejemplo de estos bots de atención al cliente es: AnswerBot; diseñado para ser personalizable según los requerimientos de la empresa. Admite acceder al sitio en línea de dicha empresa; a través de la extracción de la información publicada en el mismo y desvía la comunicación del cliente con un agente de soporte en caso de no entender la intención del usuario. (Nguyen 2017)

De igual manera algunos gobiernos emplean los bots de charla con inteligencia artificial en aplicaciones móviles para la recopilación y análisis de datos en la nación. Lo que les permite generar estadísticas e infografías del estado de la misma. Asimismo les es posible, a través de bots orientados a encuestas: obtener datos asociados a la satisfacción popular y apoyo de los ciudadanos al gobierno. (Steiner 2014)

Estos elementos de suma importancia pueden ser de apoyo en la toma de decisiones en los diferentes sectores en los que se asocia. Por ejemplo, en la economía: en marzo del 2016, Australia desplegó en el sitio oficial del gobierno el bot Alex. Este bot fue diseñado con el objetivo de responder las interrogantes de los ciudadanos relacionadas a los impuestos del estado. (Criterion et al., 2016)

Otro ejemplo es el caso del bot Textizen, desplegado como una aplicación móvil en el estado de Atlanta en los Estados Unidos, con el objetivo de ofrecer un medio informativo sobre los servicios de transportación del metro en esta región. (Krishnamurthy 2017)

Los ejemplos de bots de charlas citados, fueron implementados según la infraestructura definida por el modelo de inteligencia artificial, a partir de la restricción del acceso que establecen a las fuentes de información. Ya que acceden únicamente a fuentes vinculadas a servicios de pago. Por otra parte cuentan con la toda la infraestructura de servidores externos a las aplicaciones, por lo que su empleo se limita sobre

mecanismos de pagos por consultas. Debido a estas limitaciones se descartan como posibles soluciones al problema planteado.

1.3.5. Estado de los bots a nivel nacional

Hasta el presente año en Cuba no se habían implementado aplicaciones que brindaran servicios de mensajería instantánea. Lo que ha imposibilitado el empleo de bots conversacionales orientados a enriquecer los servicios de las aplicaciones y por consecuente, explotar los intereses, comportamientos y tendencias de sus usuarios. Sin embargo se consumen servicios de bots implementados por grupos y empresas internacionales; con el objetivo de publicar de manera automática y generar presencia en las redes sociales. Por ejemplo Cubadebate, un sitio web de noticias y actualidad en línea emplea con este propósito el bot phpSFP (*Schedule Facebook Posts* en español Publicación Programada en Facebook). Una aplicación gratuita y de licencia libre, mediante la cual pueden compartir en la red social Facebook los vínculos de los artículos publicados en el sitio de forma programada. Este bot, a pesar de no ser configurable en aplicaciones de mensajería instantánea, permite registrar los datos de interacción de los usuarios con las publicaciones. El bot cuenta con la infraestructura de los servidores de datos, externa e este por lo que el acceso institucional a la información, no es posible. (Litetech, 2017)

Otro bot que se emplea en sitios de publicaciones de noticias y actualidad en Cuba es BOTIZE, enfocado a aumentar la presencia en cuentas de la red social Twitter. El bot permite compartir en esta red social: enlaces, imágenes, videos y artículos. De uso gratuito, licencia libre y contando con posibilidades de adaptación a casi cualquier servicio web. Al igual que phpSFP, este bot es implementado y hospedado en servidores externos a las aplicaciones, por lo que la extracción de los datos recopilados es imposible, debido a políticas de estas aplicaciones y el bloqueo impuesto a Cuba que impide el acceso institucional de estos servicios. (Toledo et al., 2014)

1.4. Tecnologías a utilizar

Con el propósito de cumplir el objetivo planteado y desarrollar la propuesta de solución, se seleccionaron un conjunto de herramientas y tecnologías informáticas; las que se describen a continuación.

1.4.1. Metodologías de desarrollo de software

AUP-UCI

El Proceso Unificado Ágil o Agile Unified Process (AUP)[8] en inglés, en su variación para la UCI, define cuatro escenarios posibles, en los que la propuesta de solución puede enmarcarse. Permite su adaptación según

[8] El **Proceso Unificado Ágil** de Scott Ambler o **Agile Unified Process** (AUP) en inglés es una versión simplificada del Rational Unified Process (RUP). Este describe de una manera simple y fácil de entender la forma de desarrollar aplicaciones de software de negocio usando técnicas ágiles y conceptos que aún se mantienen válidos en RUP.

las características de cada proyecto (equipo de desarrollo, recursos, etc.). Por lo que asegura que el proceso de desarrollo sea configurable y aumentar la calidad del software que se produce al aplicar las buenas prácticas en el proceso de desarrollo. La variación para la UCI de la metodología AUP es flexible y puede ajustarse en dependencia del proyecto y el software a implementar; definiendo tres fases: Inicio, Ejecución y Cierre. Durante el inicio se realizan actividades relacionadas a la planeación, definición del alcance y estimaciones de tiempo, costo y esfuerzo. En la fase de ejecución, se modela el negocio, se obtienen los requisitos, se elabora la arquitectura y diseño; se implementa y libera el producto. En el cierre se analizan los resultados del proyecto y su ejecución y se realizan las actividades formales de cierre del proyecto. (Sánchez Rodríguez, 2012)

Durante la fase de ejecución en la disciplina Requisitos AUP-UCI, existen tres formas de encapsular los requisitos y, a partir del Modelo Conceptual, se definen los escenarios posibles. El desarrollo de la solución que se propone, se enmarca en el Escenario número 4 de la metodología antes definida. Debido a que el proyecto no es extenso, se muestra un alto vínculo entre el cliente y el desarrollador; y el negocio a informatizar está bien definido.

1.4.2. Herramientas CASE

Visual Paradigm

Es una aplicación de modelado que emplea el lenguaje unificado de modelado (**UML**, por sus siglas en inglés, *Unified Modeling Language*), esta herramienta soporta el modelado de todos los diagramas UML, además genera documentación del sistema en varios formatos como PDF, HTML y Word, asimismo permite la generación de código a partir de algunos diagramas. La que puede ser utilizada en el proceso de modelado de aplicaciones informáticas que sigan la filosofía de software libre. (Aysolmaz y Demirörs, 2014)

Mediante el empleo de esta aplicación es posible realizar los procesos de ingeniería tanto inversa como directa, ya que a partir de un modelo relacional es capaz de desplegar todas las clases asociadas a las tablas. Permite el control de versiones y es multiplataforma.

Por lo anteriormente descrito se decide utilizar Visual Paradigm como herramienta CASE para el modelado de la solución.

1.4.3. Protocolo de comunicación

XMPP

Protocolo extensible de mensajería y comunicación de presencia, en inglés *Extensible Messaging and Presence Protocol*, es un protocolo abierto y extensible basado en leguaje de enmarcado extensible (XML en inglés *Extensible Markup Language*). (Bray et al. 2006)

Fue originalmente desarrollado en la comunidad de código abierto Jabber con el fin de proveer una alternativa abierta y descentralizada ante los servicios existentes del momento. Representa un protocolo libre, abierto, público y de fácil entendimiento; además existen múltiples implementaciones de clientes, servidores, componentes y librerías. A través de la formalización de una arquitectura similar a la de correos electrónicos, permite implementar mecanismos de protección como el protocolo criptográfico TLS (Seguridad en la Capa de Transporte en inglés *Transport Layer Security*; asimismo, al emplear XML, es posible implementar funcionalidades personalizadas según se requiera. (Saint-Andre, 2005)

Se empleará el protocolo XMMP en el desarrollo de la solución por las características y funcionalidades antes descritas, además de ser la alternativa utilizada por la mayoría de los servicios de mensajería instantánea.

1.4.4. Lenguaje de programación

Python 2.7.5
Es un lenguaje de programación interpretado, multiplataforma y multiparadigma, ya que soporta la programación orientada objeto, imperativa y en menor medida la programación funcional. Hecho que fuerza a los desarrolladores a adoptar varios estilos de codificación. Posee una licencia de código abierto y está en constante evolución. Permite la resolución dinámica de nombres, es decir, que enlaza un método y un nombre de variable durante la ejecución del programa. En sus versiones 2 y 3, Python cuenta con disímiles librerías desarrolladas por la comunidad en línea de programadores, lo que permite el crecimiento y expansión de las soluciones implementadas en el lenguaje; así como abundante documentación. Estas bibliotecas tienen Python como lenguaje base, y pueden vincularse con protocolos, servicios y otros lenguajes, con el fin de facilitar el modelado de la aplicación. (SANNER, 2007)

Se empleará Python 2.7.5 para el desarrollo de la solución por las características anteriormente descritas y por la alta integración del protocolo XMPP con el lenguaje.

SleekXMPP
Es una librería bajo licencia de software libre permisiva desarrollada para Python 2.6/3.1+ y el protocolo de comunicación XMPP; esta permite el desarrollo de aplicaciones sobre el protocolo XMPP, como es el caso de bots y extensiones, facilitando la administración y experiencia de usuario en los servidores de aplicaciones de mensajería instantánea, su filosofía se basa en generar el menor número de dependencias posibles al contar con módulos terciarios incluidos en la librería. (Fritz y Stout, 2009)

Para el desarrollo de la solución se empleará la librería de Python SleekXMPP, debido a que en esta se definen las principales pautas de implementación de la solución.

1.4.5. Entorno de desarrollo de software

JetBrains PyCharm 2017.3.3 Edu
Es un entorno de desarrollo integrado (IDE por sus siglas en inglés *Integrated Development Environment*) de código abierto bajo la versión 2 de Apache, enfocado al lenguaje Python, permite el análisis del código a través de un intérprete del lenguaje y la integración como variable de entorno en el sistema operativo; ofrece características como el autocompletado, refactorización avanzada (renombramiento en todo el proyecto, extracción de métodos, introducción de variables, etc.); además cuenta con integración a los sistemas de control de versiones Git y Suvbersion. Es fácil de configurar y permite el desarrollo de aplicaciones con varias versiones de Python; se integra satisfactoriamente con las librerías del lenguaje. (Islam, 2015)

Para la implementación de la propuesta de solución se utilizará JetBrains PyCharm 2017.3.3 Edu, puesto que el IDE se adecua al lenguaje y la librería seleccionada, brindando la posibilidad de explotar las características definidas y simplificando en medida, parte de la etapa de implementación de la solución que se propone.

1.5. Conclusiones parciales

Como resultado de la investigación y el análisis bibliográfico realizado durante el desarrollo del presente capítulo, han sido expuestos los elementos fundamentales que componen los bots de charla en aplicaciones de mensajería instantánea. Se confeccionó un marco teórico que permite avanzar a la realización de la propuesta de solución que se integrará a servicios de mensajería instantánea, cumpliendo con los objetivos del presente trabajo. Se trataron los aspectos relacionados al desarrollo de bots de charla, técnicas de desarrollo y las principales tendencias de diseño e implementación. Se definieron las metodologías de desarrollo de software y las herramientas que se emplearán durante la implementación de la solución que se propone; teniendo como resultado las que se presentan a continuación:

- Metodología de desarrollo de software: Proceso Unificado Ágil – Variante para la UCI (AUP-UCI por sus siglas en inglés Agile Unified Process)
- Herramienta CASE: Visual Paradigm
- Lenguaje de programación: Python 2.7.5
- Entorno de desarrollo: JetBrains PyCharm 2017.3.3 Edu

Luego de la realización de un estudio sobre las herramientas y metodologías definidas anteriormente y el análisis de los elementos fundamentales que se abracarán en el presente trabajo; quedan definidas las bases para el desarrollo de una entidad conversacional, basada en elementos de inteligencia artificial para servicios de mensajería instantánea, como propuesta de solución.

Capítulo 2 : Propuesta de solución

En el presente capítulo se describen detalladamente las características con las que debe contar el bot de charla. Con el fin de garantizar su correcto funcionamiento e integración a los servicios de mensajería instantánea. Se fundamentará además de manera general el objeto de automatización, del que formará parte la propuesta de solución y se describe el modelo de dominio. Se presenta la propuesta de solución que podrá ser integrada en los servicios de mensajería instantánea. A partir de la descripción de los requisitos funcionales y no funcionales que debe cumplir el bot de charla. Se definen los componentes que interactúan en el mismo y las relaciones que existen entre ellos; se modelan y describen las historias de usuarios y los elementos descriptivos que las componen.

2.1. Análisis de la propuesta

Con el fin de concretar la propuesta de solución y darle respuesta al objetivo de la presente investigación se define como línea principal para la investigación:

- Desarrollar una entidad conversacional artificial especializada en responder instrucciones de los usuarios en los servicios de mensajería instantánea a partir de consultas en tiempo real a fuentes de información.

A partir de esta pauta de trabajo, el presente trabajo de diploma centrará los esfuerzos en lograr valorar el funcionamiento de la propuesta. La implementación de la entidad conversacional artificial especializada que propone la investigación para la integración a los servicios de mensajería instantánea.

2.2. Objeto de automatización

Como parte de la integración de servicios complementarios a las aplicaciones que ofrecen servicios de mensajería instantánea. Se desarrollará un bot de charla que permitirá automatizar los procesos de consultas a fuentes de información especializadas desde la aplicación. Los que lograrían la afinación de los clientes al ofrecerles estas facilidades desde la misma, sin necesidad de ejecutar terceras aplicaciones.

2.3. Propuesta del chatBot

Con el propósito de dar cumplimiento al problema planteado se propone el desarrollo de un bot de charla que sea posible integrarlo a los servicios de mensajería instantánea. Que permita el enriquecimiento de las funcionalidades de estas. Posibilite la realización de consultas especializadas a fuentes de información desde la misma aplicación y permita abstraer a los usuarios de este proceso.

El bot de charla propuesto en la presente investigación está desarrollado sobre el lenguaje de programación Python en su versión 2.7.5. A partir del empleo fundamental de la biblioteca SleekXMPP del propio lenguaje y el protocolo de comunicación XMPP. El bot será parte de los servicios complementarios que brindarían

las aplicaciones de mensajería instantánea y los usuarios interactuarán con este de manera simple, a través de instrucciones en formato de mensaje de texto. Durante el funcionamiento del mismo se establecerá una interacción con un almacén de intenciones en los servidores de las aplicaciones. En estos se registrarán las intenciones extraídas de las interacciones de los usuarios con el bot de charla, ya que representan elementos necesarios para el análisis y conformación de grandes volúmenes de información y Big Data.

El bot brindará a los usuarios respuestas en consecuencia de las intenciones detectadas. Si no es posible interpretar la instrucción recibida, el bot será capaz de realizarle al usuario una petición de reformulación de la instrucción enviada. Proceso que es iterando hasta detectar la intención de la consulta realizada. Una vez extraída la intención, se realiza la consulta y como respuesta, el bot envía al usuario en el cuerpo de un mensaje de texto el resultado encontrado. Si no es posible establecer la conexión con la fuente de información especializada, el bot enviará una notificación al usuario con dicha información.

2.4. Modelo de dominio

Al enmarcarse la propuesta de solución en el escenario cuatro de la variante para la UCI de AUP, no se realizará modelación del negocio. Debido a la respectiva simplicidad del entorno que caracteriza al sistema. El conocimiento que se posee acerca de su funcionamiento y la continua presencia del cliente durante el desarrollo de la misma, para convenir los detalles de los requisitos, y así poder implementarlos, probarlos y validarlos. Consecuentemente no es necesario realizar el modelo de negocio completo para comprender la problemática que ha de resolverse. Siendo suficiente la ejecución de un modelo de dominio o conceptual. (Sánchez Rodríguez, p. 13, 2012)

A través del modelo conceptual se representó visualmente los principales conceptos u objetos del mundo real, significativos para la construcción de un bot. Es de fundamental soporte para los desarrolladores y usuarios, ya que de esta forma se emplea un vocabulario común y se puede entender el contexto en que se enmarca. (García Peñalvo y García Holgado 2018)

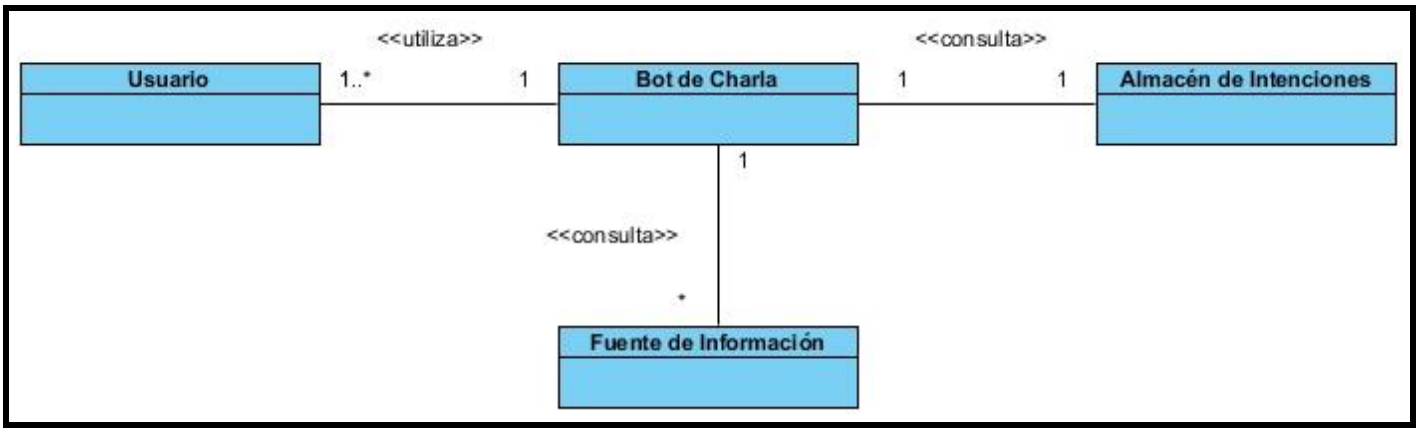

Imagen 2.1 Modelo de Dominio

Definición de las clases del Modelo de Dominio

Usuario: Persona que interactúa con el sistema, hace papel de actor en el mismo y es iniciador de las historias de usuarios.

Bot de Charla: Propuesta de entidad conversacional artificial para los servicios de mensajería instantánea.

Fuentes de Información: Son las fuentes de información especializada en línea sobre las que el bot debe realizar las consultas.

2.5. Especificación de requisitos

Se realizó una captura de los requerimientos funcionales de la propuesta de solución, con el propósito de describir las actividades que este debe realizar, ya que permitió describir el comportamiento o función particular cuando se cumplen ciertas condiciones. (Rodríguez et al., 2008)

A continuación se enumeran los que han sido capturados para el desarrollo de la presente investigación.

2.5.1. Requerimientos funcionales

- **RF1.** Recibir mensaje de consulta especializada
    - Obtener el mensaje de texto vía XMPP
    - Extraer la intención del cuerpo del mensaje
- **RF2.** Procesar intención
    - Verificar intención en la base de datos de intenciones
    - Verificar resultados de la consulta
    - Extraer resultados de la consulta
- **RF3.** Enviar respuesta de consulta

2.5.2. Requerimientos no funcionales

Se realizó una captura de los requerimientos no funcionales con el propósito de especificar los criterios para evaluar las operaciones del bot de charla; debido a que definen las características o cualidades generales que debe cumplir. (Serna-Montoya 2012)

A continuación se muestran separados por categorías.

Apariencia o interfaz externa:

- **RNF1.** Los mensajes y respuestas mostradas al usuario deben seguir una estructura y lenguaje natural.

Usabilidad

- **RNF2.** El bot de charla podrá ser utilizado de forma fácil por cualquier usuario.
- **RNF3.** Se emplea el idioma español para los mensajes y respuestas del bot.
- **RNF4.** El bot de charla debe enviar mensajes de texto a los usuarios definiendo su funcionamiento.

Accesibilidad

- **RNF5.** Las funcionalidades del bot estarán disponibles y los usuarios podrán acceder a ellas en todo momento.

Disponibilidad

- **RNF6.** El bot estará disponible dependiendo solo de la operatividad de la aplicación de mensajería instantánea.

Rendimiento

- **RNF7.** El bot de charla permitirá que múltiples usuarios lo empleen a la vez.
- **RNF8.** Los tiempos de respuesta y velocidad de procesamiento de las intenciones, no excederán los 10 segundos.

Legales

- **RNF9.** Las herramientas seleccionadas para el desarrollo de la propuesta de solución están respaldadas por licencias libres, bajo las condiciones de software libre. La herramienta Visual Paradigm se utiliza bajo la licencia que posee la Universidad de las Ciencias Informáticas.

Software

- **RNF10.** Los dispositivos deben contar con un cliente XMPP para la conexión con el servidor.

Soporte

- **RNF11.** Realizar pruebas y mantenimiento necesarios para lograr el mejoramiento y evolución en el tiempo.

2.6. Historias de usuarios

Se describieron los requerimientos funcionales mediante tres historias de usuarios (HU), ya que estas representan los requisitos o funcionalidades que la propuesta debe realizar. Aportan valor al usuario; a través de su empleo en la metodología definida para guiar el desarrollo de la propuesta de solución y se caracterizan por ser independientes entre estas, estimables y verificables. (Molina, 2014)

En estas se definieron lo que se debe construir en la solución; a partir de una prioridad definida por el cliente de manera que fue posible indicar cuales fueron las más importantes para el resultado final. Están divididas en tareas de ingenierías y el tiempo de ejecución fue estimado por los desarrolladores. El empleo de HU permitió a los desarrolladores estrechar la relación con el cliente por la inclusión de este en la fase de implementación, ya que facilitaron la discusión de los detalles de los requisitos. A cada HU se le asociaron pruebas que permiten la valoración del cumplimiento del requerimiento, lo que asegura su desarrollo exitoso en etapas posteriores. (Juan Quijano, 2009)

A continuación se definen los elementos que las conformaron:

- **Número:** Representa el identificador de la historia de usuario, con el fin de emplearlo como referencia de esta en la fase de validación.
- **Nombre del requisito:** Define el nombre del requisito que describe.
- **Programador:** Especifica el nombre y apellidos del programador que se encargará de desarrollar el requisito descrito.
- **Iteración asignada:** Enumera según la metodología la iteración en la que se encuentra la implementación del requisito.
- **Riesgo en desarrollo:** Define los riesgos que pudieran imposibilitar la implementación del requisito.
- **Tiempo estimado:** Establece el tiempo en días que se estimó para el desarrollo del requisito.
- **Tiempo real:** Especifica el tiempo en horas que tomó al programador, la implementación del requisito.
- **Descripción:** Realiza una descripción detallada de las condiciones que debe cumplir el requisito según exige el cliente en diferentes escenarios posibles.
- **Observaciones:** Se especifican los elementos bajo los que el requisito funcional debe ejecutarse.

Para la propuesta de solución no se describieron prototipos de interfaces, dado que la misma no los requiere, sino que describen el flujo de interacción del usuario con la entidad conversacional. Por otra parte, proporcionaron el detalle suficiente para que fuera posible la estimación de cuánto tiempo fue necesario para la implementación de las mismas. Estas difieren de los casos de uso, debido que estos son descritos por el cliente, no por los desarrolladores y emplean la terminología del cliente. (Adrián Anaya Villegas 2009)

A continuación se definen los elementos que complementan las HU:

- Tarjeta Clase – Responsabilidad - Colaboración: es en donde se define y recopila suficiente información para identificar y detallar la historia de usuario.
- Tareas de ingeniería: formalizan la conversación y el vínculo establecido entre el cliente y el desarrollador, con el fin de ampliar los detalles de la historia de usuario.

2.6.1. Descripción de historias de usuarios

En la presente investigación de definieron tres historias de usuarios, las que representan los requisitos funcionales críticos identificados para la solución propuesta. A continuación se presentan estas HU:

Tabla 2.1 HU 1: Recibir mensaje de consulta especializada

Número: HU 1	**Nombre del requisito:** Recibir mensaje de consulta especializada
Programador: Yosvani Vázquez Cruz	**Iteración asignada:** 1
Prioridad: Alta	**Tiempo estimado:** 72 horas
Riesgo en desarrollo: Fallo de conexión con el servidor	**Tiempo real:** 96 horas
Descripción: El usuario deberá enviarle un mensaje al bot mediante un cliente XMPP. Este debe realizar la extracción de la instrucción en el mensaje del usuario con el fin de generar la consulta especializada. Una vez hecho esto, el bot dará paso a la extracción de la intención mediante el análisis léxico del texto y empleo del lenguaje natural. Si el mensaje de texto tiene una estructura incoherente, o no se puede definir una intención. Se le enviará un mensaje al usuario requiriéndole que repita la instrucción según el patrón: “pregunta + objeto”; por ejemplo: “*¿Quién fue Carlos J. Finlay?*”	
Observaciones: El usuario deberá estar autenticado en la aplicación de mensajería instantánea.	

Tabla 2.2 HU2: Procesar Intención

Número: HU 2	**Nombre del requisito:** Procesar Intención
Programador: Yosvani Vázquez Cruz	**Iteración asignada:** 1
Prioridad: Alta	**Tiempo estimado:** 102 horas
Riesgo en desarrollo: Fallo de conexión con el servidor	**Tiempo real:** 90 horas
Descripción: Una vez extraída la intención del cuerpo del mensaje de texto. Se da paso al procesamiento de la misma. Primeramente se realiza una verificación en la base de datos de intenciones de la existencia de esta en la misma; de no ser así se agregaría dicha intención a la base de datos. Seguidamente se realiza una verificación de los resultados de la consulta en la caché de búsquedas en el servidor de la aplicación. De no encontrar posibles resultados, se realiza la extracción de la información requerida por el usuario en la intención. La extracción de los resultados se realiza mediante el análisis de las fuentes de información y los datos de la consulta enviados por el usuario. Si no es posible acceder a esta fuente de información especializada, se le enviará al usuario una notificación del error de acceso.	
Observaciones: NA	

Tabla 2.3 HU3: Enviar respuesta de consulta

Número: HU 3	**Nombre del requisito:** Enviar respuesta de consulta
Programador: Yosvani Vázquez Cruz	**Iteración asignada:** 2
Prioridad: Alta	**Tiempo estimado:** 28 horas
Riesgo en desarrollo: Fallo de conexión con el servidor	**Tiempo real:** 24 horas
Descripción: Una vez obtenida la respuesta de la consulta. Se transforma al lenguaje de maquetado extensible. A partir de la estructura de los paquetes de mensajería instantánea. Durante este proceso se le asigna un identificador a cada mensaje que es emitido. Lo que posibilita la interpretación de la aplicación y posteriores análisis. Luego se da paso al envío e interpretación del resultado de la consulta al usuario mediante un mensaje de texto por el mismo cliente del protocolo extensible de mensajería y envío de presencia.	
Observaciones: El usuario deberá estar autenticado en la aplicación de mensajería instantánea.	

2.6.2. Tarjetas CRC

Se definieron como elementos descriptivos de las historias de usuario dos Tarjetas CRC (Clase-Responsabilidad-Colaboración) ya que estos artefactos permiten expresar la interrelación entre las clases, las responsabilidades de estas y las otras clases que interactúan en la composición de la propuesta de solución. Están divididas por tres secciones que contienen las clases analizadas independientemente del resto, los objetivos de la clase o responsabilidades y las clases que colaboran en el cumplimiento de dichas responsabilidades. (Gómez, Duarte y Güevara, 2014)

A continuación se presentan las tarjetas CRC definidas en el diseño de la propuesta de solución:

Tabla 2.4 Tarjeta CRC 1: ChatBot

ChatBot	
- **Identificador de bot** - **Clave de autenticación** - **Mensaje** Autenticar el bot de charla en el servidor de aplicación. Extraer el cuerpo del mensaje enviado por el usuario. Extraer del cuerpo del mensaje la intención del usuario. Analizar la intención en búsqueda de entidades.	InfoManage

Tabla 2.5 Tarjeta CRC 2: InfoManage

InfoManage	
- **Intención** - **Entidad** Verificar si la intención ya está registrada en la base datos. Almacenar la intención y entidad en caso de inexistencia en la base de datos. Consultar la caché de los resultados de búsqueda en el servidor de la aplicación. Extraer contenido de la fuente de información (EcuRed).	ChatBot

Retornar resultados de la búsqueda. Retornar resultados alternativos de la consulta.	

2.6.3. Tareas de ingeniería

Se describieron ocho tareas de Ingeniería que permitieron representar un nivel descriptivo más profundo de las historias de usuarios. Están asociadas independientemente de las historias de usuario y definieron los detalles esenciales que las componen. (Bedregal y Raúl, 2017)

El empleo de este artefacto permitió agilizar los procesos de diseño e implementación, debido a que en estas se detallaron los siguientes elementos:

- **Número de tarea:** Se especifica un valor numérico que representa el identificador de la tarea.
- **Historia de usuario (Número y Nombre):** Representa la HU asociada a la tarea.
- **Nombre de tarea:** Define el nombre de la tarea.
- **Tipo de tarea:** Las tareas de ingeniería pueden ser de tipo: Desarrollo, Corrección, Mejora u otra especificación definida según las necesidades.
- **Puntos estimados:** Estos valores están dados según la estimación de tiempo realizada por el desarrollador; teniendo en cuenta la complejidad de la tarea y que cada punto representa el transcurso de una semana.
- **Fecha inicio:** Fecha en la que se comienza el desarrollo de la tarea.
- **Fecha fin:** Fecha en la que se culmina el desarrollo de la tarea.
- **Programador responsable:** Se especifica el nombre del programador encargado de su desarrollo.
- **Descripción:** Se describe formalmente los elementos que debe cumplir la tarea.

A continuación se presentan las tareas de ingeniería definidas en la presente investigación:

Tabla 2.6 Tarea de Ingeniería TI1: Autenticar bot de charla

Número de tarea: TI1	**Historia de Usuario:** HU 1 Recibir mensaje de consulta especializada.
Nombre de tarea: Autenticar bot de charla.	
Tipo de tarea: Desarrollo.	**Puntos estimados:** 0.2
Fecha inicio: 16/02/2018	**Fecha fin:** 19/02/2018
Programador responsable: Yosvani Vázquez Cruz	
Descripción: El bot de charla deberá conectarse al servidor de la aplicación de mensajería en forma de cliente, por lo que deberá autenticarse haciendo uso de las credenciales asignadas por la aplicación: jid: identificador (en inglés, *jabber identification*) y jpassword: contraseña (en inglés, *jabber password*).	

Tabla 2.7 Tarea de Ingeniería TI2: Extraer cuerpo del mensaje

Número de tarea: TI2	**Historia de usuario:** HU 1 Recibir mensaje de consulta especializada.
Nombre de tarea: Extraer cuerpo del mensaje.	
Tipo de tarea: Desarrollo.	**Puntos estimados:** 0.2
Fecha inicio: 20/02/2018	**Fecha fin:** 22/02/2018
Programador responsable: Yosvani Vázquez Cruz	
Descripción: El bot deberá extraer del mensaje de texto enviado por el usuario, el cuerpo del mismo siguiendo la codificación XML, específicamente la información enmarcada en las etiquetas <*message*> y <*from*>en donde se registra el texto enviado por el usuario y el identificador del usuario remitente.	

Tabla 2.8 Tarea de Ingeniería TI3: Extraer Intención del mensaje

Número de tarea: TI3	**Historia de Usuario:** HU 1 Recibir mensaje de consulta especializada.
Nombre de tarea: Extraer intención y entidad del mensaje.	
Tipo de tarea: Desarrollo.	**Puntos estimados:** 0.2
Fecha inicio: 23/02/2018	**Fecha fin:** 25/02/2018
Programador responsable: Yosvani Vázquez Cruz	

Descripción: Una vez extraído el contenido del cuerpo del mensaje, se extrae la intención y entidad o entidades del mismo, aplicando algoritmos de análisis léxico de textos y el lenguaje natural. Teniendo en cuenta que el contenido del mensaje puede ser incoherente, se realiza una petición de reformulación de la instrucción, en caso de no cumplir con los estándares definidos.

Tabla 2.9 Tarea de Ingeniería TI4: Verificar intención

Número de tarea: TI4	**Historia de Usuario:** HU 2 Procesar intención.
Nombre de tarea: Verificar intención.	
Tipo de tarea: Desarrollo.	**Puntos estimados:** 0.2
Fecha Inicio: 1/03/2018	**Fecha fin:** 3/03/2018
Programador responsable: Yosvani Vázquez Cruz	
Descripción: Se realiza una consulta a la base de datos de intenciones en búsqueda de existencia de la intención, si no está registrada se da paso al registro de la misma.	

Tabla 2.10 Tarea de Ingeniería TI5: Consultar caché de búsqueda

Número de tarea: TI5	**Historia de Usuario:** HU 2 Procesar intención
Nombre de tarea: Verificar intención	
Tipo de tarea: Desarrollo	**Puntos estimados:** 0.2
Fecha Inicio: 4/03/2018	**Fecha fin:** 6/03/2018
Programador responsable: Yosvani Vázquez Cruz	
Descripción: Se realiza una consulta a la caché de búsqueda en el servidor de la aplicación de mensajería, teniendo en cuenta la intención y entidad identificadas en la instrucción del usuario, en caso de existencia, se extrae el resultado de la consulta.	

Tabla 2.11 Tarea de Ingeniería TI6: Extraer contenido de fuente de información

Número de tarea: TI6	**Historia de Usuario:** HU 2 Procesar intención.
Nombre de tarea: Extraer contenido de fuente de información.	

Tipo de tarea: Desarrollo.	**Puntos estimados:** 0.3
Fecha Inicio: 7/03/2018	**Fecha Fin:** 10/03/2018
Programador Responsable: Yosvani Vázquez Cruz	
Descripción: Se establece la conexión a nivel de servidor, con la fuente de información (EcuRed) y se realiza la consulta al mismo teniendo en cuenta la intención y entidad extraída de la instrucción y se extraen los resultados de la consulta.	

Tabla 2.12 Tarea de Ingeniería TI7: Procesar resultados de consulta

Número de tarea: TI7	**Historia de Usuario:** HU 2 Procesar intención.
Nombre de tarea: Procesar resultados de consulta.	
Tipo de tarea: Desarrollo.	**Puntos estimados:** 0.3
Fecha inicio: 11/03/2018	**Fecha fin:** 14/03/2018
Programador responsable: Yosvani Vázquez Cruz	
Descripción: Los resultados de la consulta son almacenados en la caché de búsqueda del servidor de la aplicación de mensajería instantánea; posteriormente son analizados con el fin de detectar su naturaleza, es decir, si es directo, alternativo o inexistente.	

Tabla 2.13 Tarea de Ingeniería TI8: Enviar respuesta de consulta

Número de tarea: TI8	**Historia de Usuario:** HU 3 Enviar respuesta de consulta.
Nombre de tarea: Enviar respuesta de consulta.	
Tipo de tarea: Desarrollo.	**Puntos estimados:** 0.3
Fecha Inicio: 15/03/2018	**Fecha fin:** 18/03/2018
Programador responsable: Yosvani Vázquez Cruz	
Descripción: El resultado de la consulta a la fuente de información se transforma en el cuerpo de un mensaje de texto, teniendo en cuenta la codificación XML; y se retorna al usuario.	

2.7. Diseño

Durante el diseño de la propuesta de solución se especificaron detalles de los elementos que la componen según plantea la metodología de desarrollo de software empleada El análisis independiente de estos, permite profundizar en los aspectos que se deben tener en cuenta durante esta etapa.

2.7.1. Arquitectura del sistema

Se refirió como arquitectura de software a la estructuración del bot de charla que incluye los componentes principales del mismo. El comportamiento de esos componentes desde la visión del resto de la propuesta y la forma en la que estos interactúan y se coordinan para alcanzar la misión de la misma. La arquitectura de un sistema consiste en la vista conceptual de toda su estructura. (Humberto Cervantes, 2008)

Estilos arquitectónicos

Las soluciones de diseños arquitectónicos que son comunes y reusables a lo largo de los años de experiencia, se han agrupados en lo que se les llama actualmente como estilos. El éxito del diseño de la arquitectura de software depende de los estilos que se decidan utilizar para el desarrollo de la misma. Los estilos expresan la arquitectura en el sentido más formal y teórico, describen una clase de arquitectura, o piezas identificables de las arquitecturas empíricamente dadas. Una vez identificados los estilos es lógico y natural que sean reutilizables en situaciones donde se presenten elementos semejantes en el futuro. (César Julio Bustacara Medina, p. 11-13, 2015), (Reynoso y Kicillof, p. 4, 2004)

Durante el presente trabajo investigativo se decidió utilizar el estilo arquitectónico, basado en capas; por las facilidades que ofrece en la etapa de implementación. La posibilidad de separar cada componente de la solución que se propone en una capa independiente y la compatibilidad que ofrece el empleo de este estilo arquitectónico con los servicios de mensajería instantánea. (Juan Peláez, 2015)

Este estilo arquitectónico es del tipo llamada y retorno, en el mismo cada capa proporciona servicios a la capa superior y se sirve de las presentaciones que brinda las inferiores. Al dividir un sistema en capas, cada capa puede tratarse de forma independiente, sin determinar detalles de las demás. La estructuración de la propuesta en capas facilitó el diseño modular; en el que cada capa encapsuló un aspecto concreto del bot de charla y permitió además la construcción de un bot con componentes débilmente acoplados. Lo que expresa que si se minimizan las dependencias entre capas, resulta más fácil sustituir la implementación de una capa sin afectar al resto del sistema. (Eliazar Lopez, 2011)

Entre las variantes reconocidas de la arquitectura basada en capas, se decide utilizar la arquitectura basada en tres capas. Esta especialización es empleada fundamentalmente en aplicaciones que consumen

servicios de un servidor en línea desde un cliente. Ya que facilita la separación de la capa asociada a los datos de la de presentación al usuario.

Una arquitectura tres capas, consta con una capa superior que interactúa con una capa inferior mediante interfaces que definen las funcionalidades que debe brindar. Usualmente todas las capas no residen en un único ordenador, debido a la existencia de múltiples modelos donde reside la capa de presentación. Las capas de negocio y datos pueden residir en un mismo ordenador que funcione como servidor de la aplicación. Aunque si existe un crecimiento de las necesidades, es posible aislar las capas en más de un ordenador. De esta manera si el tamaño y la complejidad de la base de datos aumentan, se puede separar en varios ordenadores que recibirán las peticiones del ordenador en que resida la capa de negocio. Análogamente si fuese la complejidad de la capa de negocio la que obligase la separación, esta capa de negocio podría residir en uno o más ordenadores que realizarían solicitudes a una única base de datos. En sistemas de alta complejidad se llega a contar con una serie de ordenadores sobre los que corre la capa de datos, y otra serie de ordenadores sobre los cuales corre la base de datos. (García-Holgado y García-Peñalvo, 2014)

En la arquitectura tres capas utilizada en la presente investigación se definieron las capas siguientes: presentación (cliente XMPP), lógica de negocio o dominio (tareas y reglas que rigen el proceso), acceso a datos o gestión de datos (mecanismos de almacenamiento persistentes).

Cada una de las que encapsula los siguientes elementos:

- **Capa de presentación**: Es la que permite la interacción del usuario con el bot de charla, posibilita la captura de información introducida por este y realiza las peticiones a la capa inferior, ya que muestra al usuario la respuesta proveniente de ésta. Únicamente se comunica con la capa de negocio.
- **Capa de lógica de negocio**: Está conformada por los paquetes que integran el sistema; los que se ajustan a los requisitos funcionales arquitectónicamente significativos y a los requisitos no funcionales. Desde el punto de vista del diseño, esta capa es contenedora de las clases entidades y controladoras. Se comunica con la capa de acceso a datos y brinda información a la capa de presentación.
- **Capa de acceso a datos**: Contiene componentes que interactúan con las bases de datos y permiten, a través de la utilización de procedimientos almacenados y generados previamente. Realizan las operaciones con las bases de datos de forma transparente para la capa de negocio.

2.7.2. Patrones de diseño

Patrones GRASP

Entre los patrones de diseño más empleados en el proceso de desarrollo de software se encuentran los Patrones de Software para la Asignación general de Responsabilidades (GRASP, por sus siglas en inglés *General Responsibility Assignment Software Patterns*), los que se dividen en dos grupos, cinco fundamentales (Bajo Acoplamiento, Alta Cohesión, Experto, Creador y Controlador) y cuatro de apoyo (Polimorfismo, Fabricación Pura, In-dirección y No hables con extraños). En el presente epígrafe se analizarán los patrones que se espera que apoyen el proceso de diseño de la solución propuesta. (Mall, 2014)

Controlador

El patrón controlador se desempeña como intermediario entre una determinada interfaz y el algoritmo que la implementa, asigna la responsabilidad de controlar el flujo de eventos del sistema, a clases específicas. Esto facilita la centralización de actividades (validaciones, consultas, etc.), de tal forma que es la que recibe los datos del usuario y la que los envía a las distintas clases según el método llamado. El patrón se manifiesta en la solución propuesta, cuando la clase *ChatBot* recibe la instrucción enviada por el usuario en un mensaje a través de un cliente XMPP, extrae la intención del cuerpo del mensaje y la envía a la clase *InfoManage* la que se encarga del procesamiento de la información.

Alta cohesión

Este patrón se encarga de regular la coherencia de la información que manejan las clases; donde cada elemento del diseño debe realizar una labor única dentro del sistema. No desempeñada por el resto de los elementos y auto-identificable. La utilización del presente patrón asegura la creación de clases agrupadas por funcionalidades que permiten su simple reutilización. Durante el diseño de las clases se manifestó la presencia de este patrón pues existen clases definidas por la biblioteca *SleekXMPP* que se reutilizaron durante la implementación de la solución propuesta. (Escalante, 2016)

Bajo acoplamiento

El empleo de este patrón garantiza la creación de clases con mínimas dependencias entre ellas, lo que permite que en caso de modificación en alguna de ellas, se tenga la mínima repercusión posible en el resto de las clases. Efecto que potencia en conjunto con una alta cohesión, la reutilización de las clases en otros sistemas. En la propuesta de solución se evidencia el empleo del patrón por la mínima relación de las clases, definida en el diseño de las clases de la aplicación.

2.7.3. Diagrama de clases del diseño

En la fase de ejecución de la variante AUP-UCI se elaboraron análisis de los elementos significativos de la arquitectura hasta diseñar todos sus elementos. El diseño es el centro de atención al final de la fase de

ejecución y dictó el comienzo para la implementación del producto. Esto contribuye a una arquitectura estable, sólida y crea un plano para el modelo de implementación.

En el diseño se confeccionaron los diagramas de clases del diseño. Los elementos básicos que se pueden encontrar en este diagrama son las clases y las relaciones que existen entre las mismas. (Larman, p. 224, 2005)

A continuación se representa el diagrama de clases de diseño del sistema.

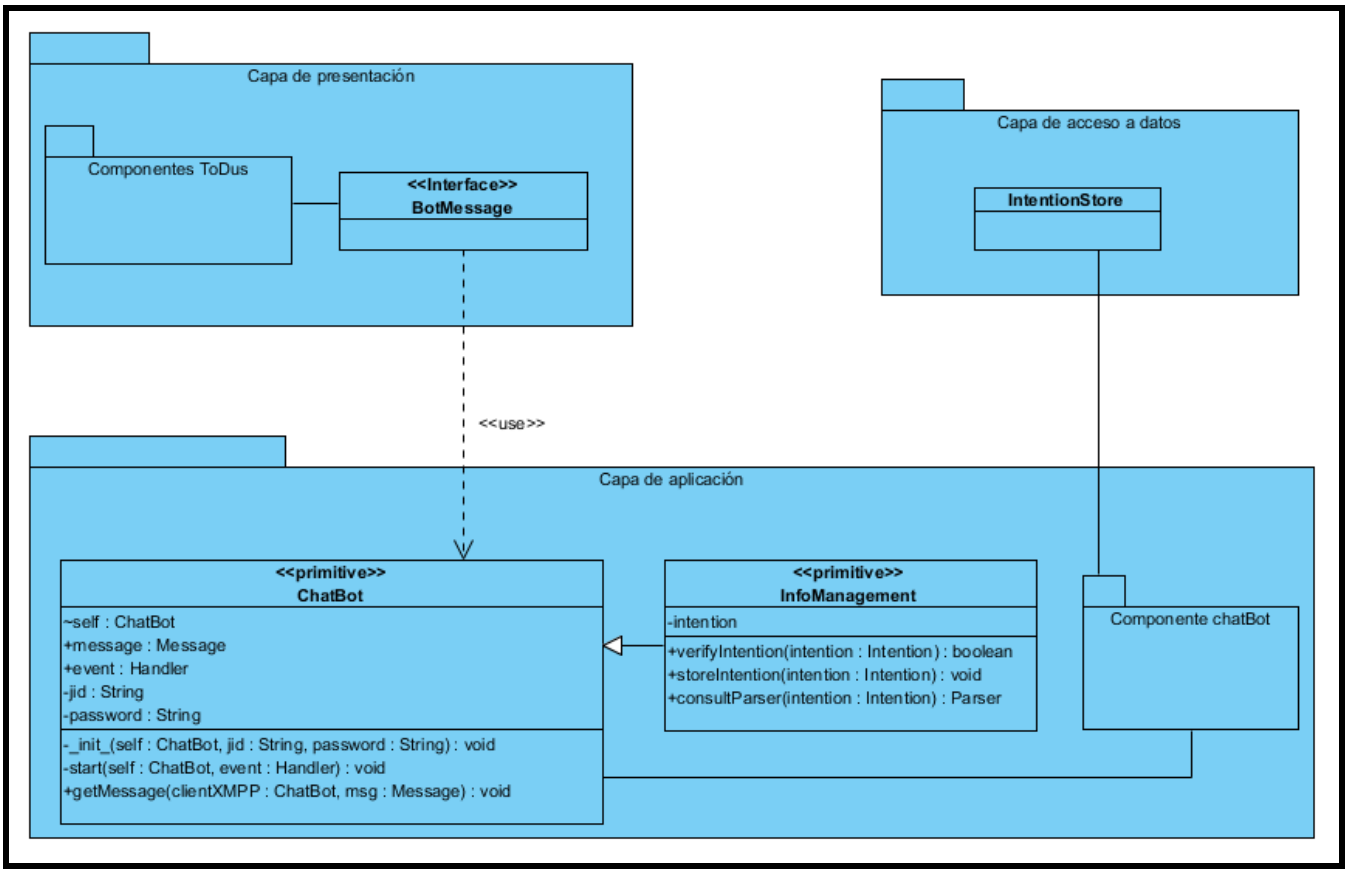

Imagen 2.2 Diagrama de clases del diseño

En el diagrama se muestran las relaciones existentes entre las clases o componentes de cada una de las capas. En la capa presentación se encuentra la interfaz de la aplicación de mensajería instantánea como cliente XMPP. La que cuenta entre las acciones que se muestran al usuario “Mensaje del Bot”, que inicia el flujo de diálogo con la entidad conversacional, es manejada por la clase “*ChatBot*”, la que se encarga de controlar el flujo de diálogo con el usuario, encapsulándose en la capa de negocio. Durante este proceso es necesario consultar la información contenida en el almacén de intenciones. Esto se realiza mediante la clase “*InfoManage*”; la que envía la solicitud de los datos al fichero de información ubicado en la capa de acceso a datos.

2.8. Implementación

Durante el diseño de la implementación de la solución que se propone se describieron elementos fundamentales tenidos en cuenta para la ejecución de esta fase y resultan de alto valor en análisis posteriores. Por otra parte, representan formalmente a las guías de implementación utilizadas en el desarrollo, ya que apoyan a la comprensión de la codificación.

2.8.1. Diagrama de despliegue

El diagrama de despliegue describe el despliegue físico de información generada por el software en los componentes de hardware. Muestra dónde y cómo se desplegará el sistema; las máquinas físicas y los procesadores se representan como nodos. La construcción interna puede ser representada por nodos o artefactos embebidos, en los que los estereotipos permiten precisar la naturaleza del equipo: dispositivos, procesadores y memoria. (Booch et al., 1999)

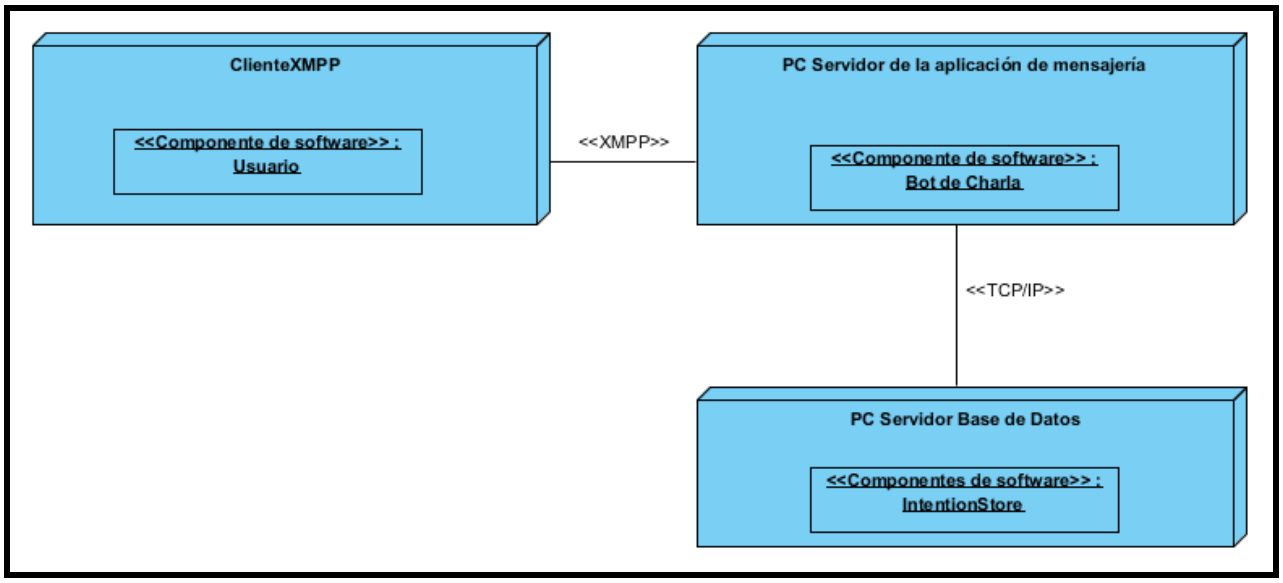

Imagen 2.3 Diagrama de despliegue

Descripción del diagrama de despliegue

El diagrama de despliegue representado que muestra la siguiente distribución:

- ClienteXMPP: Dispositivo cliente, capaz de conectarse al servidor de la aplicación de mensajería instantánea empleando un cliente de mensajería instantánea sobre el protocolo XMPP.
- PC Servidor de la aplicación de mensajería instantánea: Ordenador en que se encuentran los servidores de la aplicación de mensajería instantánea, donde se gestionará todo el flujo de diálogo

del bot de charla con el usuario. Estos servidores establecerán comunicación con los dispositivos clientes mediante el protocolo XMPP y con el servidor de base de datos por medio del protocolo TCP/IP.

- PC Servidor de Base de Datos: Ordenador en que se encuentra el almacén de intenciones gestionado por un gestor de base de datos, el que es capaz de mantener persistente la información almacenada y a utilizar por la aplicación. El mismo establece comunicación con los servidores de la aplicación de mensajería instantánea mediante el protocolo TCP/IP.

2.8.2. Diagrama de componentes

En el diagrama de componentes se representaron las dependencias entre los componentes de software, incluyendo componentes de código fuente, componentes del código binario, y componentes ejecutables. Dado que estos forman parte de la vista física de la propuesta, la que modela la estructura de implementación de la aplicación por sí misma y proporciona la oportunidad de establecer correspondencias entre las clases y los componentes de la implementación. (Rodríguez y Dorado, 2015)

En la fase de implementación de la solución propuesta se decide modelar las clases como parte de los componentes de la misma. Lo que permite una mayor interpretación de la estructura durante la implementación del bot de charla; a continuación se muestra el diagrama de componentes.

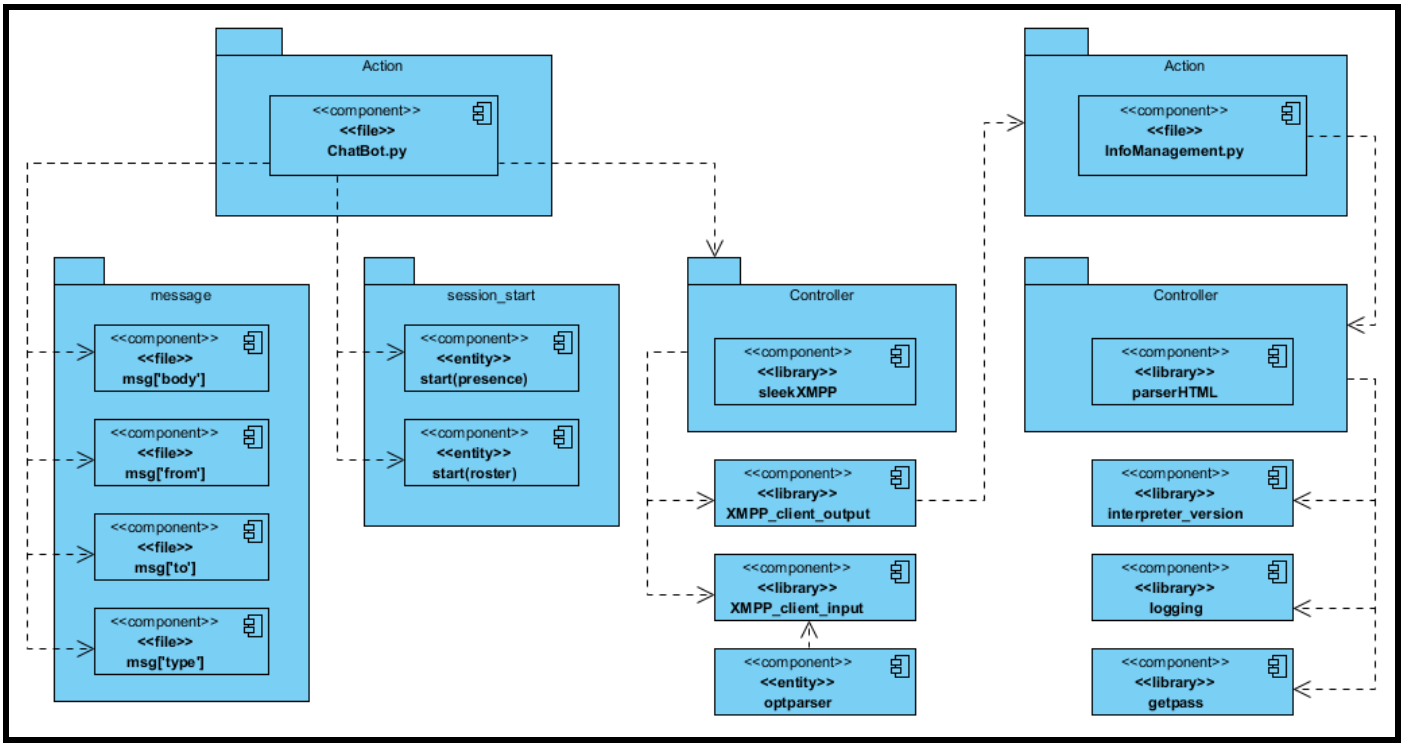

Imagen 2.4 Diagrama de componentes

2.8.3. Descripción de componentes

En el diagrama anterior se mostraron varios componentes que conforman la estructura del bot de charla. Debido al grado de aplicación en el desarrollo del mismo se hace preciso una descripción detallada de algunos de ellos; ya que son estos los que dan respuesta a los requisitos funcionales definidos previamente. Seguidamente se representa la descripción de los componentes *ChatBot.py* e *InfoManagement.py*, los que representan clases implementadas en el lenguaje Python 2.7 y que dan solución al proceso de control del flujo de diálogo.

Tabla 2.14 Descripción del componente ChatBot

Nombre: ChatBot	
Tipo de clase: Controladora	
Atributo	**Tipo**
self	ChatBot
jid	String
password	String
event	handler
message	Message
Para cada responsabilidad:	
Nombre:	__init__(ChatBot self, String jid, String password)
Descripción:	Inicializa los atributos de la clase y define los parámetros de acceso.
Nombre:	start(ChatBot self, handler event)
Descripción:	Inicia la conexión con el servidor de la aplicación de mensajería instantánea, obtiene la lista de contactos (roster) y envía el mensaje de estado (precense)
Nombre:	getMessage(ChatBot self, message msg)
Descripción:	Extrae de un mensaje, el cuerpo del mensaje, remitente, destinatario y tipo de mensaje. Analiza el léxico del cuerpo del mensaje y extrae la intención del mismo. Envía al remitente el resultado de la consulta de dicha intención.

	En la ejecución de este método se hace llamada a los métodos de la clase InfoManage

Tabla 2.15 Descripción del componente InfoManage

Nombre: InfoManage	
Tipo de clase: Controladora	
Atributo	**Tipo**
intention	Intention
Para cada responsabilidad:	
Nombre:	verifyIntention(Intention intention)
Descripción:	Consulta en la base de datos de intenciones, la existencia o no de la intención recibida.
Nombre:	ecuSearch(Parser self, String subject, Boolean Url)
Descripción:	Si el *url* es falso, realiza una consulta en EcuRed con el valor de la variable *subject* y retorna un objeto del tipo Parser
Nombre:	storeIntention(Intention intention)
Descripción:	Almacena en la base de datos de intenciones la intención recibida.
Nombre:	consultParser(Intention intention)
Descripción:	Consulta en la caché de búsqueda, los valores de la intención recibida. Realiza la búsqueda en las fuentes de información especializadas, relacionadas con el valor de la intención. Analiza gramaticalmente el resultado de la búsqueda y la envía a la clase ChatBot.
Nombre:	getSumary(Parser self, String find)
Descripción:	Retorna el valor del primer párrafo de la consulta en EcuRed
Nombre:	getAlternative(Parser self, String find)

Descripción:	Retorna el valor alternativo si existe de una consulta no exitosa en EcuRed
Nombre:	getResults(Parser self, String find)
Descripción:	Retorna los valores alternativos de posibles resultados de una consulta no exitosa en EcuRed

2.9. Pruebas

En el desarrollo de la propuesta de solución, el proceso de prueba fue clave a la hora de detectar errores o fallas, ya que representan cierto grado de importancia en la garantía de la propuesta. Una selección cuidadosa de los datos de prueba puede ofrecer mucha confianza en cuanto al desempeño que posee el bot. La selección de un determinado mecanismo de comprobación de errores, puede producir un software más confiable. El objetivo principal de la realización de pruebas es descubrir la mayor cantidad posible de defectos de la solución que se propone. (Cindy Campos Chiu, 2015)

2.9.1. Pruebas a aplicar

Con el propósito de valorar el funcionamiento de la propuesta de solución se decide emplear una estrategia de pruebas que permita comenzar la etapa de valoración desde el nivel más bajo de la aplicación, es decir, el código en sí. Para esto se decide implementar y ejecutar las pruebas unitarias, debido a que estas aseguran la comprobación del correcto funcionamiento de los métodos y funciones de forma aislada. (Paz y Andrés, 2016)

En una segunda etapa de la estrategia de valoración de la propuesta, se da paso al siguiente nivel de la valoración de aplicación. La comprobación de que la interacción y desempeño entre las unidades de software o componentes de esta, funcionan correctamente. Con este objetivo se decide implementar y aplicar pruebas de integración; ya que permiten la continuidad a la tercera etapa de la estrategia de pruebas definida: las pruebas funcionales.

En el presente capítulo se realizó la descripción de los requerimientos funcionales mediante historias de usuarios, según plantea el escenario cuatro de la variación AUP-UCI. Los artefactos definidos por la metodología valoran el correcto funcionamiento de los requisitos mediante el cumplimiento de las observaciones, lo que representa el elemento que detalla lo que debe hacer el requerimiento en la propuesta de solución. Un método de pruebas funcionales son las pruebas de caja negra, en el que se verifica la funcionalidad sin tomar en cuenta la estructura interna del código, detalles de implementación o escenarios de ejecución internos en el software. (Escobar-Sánchez y Fuertes-Díaz, 2015)

En la realización de las pruebas mediante caja negra, se empleará la técnica de partición de equivalencia, debido a que la misma permite examinar los valores válidos en inválidos de las entradas existentes en el software.

2.10. Conclusiones parciales

En el presente capítulo se realizó un estudio de la propuesta de solución que se realizará. Se definió un flujo de interacciones ajustado a una entidad conversacional basada en el modelo de inteligencia artificial que sea posible integrarla a los servicios de mensajería instantánea. Capaz de atender las consultas especializadas de los usuarios. Se mostraron los principales elementos que conforman el dominio del sistema para una mayor comprensión. Se analizaron sus características y funciones fundamentales, las que fueron representadas mediante tres Historias de Usuario. Se especificó sobre el diseño, la arquitectura en capas y como estilo de esta, la basada en tres capas. Como modelador de la estructura de la solución propuesta y se describió en cada capa de presentación, negocio y acceso a datos, cada una de las clases del diseño respectivamente. La captura y descripción de los requisitos funcionales y no funcionales, sentó las bases para la implementación del bot de charla propuesto como solución del problema. Durante esta fase se expuso el diagrama de despliegue y la descripción del mismo. Además se fundamentó en el diagrama de componente, las descripciones de las clases involucradas en el cumplimento de los requisitos funcionales descritos. Se definió la estrategia de pruebas a ejecutar y se mencionaron, las que se ejecutarían en cada etapa con el fin de valorar la solución que se propone: pruebas unitarias, pruebas de integración y pruebas de caja negra con la técnica de particiones de equivalencia. La culminación de estas acciones permitió definir las bases para la ejecución de la valoración de la solución que se propone.

Capítulo 3 : Implementación y realización de pruebas

En el presente capítulo se abordarán los detalles de las fases de implementación y prueba de la solución propuesta, partiendo de la conceptualización del análisis y diseño definidos en el capítulo anterior. Una vez concluida la fase de implementación se realizará el diseño y estudio de los diferentes casos de prueba a aplicar en la solución con el fin de asegurar el cumplimiento de los estándares de calidad y funcionamiento de la misma.

3.1. Implementación

La implementación de la propuesta de solución desarrollada se corresponde con la fase de ejecución definida por la metodología AUP-UCI, esta fase posibilita la realización de iteraciones y la obtención de resultados incrementales. (Sánchez Rodríguez, p. 13, 2012)

El escenario adoptado en el desarrollo de la solución que se propone, especifica que la realización de las disciplinas de requisitos, análisis y diseño, implementación y pruebas internas se desarrollen en tantas iteraciones como sean necesarias para el cumplimiento del objetivo; lo que permite la repetición del flujo de trabajo. De esta forma se asegura que se brinde un resultado más completo para un producto final de manera creciente, teniendo en cuenta que cada requisito debe desarrollarse completamente durante el transcurso de estas iteraciones.

3.1.1. Estándar de codificación

Como parte de la adopción de las buenas prácticas en la construcción de la solución planteada, se decide utilizar el estándar o estilo de programación de codificación del lenguaje Python: PEP 8 (en inglés, *Python Enhancement Proposal*). La utilización de este estándar asegura que el código exprese claramente el propósito del mismo y agilice el proceso de refactorización, que sea legible, entendible y que refleje un estilo armonioso. (Savarimuthu, 2016)

El estilo de programación PEP 8 define esencialmente:

- Empleo de cuatro espacios para indentar.
- Limitación de los tamaños de línea a 79 caracteres como máximo, por ejemplo:

```
self.send_message(mto=msg['from'],
                  mbody='No encontre exactamente ' + raw_string +
                        ', pero al parecer tiene que ver con esto: ',
                  mtype=msg['type'])
```

Imagen 3.1 PEP 8: Máximo de caracteres por línea

- Se debe separar las funciones de nivel superior y las clases con dos líneas en blanco, mientras que los métodos dentro de las clases se separan con una sola línea. Se permite la utilización de líneas en blanco dentro de las funciones para separar bloques que guardan cierta correlación lógica, por ejemplo:

```
raw_input = input


class EchoBot(sleekxmpp.ClientXMPP):
    def __init__(self, jid, password):
        sleekxmpp.ClientXMPP.__init__(self, jid, password)
        self.add_event_handler("session_start", self.start)
```

Imagen 3.2 PEP 8: Separación entre funciones de nivel superior y clases

- Las sentencias de *import* deben estar separadas una en cada línea, por ejemplo:

```
import datetime
import os
import unicodedata
from optparse import OptionParser
import sleekxmpp
from ChatDictionary import ChatDict
from InfoManage import Parser
from LogWriter import Writer
```

Imagen 3.3 PEP 8: Sentencias de import, separadas en líneas

- Utilizar espacios alrededor de los operadores aritméticos.
- No utilizar espacios alrededor del signo igual cuando se encuentre en un listado de argumentos de una función.
- No se deben incluir comentarios obvios.
- No se deben comparar booleanos mediante *"==*", por ejemplo:

```
if url is False:
        response = requests.get('https://www.ecured.cu/EcuRed:Enciclopedia_cubana/',
                                params={'search': subject},
                                headers={
```

Imagen 3.4 PEP 8: Comparación de valores booleanos

- Las clases deben nombrarse utilizando la convención *"CapWords"* (palabras que comienzan con mayúscula). Las clases para uso interno deben tener un guion bajo como prefijo, por ejemplo:

```
class EchoBot(sleekxmpp.ClientXMPP):
```

Imagen 3.5 PEP 8: Nombramiento "CapWords" para las clases.

- Las funciones deben nombrarse utilizando la convención *"lower_case"* (todo en minúscula).
- Siempre se utiliza *self*, para el primer argumento de los métodos de instancia, excepto para los métodos definidos como estáticos.

3.2. Pruebas de software

En el Capítulo 2: Propuesta de solución se mencionaron de manera general como parte del diseño, la estrategia de pruebas de software a aplicar a la propuesta de solución, en vista de valorar el funcionamiento de la misma. En el presente epígrafe se abordará este tema con profundidad, a partir de la descripción de cada uno de los tipos de pruebas a diseñar y ejecutar.

3.2.1. Pruebas unitarias

Las pruebas unitarias se aplican en el nivel más bajo de la aplicación, el de la programación, independiente de los módulos y funciones de las clases. Lo que permite medir las unidades de software independientemente. Estas pruebas se describen para comprobar si un método de una clase funciona correctamente de forma aislada.

Como características fundamentales, las pruebas unitarias corresponden a la visión de los desarrolladores, siendo estos los responsables de su elaboración y ejecución; ya que estos son los más conocedores del código y su correcto funcionamiento. Las dependencias complejas o interacciones con el exterior se gestionan realizando *stubs* (objetos que tienen un comportamiento programado ante ciertas llamadas de un *test* completo) o *mocks* (objetos ya programados, con los datos que se espera recibir). Como ventaja de aplicar estas pruebas, la ejecución de los casos de pruebas o *test* tardan menos en ejecutarse. Por lo que se tiende a lanzarlos más a menudo. Fuerzan la codificación de las clases menos acopladas, es decir, con menos dependencias entre sí, hecho que mejora el diseño del software.

Durante el proceso de aplicación de pruebas unitarias, el autor se enfocó en cómo funciona la unidad, no la interacción entre componentes. Ya que esto corresponde a las pruebas de integración.

El lenguaje de programación utilizado durante el desarrollo de la propuesta de solución *Python 2.7.5* incluye bibliotecas que posibilitan implementar pruebas unitarias durante la etapa de desarrollo. Ya que asegura la comprobación de las unidades de software desde etapas tempranas en la implementación. Las bibliotecas más utilizadas y con mayor documentación existente son: *doctest* y *unittest.* La implementación de las pruebas unitarias haciendo uso de la biblioteca *doctest* se realiza junto con la documentación de una función o clase, ejecutando todas la operaciones que se encuentran luego del bloque “>>>” y las compara con el resultado inmediatamente siguiente hasta otra operación o bien en un espacio en blanco. Esto resulta de gran comodidad ya que la documentación de la función o clase cumple con un objetivo dual: ilustrar con un ejemplo y servir como prueba unitaria. (Sale, 2014)

La biblioteca *unittest* por otra parte, permite implementar las pruebas unitarias en un fichero externo del lenguaje en forma de métodos. Estos componentes de prueba deben ser implementados bajo la condición

de ser nombrados primeramente con la palabra "*test_*". El empleo de esta biblioteca permite la implementación de tantos casos de pruebas unitarias como fueran necesarias y facilitando su análisis posterior. Por lo que se ajusta en gran medida lo requerido en el entorno en el que se encuentra el desarrollo de la solución que se propone. (Arbuckle, 2014)

Con el propósito de probar independientemente los métodos que componen el bot de charla, se desarrollaron y ejecutaron 12 pruebas unitarias y se empleó la biblioteca *unittest* del lenguaje *Python 2.7.5*.

A continuación se muestra una parte de los métodos implementados en la clase *TestClass*, ubicada en el fichero *Testing.py*. Con el propósito de realizar pruebas unitarias a las funciones críticas involucradas en el desempeño del bot de charla:

El método siguiente comprueba si el resultado de la búsqueda en EcuRed de la palabra "cuba" es equivalente a una búsqueda sin acceso a la enciclopedia:

```
def test__ecu__search(self):
    parser = Parser()
    soup = bs4.BeautifulSoup(open("search_cache/offline.html"), "html.parser")
    word = parser.ecu__search('cuba')
    self.assertEqual(soup,word)
```

Imagen 3.6 Prueba Unitaria para el método: ecu__search

El método siguiente comprueba que el resultado de la búsqueda en EcuRed de la palabra "cuba" contiene un valor real y distinto a "*404*", que representa error en la conexión:

```
def test__get_summary(self):
    parser = Parser()
    self.assertTrue(parser.get_summary('cuba'))
    self.assertNotEquals(parser.get_summary('cuba'),404)
```

Imagen 3.7 Prueba Unitaria para el método: get_summary

El método siguiente comprueba si el resultado de la búsqueda en EcuRed de la palabra "cuba" contiene valores alternativos y distintos al valor predeterminado del sistema: "*no_alternative*":

```
def test__get_alternative(self):
    parser = Parser()
    self.assertTrue(parser.get_alternative('cuba'))
    self.assertNotEquals(parser.get_alternative('cuba'),'no_alternative')
```

Imagen 3.8 Prueba Unitaria para el método: get_alternative

El método siguiente comprueba si el resultado de la búsqueda en EcuRed de la palabra "cuba" contiene resultados similares en otras páginas y distintos al valor predeterminado del sistema: "*no_links*":

```
def test__get_results(self):
    parser = Parser()
    self.assertTrue(parser.get_results('cuba'))
    self.assertNotEquals(parser.get_results('cuba'),'no_links')
```

Imagen 3.9 Prueba Unitaria para el método: get_results

```
def test__is_a_date(self):
    parser = Parser()
    date = parser.is_a_date('28 de enero')
    not_a_date = parser.is_a_date('28 enero')
    self.assertTrue(date)
    self.assertFalse(not_a_date)
```

Imagen 3.10 Prueba Unitaria para el método: is_a_date

El resto de los métodos implementados con el fin de realizar las pruebas unitarias se encuentran en la sección de anexos: Anexo 1: Pruebas unitarias

3.2.2. Pruebas de integración

Las pruebas integrales o pruebas de integración son aquellas que realizadas en el ámbito del desarrollo de la propuesta de solución una vez que se han ejecutados las pruebas unitarias. Consisten en realizar pruebas para valorar que la interacción de un gran conjunto de unidades de software funciona correctamente. (Shahabuddin y Prasanth 2016)

En las pruebas de integración, los módulos individuales del software son combinados en métodos y probados como un grupo. A partir de lo que se expone a continuación:

- Se deben ensamblar o integrar los módulos para formar el paquete de software completo.
- Las pruebas de integración se dirigen a todos los aspectos asociados con el doble de problema de verificación y construcción del programa.
- Las técnicas que más prevalecen son las de diseño de casos de prueba de caja negra.

De manera similar que las pruebas unitarias, descritas en el epígrafe anterior. Las pruebas de integración se pueden realizar a través de la biblioteca *unittest* incluida en el lenguaje *Python 2.7.5*, mediante el diseño e implementación de los casos de prueba. Con la exclusividad, de que durante el proceso debe ser probada la interacción entre dos o más unidades de software críticas que componen la propuesta de solución. (Vasilescu et al., 2014), (Arbuckle, 2010)

Con el propósito de comprobar la correcta integración entre las unidades de software críticas que componen el bot de charla, se ejecutaron y desarrollaron cuatro pruebas de integración, empleando la biblioteca *unittest* del lenguaje *Python 2.7.5*.

A continuación se muestra una parte de los métodos implementados en la clase *TestClass*, ubicada en el fichero *Testing.py*; con el propósito de realizar pruebas de integración entre las funciones críticas involucradas en la solución:

El método mostrado a continuación comprueba la integración entre los métodos encargados de proveer los resultados previamente definidos en el diccionario y el método encargado de realizar la lectura del diccionario; el mismo comprueba la lectura del método encargado de la escritura en la consola de *logs*[9]:

```
def test__casual_dict_integration(self):
    writer = Writer()
    dicto = ChatDict()
    self.assertTrue(dicto.casual_dict('@ayuda.'))
    self.assertEquals(dicto.casual_dict('@ayuda.'), '\n @ayuda. | Muestra '
                                                    'los comandos disponibles.'
                                                    '\n @msg. [Mensaje] | Envia '
                                                    'un mensaje al desarrollador.')
    self.assertTrue(writer.print_dict('dict_sms'))
    self.assertEquals(writer.print_dict('dict_sms'), "  > seems like I'm chatting using"
                                                     " the dictionary :D")
    self.assertEquals(writer.ending_with('user'), None)
```

Imagen 3.11 Prueba de Integración entre los métodos: dicto.casual_dict, writer.print_dict, writer.ending_with

El método siguiente comprueba la integración entre los métodos relacionados a la comprobación y extracción de los resultados del tipo fecha:

```
def test__date_results_integration(self):
    parser = Parser()
    self.assertTrue(parser.is_a_date('28 de enero'))
    date_results = parser.date_results('28 de enero')
    self.assertNotEquals(date_results, 'nop')
```

Imagen 3.12 Prueba de Integración entre los métodos: parser.is_a_date; parser.date_results

El resto de los métodos implementados con el fin de realizar las pruebas de integración se encuentran en la sección de anexos: Anexo 2: Pruebas de Integración

3.2.3. Pruebas funcionales

Pruebas de caja negra

Durante el proceso de aplicación de las pruebas de caja negra, el autor solamente se enfocó en las entradas y salidas del sistema. Sin la necesidad de recurrir al conocimiento de la estructura interna del programa de

[9] En informática, se usa el término ***log***, **historial de *log*** o **registro** a la grabación secuencial en un archivo o en una base de datos de todos los acontecimientos (eventos o acciones) que afectan a un proceso particular (aplicación, actividad de una red informática, etc.). De esta forma constituye una evidencia del comportamiento del sistema.

software. Como punto de partida se establecieron los requisitos de software y especificaciones funcionales, para la obtención en detalle de las entradas y salidas esperadas.

Las pruebas de caja negra, también denominadas por el ISTQB (en inglés, *International Software Testing Qualifications Boards*)[10] como técnicas basadas en especificaciones, son una forma de derivar y seleccionar condiciones, datos y casos de prueba. A partir de la documentación de los requerimientos del sistema; el presente caso se refiere a los artefactos previamente definidos que describen dichos requerimientos, los que serían: las Historias de Usuario, Tareas de ingeniería y Tarjetas CRC. (Graham, Van Veenendaal y Evans, 2008)

Estas pruebas no utilizan ninguna información interna de los componentes de software o sistemas que se van a probar. Sino que consideran el comportamiento del software, desde el punto de vista de un observador externo, es decir, como un usuario del sistema.

Técnicas de pruebas de caja negra

En los estándares definidos por el ISTQB para realizar las pruebas de software, las técnicas de prueba de caja negra son utilizadas para el guiar el desempeño de las pruebas funcionales. Basadas en las funciones y características del sistema y su integración con otros sistemas o componentes. Estas pueden ser utilizadas también con el fin del identificar las condiciones y casos de prueba a partir de la funcionalidad del software, como ocurre durante el empleo de la técnica de la Partición de equivalencias. (López, Marticorena y Martín, 2005)

Partición de equivalencia

Esta técnica describe los datos de entrada y los resultados de salida en clases diferentes. En las que todos los miembros de dicha clase están relacionados. Cada una de estas clases es una partición de equivalencia en la que el programa se comporta de la misma forma para cada miembro de la clase. Lo que supone que la prueba de un valor representativo de cada clase sea equivalente a la prueba de cualquier otro valor.

Los pasos ejecutados para el empleo de la técnica se muestran a continuación:

1. Identificar las clases de equivalencia
2. Definir los casos de prueba

En el primer paso se tomó cada condición de entrada o salida (generalmente una frase o sentencia en la especificación) y se dividió en uno o más grupos. Se identificaron dos tipos de clase de equivalencias:

[10] Es una asociación jurídica trans-sectorial y trans-nacional sin fines de lucro fundada por empresas, instituciones, organizaciones y personas especializadas en el campo del testeo y la industria del software.

válidas e inválidas. Lo permite que una cierta combinación de clases de entrada dará como resultado una combinación de clases de salida. (Bueno 2014)

En el segundo paso se definió para cada dato de entrada y salida, un valor perteneciente a una clase de equivalencia. Lo que permitió la conformación del caso de una prueba para cada dato de entrada y salida, es decir, el resultado esperado.

Esta técnica tiene las siguientes características fundamentales:

- Clasifica las entradas de datos del sistema en grupos que representan un comportamiento similar, por los que son procesadas de igual forma.
- Es posible definir particiones tanto para datos validos como no válidos.
- Las particiones también pueden definirse en función de las salidas de datos, valores internos, valores relacionados antes o después de ciertos eventos, y también para los valores que reciben las interfaces.
- Es posible aplicar la técnica a entradas de datos realizados por personas o vía interfaces con otros sistemas.

Lo esperado en el proceso de aplicación de las pruebas de caja negra mediante las particiones de equivalencia es realizar una cobertura completa. En la mayoría de los casos no es suficiente, por lo que es necesario combinarlas con pruebas de integración. Debido a que por muy satisfactorio que sea el funcionamiento de los datos de entrada y salida, pueden existir ocurrencias de defectos que no se están teniendo en cuenta. Estos defectos pueden no acarrear problemas a corto plazo, pero si pudieran afectar el desempeño del bot más adelante. (Tsui, Karam y Bernal, 2016)

Para comprobar la calidad de la funcionalidad del bot de charla desarrollado, se realizaron pruebas de caja negra, con el objetivo de demostrar que el mismo cumple con los requisitos previamente definidos. En la realización de esta prueba, la técnica empleada fue las particiones de equivalencias. Dado que permite examinar valores válidos e inválidos de las posibles entradas al bot de charla.

A continuación se describen los casos de prueba desarrollados con el objetivo de medir los requerimientos funcionales críticos especificados en las Historias de Usuario de la sección: 2.6.1. Descripción de las historias de usuario. Se especificó información de entrada, los resultados obtenidos una vez ejecutado el caso de prueba y el flujo central que debe cumplirse mientras este es ejecutado.

Tabla 3.1 Caso de Prueba: Recibir mensaje de consulta especializada

Escenario	**Descripción**	**Nombre**	**Respuesta del bot**	**Flujo central**
EC 1.1 Recibir mensaje de consulta especializada.	Se intenta recibir un mensaje de consulta especializada enviado al bot de charla.	**V** (¿Quien fue Carlos J. Finlay?)	Recibe el mensaje de consulta especializada enviado por el usuario.	Una vez recibido el mensaje de consulta especializada enviado por el usuario, se prepara el cuerpo del mensaje para realizar el resto de las operaciones.
EC 1.2 Recibir mensaje de consulta especializada. / Vacío	Se intenta recibir un mensaje de consulta especializada vacío, enviada al bot de charla.	**N** (Vacío)	Mensaje de consulta especializada no recibido	Una vez recibido el mensaje de consulta especializada enviado por el usuario, el bot de charla detecta que el mensaje está vacío y se envía un mensaje al usuario: "No puedo realizar una consulta vacía". El bot de charla queda en espera de nuevos mensajes.
EC 1.3 Recibir mensaje de consulta especializada. (Extraer intención)	Se intenta extraer la intención en el mensaje de consulta especializada enviado al bot de charla	**V** (¿Quien fue Carlos J. Finlay?)	Extrae la intención en el mensaje de consulta especializada recibido	Una vez recibido el mensaje de consulta especializada enviado por el usuario, el bot de charla, prepara el cuerpo de este y extrae la intención en el mismo
EC 1.4 Recibir mensaje de consulta especializada. (Extraer intención) / Vacío	Se intenta extraer la intención en el mensaje de consulta especializada vacío enviado al bot de charla	**N** (Vacío)	No se puede extraer la intención del mensaje de consulta especializada recibido	Una vez recibido el mensaje de consulta especializada enviado por el usuario, el bot de charla detecta que el mensaje está vacío, no se puede extraer la intención y se envía un mensaje al usuario: "No puedo realizar una consulta vacía". El bot de charla queda en espera de nuevos mensajes.

Tabla 3.2 Caso de Prueba: Procesar Intención

Escenario	Descripción	Nombre	Respuesta del bot	Flujo central
EC 1.1 Procesar intención.	Se intenta verificar que la intención existe en la base de datos de intenciones.	**V** (Carlos J. Finlay)	Verifica la existencia en la base de datos de intenciones, la intención extraída del cuerpo del mensaje de consulta especializada.	Una vez extraída la intención del cuerpo del mensaje se verifica que la intención existe en la base de datos de intenciones. Se verifica que exista un resultado para esa intención en la caché de búsqueda.
EC 1.2 Procesar intención. / (Intención no registrada)	Se intenta verificar que la intención no existe en la base de datos de intenciones.	**V** (Carlos J. Finlay)	Verifica la existencia en la base de datos de intenciones, la intención extraída del cuerpo del mensaje de consulta especializada.	Una vez extraída la intención del cuerpo del mensaje se verifica que la intención no existe en la base de datos de intenciones y se registra como una nueva intención. Se realiza la consulta a la fuente de información utilizando dicha intención. Se extraen los resultados de la consulta. Se almacena en la caché de búsqueda, el resultado extraído en la consulta.
EC 1.3 Procesar intención. / (Resultado no registrado en la cache de búsqueda)	Se intenta verificar que no existe un resultado asociado a la intención, registrado en la cache de búsqueda.	**V** (Pedro Kouri)	Verifica que no existe en la caché de búsqueda un resultado asociado a la intención registrada.	Una vez registrada la intención. Se verifica que no exista un resultado para esa intención en la caché de búsqueda. Se realiza la consulta a la fuente de información utilizando dicha intención. Se extraen los resultados de la consulta. Se almacena en la caché de búsqueda, el resultado extraído en la consulta.
EC 1.4 Procesar intención. / (Resultado no registrado en la caché de búsqueda, sin acceso a la fuente de información)	Se intenta verificar que no existe un resultado asociado a la intención, registrado en la cache de búsqueda y es imposible acceder a la fuente de información.	**V** (Pedro Kouri)	Verifica que no existe en la caché de búsqueda un resultado asociado a la intención registrada, y es imposible acceder a la fuente de información	Una vez registrada la intención. Se verifica que no exista un resultado para esa intención en la caché de búsqueda. No se puede acceder a fuente de información y le envía un mensaje al usuario: “En estos no tengo acceso a internet..."

Tabla 3.3 Caso de Prueba: Enviar respuesta de consulta

Escenario	Descripción	Nombre	Respuesta del bot	Flujo central

EC 1.1 Enviar respuesta de consulta.	Se intenta enviar al usuario el resultado de la consulta realizada.	**V** (Párrafo resumen de resultado)	Se envía al usuario el resultado de la consulta realizada.	Se procesa el resultado de la consulta en el lenguaje de maquetado extensible, asociándole un identificador al mensaje. Se envía el resultado al usuario.
EC 1.2 Enviar respuesta de consulta. / (Tipo de resultado: No encontrado)	Se intenta enviar al usuario un resultado no encontrado de la consulta realizada.	**V** (Mensaje informativo)	Se envía al usuario el resultado de la consulta realizada.	Se procesa el resultado no encontrado de la consulta en el lenguaje de maquetado extensible, asociándole un identificador. Se envía un mensaje al usuario: "No encontré nada referente a..."
EC 1.3 Enviar respuesta de consulta. / (Tipo de resultado: Fecha)	Se intenta enviar al usuario un resultado de la consulta realizada asociado a una fecha	**V** (Lista de acontecimientos)	Se envía al usuario el resultado de la consulta realizada.	Se procesa el resultado de la consulta asociado a una fecha en el lenguaje de maquetado extensible, asociándole un identificador. Se envía como resultado al usuario, una lista de acontecimientos.
EC 1.4 Enviar respuesta de consulta. / (Tipo de resultado: Desambiguación)	Se intenta enviar al usuario un resultado de la consulta realizada con artículos similares a la búsqueda.	**V** (Lista de artículos similares	Se envía al usuario el resultado de la consulta realizada.	Se procesa el resultado de la consulta asociado a artículos similares a la búsqueda, en el lenguaje de maquetado extensible, asociándole un identificador. Se envía como resultado al usuario, una lista de posibles referencias.
EC 1.5 Enviar respuesta de consulta. / (Tipo de resultado: Alternativo)	Se intenta enviar al usuario un resultado de la consulta realizada con un valor alternativo.	**V** (Valor alternativo)	Se envía al usuario el resultado de la consulta realizada.	Se procesa el resultado de la consulta asociado a un valor alternativo de la búsqueda, en el lenguaje de maquetado extensible, asociándole un identificador. Se envía como resultado al usuario un posible valor alternativo de la consulta.
EC 1.6 Enviar respuesta de consulta. / (Tipo de resultado: Artículos Relacionados)	Se intenta enviar al usuario un resultado de la consulta realizada con posibles artículos relacionados.	**V** (Lista de artículos relacionados)	Se envía al usuario el resultado de la consulta realizada.	Se procesa el resultado de la consulta asociado a artículos relacionados a la búsqueda, en el lenguaje de maquetado extensible, asociándole un identificador. Se envía como resultado al usuario una lista de artículos relacionados a la búsqueda.

Durante el proceso de ejecución de pruebas de caja negra a la propuesta de solución se aplicaron tres iteraciones con el objetivo de medir su correcta ejecución y cumplimiento de las funcionalidades descritas. Se obtuvo un total de seis no conformidades, las que se describen a continuación.

Iteración 1

Durante la ejecución de la primera iteración de pruebas se detectaron cuatro no conformidades, la que se describen a continuación:

Tabla 3.4 Resultados de la primera iteración de pruebas

Funcionalidad	Resultado	Satisfactorio / No satisfactorio
Recibir mensaje de consulta especializada. / (Extraer intención)	Se recibió un mensaje de consulta especializada, pero no se pudo extraer la intención.	No satisfactorio
Procesar intención. / (Registrar intención)	Se procesó la intención y fue registrada en el almacén de intenciones.	Satisfactorio
Procesar intención. / (Extraer resultado de fuente de información)	Se procesó la intención, se accedió y extrajo el resultado de la fuente de información.	Satisfactorio
Procesar intención. / (Registrar resultado en cache de búsqueda)	Se procesó la intención, y el resultado en la caché de búsqueda no fue posible.	No satisfactorio
Procesar intención. / (Extraer resultado en cache de búsqueda)	Se procesó la intención, y se extrajo el resultado en la caché de búsqueda.	Satisfactorio
Enviar respuesta de consulta. / (Procesar resultado no encontrado)	Se procesó el tipo de resultado No encontrado, y se envió la respuesta de la consulta al usuario.	Satisfactorio
Enviar respuesta de consulta. / (Procesar resultado de fecha)	Se procesó el tipo de resultado Fecha, y se envió la respuesta de la consulta al usuario.	Satisfactorio
Enviar respuesta de consulta. / (Procesar resultado de desambiguación)	No se pudo procesar el tipo de resultado Desambiguación, y no fue posible enviar la respuesta de la consulta al usuario.	No satisfactorio
Enviar respuesta de consulta. / (Procesar resultado alternativo)	Se procesó el tipo de resultado Alternativo, y se envió la respuesta de la consulta al usuario.	Satisfactorio

Enviar respuesta de consulta. / (Procesar resultado de artículos relacionados)	Se procesó el tipo de resultado Artículos relacionados, pero la respuesta de la consulta al usuario no pudo ser procesada para el envío.	No satisfactorio

Iteración 2

Durante la ejecución de la segunda iteración de pruebas fueron corregidas las no conformidades de la primera iteración; y se detectaron dos nuevas no conformidades, las que se describen a continuación:

Tabla 3.5 Resultados de la segunda iteración de pruebas

Funcionalidad	Resultado	Satisfactorio / No satisfactorio
Recibir mensaje de consulta especializada. / (Extraer intención)	Se recibió un mensaje de consulta especializada y la intención fue extraída correctamente.	Satisfactorio
Procesar intención. / (Registrar intención)	Se procesó la intención y fue registrada en el almacén de intenciones.	Satisfactorio
Procesar intención. / (Extraer resultado de fuente de información)	Se procesó la intención pero no fue posible el acceso a la fuente de información por lo que el resultado no pudo ser extraído.	No satisfactorio
Procesar intención. / (Registrar resultado en cache de búsqueda)	Se procesó la intención, y se registró el resultado en la caché de búsqueda.	Satisfactorio
Procesar intención. / (Extraer resultado en cache de búsqueda)	Se procesó la intención, y se extrajo el resultado en la caché de búsqueda.	Satisfactorio
Enviar respuesta de consulta. / (Procesar resultado no encontrado)	El tipo de resultado No encontrado, no pudo ser procesado correctamente y se envió una respuesta inválida al usuario.	No satisfactorio
Enviar respuesta de consulta. / (Procesar resultado de fecha)	Se procesó el tipo de resultado Fecha, y se envió la respuesta de la consulta al usuario.	Satisfactorio

Enviar respuesta de consulta. / (Procesar resultado de desambiguación)	Se procesó el tipo de resultado Desambiguación, y se envió la respuesta de la consulta al usuario.	Satisfactorio
Enviar respuesta de consulta. / (Procesar resultado alternativo)	Se procesó el tipo de resultado Alternativo, y se envió la respuesta de la consulta al usuario.	Satisfactorio
Enviar respuesta de consulta. / (Procesar resultado de artículos relacionados)	Se procesó el tipo de resultado Artículos relacionados, y se envió la respuesta de la consulta al usuario.	Satisfactorio

Iteración 3

Durante la ejecución de la tercera iteración de pruebas se corrigieron las no conformidades detectadas en la segunda iteración, cumpliendo correctamente con el desempeño de todas las funcionalidades especificadas para la solución que se propone; las que se describen a continuación:

Tabla 3.6 Resultados de la tercera iteración de pruebas

Funcionalidad	Resultado	Satisfactorio / No satisfactorio
Recibir mensaje de consulta especializada. / (Extraer intención)	Se recibió un mensaje de consulta especializada y la intención fue extraída correctamente.	Satisfactorio
Procesar intención. / (Registrar intención)	Se procesó la intención y fue registrada en el almacén de intenciones.	Satisfactorio
Procesar intención. / (Extraer resultado de fuente de información)	Se procesó la intención, se accedió y extrajo el resultado de la fuente de información.	Satisfactorio
Procesar intención. / (Registrar resultado en cache de búsqueda)	Se procesó la intención, y se registró el resultado en la caché de búsqueda.	Satisfactorio
Procesar intención. / (Extraer resultado en cache de búsqueda)	Se procesó la intención, y se extrajo el resultado en la caché de búsqueda.	Satisfactorio
Enviar respuesta de consulta. / (Procesar resultado no encontrado)	Se procesó el tipo de resultado No encontrado, y se envió la respuesta de la consulta al usuario.	Satisfactorio

Enviar respuesta de consulta. / (Procesar resultado de fecha)	Se procesó el tipo de resultado Fecha, y se envió la respuesta de la consulta al usuario.	Satisfactorio
Enviar respuesta de consulta. / (Procesar resultado de desambiguación)	Se procesó el tipo de resultado Desambiguación, y se envió la respuesta de la consulta al usuario.	Satisfactorio
Enviar respuesta de consulta. / (Procesar resultado alternativo)	Se procesó el tipo de resultado Alternativo, y se envió la respuesta de la consulta al usuario.	Satisfactorio
Enviar respuesta de consulta. / (Procesar resultado de artículos relacionados)	Se procesó el tipo de resultado Artículos relacionados, y se envió la respuesta de la consulta al usuario.	Satisfactorio

Las pruebas de caja negra fueron ejecutadas en un total de tres iteraciones. Se hizo uso de las particiones de equivalencia como técnica y apoyándose continuamente en los métodos que definen los casos pruebas de integración implementadas y ejecutas anteriormente. Durante este periodo se detectaron seis no conformidades, las que fueron corregidas en su totalidad en el transcurso de las iteraciones, valorando como correcto el cumplimento de los requerimientos funcionales de la aplicación.

3.3. Conclusiones parciales

En el presente capítulo se describieron los elementos fundamentales del estándar de codificación para servicios en Python: PEP 8. Que permitieron guiar el proceso de codificación de la propuesta de solución, lo que posibilitó la estandarización y adopción de las buenas prácticas en la etapa de implementación del bot de charla. Se tomó como punto de partida la especificación de las pruebas a realizar en la etapa de diseño. Se implementaron y ejecutaron 12 pruebas unitarias, con el propósito de medir independientemente, el desempeño de los métodos y funciones que componen las unidades de software del bot. Durante una segunda etapa se implementaron y ejecutaron cuatro casos de pruebas de integración que permitieron comprobar la interacción entre estas unidades de software. De esta forma se estableció el punto de partida para la tercera etapa de la estrategia de pruebas. En esta etapa se diseñó y ejecutó la comprobación de las funcionalidades empleando las pruebas de caja negra y las particiones de equivalencia como técnica. Se ejecutaron en un total de tres iteraciones donde se detectaron y corrigieron seis no conformidades. La ejecución de las pruebas en varios niveles de la aplicación permitió validar su correcto funcionamiento desde la codificación hasta la interacción entre los componentes; así como el cumplimiento de las historias de usuarios definidas en el Capítulo 2.

Conclusiones

En la presente investigación se caracterizaron las entidades conversacionales utilizadas en los sistemas de mensajería instantánea. Se identificaron sus tipos de infraestructura, funcionamiento y el estado de los mismos a nivel nacional e internacional, lo que permitió determinar las funcionalidades y procedimientos de la propuesta de solución.

Se diseñó e implementó una entidad conversacional capaz de recibir, analizar y procesar las interacciones de los usuarios; ofrecer respuestas automáticas en lenguaje natural y ofrecer un servicio complementario de consultas a fuentes externas de información.

La estrategia de pruebas utilizadas para evaluar la propuesta de solución permitió valorar de manera positiva su funcionamiento e integración en las aplicaciones que brindan servicios de mensajería.

Ofrecida como un servicio complementario fiable de los sistemas de mensajería instantánea, la propuesta de solución apoyará de forma determinante la afinación y fidelización de los usuarios que utilicen estos servicios.

Recomendaciones

Durante el desarrollo de la presente investigación se abordaron algunos temas que divergen en otros enfoques investigativos, por lo que fueron mencionados en su forma general con el propósito de continuar el desarrollo del objetivo planteado. Por este motivo se recomienda:

- Integrar a la propuesta de solución en futuras investigaciones, algoritmos que permitan realizar complejos análisis léxicos de las entradas de los usuarios. A partir de consultas a diccionarios en diversos lenguajes. Que permita la extracción exitosa de intenciones y entidades en las interacciones de los usuarios de mayor complejidad.
- Desarrollar e integrar métodos que permitan realizar búsquedas avanzadas en varias fuentes de información en cada iteración de los diálogos. Que permita asegurar a partir de un análisis de todos los resultados exitosos, el mejor y más completo de estos.
- Implementar un mecanismo a nivel de servidores en las aplicaciones que brindan servicios de mensajería instantánea, que permita una mejor interacción con la caché de resultados e intenciones exitosas detectadas por el bot de charla. Con el propósito de mejorar el rendimiento y tiempo de ejecución de las consultas realizadas.

Referencias

ADRIÁN ANAYA VILLEGAS, 2009. A propósito de programación extrema XP (eXtreme Programming) (página 2) - Monografias.com. [en línea]. [Consulta: 5 abril 2018]. Disponible en: http://www.monografias.com/trabajos51/programacion-extrema/programacion-extrema2.shtml.

ARBUCKLE, D., 2010. *Python Testing: Beginner's Guide*. S.l.: Packt Publishing Ltd.

ARBUCKLE, D., 2014. *Learning Python Testing*. S.l.: Packt Publishing Ltd. ISBN 978-1-78355-322-8.

AYSOLMAZ, B. y DEMIRÖRS, O., 2014. Unified Process Modeling with UPROM Tool. *Information Systems Engineering in Complex Environments* [en línea]. S.l.: Springer, Cham, Lecture Notes in Business Information Processing, pp. 250-266. [Consulta: 2 junio 2018]. ISBN 978-3-319-19269-7. Disponible en: https://link.springer.com/chapter/10.1007/978-3-319-19270-3_16.

BEDREGAL, V. y RAÚL, H., 2017. *Modelo para la estimación del esfuerzo de desarrollo en tareas de ingeniería de proyectos de software empleando aprendizaje automático* [en línea]. S.l.: Universidad de Granada. [Consulta: 2 junio 2018]. ISBN 978-84-9163-136-1. Disponible en: http://digibug.ugr.es/handle/10481/45264.

BEILBY, L.J., ZAKOS, J. y MCLAUGHLIN, G.A., 2014. Chatbots [en línea]. US8630961B2. [Consulta: 2 junio 2018]. US8630961B2. Disponible en: https://patents.google.com/patent/US8630961B2/en. US13143887

BOOCH, G., RUMBAUGH, J., JACOBSON, I., MARTÍNEZ, J.S. y MOLINA, J.J.G., 1999. *El lenguaje unificado de modelado*. S.l.: Addison Wesley Madrid.

BRAY, T., PAOLI, J., SPERBERG-MCQUEEN, C.M., MALER, E., YERGEAU, F. y COWAN, J., 2006. Extensible Markup Language (XML) 1.1 (Second Edition). , pp. 50.

BUENO, C.B., 2014. *Ingeniería del software II*. S.l.: Obtenido de http://ocw. unican. es/ensenanzas-tecnicas/ingenieria-del-softwareii/materiales/tema1-pruebasSistemasSoftware. pdf.

CÉSAR JULIO BUSTACARA MEDINA, 2015. Estilos Arquitectónicos. [en línea]. Pontificia Universidad Javeriana. [Consulta: 20 febrero 2018]. Disponible en: https://sophia.javeriana.edu.co/~cbustaca/docencia/DEAS-2015-03/presentaciones/Estilos_arquitecturales.pdf.

CINDY CAMPOS CHIU, 2015. *Las pruebas en el desarrollo de software* [en línea]. Investigacion. Mexico: UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO. [Consulta: 5 marzo 2018]. Disponible en: http://www.ptolomeo.unam.mx:8080/xmlui/bitstream/handle/132.248.52.100/7627/Las%20pruebas%20en%20el%20desarrollo%20de%20software.pdf?sequence=1.

CRAWFORD, K., 2014. Critiquing Big Data: Politics, Ethics, Epistemology | Special Section Introduction. , pp. 10.

CRITERION, J.F.J. is part of the marketing team at, CONTENT, S. in, IREL, social media O. from, TRAVELLER, S. an A. y ASIA, moved to S. after a year spent living out of a backpack in, 2016. Are virtual assistants the future of Public Sector customer service? *Criterion Conferences* [en línea]. [Consulta: 2 junio 2018]. Disponible en:

https://www.criterionconferences.com/blog/government/virtual-assistants-future-public-sector-customer-service/.

DEWEY, J., 2014. *Naturaleza humana y conducta: Introducción a la psicología social*. S.l.: Fondo de Cultura Economica. ISBN 978-607-16-2298-3.

DUAN, X., 2014. Chatbot system and method with enhanced user communication [en línea]. US20140122056A1. [Consulta: 2 junio 2018]. US20140122056A1. Disponible en: https://patents.google.com/patent/US20140122056A1/en. US13661042

ELIAZAR LOPEZ, 2011. Arquitectura de n capas. [en línea], [Consulta: 20 febrero 2018]. Disponible en: http://www.academia.edu/10102692/Arquitectura_de_n_capas.

ESCALANTE, L.C., 2016. Aplicación de Patrones de Diseño para Garantizar Alta Flexibilidad en el Software. *Tecnología & Desarrollo (Trujillo)*, vol. 12, no. 1, pp. 77–82.

ESCOBAR-SÁNCHEZ, M.E. y FUERTES-DÍAZ, W.M., 2015. Modelo formal de pruebas funcionales de software para alcanzar el Nivel de Madurez Integrado 2. , no. 39, pp. 31. ISSN 0121-1129.

FRITZ, N. y STOUT, L., 2009. *SleekXMPP*. S.l.: s.n.

GARCÍA PEÑALVO, J. y GARCÍA HOLGADO, A., 2018. Modelo de dominio. [en línea]. Universidad de Salamanca. [Consulta: 2 junio 2018]. Disponible en: https://repositorio.grial.eu/bitstream/grial/1153/1/8.%20Modelo%20de%20dominio.pdf.

GARCÍA-HOLGADO, A. y GARCÍA-PEÑALVO, F.J., 2014. *Patrón arquitectónico para la definición de ecosistemas de eLearning basados en desarrollos open source* [en línea]. S.l.: s.n. [Consulta: 2 junio 2018]. ISBN 978-84-16125-41-8. Disponible en: https://gredos.usal.es/jspui/handle/10366/125069.

GÓMEZ, A.R., DUARTE, A.Q. y GÜEVARA, C.D.M., 2014. Desarrollo ágil de software aplicando programación extrema. *Revista Ingenio UFPSO*, vol. 5, no. 1, pp. 24-29. ISSN 2389-864X.

GONZALES, H.M.S. y GONZÁLEZ, M.S., 2017. Los bots como servicio de noticias y de conectividad emocional con las audiencias. El caso de Politibot1. , pp. 22.

GRAHAM, D., VAN VEENENDAAL, E. y EVANS, I., 2008. *Foundations of software testing: ISTQB certification*. S.l.: Cengage Learning EMEA.

GREENWOOD, S., PERRIN, rew y DUGGAN, M., 2016. Social Media Update 2016. *Pew Research Center: Internet, Science & Tech* [en línea]. [Consulta: 2 junio 2018]. Disponible en: http://www.pewinternet.org/2016/11/11/social-media-update-2016/.

HATWAR, H., PATIL, A. y GONDANE, D., 2016. AI BASED CHATBOT. [en línea], vol. 3. [Consulta: 2 junio 2018]. ISSN 2349-6967. Disponible en: http://www.ijeebs.com/upload/pdf/5bhejhiN.PDF.

HUMBERTO CERVANTES, 2008. Arquitectura de Software | SG Buzz. [en línea]. [Consulta: 20 febrero 2018]. Disponible en: https://sg.com.mx/revista/27/arquitectura-software.

ISLAM, Q.N., 2015. *Mastering PyCharm*. S.l.: Packt Publishing Ltd. ISBN 978-1-78355-132-3.

JOHN WALKER, S., 2014. Big Data: A Revolution That Will Transform How We Live, Work, and Think. *International Journal of Advertising*, vol. 33, no. 1, pp. 181-183. ISSN 0265-0487, 1759-3948. DOI 10.2501/IJA-33-1-181-183.

JUAN PELÁEZ, 2015. Arquitectura basada en capas. – Blog de Juan Peláez en Geeks.ms. *Arquitectura basada en capas* [en línea]. [Consulta: 20 febrero 2018]. Disponible en: https://geeks.ms/jkpelaez/2009/05/30/arquitectura-basada-en-capas/.

JUAN QUIJANO, 2009. Historias de usuario, una forma natural de análisis funcional. [en línea]. [Consulta: 20 febrero 2018]. Disponible en: https://www.genbetadev.com/metodologias-de-programacion/historias-de-usuario-una-forma-natural-de-analisis-funcional.

KAR, R. y HALDAR, R., 2016. Applying Chatbots to the Internet of Things: Opportunities and Architectural Elements. En: arXiv: 1611.03799, *arXiv:1611.03799 [cs]* [en línea], [Consulta: 2 junio 2018]. DOI 10.14569/IJACSA.2016.071119. Disponible en: http://arxiv.org/abs/1611.03799.

KRISHNAMURTHY, K.C.D. and R., 2017. Chatbots move public sector toward artificial intelligence. *Brookings* [en línea]. [Consulta: 2 junio 2018]. Disponible en: https://www.brookings.edu/blog/techtank/2017/06/02/chatbots-move-public-sector-towards-artificial-intelligence/.

LANZARINI, L.C., HASPERUÉ, W., ESTREBOU, C.A., RONCHETTI, F., VILLA MONTE, A., AQUINO, G.O., QUIROGA, F., ROJAS, L. y JIMBO SANTANA, P., 2015. Redes neuronales artificiales. *XVII Workshop de Investigadores en Ciencias de la Computación (Salta, 2015)* [en línea]. S.l.: s.n., [Consulta: 2 junio 2018]. Disponible en: http://hdl.handle.net/10915/45598.

LARMAN, C., 2005. *UML y Patrones*. S.l.: Prentice Hall.

LITETECH, 2017. phpSFP - Schedule facebook posts. *CodeCanyon* [en línea]. [Consulta: 2 junio 2018]. Disponible en: https://codecanyon.net/item/phpsfp-schedule-facebook-posts/5177393.

LIU, M., NAVARRETE, C.C. y WIVAGG, J., 2014. Potentials of Mobile Technology for K-12 Education: An Investigation of iPod touch Use for English Language Learners in the United States. *Journal of Educational Technology & Society*, vol. 17, no. 2, pp. 115-126. ISSN 1176-3647.

LÓPEZ, C., MARTICORENA, R. y MARTÍN, D., 2005. Pruebas de caja negra: una experiencia real en laboratorio. ,

LY PICHPONREAY, JIN-HYUK KIM, CHI-HWAN CHOI, KYUNG-HEE LEE y WAN-SUP CHO, 2016. Smart answering Chatbot based on OCR and Overgenerating Transformations and Ranking. [en línea]. S.l.: IEEE, pp. 1002-1005. [Consulta: 2 junio 2018]. ISBN 978-1-4673-9991-3. DOI 10.1109/ICUFN.2016.7536948. Disponible en: http://ieeexplore.ieee.org/document/7536948/.

MALL, R., 2014. *Fundamentals of software engineering*. S.l.: PHI Learning Pvt. Ltd.

MASLOW, A., 2016. *El hombre autorrealizado: Hacia una psicología del Ser*. S.l.: Editorial Kairós. ISBN 978-84-7245-893-2.

MAXINEZ, D.G., RANGEL, F.J.S. y GARRIDO, L.C., 2010. ROBOTICA, ECONOMIA & EDUCACION. , pp. 10.

MOLINA, S.G.R., 2014. Metodologías ágiles enfocadas al modelado de requerimientos. *Informes Científicos - Técnicos UNPA*, vol. 5, no. 1, pp. 1-29. ISSN 1852-4516.

MONAR, L. y VICENTE, L., 2017. La comunicación asertiva en el comportamiento organizacional de los trabajadores de la Empresa Prodegel S.A del cantón Ambato, provincia de Tungurahua. [en línea], [Consulta: 2 junio 2018]. Disponible en: http://repositorio.uta.edu.ec/jspui/handle/123456789/26510.

MONTAG, C., BŁASZKIEWICZ, K., SARIYSKA, R., LACHMANN, B., ANDONE, I., TRENDAFILOV, B., EIBES, M. y MARKOWETZ, A., 2015. Smartphone usage in the 21st century: who is active on WhatsApp? *BMC Research Notes*, vol. 8, pp. 331. ISSN 1756-0500. DOI 10.1186/s13104-015-1280-z.

MUÑIZ PASARÍN, L., 2018. Bot de Telegram para la comunicación con PLCs de Beckhoff. [en línea], [Consulta: 2 junio 2018]. Disponible en: http://digibuo.uniovi.es/dspace/handle/10651/45577.

MUÑOZ, M.M., 2017. Privacidad y procesado automático de datos personales mediante aplicaciones y bots. *Dilemata*, no. 24, pp. 1-23. ISSN 1989-7022.

NGUYEN, M.-H., 2017. Customer service and virtual assistant bots will be prevalent for online businesses in many markets. *Business Insider* [en línea]. [Consulta: 2 junio 2018]. Disponible en: http://www.businessinsider.com/customer-service-chatbots-websites-2017-10.

OGHUMA, A.P., LIBAQUE-SAENZ, C.F., WONG, S.F. y CHANG, Y., 2016. An expectation-confirmation model of continuance intention to use mobile instant messaging. *Telematics and Informatics*, vol. 33, no. 1, pp. 34-47. ISSN 0736-5853. DOI 10.1016/j.tele.2015.05.006.

PAZ, M. y ANDRÉS, J., 2016. Análisis del proceso de pruebas de calidad de software. [en línea], [Consulta: 2 junio 2018]. Disponible en: http://repository.ucc.edu.co/handle/ucc/962.

PONCE, I., 2016. MONOGRÁFICO: Redes Sociales - Clasificación de redes sociales. [en línea], [Consulta: 2 junio 2018]. Disponible en: http://recursostic.educacion.es/observatorio/web/es/internet/web-20/1043-redes-sociales?start=3.

REYNOSO, C. y KICILLOF, N., 2004. *Estilos y Patrones en la Estrategia de Arquitectura de Microsoft*. S.l.: s.n.

ROBALINO, M. y JAVIER, A., 2017. Desarrollo de un prototipo de “BOT conversacional” empleando procesamiento de lenguaje natural. [en línea], [Consulta: 2 junio 2018]. Disponible en: http://repositorio.espe.edu.ec/jspui/handle/21000/13257.

ROBERT, C., 2014. Machine Learning, a Probabilistic Perspective. *CHANCE*, vol. 27, no. 2, pp. 62-63. ISSN 0933-2480. DOI 10.1080/09332480.2014.914768.

RODRÍGUEZ, C. y DORADO, R., 2015. ¿Por qué implementar Scrum? *Revista Ontare*, vol. 3, no. 1, pp. 125-144. ISSN 23823399. DOI 10.21158/23823399.v3.n1.2015.1253.

RODRÍGUEZ, P., YAGÜE, A., ALARCÓN, P.P. y GARBAJOSA, J., 2008. Metodologías Ágiles desde la Perspectiva de la Especificación de Requisitos Funcionales y No-Funcionales. *JISBD*. S.l.: s.n., pp. 333–344.

RODRÍGUEZ ROA, S., 2016. Estudio y uso de chatbots para el aprendizaje de inglés. [en línea]. [Consulta: 2 junio 2018]. Disponible en: http://oa.upm.es/43327/.

ROSAS, R.E., 2015. Redes sociales y pobreza: mitos y realidades. *Revista de Estudios de Género, La Ventana E-ISSN: 2448-7724*, vol. 1, no. 11, pp. 36-72. ISSN 24487724.

SAINT-ANDRE, P., 2005. Streaming XML with Jabber/XMPP. *IEEE Internet Computing*, vol. 9, no. 5, pp. 82-89. ISSN 1089-7801. DOI 10.1109/MIC.2005.110.

SALE, D., 2014. *Testing Python: Applying Unit Testing, TDD, BDD and Acceptance Testing*. S.l.: John Wiley & Sons.

SÁNCHEZ RODRÍGUEZ, T., 2012. *Metodología de desarrollo para la Actividad productiva de la UCI*. 2012. S.l.: s.n.

SANNER, M.F., 2007. PYTHON: A PROGRAMMING LANGUAGE FOR SOFTWARE INTEGRATION AND DEVELOPMENT. , pp. 7.

SAVARIMUTHU, T., 2016. PROCESS COMPLIANCE IN OPEN SOURCE SOFTWARE DEVELOPMENT – A STUDY OF PYTHON ENHANCEMENT PROPOSALS (PEPS). , pp. 18.

SERNA-MONTOYA, É., 2012. Estado actual de la investigación en requisitos no funcionales. *Ingeniería y Universidad*, vol. 16, no. 1, pp. 225-246. ISSN 0123-2126.

SHAHABUDDIN, S.M. y PRASANTH, Y., 2016. Integration Testing Prior to Unit Testing: A Paradigm Shift in Object Oriented Software Testing of Agile Software Engineering. *Indian Journal of Science and Technology* [en línea], vol. 9, no. 20. [Consulta: 2 junio 2018]. ISSN 0974 -5645. DOI 10.17485/ijst/2016/v9i20/91223. Disponible en: http://www.indjst.org/index.php/indjst/article/view/91223.

SIGNIFICADOS.COM, 2016. Significado de El hombre es un ser social por naturaleza. *Significados* [en línea]. [Consulta: 2 junio 2018]. Disponible en: http://www.significados.com/el-hombre-es-un-ser-social-por-naturaleza/.

STEINER, T., 2014. Bots vs. Wikipedians, Anons vs. Logged-ins. *Proceedings of the 23rd International Conference on World Wide Web* [en línea]. New York, NY, USA: ACM, pp. 547–548. [Consulta: 2 junio 2018]. ISBN 978-1-4503-2745-9. DOI 10.1145/2567948.2576948. Disponible en: http://doi.acm.org/10.1145/2567948.2576948.

TÓALA, C., ALEXANDER, M., INDIO, D. y TEODORO, J., 2017. Propuesta tecnológica de una página web con la implementación de Bots para la gestión de relaciones con el cliente en la empresa Vipcell Electrónics. [en línea], [Consulta: 2 junio 2018]. Disponible en: http://repositorio.ug.edu.ec/handle/redug/21898.

TOLEDO, E., COMBA, S., CARRERAS, M.I., DUYOS, L., RUCQ, J., FRANA BISANG, A., SCHOO LASTRA, S. y VINOCUR, E., 2014. *Comunicación, educación y TICs: manual de recursos para la enseñanza con herramientas digitales. Aprendiendo a enseñar con facebook, twitter y youtube* [en línea]. S.l.: Facultad de Ciencia Política y Relaciones Internacionales. Escuela de Comunicación Social. Secretaría de Extensión. [Consulta: 2 junio 2018]. ISBN 978-987-33-5104-4. Disponible en: http://rephip.unr.edu.ar/xmlui/handle/2133/3299.

TSUI, F., KARAM, O. y BERNAL, B., 2016. *Essentials of software engineering*. S.l.: Jones & Bartlett Learning.

VASILESCU, B., VAN SCHUYLENBURG, S., WULMS, J., SEREBRENIK, A. y VAN DEN BRAND, M.G.J., 2014. Continuous Integration in a Social-Coding World: Empirical Evidence from GitHub. [en línea]. S.l.: IEEE, pp. 401-405. [Consulta: 2 junio 2018]. ISBN 978-1-4799-6146-7. DOI 10.1109/ICSME.2014.62. Disponible en: http://ieeexplore.ieee.org/document/6976106/.

WEN, T.-H., VANDYKE, D., MRKSIC, N., GASIC, M., ROJAS-BARAHONA, L.M., SU, P.-H., ULTES, S. y YOUNG, S., 2016. A Network-based End-to-End Trainable Task-oriented Dialogue System. En: arXiv: 1604.04562, *arXiv:1604.04562 [cs, stat]* [en línea], [Consulta: 2 junio 2018]. Disponible en: http://arxiv.org/abs/1604.04562.

Anexos

Anexo 1: Pruebas unitarias

Para analizar descripción y contenido de este anexo revisar: *3.2.1. Pruebas unitarias*

El método siguiente comprueba si la entrada “28 de enero”, considerada por el bot de charla como una fecha contiene valores de fecha en el resultado y distintos del valor predeterminado del sistema “no”:

```
def test__date_results(self):
    parser = Parser()
    self.assertTrue(parser.date_results('28 de enero'))
    self.assertNotEquals(parser.date_results('28 de enero'),'no')
```

Imagen A.1 Prueba Unitaria para el método: date_results

El método siguiente comprueba si el resultado de la búsqueda en EcuRed de la palabra “cuba” contiene resultados diferentes en otros artículos y distintos al valor predeterminado del sistema “*no_related*”:

```
def test__related_references(self):
    parser = Parser()
    self.assertTrue(parser.related_references('cuba'))
    self.assertNotEquals(parser.related_references('cuba'), 'no related')
```

Imagen A.2 Prueba Unitaria para el método: related_references

El método siguiente comprueba si el fichero de caché de extensión “xml” existe y sigue la estructura definida por el sistema, también comprueba si es posible crearlo.

```
def test__build_xml_structure(self):
    parser = Parser()
    now = datetime.datetime.now()
    self.assertTrue(parser.build_xml_structure(now, 'search_cache', False))
    self.assertNotEquals(parser.build_xml_structure(now, 'search_cache', False), 'no built')
    self.assertEquals(parser.build_xml_structure(now, 'search_cache', False), 'built')
```

Imagen A.3 Prueba Unitaria para el método: build_xml_structure

El método siguiente comprueba si la palabra “cuba”, contiene valores de resultados de búsqueda en el fichero de caché y son distintos del valor predeterminado “nop”.

```
def test__search_in_cache(self):
    parser = Parser()
    self.assertTrue(parser.search_in_cache('search_cache','cuba'))
    self.assertNotEquals(parser.search_in_cache('search_cache','cuba'),'nop')
```

Imagen A.4 Prueba Unitaria para el método: search_in_cache

El método siguiente comprueba si la entrada “*@msg”* es un mensaje para el desarrollador, al mismo tiempo comprueba las posibles entradas relacionadas a la instrucción, dígase: *@ms., @mg.*; incluyendo la comprobación de una entrada que no cumple con la instrucción.

```
def test__is_msg(self):
    writer = Writer()
    self.assertEquals(writer.is_msg('@msg. '),'is_msg')
    self.assertEquals(writer.is_msg('@ms. '),'almost_msg')
    self.assertEquals(writer.is_msg('@mg. '),'almost_msg')
    self.assertEquals(writer.is_msg('msg'),'no_msg')
```

Imagen A.5 Prueba Unitaria para el método: is_msg

El método siguiente comprueba si la función *log_sms_dict* recibe una entrada del tipo *'showing_sms'*, retorna el valor correspondiente según el diccionario previamente establecido.

```
def test__log_sms_dict(self):
    writer = Writer()
    self.assertTrue(writer.log_sms_dict('showing_sms'))
    self.assertEquals(writer.log_sms_dict('showing_sms'),"   > showing results found...")
```

Imagen A.6 Prueba Unitaria para el método: log_sms_dict

El método siguiente comprueba si la ruta para el almacenamiento de los ficheros de caché es correcta y existe.

```
def test__log_path(self):
    writer = Writer()
    self.assertTrue(writer.log_path('search_cache'))
```

Imagen A.7 Prueba Unitaria para el método: log_path

Anexo 2: Pruebas de integración

Para analizar descripción y contenido de este anexo revisar: *3.2.2. Pruebas de integración*

El siguiente método comprueba la integración entre los métodos encargados de analizar y devolver la existencia en caché de los resultados, además comprueba la interacción entre estos métodos y los de escritura en la consola de *logs*.

```
def test__search_in_cache_integration(self):
    writer = Writer()
    parser = Parser()
    # Found in cache situation
    self.assertTrue(parser.search_in_cache('search_cache', 'cuba'))
    self.assertEquals(writer.found_cache_sms('cuba'), None)
    # Not found in cache situation
    self.assertEquals(writer.not_cache_sms('cuba'), None)
    self.assertTrue(writer.print_dict('net_issue'))
```

Imagen A.8 Prueba de Integración entre los métodos: parser.search_in_cache; writer.found_cache_sms…

El método que se muestra a continuación constituye la prueba de integración de mayor importancia; en el presente, se comprueba la interacción entre los métodos involucrados en la extracción de los resultados del tipo: sumario, alternativa, lista de posibles resultados y los métodos de escritura de *logs* asociados:

```
def test__all_result_integrations(self):
    writer = Writer()
    parser = Parser()
    summary = parser.get_summary('cuba')
    results = parser.get_results('cuba')
    alternative = parser.get_alternative('cuba')
    if summary == 'no_sumary' and alternative == 'no_alternative' and results == 'no_links':
        self.assertTrue(writer.empty_element('cuba'))
    elif summary == 'no_sumary':
        if alternative == 'no_alternative':
            self.assertTrue(parser.print_list(results))
        elif results == 'no_links':
            self.assertTrue(writer.with_alt_element_sms('cuba'))
        else:
            self.assertTrue(writer.with_alt_link_element_sms('cuba'))
            self.assertTrue(parser.print_list(results))
    else:
        self.assertTrue(writer.print_dict('summary_result'))
        if parser.search_in_cache('search_cache', 'cuba') is "nop" or summary == "404":
            self.assertTrue(writer.print_dict('write_cache'))
            parser.write_cache('search_cache', summary, 'cuba')
        else:
            writer.alredy_cahe('cuba')
```

Imagen A.9 Prueba de Integración entre los métodos: parser.get_summary; parser.get_results; parser.get_alternative