UNIVERSIDAD TÉCNICA PARTICULAR DE LOJA

La Universidad Católica de Loja

ÁREA TÉCNICA

TÍTULO DE INGENIERO EN SISTEMAS INFORMÁTICOS Y COMPUTACIÓN

Desarrollo de un chatbot que ayude a responder a preguntas frecuentes referentes a becas en la Universidad Técnica Particular de Loja.

TRABAJO DE TITULACIÓN

AUTOR: Toledo Cambizaca, Alcides Francisco

DIRECTOR: Cordero Zambrano Jorge Marcos, Mgtr.

LOJA - ECUADOR

2018

APROBACIÓN DEL DIRECTOR DEL TRABAJO DE TITULACIÓN

Magister

Jorge Marcos Cordero Zambrano

DOCENTE DE LA TITULACIÓN

De mi consideración

El presente trabajo de titulación: **Desarrollo de un chatbot que ayude a responder a preguntas frecuentes referentes a becas en la Universidad Técnica Particular de Loja**, realizado por Toledo Cambizaca Alcides Francisco, ha sido orientado y revisado durante su ejecución, por cuanto se aprueba la presentación del mismo.

Loja, febrero de 2018

f)……..……..

DECLARACIÓN DE AUTORÍA Y CESIÓN DE DERECHOS

“Yo Toledo Cambizaca Alcides Francisco declaro ser autor del presente trabajo de titulación: Desarrollo de un chatbot que ayude a responder a preguntas frecuentes referentes a becas en la Universidad Técnica Particular de Loja, de la Titulación de Sistemas Informáticos y Computación, siendo Jorge Marcos Cordero Zambrano director del presente trabajo; y eximo expresamente a la Universidad Técnica Particular de Loja y a sus representantes legales de posibles reclamos o acciones legales. Además, certifico que las ideas, conceptos, procedimientos y resultados vertidos en el presente trabajo investigativo, son de mi exclusiva responsabilidad.

Adicionalmente declaro conocer y aceptar la disposición del Art. 88 del Estatuto Orgánico de la Universidad Técnica Particular de Loja que en su parte pertinente textualmente dice: “Forman parte del patrimonio de la Universidad la propiedad intelectual de investigaciones, trabajos científicos o técnicos y tesis de grado o trabajos de titulación que se realicen con el apoyo financiero, académico o institucional (operativo) de la Universidad”

f. .....

Autor: Toledo Cambizaca Alcides Francisco

Cédula: 1105117798

DEDICATORIA

El presente trabajo de titulación está dedicado especialmente a Dios por darme la sabiduría y la fortaleza para poder cumplir con las metas y objetivos que me propuse.

A mi padre Manuel, a mi madre Rosa América por ser los principales pilares en mi vida, por su incondicional apoyo en todos estos años; a mis hermanos Cristiam y Michael, que son una inspiración para seguir adelante, a toda mi familia por esta presentes en cada momento de mi vida, por siempre estar dispuestos a brindarme su apoyo.

A mi amigos y compañeros, que siempre estuvieron presentes y fuero apoyo para alcanzar este objetivo, que siempre compartieron sus conocimientos y sin pedir nada a cambio, y a las demás personas que con algún consejo me dieron su apoyo incondicionalmente.

Alcides Francisco Toledo Cambizaca

AGRADECIMIENTO

Agradezco a Dios por darme las fuerzas, iluminar mi mente y por haber puesto en mi camino a las personas indicadas y que sirvieron de ejemplo para ser una mejor ser humano.

A mis padres que siempre están conmigo, en los buenos momentos, y más aún en los malos, gracias por todo, por darme la oportunidad de ser alguien mejor, por creer en mí y por darme su apoyo incondicional y su infinito amor.

A mis hermanos por sus consejos, por estar siempre presentes, acompañándome en todo momento para poder culminar este objetivo.

A mis amigos y demás familiares, gracias por estar conmigo todo este tiempo en donde he vivido momentos felices y tristes, siempre los llevare en mi corazón.

A mi director de tesis Mgtr. Jorge Cordero Zambrano por compartir sus conocimientos, brindarme su apoyo y ser una guía en el desarrollo de este trabajo.

Al Mgtr. Rodrigo Barba, al Mgtr. Franco Guamán, a todos los docentes de la titulación de Sistemas Informáticos y Computación y a los demás docentes de la universidad por haber compartido sus conocimientos y haber sido parte de mi formación en todos estos años.

ÍNDICE DE CONTENIDOS

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

RESUMEN

El presente trabajo de titulación surge de la iniciativa de contar con nuevas herramientas o canales de comunicación que resuelvan las dudas o inquietudes que tengan los usuarios sobre becas de la UTPL. Por este motivo se desarrolló e implementó un chatbot que es capaz de simular una conversación, además, involucra computación cognitiva para interpretar la información que proporciona el usuario en lenguaje natural y así responder en tiempo real y de forma eficaz.

Para el desarrollo del chatbot se utilizó la metodología ágil XP, además, para la implementación se usó la plataforma de IBM Bluemix que provee los servicios para conectar con distintas interfaces web, particularmente el servicio de Watson Conversation para reconocer la consulta de los usuarios y responder de forma eficaz.

También, se realizó pruebas de aceptación, unitarias y calidad de software usando herramientas como SonarQube, Mocha y Qunit. Finalmente, se aplicó la encuesta SUS (System Usability Scale) para determinar el nivel de aceptabilidad que tiene el chatbot, obteniendo como resultado la puntuación de 76/100, lo que significa que tiene una buena aceptación.

PALABRAS CLAVE: chatbot, agente conversacional, inteligencia artificial, Watson Conversation, text-to-speech.

ABSTRACT

The present title work arises from the initiative to have new tools or communication channels that resolve the doubts or concerns that users have about UTPL scholarships. For this reason, a chatbot was developed and implemented that is capable of simulating a conversation. It also involves cognitive computing to interpret the information provided by the user in natural language and thus respond in real time and efficiently.

The agile XP methodology was used for the development of the chatbot, and for the implementation, the IBM Bluemix platform was used, which provides the services to connect with different web interfaces, particularly the Watson Conversation service to recognize the users' query and answer effectively.

Also, acceptance, unit, and software quality tests were carried out using tools such as SonarQube, Mocha, and Qunit. Finally, the SUS (System Usability Scale) survey was applied to determine the acceptability level of the chatbot, obtaining, as a result, the score of 76/100, which means that it has a good acceptance.

KEYWORDS: chatbot, conversational agent, artificial intelligence, Watson Conversation, text-to-speech.

INTRODUCCIÓN

En la actualidad se ha proliferado la inclusión de agentes conversacionales como sistemas de interacción hombre-máquina, que comúnmente se los conoce como chatbot o asistentes virtuales. Estos chatbots son programas diseñados para simular conversaciones con humanos, son una solución de inteligencia artificial que puede mejorar la experiencia de los usuarios al permitir una interacción con una interfaz mediante texto o voz. El chatbot puede facilitar el trabajo al momento de obtener información de un tema en específico, además de responder de manera rápida, permanecer activo durante las veinticuatro horas del día y mantener un dialogo con varias personas simultáneamente. Por eso algunas compañías como Facebook, IBM, Google, proveen de API's para construir chatbots a medida, esto permite a las empresas disminuir costos y aumentar beneficios.

Las preguntas que los estudiantes realizan con frecuencia sobre las becas generalmente son: a quien va dirigida, cuáles son los montos o descuentos que se otorga a cada beca, cuando son las fechas de postulaciones, entre otras. Los estudiantes desean obtener información y puede consumir un tiempo significativo del estudiante que tiene que buscar en sitios web de la institución, comunicarse al *Call Center* o estar físicamente en el balcón de servicios, en los dos últimos casos es necesario esperar un funcionario para resolver la inquietud. De la misma forma, la persona encargada de resolver estas inquietudes solo puede interactuar con una sola persona a la vez, y en ocasiones necesita buscar la información para dar una respuesta acertada.

Con este antecedente, en el presente trabajo se propone la implementación de un prototipo de chatbot que responda a las preguntas frecuentes referentes a las becas, con la finalidad de que Balcón de Servicios Estudiantiles y el Área de Bienestar Estudiantil cuenten con una herramienta que trabaje de forma conjunta y ofrezcan una adecuada atención a los usuarios en general. Además, ofrecer a los estudiantes una alternativa para acceder a dicha información.

Para una mejor comprensión del proyecto y la forma de solucionarlo, se revisó varios trabajos relacionados y se han considerado las diversas plataformas que utilizan computación cognitiva como: Dialogflow, LUIS y Watson Conversation. Para el diseño del chatbot se seleccionó la plataforma IBM Watson Conversation y la implementación del prototipo se despliega en un servidor Node.js que proporciona una interfaz web para los usuarios. Además, se integra con una base de datos que cuenta con dos tablas para almacenar la información de los usuarios, la primera tabla almacena información acerca de la consulta realizada y la

segunda tabla almacena el puntaje que otorga el usuario al chatbot, de esta manera se obtiene retroalimentación y así mejorar el prototipo.

Objetivos

General

Implementar un prototipo de chatbot que ayude a responder a preguntas frecuentes referentes a becas en la UTPL

Específicos

- Investigar aspectos teóricos y prácticos relacionados con chatbots.
- Análisis de sistemas de preguntas y respuestas, para generar el protocolo de dialogo.
- Planeación, diseño y codificación de la propuesta.
- Entrenar al chatbot para que pueda responder de manera rápida y correcta.
- Evaluar el prototipo chatbot en colaboración con personal del balcón de servicios para garantizar la calidad de respuestas.
- Implementar el prototipo en un sitio web.

Metodología de desarrollo

Para el desarrollo del presente Trabajo de Titulación se utilizó la metodología de desarrollo ágil XP en la que el cliente forma parte del equipo de desarrollo, la sencillez y la comunicación entre todos los miembros del equipo, de esta manera facilita que los desarrolladores respondan con eficacia a cambios que se produzcan durante el desarrollo del proyecto. Además, la plataforma de servicios IBM Bluemix, en la cual se puede integrar un servidor en Node.js que se conecta a los servicios computación cognitiva de Watson IBM, este servidor presenta una página web que puede ser accedida desde cualquier navegador en la Internet, con la finalidad de realizar preguntas acerca de las becas ofertadas en periodo actual de clases.

Descripción de los capítulos

A continuación, se describen brevemente los capítulos:

En el Capítulo 1, se plasma información sobre el marco teórico, que da fundamento al presente trabajo. Se describen conceptos básicos sobre chatbots, sus características, arquitecturas, plataformas de desarrollo y la metodología seleccionada para la creación del chatbot.

En el Capítulo 2, se define el protocolo de atención al usuario en el que se basa el chatbot para responder a las consultas de los clientes y el proceso que se sigue para obtener la información referente al tema de dominio.

En el Capítulo 3, se describe los resultados obtenidos en cada fase de desarrollo del chatbot como: los interesados del proyecto, las historias de usuario, la arquitectura solución, entre otros artefactos, las pruebas que se realizaron luego de la implementación de la aplicación, así como también los resultados obtenidos al aplicar la encuesta SUS a los estudiantes de la materia de Inteligencia Artificial Avanzada de la titulación de Sistemas Informáticos y Computación.

Finalmente, se presentan las conclusiones y recomendaciones obtenidas durante el desarrollo del trabajo de titulación.

CAPITULO I.
MARCO TEÓRICO

1.1. Chatbot o agente conversacional

Un chatbot, o agente conversacional, se puede definir como un programa informático que interactúa con usuarios que usan lenguaje natural, a veces conoce un determinado dominio o en un determinado tema, y posiblemente tiene un avatar y un módulo de procesamiento de voz (Van Woudenberg, 2014, p. 1). Además, (Cobos T, 2013) agrega que "se integra técnicas de lingüística computacional y la comunicación se puede establecer a través de la Internet, Mensajes instantáneos, e-mail, foros, entre otros" (p. 9) . Logra que, un programa debe procesar las oraciones escritas por un usuario real que está interactuando con el chatbot. Tales programas pueden participar en el análisis complejo de una frase de entrada del usuario para determinar su estructura gramatical, a partir de ahí logra determinar cómo responder a la oración.

El término *agente* se puede definir como un sistema o entidad física (actúan en el mundo real como seres humanos) o virtual (que no tiene existencia física), situada en algún ambiente, que es capaz de actuar de manera autónoma y flexible y cuyo comportamiento está orientado por un conjunto de tendencias que pueden estar dadas por la satisfacción de ciertos objetivos individuales (Giugni, Vera, Díaz, & Cattafi, 2007, párr. 22), que pueden ser:

- *Reactivo*, donde responde al entorno en que se encuentra.
- *Proactivo*, donde puede ser capaz de intentar cumplir sus propios objetivos.
- *Social,* donde es capaz de comunicarse con otros agentes mediante algún tipo de lenguaje (Cobos T, 2013, p. 9).

La definición de *conversacional* es "Que tiene las características de una conversación o del lenguaje informal empleado en ella" (WordReference, 2017).

Entonces un agente conversacional es una entidad virtual que está diseñada para tener conversaciones con los seres humanos o con otros agentes. Estos agentes conversacionales se los conocen generalmente como asistentes virtuales, chatbot o chatterbot, y dependiendo del contexto pueden ser usados para entretenimiento o agentes más técnicos que den respuesta a un tema en concreto.

1.2. Características de un chatbot

Según (Cobos T, 2013; Inglada, 2002) cada agente inteligente tiene sus propias características, sin embargo, las que diferencian a los chatbots son las siguientes:

- *Autonomía*: es la capacidad de actuar basándose en la experiencia, sin la intervención de un ser humano u otro agente. El agente es capaz de continuar, aunque el entorno cambie severamente. Por otra parte, una definición menos estricta de autonomía cuando el agente percibe el entorno.

- *Sociabilidad:* este atributo permite a un agente comunicar con otros agentes o incluso con otras entidades, lo que convierte en una característica diferenciadora.
- *Racionalidad:* esta característica permite que el agente siempre realice lo correcto a partir de los datos que percibe del entorno. Sabe que respuesta formular analizando la frase introducida por el otro agente
- *Reactividad*: un agente actúa como resultado de cambios en su entorno. En este caso, un agente percibe el entorno y esos cambios dirigen el comportamiento del agente.
- *Pro-actividad*: un agente es pro-activo cuando es capaz de controlar sus propios objetivos a pesar de cambios en el entorno. Esta definición no contradice la de reactividad. Los chatbots pueden ser programados para entender el contexto la conversación y situarse en la pregunta correcta.
- *Adaptabilidad*: está relacionado con el aprendizaje que un agente es capaz de realizar y si puede cambiar su comportamiento basándose en ese aprendizaje. Esta característica aún está en desarrollo, ya que permitiría que un agente conversacional entienda de una mejor manera el contexto para la cual fue programado.
- *Movilidad:* capacidad de un agente de trasladarse a través de una red telemática.
- *Veracidad:* asunción de que un agente no comunica información falsa a propósito. Para lo cual el desarrollador debe alimentar el chatbot con información de fuentes confiables.
- *Personalidad:* cada agente es único y depende del programador las características que quiera darle como emociones, comportamiento no verbal, entre otros.

Estas características deberían ser las necesarias que debería tener los agentes conversacionales

1.3. Arquitectura de los chatbots

En la figura 1 se describe la arquitectura básica de cómo funciona un chatbot, el proceso es el siguiente: en primer lugar, se debe ingresar el conocimiento del experto humano a la base del conocimiento en función de la reglas y patrones que exija la plataforma del chatbot, proceso conocido como ingeniería del conocimiento. Posteriormente, el usuario ingresa la consulta mediante la interfaz, y esta es enviada al motor de inferencia. El motor de inferencia establece los objetivos y obtiene la respuesta de acuerdo a la base de conocimiento, que al final sugiere un consejo o responde a una cuestión, enviado de vuelta a la interfaz de usuario (Ruiz, 2009, p. 27).

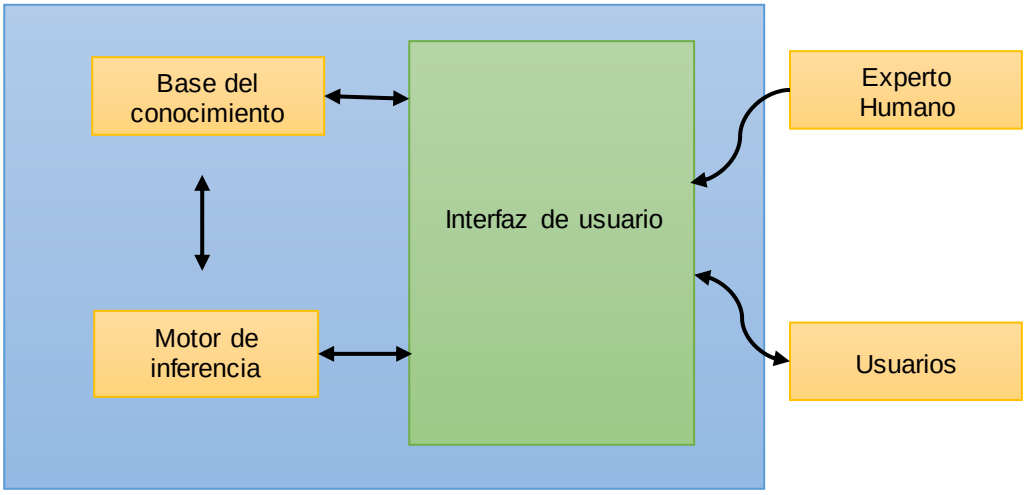


Figura 1. Arquitectura básica de un Chatbot
Fuente: (Ruiz, 2009)
Elaboración: (Ruiz, 2009)

1.4. Tipos de chatbots

Los chatbots pueden usarse en diferentes áreas como en Banca y Finanzas, Seguridad Industrial, Centros Médicos, en Telecomunicaciones, etc. Y dependiendo del rol que vaya a cumplir en determinada área, puede ser:

- ***Asistentes Virtuales Sociales:*** Su propósito es conversar libremente acerca de algo con cualquier usuario, así como lo haría con un amigo. Son usados en línea para entretenimiento.
- ***Asistentes Virtuales Educacionales:*** Su propósito es ayudar a aprender acerca de algo, como un nuevo lenguaje, historia, geografía, etc. Son usados en las escuelas.
- ***Asistentes Virtuales Orientados a Servicios:*** Estos generalmente son utilizados por empresas que ofrecen servicios en línea. Su propósito es ayudar a los clientes a encontrar el camino en el sitio web, así como contestar preguntas acerca de sus productos y servicios (Morales-Rodríguez & Domínguez-Martínez, 2011, párr. 10).

El rol que cumplirá el chatbot será de un *Asistente Virtual Orientado a Servicios,* ya que el objetivo de este será la de contestar preguntas referentes a las becas que oferta la Universidad Técnica Particular de Loja.

Según (Bhatia, Gavaldà, & Einolghozati, 2017) históricamente existen dos clases principales de modelos conversacionales: modelos basados en la recuperación y generativos.

- ***Modelos basados en la recuperación:*** Estos modelos utilizan un repositorio de respuestas predefinidas y algunas heurísticas para elegir una respuesta adecuada basada en la entrada y el contexto. La heurística podría ser tan simple como una

expresión basada en reglas de coincidencia, o tan compleja usando aprendizaje automático. Estos sistemas no generan ningún texto nuevo y sólo seleccionan una respuesta de un conjunto determinado de respuestas. En la Figura 2 se observa el modelo de un chatbot basado en la recuperación, que dependiendo del mensaje y del contexto en el que se encuentre, a través de algoritmos heurísticos seleccionara la respuesta más idónea de un conjunto de posibles resultados

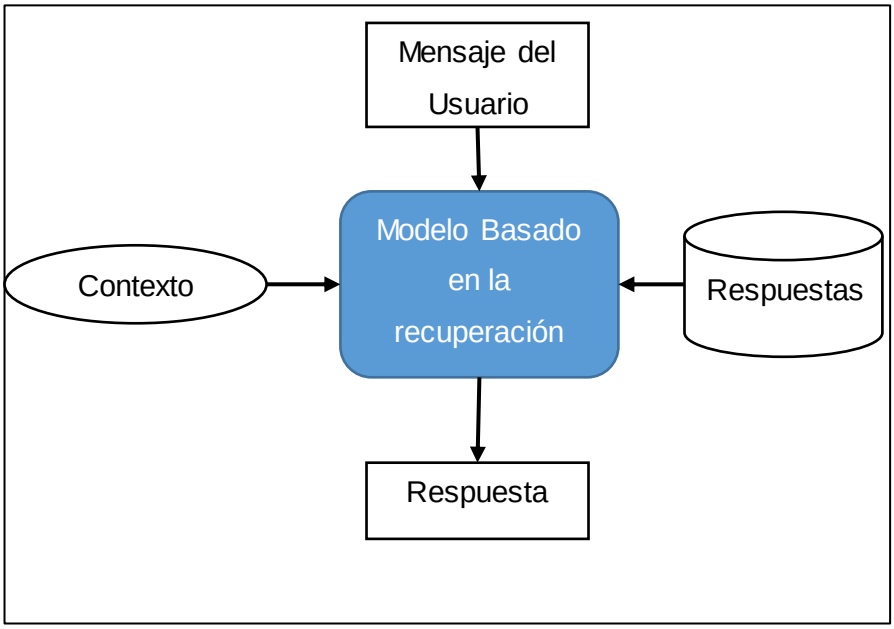


Figura 2. Modelo Basado en la recuperación
Fuente: (Bhatia et al., 2017)
Elaboración: Alcides Toledo

- ***Modelos Generativos:*** Estos modelos van más allá de las respuestas predefinidas y son capaces de generar nuevas respuestas desde cero. Tales modelos suelen basarse en técnicas de traducción automática, pero en lugar de traducir, "traducen" a partir de un texto de entrada a una respuesta. Los modelos generativos son más difíciles de construir y entrenar. Normalmente requiere millones de ejemplos para entrenar un modelo de aprendizaje profundo para obtener una calidad de conversación decente, y aun así no puede estar totalmente seguro de qué respuestas generará el modelo. En la Figura 3 se muestra cómo funciona un chatbot basado en un modelo generativo, el usuario ingresa un mensaje y dependiendo de los mensajes que se usaron para el entrenamiento, dará una respuesta lo más acertada posible

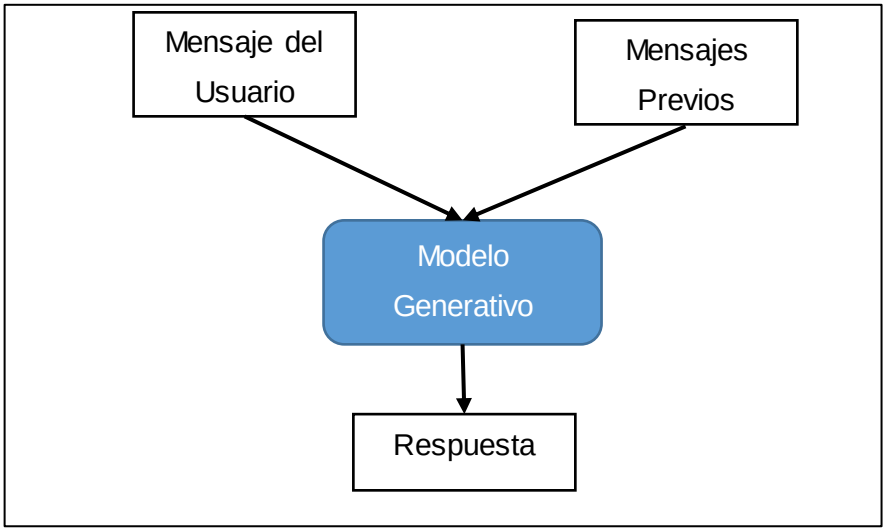


Figura 3. Modelo Generativo
Fuente: (Bhatia et al., 2017)
Elaboración: Alcides Toledo

Ambos enfoques tienen algunos ventajas y desventajas claros. Debido al repositorio de respuestas, los métodos basados en la recuperación no cometen errores gramaticales. Sin embargo, pueden ser incapaces de manejar casos no vistos para los que no existe una respuesta predefinida apropiada. Por las mismas razones, estos modelos no pueden referirse a información de entidad contextual como los nombres mencionados anteriormente en la conversación. Los modelos generativos son "más inteligentes". Pueden referirse a las entidades en la entrada y dar la impresión de que usted está hablando con un ser humano. Sin embargo, estos modelos son difíciles de entrenar, es muy probable que cometan errores gramaticales (especialmente en frases más largas), y normalmente requieren grandes cantidades de datos de entrenamiento.

1.5. Computación cognitiva

Según la (RAE, 2017b) el término *computación,* también llamada *informática,* es el conjunto de conocimientos científicos y técnicas que hacen posible el tratamiento automático de la información por medio de computadoras

La definición de (RAE, 2017a) *cognitiva* que pertenece o es relativo al conocimiento, y está relacionado con el proceso de adquisición de conocimiento (cognición) mediante la información recibida por el ambiente y el aprendizaje. La cognición implica muchos factores como el pensamiento, el lenguaje, la percepción, la memoria, el razonamiento, la atención, la resolución de problemas, la toma de decisiones, etc., que forman parte del desarrollo intelectual y de la experiencia.

En (Kelly III, 2015) define la Computación Cognitiva como sistemas que aprenden a escala, razonan con un propósito e interactúan con humanos de manera natural. En vez de ser explícitamente programados, aprenden y razonan desde su interacción con nosotros y desde sus experiencias con el entorno (p. 2).

Por lo tanto, la computación cognitiva significa informática basada en computadores capaces de sentir, adaptarse y aprender que tratan de simular la forma de pensamiento humano. Se basan en algoritmos de aprendizaje profundo y redes neuronales para procesar la información comparándola con un conjunto de datos disponibles o aprendidos anteriormente.

Con el fin de describir y cubrir verdaderamente el amplio espectro de tecnologías involucradas en la Computación Cognitiva, en (Bhilegaonkar, 2016) se propone seis categorías de Tecnologías de Computación Cognitivas, y son las siguientes:

- ***Análisis predictivo:*** Sistemas inteligentes que tratan sobre todo con datos estructurados. Estos datos suelen ser en forma de números, por ejemplo, transacciones financieras o datos de ventas. En este campo se aplican un amplio espectro de técnicas de análisis estadístico o matemático a los números para encontrar patrones, conocimientos o tendencias. Las tecnologías de análisis utilizan técnicas de visualización de datos para comunicar las ideas que pueden conducir a un rendimiento mejorado o la optimización del sistema
- ***Reconocimiento de imágenes / Visión por computadora:*** Recibir imágenes que forman el mundo real y luego procesarlas representándolas en forma de números, realizar análisis y luego tomar decisiones de acuerdo con los valores obtenidos para luego realizar una interpretación.
- ***Reglas de Negocio:*** Las reglas expresan la lógica de negocio de una manera estructurada usando construcciones simples tales como declaraciones de *Si/Entonces*. Los sistemas expertos fueron programados usando este tipo de reglas de negocio para emular la capacidad de toma de decisiones de un experto humano. Un sistema experto podría emplear una regla de negocio que dicta cómo un cliente debe ser tratado cuando se aplicó para una hipoteca. Si la solicitud de crédito se acepta o rechaza dependerá de un umbral conocido como un puntaje de crédito.
- ***Aprendizaje Automático:*** Estos son conceptos más avanzados en el análisis de datos, pero también implican la creación de un modelo a partir de la adaptación de datos o modelos. Muchos enfoques para el aprendizaje automático incluyen redes neuronales, redes neurales profundas, clasificadores bayesianos, máquinas de vectores de apoyo, etc.

- ***Procesamiento del lenguaje natural:*** El PLN se ocupa de la interacción de la computadora humana. Reconocimiento de voz, lectura de contenido de texto y derivar el significado de la información textual procesada. IBM Watson, Apple Siri y Google Now son ejemplos de sistemas que intentan entender las lenguas habladas y actuar sobre la información.
- **Procesamiento complejo de eventos (CEP):** Los CEP tratan datos en tiempo real de un conjunto diverso de fuentes y tipos. A continuación, tratan de agregar la información, entender y tomar medidas. Los sistemas de CEP son ampliamente utilizados en el ámbito financiero, como el comercio de valores, la detección de fraudes de crédito, etc (p. 18).

1.6. Plataformas de Chatbots

A continuación, en esta sección se presentan tres plataformas de chatbot que usan computación cognitiva para comprender el lenguaje natural y responden a los usuarios dependiendo de su consulta.

1.6.1. IBM Watson

Watson Conversation es un servicio que puede crear una aplicación que comprende la entrada de lenguaje natural y utiliza el aprendizaje automático para responder a los clientes de una manera que simule una conversación entre los seres humanos (IBM, 2017).

En la Figura 4 se muestra la arquitectura general de cómo funciona el servicio.

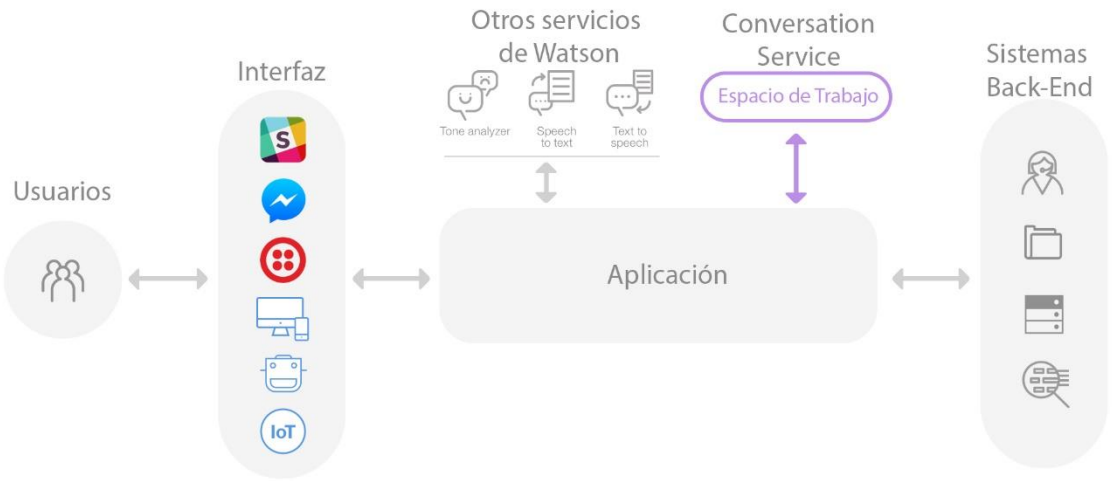


Figura 4. Arquitectura general de Conversation Service
Fuente: (IBM, 2017)
Elaboración: (IBM, 2017)

Los usuarios interactúan con su aplicación a través de la interfaz de usuario que implementa. Por ejemplo, una ventana de chat simple o una aplicación móvil, o incluso un robot con una interfaz de voz pueden funcionar como clientes. Luego la aplicación envía la entrada del usuario al servicio de conversación. La aplicación se conecta a un espacio de trabajo, que es un contenedor para el protocolo de diálogo y los datos de formación del chatbot.

Puede conectar servicios adicionales de Watson para analizar la entrada de usuarios, como Tone Analyzer o Text to Speech.

La aplicación puede interactuar con los sistemas de back-end según las intenciones del usuario y la información adicional. Por ejemplo, responder preguntas, abrir tiques, actualizar la información de la cuenta o realizar pedidos.

Watson Conversation proporciona un entorno gráfico fácil de usar para crear flujos naturales de conversación entre sus aplicaciones y sus usuarios. La creación de la primera conversación mediante el servicio de Conversación IBM Watson implica los siguientes pasos:

- Entrenar a Watson para que entienda la entrada de sus usuarios con expresiones de ejemplo: *Intenciones* y algunos ejemplos de esas intenciones.
- Identificar los términos que pueden variar en la entrada de sus usuarios: *Entidades*.
- Crear las respuestas a las preguntas de un usuario: Constructor del Dialogo.

1.6.2. LUIS (Language Understanding Intelligent Service)

El servicio inteligente (LUIS) de Microsoft permite a los desarrolladores crear aplicaciones inteligentes que pueden comprender el lenguaje humano y reaccionar en consecuencia a las solicitudes de los usuarios. LUIS utiliza el poder del aprendizaje automático para resolver el difícil problema de extraer significado de la entrada de un mensaje en lenguaje natural. Cualquier aplicación cliente que necesite conversar con los usuarios, el desarrollador puede pasar la entrada del usuario a una aplicación LUIS y recibir resultados que proporcionen comprensión del lenguaje natural.

En la Figura 5 se observa el proceso de una aplicación LUIS donde se toma un enunciado del usuario y se extrae intenciones y entidades que corresponden a actividades en la lógica de la aplicación cliente, y esta la aplicación puede tomar la acción apropiada basándose en las intenciones del usuario que LUIS puede reconocer del mensaje (Microsoft, 2017).

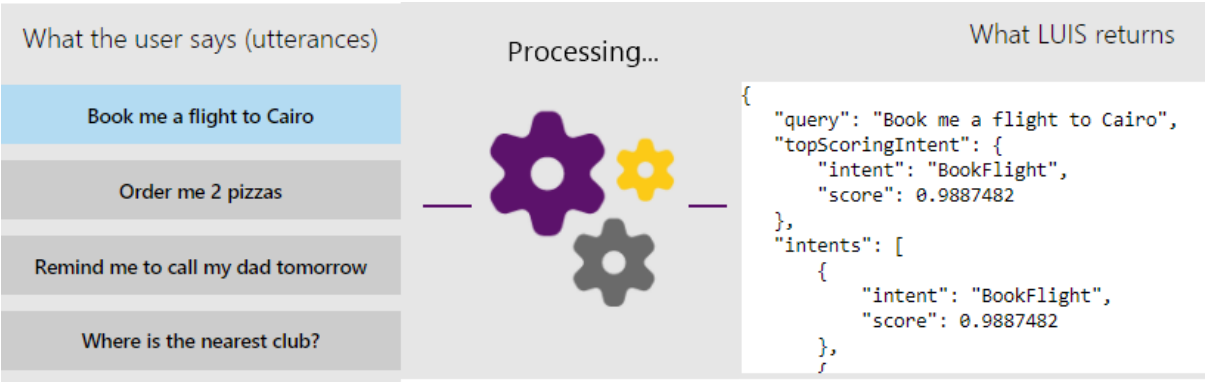


Figura 5. Arquitectura general de LUIS
Fuente: (Microsoft, 2017)
Elaboración: (Microsoft, 2017)

Las *intenciones* son como verbos en una oración. Una intención representa las acciones que el usuario desea realizar. Es un propósito u objetivo expresado en el mensaje de un usuario, como reservar un vuelo, pagar una factura o encontrar un artículo de noticias.

Si las *intenciones* son verbos, las *entidades* son sustantivos. Una entidad representa una instancia de una clase de objeto que es relevante para la intención de un usuario. En el enunciado "Reserva un billete para París", "París" es una entidad de tipo localización. Al reconocer las entidades que se mencionan en la entrada del usuario, LUIS le ayuda a elegir las acciones específicas que debe tomar para cumplir una intención.

Para crear el chatbot Microsoft proporciona *Bot Framework*. Con *Bot Framework*, es posible escribir un flujo de dialogo usando Node.js o C#. En Node.js, la lógica empresarial y el flujo de conversación se escriben dentro de callbacks que escuchan eventos; Cuando LUIS reconoce las intenciones y las entidades, emite los eventos relacionados y las devoluciones de llamada se ejecutan. El framework necesita ser manejado por el desarrollador del chatbot, pero es mejor tener la flexibilidad del framework y no tener que construir uno desde cero.

En la figura 6 se observa el proceso de cómo interactúa el chat frente a los usuarios.

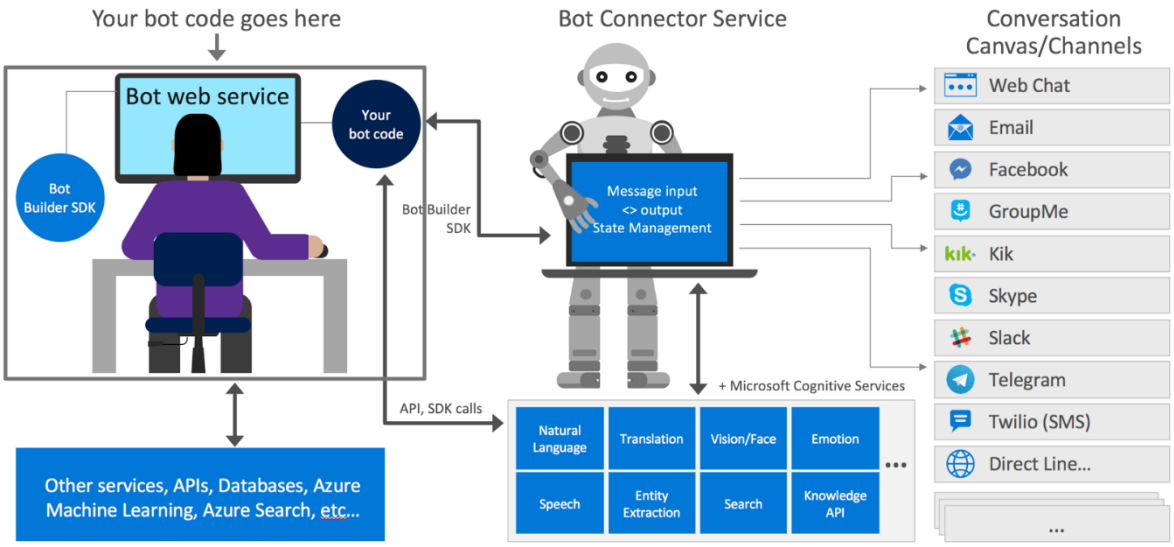


Figura 6. Funcionamiento de Bot Framework
Fuente: (Microsoft, 2017)
Elaboración: (Microsoft, 2017)

La sección de Bot web service está integrada con *Bot Framework,* el que incluye Bot Builder SDK. Los SDK proporcionan bibliotecas, ejemplos y herramientas para ayudarlo a crear y depurar chatbots. Los SDK contienen flujos de diálogo integrados para manejar las interacciones del usuario que van desde un Sí/No básico a una conversación compleja.

La plataforma se puede utilizar para crear una amplia variedad de chatbots, desde avisos simples hasta conversaciones de forma libre. La lógica de conversación para el chatbot se aloja como un servicio web.

El servicio Bot Connector conecta un bot a uno o más canales y maneja el intercambio de mensajes entre ellos. Este servicio de conexión permite al chatbot comunicarse a través de muchos canales sin diseñar manualmente un mensaje específico para el esquema de cada canal.

Bot Framework proporciona varios canales para comunicación del chatbot y otras aplicaciones de mensajería como Skype, Slack, Facebook Messenger, correo de Office 365 y otros. Se debe usar el Portal de desarrolladores para poder configurar cada canal en el que desea que el chatbot esté disponible. Los canales de Skype y de chat en la web se preconfiguran automáticamente.

1.6.3. Dialogflow

Dialogflow es una plataforma de comprensión del lenguaje natural que facilita que los desarrolladores (y no los desarrolladores) diseñen e integren interfaces de usuario conversacionales inteligentes y sofisticadas en aplicaciones móviles y aplicaciones web. El

objetivo es hacer que el proceso de creación e integración de sofisticadas interfaces de usuario de conversación sea lo más simple posible (Dialogflow, 2017b).

En la Figura 7 se muestra cómo Dialogflow está relacionado con otros componentes y cómo procesa los datos:

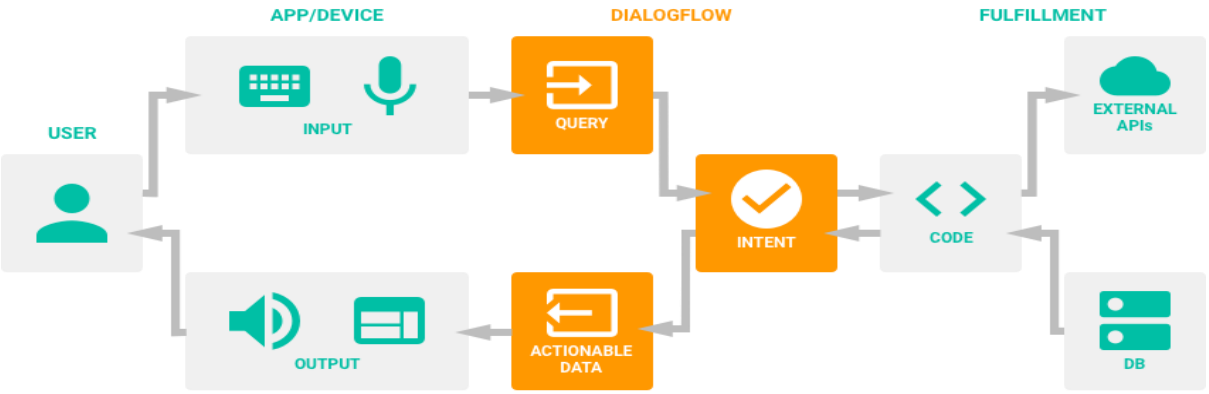


Figura 7. Arquitectura general de Dialogflow
Fuente: (Dialogflow, 2017a)
Elaboración: (Dialogflow, 2017a)

La sección de Usuario y App representa el cliente que consumirá los servicios proporcionados por la plataforma, dicha aplicación puede ser dispositivos móviles donde puede ingresar el enunciado del usuario por medio de texto o voz, posteriormente la plataforma reconoce el método de entrada de la información, la procesa y retorna la respuesta ya sea por un mensaje de texto o voz, de la misma forma trabaja la integración con una página web donde el dispositivo tiene que contar con un micrófono o un teclado para poder ingresar el enunciado y la plataforma retorna la respuesta adecuada.

La sección verde es proporcionada por la plataforma Dialogflow. El código de chatbot proporciona los métodos de entrada y salida y responde a los datos procesables. También puede proporcionar una implementación opcional que Dialogflow utiliza para conectarse a su servicio web. Su servicio web puede realizar la lógica de negocios, llamar a API's externas o acceder a una base de conocimientos y hacer coincidir la consulta con la intención más adecuada basada en la información contenida en la intención (ejemplos, entidades utilizadas para anotaciones, contextos, parámetros, eventos) y el modelo de aprendizaje automático del agente. Dialogflow transforma el texto de la consulta en datos ejecutables y devuelve los datos de salida como un objeto de respuesta JSON.

Los principales diferenciadores de *Procesamiento de Lenguaje Natural* y *Machine Learning* de Dialogflow se basan en el volumen de diálogos procesados (procesan millones de solicitudes al día), técnicas de capacitación (no es necesario proporcionar muchos ejemplos inicialmente, aprovechar los datos existentes) y todo el flujo de trabajo de creación de diálogo.

Lo interesante es que Dialogflow tiene dominios integrados de conocimiento (*Intenciones* con *Entidades* e incluso sugiere respuestas) sobre temas como las fechas específicas, el tiempo, alarmas y gestión sobre otras aplicaciones si se desarrolla para una aplicación móvil. Esto significa que el agente en el sistema puede reconocer todas estas intenciones sin ningún entrenamiento adicional e incluso se puede proporcionar el texto de respuesta que puede usar y guiar al usuario dentro del flujo conversacional (Dialogflow, 2017a). Cuenta con soporte para 13 diferentes idiomas con soporte completo entre ellos esta inglés, chino, francés, portugués, alemán y español.

1.7. Comparativa de Watson Conversation, LUIS.AI y Dialogflow

En la Tabla 1, se describe las características de las plataformas más conocidas para el desarrollo de Chatbots, en la investigación de (Davydova, 2017) se encuentran los Chatbots mencionados anteriormente, además de otras 21 plataformas describiendo características, canales de comunicación, lenguajes de programación para poder realizar la integración con otros recursos, etc.

Tabla 1. Comparativa de Chatbots

Chatbot	IBM Watson Conversation Service	Dialogflow	Microsoft Bot Framework	Microsoft Language Understanding Intelligent Service (LUIS)
Plataforma	Cloud	Cloud	Cloud	Cloud
Clasificación de texto	Construido sobre una red neuronal (mil millones de palabras de Wikipedia). Tiene tres componentes principales: Intenciones, Entidades, Dialogo	Dialogflow hace coincidir la consulta con la intención más adecuada basada en la información contenida en la intención (ejemplos, entidades utilizadas para anotaciones, contextos, parámetros, eventos) y el modelo de aprendizaje automático del agente. Dialogflow transforma el texto de la consulta en datos ejecutables y devuelve los datos de salida	Entiende la intención del usuario. Para darle al chatbot más sentidos humanos, se puede incorporar LUIS para el entendimiento del lenguaje natural, Cortana para voz y las APIs de Bing para la búsqueda.	Usos intentos y entidades. Todas las aplicaciones LUIS se centran en un tema específico del dominio o contenido relacionado. Puede utilizar preexistentes, de clase mundial, los modelos pre-construidos de Bing y Cortana. Implementación de modelos en un punto extremo HTTP

		como un objeto de respuesta JSON.		con un solo clic. LUIS devuelve JSON fácil de usar.
Lenguajes de programación / Aplicaciones / Integración	Node.js SDK, Java SDK, Python SDK, iOS SDK, Unity SDK	SDKs: Android, iOS, Cordova, HTML, JavaScript, Node.js, .NET, Unity, Xamarin, C++, Python, Ruby, PHP (community supported), Epson Moverio, Botkit, Java	Bot Builder SDK (.NET SDK and Node.js SDK.), Bot Connector, Developer Portal, Bot Directory	C# SDK, Python SDK, Node.js SDK, Android SDK
Detalles técnicos	El marco proporciona la Direct Line REST API, que puede utilizar para alojar su bot en una aplicación o sitio web.	La plataforma provee integración con 14 plataformas con un solo clic, además de una REST API.	El marco proporciona la Direct Line REST API, que puede utilizar para alojar su bot en una aplicación o sitio web.	El marco proporciona la Direct Line REST API, que puede utilizar para alojar su bot en una aplicación o sitio web.
Licencia	Gratis Standard Premium https://www.ibm.com/watson/developercloud/conversation.html#pricing-block	Gratis https://dialogflow.com/pricing/	Es de código abierto y está disponible en Github. Se debe contar con cuenta de Azure para poder desplegar en entorno de producción	Privativo https://www.microsoft.com/cognitive-services/en-us/language-understanding-intelligent-service-luis#pricingoptions

Lenguajes	Inglés, Japonés, Árabe, Portugués, Chino, Francés, Alemán, Holandés, Coreano, Español	Portugués(Brasil), chino (cantonés), chino (simplificado), chino (tradicional), inglés, holandés, francés, alemán, italiano, japonés, coreano, portugués, ruso, español, ucraniano	Traducción automática a más de 30 idiomas incluido español e ingles	Inglés, francés, italiano, alemán, español, portugués brasileño, japonés, coreano y chino.
Project Link	https://www.ibm.com/watson/developercloud/conversation.html	https://dialogflow.com/	https://docs.botframework.com/en-us/	https://www.microsoft.com/cognitive-services/en-us/language-understanding-intelligent-service-luis
Canales de comunicación	Voz, Imagen, Texto, Slack, Facebook, Web app	Agent Demo Page, Actions on Google, Facebook, Slack, Twilio IP, Messaging, Twilio, Skype, Tropo , Telegram, Kik, LINE, Spark, Alexa, Cortana, Twitter	Desde su sitio web o aplicación a texto / SMS, Skype, Slack, Facebook Messenger, correo de Office 365	Se puede incorporar LUIS para el entendimiento del lenguaje natural, Cortana para voz y las API de Bing para la búsqueda.
Campos de Implementación	Salud, Finanzas, Legal, Venta al por menor, Juegos Sociales, Educación.	Plataforma Conversacional para bots, aplicaciones, servicios y dispositivos.	Se utiliza para construir y desplegar bots de alta calidad	

Fuente: (Davydova, 2017)
Elaboración: (Davydova, 2017)

Las plataformas que fueron investigadas, las características que contienen son similares. En lenguaje de programación las cuatro plataformas soportan lenguajes que se utilizan para ambientes web. En detalles técnicos las cuatro plataformas proveen de API's para conectar con diferentes plataformas centrándose en redes sociales además de una REST API para conectar con páginas web. La licencia de Dialogflow e IBM Watson proveen del servicio del chatbot de forma gratuita pudiendo mejorar el servicio si se cuenta con una cuenta Premium, en cambio la plataforma de LUIS.AI es una plataforma gratuita, pero necesita de Bot Framework para poder desplegar el chatbot, pero se debe contar con una cuenta Premium para publicar el chatbot.

El lenguaje todas las plataformas soportan los lenguajes más hablados en la que se incluyen el español e inglés. En el atributo conexión a canales de comunicación los Dialogflow e IBM Watson se pueden conectar a la mayoría de redes sociales y a web apps, mientras que LUIS necesita conectarse a Bot Framework para poder desplegarse con redes sociales y web apps.

La diferencia que permitió decidir a IBM Watson como plataforma para este trabajo de titulación es la clasificación del texto. LUIS.AI y Bot Framework deben desplegarse de forma conjunta para obtener un resultado de las otras dos plataformas investigadas. La capacidad de detectar las intenciones y entidades de IBM Watson es mejor que Dialogflow permitiendo un rendimiento mucho mayor cuando se cuenta con conjunto de datos más grandes. Además, IBM Watson cuenta con el componente de dialogo, este permite diseñar las respuestas que retorna a los usuarios de forma más personalizada y generando un dialogo con ellos permitiendo una experiencia más agradable.

1.8. Sistemas de preguntas y respuestas

Según (Wantroba & Romero, 2016), (Pudaruth, Boodhoo, & Goolbudun, 2016) un sistema de preguntas y respuestas parte del área de PLN y de Recuperación de Información . El proceso de recuperación de información se refiere al análisis, el preprocesamiento de la gran cantidad de datos de diferentes documentos, párrafos, etc. y almacenarlo para acceder a la información específica o precisa, y es un programa de computadora para responder las preguntas formuladas por un usuario. Lo realiza mediante la consulta a una base de datos estructurada de información que también se conoce como base de conocimientos. Para devolver la mejor respuesta posible, el sistema puede formular el resultado combinando información obtenida de bases de datos locales y de documentos que se encuentren disponibles en la web, a partir de esto el sistema da respuestas correctas en lenguaje natural.

Los sistemas de preguntas y respuestas se pueden clasificar en dos categorías:

- ***Sistemas de dominio abierto:*** Los sistemas de preguntas y respuestas de dominio abierto se ocupan de las preguntas independientes del dominio, en las que las respuestas de búsqueda del sistema proceden de la gran colección de documentos o de cualquier corpus. Una de las ventajas de este sistema es la capacidad de responder una amplia gama de preguntas, pero el inconveniente que tiene una baja precisión al momento de dar sus respuestas
- ***Sistemas de dominio cerrado:*** Los sistemas de preguntas y respuestas de dominio cerrado se ocupan de la respuesta a la pregunta dependiente del tema de dominio. Busca respuestas sólo desde la colección específica de documentos de un dominio en concreto. La ventaja que tienen estos sistemas es que tienen una alta precisión al momento de responder, sus desventajas son que tiene una limitada cobertura de respuestas sobre posibles consultas y estos sistemas deben ser creados por expertos en el tema de dominio(Pathak & Mishra, 2016, p. 534).

1.8.1. Arquitectura de los Sistemas de Preguntas y Respuestas

En la Figura 8 se describe la arquitectura de los sistemas de preguntas y respuestas en la sigue una serie de procesos que comienzan tomando la pregunta del usuario como entrada y terminan con una respuesta o una lista de respuestas priorizadas con indicaciones de la fuente de la información y está formado de los siguientes componentes: análisis de la pregunta, que incluye su categorización y la construcción de la correspondiente consulta en un lenguaje adecuado para ser presentada a un motor de búsqueda; recuperación de documentos en base a dicha consulta; extracción de las respuestas candidatas relevantes, y presentación al usuario.

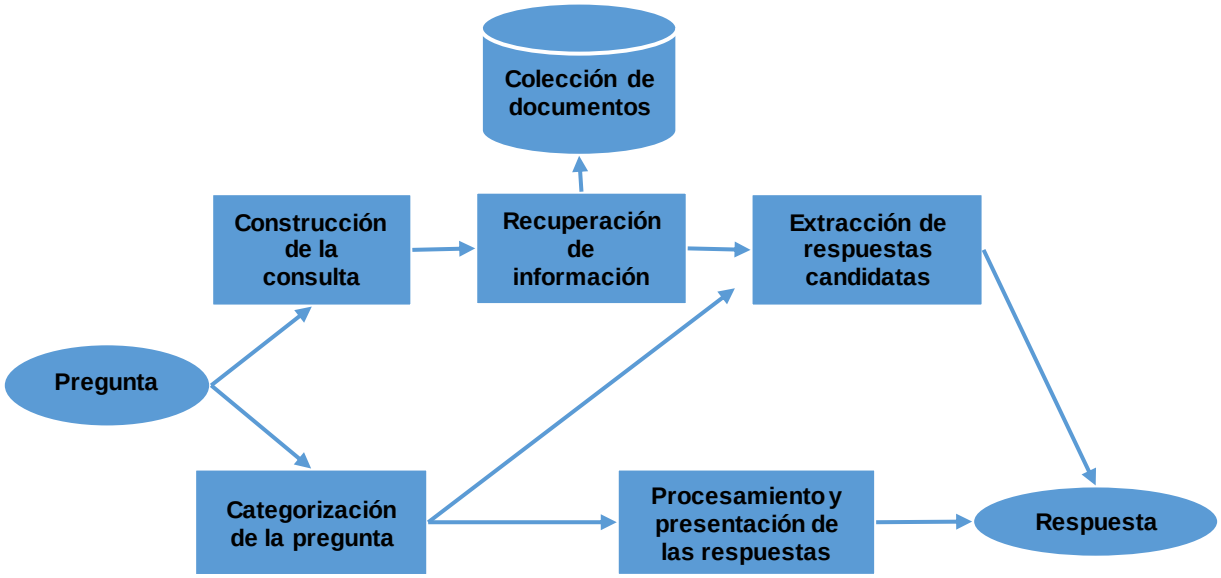


Figura 8. Arquitectura de Sistema de Preguntas y Respuestas
Fuente: (Cardoso, Bini, & Pérez Abelleira, 2015, p. 26)
Elaboración: (Cardoso et al., 2015, p. 26)

Los objetivos de los módulos de construcción de la consulta y categorización es determinar la clase de pregunta para poder concretar la clase de respuesta que tiene que dar en base a los términos que usa para realizar la consulta.

El módulo de recuperación de información retorna una colección de documentos que es la segunda parte del sistema, donde se trata de encontrar los documentos más relevantes con respecto a la pregunta entre los que se espera encontrar la respuesta. Y la información que retorne dependerá de los datos contenedores de la respuesta que pueden ser sobre datos estructurados o datos no estructurados.

En el módulo de extracción de respuestas candidatas junto al módulo de procesamiento y presentación de las respuestas se localiza los fragmentos de los documentos donde puede estar la respuesta, la dificultad en esta parte depende del tipo de respuesta esperada que pueden ser tan simple como encontrar el nombre de una persona en el fragmento relevante de un documento. Es esta parte la respuesta debe ser precisa, eficiente y simple.

La diferencia entre un Chatbot y los Sistemas de Preguntas y Respuestas es que el primero puede realizar más tareas dependiendo de cómo se plasmó el diseño conversacional, encuentra patrones para predecir los que necesita el usuario y así responder preguntas de forma más interactiva basada en una personalidad de sentido común; el segundo solo es efectivo donde las preguntas y respuestas ya estas programadas, si trata de contestar una pregunta que jamás se realizó antes, el sistema buscara una coincidencia similar a la pregunta por lo tanto dará una respuesta o listado de respuestas que podrían ser irrelevantes.

1.9. Metodología de desarrollo

Se entiende como metodología de desarrollo de software a una serie de actividades relacionada que conducen a la elaboración de un producto o servicio de software. El desarrollo de software no es una tarea fácil y para crear un proyecto son necesarias las metodologías las cuales imponen un proceso disciplinado sobre el desarrollo de software con el fin de hacerlo más eficaz y eficiente.

Existen diferentes metodologías de desarrollo, pero todas deben incluir cuatro actividades que son fundamentales para la generación de software:

- **Especificación de software.** Tienen que definirse tanto la funcionalidad del software como las restricciones de su operación.
- **Diseño e implementación del software.** Debe desarrollarse el software para cumplir con las especificaciones.
- **Validación de software.** Hay que validar el software para asegurarse de que cumple lo que el cliente quiere.
- **Evolución del software.** El software tiene que evolucionar para satisfacer las necesidades cambiantes del cliente (Sommerville, 2011, p. 28)**.**

Según la filosofía de desarrollo se pueden clasificar las metodologías en dos grupos. Las metodologías tradicionales, que se basan en una fuerte planificación durante todo el desarrollo, y las metodologías ágiles, en las que el desarrollo de software es incremental, cooperativo, sencillo y adaptado (Inteco, 2009, p. 41)

En la Tabla 2 se define esquemáticamente algunas de las principales diferencias entre estos dos grupos de metodologías. Estas diferencias que afectan no solo al proceso, sino también a la organización de los equipos de desarrollo:

Tabla 2. Diferencia entre metodologías ágiles y tradicionales

Metodologías ágiles	Metodologías tradicionales
Basadas en heurísticas provenientes de prácticas de producción de código	Basadas en normas provenientes de estándares seguidos por el entorno de desarrollo.
Especialmente preparados para cambios durante el proyecto	Cierta resistencia a los cambios
Impuestas internamente (por el equipo)	Impuestas externamente
Proceso menos controlado, con pocos principios	Proceso mucho más controlado, con numerosas políticas/normas
No existe contrato tradicional o al menos es bastante flexible	Existe un contrato prefijado

El cliente es parte del equipo de desarrollo	El cliente interactúa con el equipo de desarrollo mediante reuniones
Grupos pequeños (<10 integrantes) y trabajando en el mismo sitio	el mismo sitio Grupos grandes y posiblemente distribuidos
Pocos artefactos	Más artefactos
Pocos roles	Más roles
Menos énfasis en la arquitectura del software	La arquitectura del software es esencial y se expresa mediante modelos

Fuente: (Canós, Letelier, & Penadés, 2003, p. 4)
Elaboración: (Canós et al., 2003, p. 4)

1.9.1. Metodologías tradicionales

Las metodologías tradicionales son denominadas como metodologías pesadas, centran su atención en llevar una documentación exhaustiva de todo el proyecto y en cumplir con un plan de proyecto, definido todo esto, en la fase inicial del desarrollo del proyecto. Las metodologías tradicionales se focalizan en la documentación, planificación y procesos (Inteco, 2009, p. 42)

Estas metodologías tradicionales imponen una disciplina de trabajo sobre el proceso de desarrollo del software, con el fin de conseguir un software eficiente. Para ello, se hace énfasis en la planificación total del todo el trabajo a realizar una vez que esta todo detallado, comienza el ciclo de desarrollo del software (Hernández, 2014, p. 9)

Este tipo de metodologías su forma de trabajo, enfocándose en la planificación del proceso de desarrollo de software debido a que los procesos como plantillas, técnicas de administración, revisiones, etc. son rigurosamente controlados por normas y políticas.

Entre las metodologías tradicionales o no ágiles se puede citar las siguientes:

- RUP (Relational Unified Procces). (IBM, 1998)
- MSF (Microsoft Solution Framework).(Turner, 2006)
- Win-Win Spiral Model. (Vargas, 2015)
- Iconix. (ICONIX, 2016)

1.9.2. Metodologías ágiles

Un proceso es ágil cuando el desarrollo de software es incremental (entregas pequeñas de software, con ciclos rápidos), cooperativo (cliente y desarrolladores trabajan juntos con una cercana comunicación), sencillo y adaptable es decir que permite realizar cambios de último momento (Canós et al., 2003, p. 2).

(Canós et al., 2003) presenta las principales metodologías ágiles descritas a continuación:

- Extreme Programming.
- Scrum.
- Familia de Metodologías Crystal.
- Feature Driven Development.
- Proceso Unificado Rational, una configuración ágil.

Hay que adaptarse a la sociedad, donde se necesita tener la capacidad de proveer respuestas rápidas y que se adapte fácilmente al cambio

Las necesidades del cliente pueden sufrir cambios importantes al momento de contratación de un software y se debe aceptar los cambios y ser capaces de adaptarlos al sistema en lugar de rechazarlos. Por lo que las metodologías ágiles permiten trabajar con requisitos inestables, incapaces de predecir. Permitiendo resolverlos durante el desarrollo de prototipos evaluados por el cliente. Las metodologías ofrecen una serie de pautas y principios que junto a sus técnicas harán la entrega del proyecto menos complicado y más satisfactorio tanto para el cliente como para los equipos de entrega permitiendo que los cambios requeridos se efectúen de manera inmediata.

1.9.3. Selección de metodología

Para cumplir con los objetivos planteados en el trabajo de titulación, en la Tabla 3 se realiza una comparación de varias metodologías de desarrollo de software que es presentada en el trabajo de (Letelier & Penadés, 2006, § 2.2), donde le asigna un valor de uno a cinco, uno es las más baja y cinco la más alta, evaluando tres parámetros, :

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

Tabla 3. Comparativa Metodologías Ágiles

	CMM	ASD	Crystal	DSDM	FDD	LD	Scrum	XP
Sistema como algo cambiante	**1**	**5**	**4**	**3**	**3**	**4**	**5**	**5**
Colaboración	**2**	**5**	**5**	**4**	**4**	**4**	**5**	**5**
Características Metodología (CM)								
-Resultados	**2**	**5**	**5**	**4**	**4**	**4**	**5**	**5**
-Simplicidad	**1**	**4**	**4**	**3**	**5**	**3**	**5**	**5**
-Adaptabilidad	**2**	**5**	**5**	**3**	**3**	**4**	**4**	**3**
-Excelencia técnica	**4**	**3**	**3**	**4**	**4**	**4**	**3**	**4**
-Prácticas de colaboración	**2**	**5**	**5**	**4**	**3**	**3**	**4**	**5**
Media CM	**2.2**	**4.4**	**4.4**	**3.6**	**3.8**	**3.6**	**4.2**	**4.4**
Media Total	**1.7**	**4.8**	**4.5**	**3.6**	**3.6**	**3.9**	**4.7**	**4.8**

Fuente: (Letelier & Penadés, 2006, § 2.2)
Elaboración: (Letelier & Penadés, 2006, § 2.2)

El resultado muestra que ASD (Desarrollo Adaptable de Software) y XP tienen las puntuaciones más altas con 4.8, seguido de SCRUM con 4.7. Teniendo en cuenta que no existe una metodología que resuelva con éxito todos los proyectos, se debe usar una metodología que sea susceptible a cambios y se adapte a proyectos pequeños ofreciendo soluciones a medida. Por tal razón la metodología de desarrollo XP será seleccionada para el presente trabajo de titulación.

1.9.3.1. Programación Extrema (XP)

La metodología de desarrollo XP se centra en potenciar las relaciones interpersonales como clave para el éxito en el desarrollo de software, promoviendo el trabajo en equipo, y propiciando un buen ambiente de trabajo. Los fundamentos para el desarrollo con XP es la retroalimentación continua entre cliente y el equipo de desarrollo, comunicación clara y precisa entre todos los involucrados en el proyecto, sencillez en las soluciones implementadas y flexibilidad para enfrentar los cambios que se presenten durante todo el ciclo de vida de desarrollo del proyecto.

1.9.3.2. Definición

Los principios y prácticas son de sentido común pero llevadas al extremo, de ahí proviene su nombre XP, elegido por Kent Beck; Según la opinión de Inteco (2009) “La metodología XP es

la adopción de las mejores prácticas de metodologías de desarrollo ágil, adaptándose a lo que se pretende llevar a cabo con el proyecto y aplicarlo de manera dinámica durante todo el ciclo de vida de desarrollo" (p. 59). La metodología XP está enfocado a llevar al "extremo" las prácticas reconocidas, como el desarrollo iterativo e incluyendo al cliente como parte del equipo de desarrollo (Sommerville, 2011, p. 64), además, da prioridad a las tareas que dan resultados directos y que reducen la burocracia que hay alrededor tanto como sea posible (Ferrer, 2002, p. 2).

La Programación Extrema es una metodología de desarrollo ágil, en la que el cliente forma parte del equipo de desarrollo, la sencillez y la comunicación entre todos los miembros del equipo, permite además que los desarrolladores respondan con eficacia a cambios que se produzcan durante la marcha de desarrollo del proyecto.

1.9.3.3. Proceso XP

La programación extrema usa un enfoque orientado a objetos como paradigma de desarrollo, en la que engloba un conjunto de reglas y prácticas que ocurren en el contexto de cuatro actividades estructurales: planeación, diseño, codificación y pruebas (Pressman & Troya, 2007, p. 62). En la Figura 9 se observa algunas ideas y tareas clave asociadas con cada fase del proceso de desarrollo.

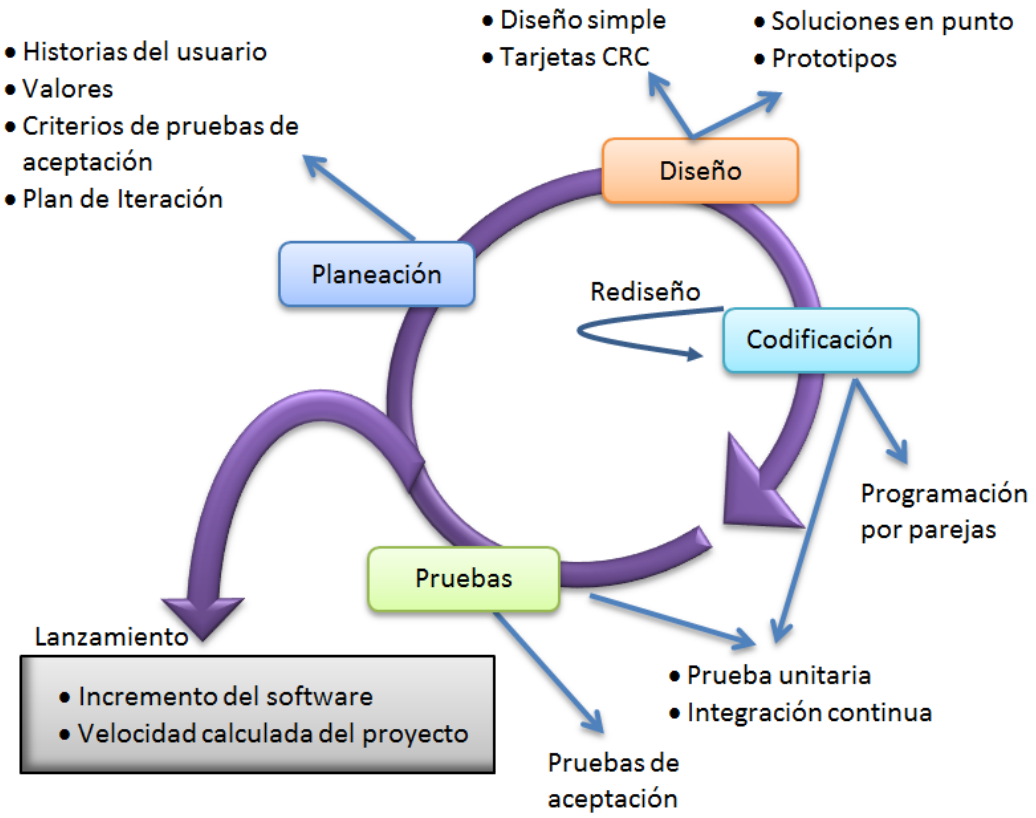


Figura 9. Proceso XP
Fuente: (Pressman & Troya, 2007, p. 63)
Elaboración: (Pressman & Troya, 2007, p63)

1.9.3.3.1. Planeación

Es un espacio frecuente de comunicación entre el cliente y los programadores. El equipo técnico realiza una estimación del esfuerzo requerido para la implementación de las historias de usuario y los clientes deciden sobre el ámbito y tiempo de las entregas y de cada iteración (Pressman & Troya, 2007, p. 62).

Aquí se planifica resolver primero las historias de usuario con más valor o más riesgo, además se pone en práctica valores, criterios de aceptación y se planifica cada iteración

- **Historias de Usuario**

Las historias de usuario son la técnica utilizada en XP para especificar los requerimientos del software. Se trata de tarjetas de papel en las cuales el cliente describe brevemente las características que el sistema debe poseer, sean r requerimientos funcionales o no funcionales. El tratamiento de las historias de usuario es muy dinámico y flexible, en cualquier momento historias de usuario pueden romperse, reemplazarse por otras más específicas o generales, añadirse nuevas o ser modificadas. Cada historia de usuario es lo suficientemente comprensible y delimitada para que los programadores puedan implementarla en unas semanas (Letelier & Penadés, 2006, § 3.1).

Cuando se realiza las historias de usuario, los desarrolladores agrupan las historias que puedan abordar en una iteración y evalúan el tiempo estimado, este servirá para las iteraciones subsecuentes, permitiendo realizar correcciones y estimar fechas de entregas más exactas.

Las historias de los usuarios se plasman en tarjetas, lo que facilita que el cliente pueda especificar la importancia relativa entre las diferentes historias de usuario. El formato de tarjeta además es muy provechoso a la hora de realizar pruebas de aceptación. Existen varias plantillas en la actualidad, de la cual se ha seleccionado la que se muestra en la Tabla 4.

Tabla 4. Formato de historia de Usuario

Historia de Usuario	
Usuario: Secretaria	
Nombre historia: Introducción de pedido (cliente preferente)	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Baja
Puntos estimados: 4	**Iteración asignada:** 1
Programador responsable: Paco Valero – David Ferrer	
Descripción: El pedido llega en un fichero de texto ASCII y es procesado automáticamente siguiendo el formato de la plantilla de pedido e introduciendo éste en la base de datos.	
Observaciones:	

Fuente: (Osuna & Usero, 2004, párr. 2)
Elaboración: (Osuna & Usero, 2004, párr. 2)

- **Valores**

Los cuatro valores de XP son: comunicación, simplicidad, realimentación y coraje. Estos valores son los necesarios para conseguir diseños y códigos simples, métodos eficientes de desarrollo y clientes contentos (Ferrer, 2002, pág. 8).

- **Criterios de aceptación**

Los criterios de las pruebas de aceptación se crean a partir de historias de usuarios. Durante una iteración, las historias de usuario seleccionadas durante la reunión de planificación de iteración se traducirán en pruebas de aceptación. El cliente especifica escenarios para probar cuando una historia de usuario se ha implementado correctamente. Una historia puede tener

una o varias pruebas de aceptación, lo que sea necesario para garantizar que la especificación funcione (Wells, 2013a, párr. 1).

- **Plan de iteración**

Cada iteración tiene una duración de 1 a 3 semanas. Las historias de usuarios son elegidas para cada iteración por el cliente y desarrolladores en orden de lo más valioso para el cliente en primer lugar, las que serán implementadas en cada versión del programa (Wells, 2013b, párr. 1).

1.9.3.3.2. Diseño

El diseño XP sigue rigurosamente el principio MS (mantenlo sencillo). Un diseño sencillo siempre se prefiere sobre una representación más compleja. Además, el diseño guía la implementación de una historia conforme se escribe: nada más y nada menos. Se desalienta el diseño de funcionalidad adicional porque el desarrollador supone que se requerirá después (Pressman & Troya, 2007, p. 63). XP recomienda el uso de tarjetas CRC para llevar un control sobre el sistema.

- **Tarjetas CRC**

El modelado clase-responsabilidad-colaborador (CRC) proporciona una manera sencilla de identificación y organización de las clases que son relevantes para los requerimientos de un sistema o producto. El objetivo es desarrollar una representación organizada de las clases. Las responsabilidades son los atributos y operaciones relevantes para la clase. En pocas palabras, una responsabilidad es “cualquier cosa que la clase sepa o haga”. Los colaboradores son aquellas clases que se requieren para dar a una clase la información necesaria a fin de completar una responsabilidad (Pressman & Troya, 2007, p. 148). El formato que se usara se muestra en la tabla 5.

Tabla 5. Tarjeta CRC

Nombre de la clase:	
Responsabilidades	**Colaboradores**
Incorpora paredes, puertas y ventanas	Pared
Muestra la posición de las cámaras de video	Cámara

Fuente: (Pressman & Troya, 2007, p. 149)
Elaboración: (Pressman & Troya, 2007, p. 149)

Si es necesario también se usarán diagramas de clases, diagramas de entidad-relación, casos de uso y arquitectura para dar solución al proyecto planteado.

- **Solución en punta**

Se refiere a buscar soluciones si en el diseño de una historia hay un problema, por lo que XP recomienda crear y evaluar inmediatamente un prototipo operativo de esa porción del diseño, esto se lo hace con el objetivo de disminuir el riesgo cuando comience la implementación verdadera y validar las estimaciones originales para la historia que contiene el problema de diseño (Wells, 2013c, párr. 1).

- **Prototipo**

El objetivo del prototipo es entregar un resultado rápido de cómo se verá, una visión preliminar del sistema final; por lo tanto, no habrá que esperar a que gran parte del proceso de desarrollo se termine para ver los resultados y permite aceptar sugerencias o adaptaciones antes de liberar el sistema.

1.9.3.3.3. Codificación

Esta fase es la más importante, ya que sin el código no se tiene nada, además la codificación se puede usar para dar a entender la solución a un problema, ya que este siempre es claro y conciso y no se puede interpretar de varias formas. XP usa el concepto de programación en parejas, este concepto se detalla a continuación

- **Programación en Parejas**

XP recomienda que dos personas trabajen juntas en una estación de trabajo con el objeto de crear código para una historia. Esto da un mecanismo para la solución de problemas en tiempo real (es frecuente que dos cabezas piensen más que una) y para el aseguramiento de la calidad también en tiempo real (el código se revisa conforme se crea). También mantiene a los desarrolladores centrados en el problema de que se trate. En la práctica, cada persona adopta un papel un poco diferente. Por ejemplo, una de ellas tal vez piense en los detalles del

código de una porción particular del diseño, mientras la otra se asegura de que se siguen los estándares de codificación (parte necesaria de XP) o de que el código para la historia satisfará la prueba unitaria desarrollada a fin de validar el código confrontándolo con la historia.

A medida que las parejas de programadores terminan su trabajo, el código que desarrollan se integra con el trabajo de los demás. En ciertos casos, esto lo lleva a cabo a diario un equipo de integración. En otros, las parejas de programadores tienen la responsabilidad de la integración. Esta estrategia de "integración continua" ayuda a evitar los problemas de compatibilidad e interfaces y brinda un ambiente que ayuda a descubrir a tiempo los errores (Pressman & Troya, 2007, p. 64).

Cabe destacar que en el presente trabajo no se usará el concepto de programación en parejas ya que solo existe una sola persona que se encuentra involucrada en el trabajo de fin de titulación.

1.9.3.3.4. Pruebas

En esta fase de desarrollo tiene como objetivo que tanto el cliente como los desarrolladores realicen pruebas con el fin de verificar el buen funcionamiento del sistema. XP recomienda realizar pruebas unitarias, pruebas de aceptación y pruebas de usabilidad que se describen a continuación:

- **Pruebas Unitarias**

Las pruebas unitarias son establecidas antes de escribir el código y son ejecutadas constantemente ante cada modificación del sistema. Los clientes escriben las pruebas funcionales para cada historia de usuario que deba validarse. En este contexto de desarrollo evolutivo y de énfasis en pruebas constantes, la automatización para apoyar esta actividad es crucial (Letelier & Penadés, 2006, § 5). Además, Wells (2013d) menciona que "Corregir pequeños problemas cada cierto número de horas toma menos tiempo que resolver problemas enormes justo antes del plazo final" (párr. 7).

- **Pruebas de aceptación**

También llamadas pruebas del cliente, son especificadas por el cliente y se centran en las características y funcionalidad generales del sistema que son visibles y revisables por parte del cliente. Las pruebas de aceptación se derivan de las historias de los usuarios que se han implementado como parte de la liberación del software (Pressman & Troya, 2007, p. 65).

- **Pruebas de usabilidad.**

Las pruebas de usabilidad se realizan para medir la experiencia de los usuarios con un sistema, preferiblemente antes de su lanzamiento para un uso real, y así encontrar cualquier problema que impida que los usuarios completen sus tareas, los desaceleren o degraden su experiencia de usuario. Tales pruebas son importantes dado que entrega información directa de cómo los usuarios reales utilizan un sistema.

Las pruebas de usabilidad consisten en seleccionar a un grupo de usuarios de un sistema y solicitarles que lleven a cabo las tareas para las cuales fue diseñada, en tanto el equipo de desarrollo toma nota de los errores y dificultades con las que se encuentren los usuarios (Hertzum, 2016, p. 86).

CAPITULO II.
DEFINICIÓN DEL PROTOCOLO CONVERSACIONAL

El chatbot está diseñado en base a dos elementos importantes para que funcione correctamente, los cuales son el motor que es Watson Conversation Service y una base de conocimientos que es recolectada para el entrenamiento del chatbot.

2.1. Definición de base de conocimientos

Las actividades realizadas para crear la base de conocimientos de Watson fueron:

2.1.1. Recopilar la información necesaria

Para crear la base de conocimientos de la aplicación desarrollada en este trabajo, inicialmente se tomó como referencia los documentos que se encuentran publicados en la página web del área de becas de la UTPL, en la que se encuentra información de cada una ellas como descripciones, quien puede beneficiarse de alguna beca en específico, los porcentajes y los requisitos para poder acceder a cada una de ellas. Así como también información de referente a fechas de postulación y preguntas frecuentes relacionadas a las becas ofertadas en el ciclo actual.

2.1.2. Categorización de la información encontrada en temas de conversación.

Después de haber recopilado la información necesaria para crear la base de conocimientos, el siguiente paso fue realizar la categorización de cada una de las becas, definiendo así el contexto de los temas de conversación del chatbot.

Cada modalidad de estudios de la UTPL, modalidad presencial y modalidad Abierta y a Distancia, cada una de ellas cuenta con categorías para definir un contexto a un grupo de becas, siendo las categorías: Becas de Apoyo Económico, Becas por Excelencia Académica, Becas a Minorías y Becas Religiosas, cada una de estas becas cuenta con una descripción que facilita su comprensión

En cada una de estas categorías se encuentra un grupo de becas, cada una de ellas no pertenece a otra categoría. En cada una de ellas se encuentra una descripción definiendo que cual es el propósito de la misma, un porcentaje de descuento o cierto monto de dinero que se realizara al momento de pagar la matricula por parte de los estudiantes, los beneficiarios que pueden o tienen la posibilidad de acceder a la beca y los documentos o requisitos que deben presentar para cada beca en particular.

A partir de esta información se realizó un diseño conversacional en la que el chatbot iniciara la conversación y poco a poco vaya resolviendo las dudas referentes a las becas de cada estudiante.

2.1.3. Desarrollar el dialogo adecuado para la base de conocimientos en Watson.

Luego de analizar y categorizar la información para la base de conocimientos, el siguiente paso fue diseñar un diagrama del flujo de la conversación que inicialmente la comenzara el chatbot, luego el estudiante puede seguir consultando sobre la información de las becas que necesita saber, desde la Figura 10 a la 13 se muestra un diagrama de flujo de la conversación que sería de forma lineal interpretando la entrada del usuario y responder de forma correcta.

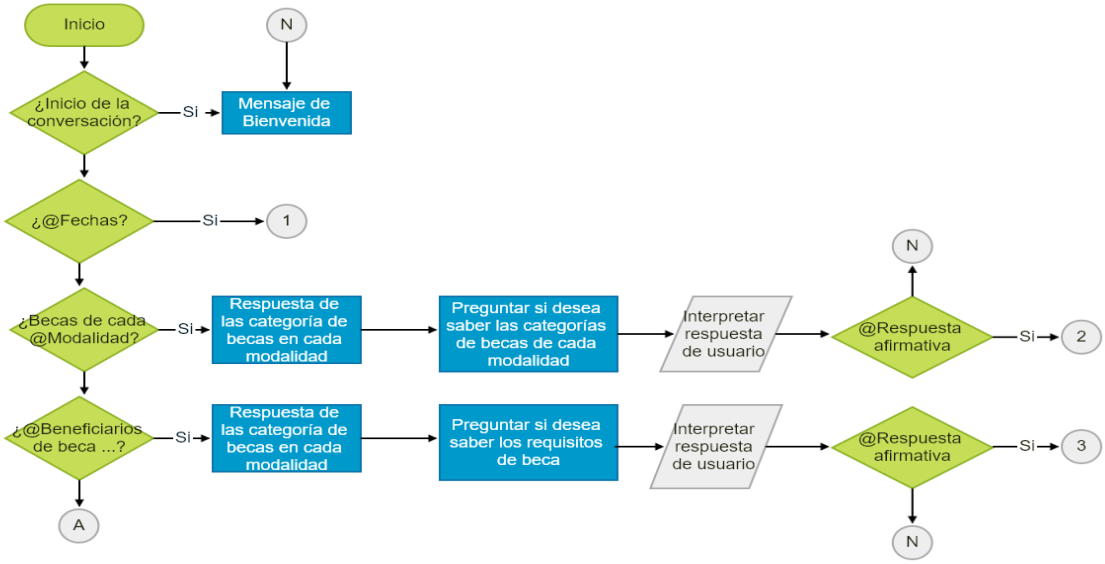


Figura 10. Proceso de información de becas según modalidad
Fuente: Autor
Elaboración: Autor

En la Figura 10 se muestra el proceso que toma el chatbot para responder dependiendo de lo que haya ingresado un usuario, si el usuario ingresa una consulta preguntando sobre becas según su modalidad el chatbot responderá adecuadamente y para continuar con la conversación le pregunta al usuario si desea saber más acerca de las becas, si la respuesta en interpretada como afirmativa se le presentara otro mensaje para continuar con la conversación, de la misma manera si la consulta es acerca sobre quién se beneficia de alguna beca en específico.

En la Figura 11, el chatbot interpreta la entrada, si es una pregunta sobre categorías, este responde con la información solicitada, luego pregunta si desea saber sobre alguna beca en específico, si la respuesta se interpreta como afirmativa o el nombre de alguna beca este contesta con la descripción de la beca mencionada, entonces se pregunta nuevamente si le interesa conocer los requisitos de la beca, si la respuesta es afirmativa, se le presenta los requisitos, si la respuesta es negativa se le pregunta si desea continuar con el dialogo, si la respuesta es afirmativa se envía un mensaje de inicio nuevamente, si la respuesta es no el

chatbot presenta un mensaje de fin de conversación. Si el mensaje no llega a ser interpretado por el chatbot se envía un mensaje de información para que el usuario corrija la consulta.

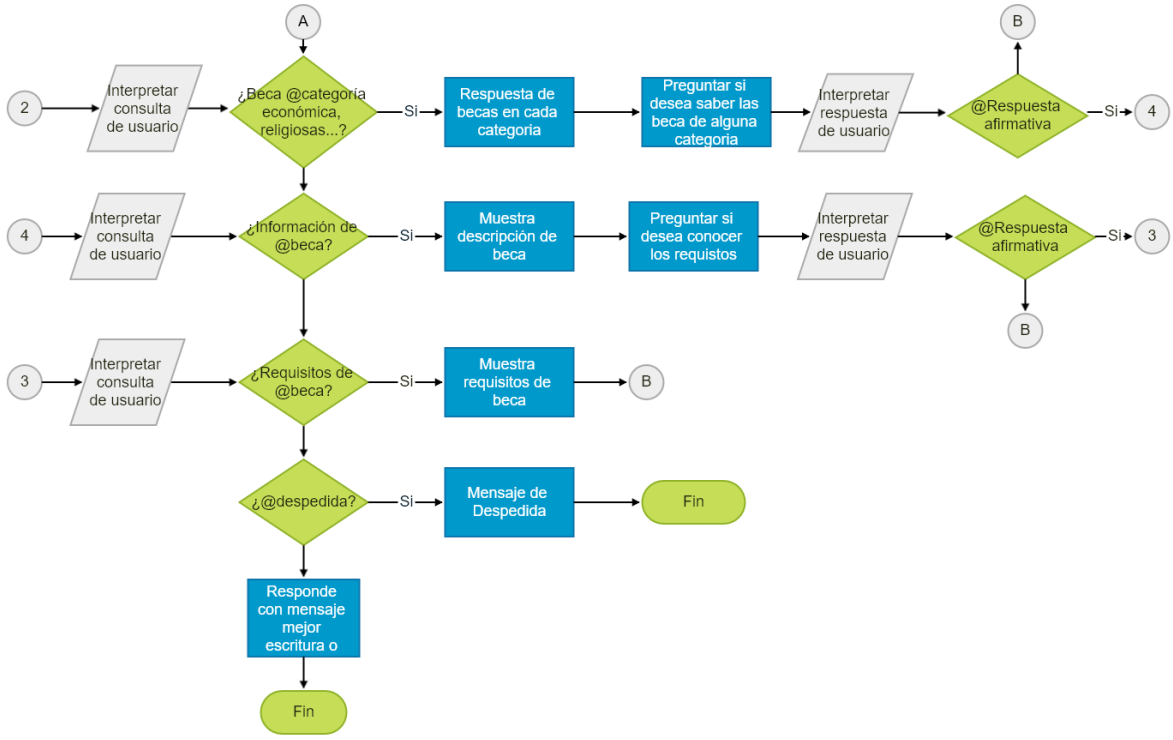


Figura 11.Proceso de información de becas, tipos de becas y requisitos
Fuente: Autor
Elaboración: Autor

En la Figura 12 se muestra el proceso del chatbot cuando interpreta la entrada del usuario como una pregunta sobre fecha de postulación o publicación de resultado, en cada caso presenta la información solicitada, si en la consulta no se especifica que desea saber sobre las fechas el chatbot responde con la información de las postulaciones y resultados.

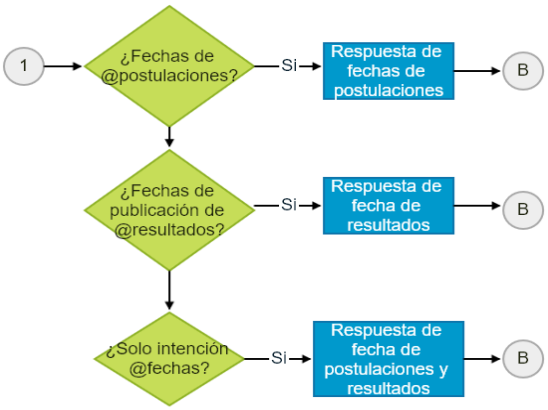


Figura 12. Proceso de información de becas
Fuente: Autor
Elaboración: Autor

La Figura 13 describe el proceso cada vez que el usuario de una respuesta negativa durante la interacción con el chatbot. El chatbot pregunta si desea continuar con la conversación, si la respuesta es afirmativa se presenta el mensaje de inicio nuevamente, si la respuesta es negativa se presenta un mensaje de agradecimiento y da fin al chatbot.

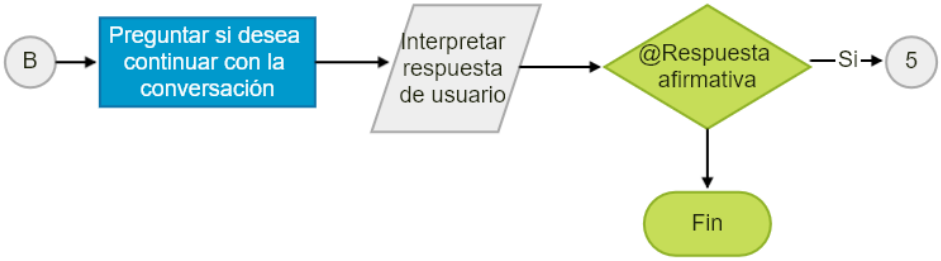


Figura 13. Proceso para continuar con la conversación
Fuente: Autor
Elaboración: Autor

2.2. Análisis de escenarios

A continuación, se describen los posibles escenarios que se podrían presentar en el prototipo y que son utilizados para entrenar la aplicación, que en función de una entrada que especifica el usuario la aplicación responda buscando la salida que mejor se adapte.

1° Escenario: Iniciando el chatbot

(Mensaje de bienvenida)

Chatbot: Saludos, seré su Asistente Virtual en esta conversación. Recuerde que estoy aquí para ayudarle con preguntas que tenga sobre las becas como: 1. Fechas de postulaciones. 2. Información sobre cada beca que oferta la UTPL. 3. Información de becas que se ofertan en cada modalidad de estudios.

2° Escenario: Saludo

Usuario: Hola

Chatbot: Hola, en que puedo ayudarte, recuerda que las preguntas deben ser sobre las becas de la UTPL.

3° Escenario: Palabras claves (Entidades)

Usuario: Apoyo solidario

Chatbot: Esta beca será asignada de acuerdo con la situación socioeconómica del estudiante, previa revisión de los documentos, informe emitido por el trabajador social y confirmación posterior con la realización de una visita domiciliaria.

4° Escenario: Oraciones con palabras claves (Entidades)

Usuario: ¿Me puedes decir cuáles son los requisitos de la beca deportiva?

Chatbot: Los requisitos son:

1. Imprimir el formulario on-line y adjuntar una foto tamaño carné.

2. Copia de la cédula de identidad del estudiante

3. Informe de desempeño del estudiante certificado por la Dirección General de Misiones universitarias-Área de Deporte Recreación y Cultura de acuerdo con los parámetros establecidos por el Comité de Becas de la UTPL.

4. En el caso de los deportistas de alto rendimiento se solicitará un informe al Área de Deporte Recreación y Cultura de la universidad.

5° Escenario: Oraciones o preguntas desconocidas

Usuario: ¿Cuál es el nombre del canciller actualmente?

Chatbot: Por favor revise su pregunta y vuelva a realizarla.

6° Escenario: Despedida

Usuario: Adiós

Chatbot: Hasta luego, espero haber ayudado.

2.3. Protocolo de atención al usuario mediante chat

Con la descripción de los posibles escenarios se puede definir un protocolo donde se ofrecen ideas y soluciones de diálogos que pueden ser utilizados para obtener un trato más amigable con el usuario, de esta manera se mejora positivamente su satisfacción y su percepción sobre la calidad de la institución.

Este protocolo contiene las respuestas que se debe seguir al momento de realizar un acercamiento con un usuario y de esta manera brindar un servicio de calidad a la persona que lo necesite y sirve como guía para la definición del flujo conversacional del chatbot.

El siguiente protocolo consta de cuatro puntos que permiten dar una atención más natural para que el usuario pueda llegar a obtener la respuesta o solución de manera clara y precisa sobre la situación de las becas en la UTPL.

Saludo inicial
- Bienvenido, Buenos días, Saludos. - Se debe aclarar el área en la que labora. - Mi nombre es …, Soy ……, en que le puedo ayudar?
En este punto es fundamental que el asistente salude al usuario de inmediato, cabe destacar que el saludo inicial es un gesto de educación y de respeto hacia los usuarios. El éxito se centra explicar sobre la situación o contexto que se va desarrollar la conversación que en este caso será sobre becas.

Análisis, comprensión y Respuesta.
Se debe esperar a que el usuario realice su consulta o duda. Si es necesario el asistente preguntar más para poder comprender bien la pregunta que formula el usuario.
Al contar con la respuesta que requiere el usuario, esta debe ser transmitida en forma precisa y clara, en un lenguaje simple y sin tecnicismos que puedan confundir al usuario, la respuesta puede ir acompañada de algún recurso informativo como, enlaces a los sitios web, documentos digitales u otros recursos multimedia. Si la consulta no es competencia del asistente, se debe indicar que soluciones puede resolver el asistente.

Verificación de la solución.
- ¿La información suministrada fue clara? - ¿Alguna otra duda o consulta? - ¿Fue clara la información suministrada?
Dado que muchos usuarios no entienden el mensaje entregado y no vuelven a preguntar, el asistente debe asegurarse de que la información entregada fue entendida. Si la información suministrada no resuelve las dudas del usuario, se debe comunicar otros medios de contacto como números telefónicos o correos electrónicos.

Despedida.
- Hasta luego, que tenga un buen día - Mensaje de despedida y recordar los diferentes puntos de contacto que se tiene a disposición
Despídase de manera cordial, con disposición de seguir atendiendo las inquietudes y comunicándole otros medios de acceso a la misma información. La despedida busca ser la

culminación de un proceso satisfactorio en cuanto a la percepción de la relación y atención que da la institución.

CAPITULO III.
DESARROLLO DEL CHATBOT

3.1. Planeación

La planeación utiliza historias de usuario para definir los requerimientos principales que debe tener el sistema mirándolo desde el punto de vista del cliente.

En la Tabla 6 se definen los roles de los interesados que están involucrados directamente con el proyecto y permiten definir las historias de usuario.

Tabla 6.Descripcion de roles en Historias de Usuario

Rol	Descripción
Administrador	Director y autor del trabajo de titulación.
Funcionario	Personas que laboran en la Institución y tienen conocimiento sobre las becas en la UTPL.
Usuario Final	Personas que usarán el chat y servirán para realizar pruebas del chat

Fuente: Autor
Elaboración: Autor

En este proyecto se ha planteado las siguientes historias de usuario que se exponen en las Tablas desde la 7 a la 13 conforme a los objetivos planteados.

Tabla 7. Historia de usuario 1: Inicio de chatbot

Historia de Usuario	
Usuario: **Usuario Final, Funcionario**	
Nombre historia: **Inicio de chatbot**	
Prioridad en negocio: **Alta**	Riesgo en desarrollo: **Medio**
Puntos estimados: **4**	Iteración asignada: 1
Programador responsable: **Alcides Toledo**	
Descripción: • El servicio de chatbot debe estar siempre en línea. • Al iniciar el chatbot o asistente virtual, este debe iniciar la conversación de inmediato con un saludo y aclarando al usuario que comenzara un dialogo con un chatbot. • Si el servicio de chatbot no se encuentra en línea, en vez de enviar el mensaje de saludo se debe enviar un mensaje aclarando que el servicio no está disponible • El chatbot debe proporcionar un campo para escribir la pregunta del usuario	
Observaciones:	

Fuente: Autor
Elaboración: Autor

Tabla 8. Historia de usuario 2: Realizar consulta

Historia de Usuario	
Usuario: Usuario Final, Funcionario	
Nombre historia: Realizar consulta	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Medio
Puntos estimados: 3	**Iteración asignada:** 1
Programador responsable: Alcides Toledo	
Descripción: • Cuando el usuario escribe su consulta en el campo de texto, el sistema debe detectar la pulsación de la tecla “Enter” o el botón a un lado del texto, para que esta se enviada al servidor y esta sea procesada debidamente.	
Observaciones:	

Fuente: Autor
Elaboración: Autor

Tabla 9. Historia de usuario 3: Respuesta de chatbot

Historia de Usuario	
Usuario: Administrador	
Nombre historia: Respuesta de chatbot	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Medio
Puntos estimados: 3	**Iteración asignada:** 2
Programador responsable: Alcides Toledo	
Descripción: • Cuando el usuario envié su consulta al servidor, el servicio debe buscar la respuesta indicada con respecto a la información de entrada. • El sistema debe responder tanto en forma de textual como auditiva. • El chatbot debe devolver una respuesta relevante a las consultas hechas, por ejemplo, si preguntas sobre “fechas de postulación”, la respuesta que devuelve debe ser acerca de las fechas. • Una vez dada la respuesta por el chatbot, este debe preguntar si desea continuar con la conversación, si el usuario responde con un “Si”, el chatbot debe responder	

de manera que se pueda continuar con el dialogo, en caso de que la respuesta sea "no", el chatbot se despedirá y termina la conversación, pero el chat seguirá activo.
Observaciones:

Fuente: Autor
Elaboración: Autor

Tabla 10. Historia de usuario 4: Almacenar dialogo

Historia de Usuario	
Usuario: Administrador, Funcionario	
Nombre historia: Almacenar dialogo	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Medio
Puntos estimados: 2	**Iteración asignada:** 2
Programador responsable: Alcides Toledo	
Descripción: • El sistema debe mantener un registro del dialogo actual de tal manera que estos registros o logs puedan ser usados en diferentes ámbitos.	
Observaciones:	

Fuente: Autor
Elaboración: Autor

Tabla 11. Historia de usuario 5: Consumo de servicio

Historia de Usuario	
Usuario: Administrador	
Nombre historia: Consumo de servicio	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Medio
Puntos estimados: 5	**Iteración asignada:** 3
Programador responsable: Alcides Toledo	
Descripción:	

• El servicio del chatbot debe ser consumido a través de una página web, esta página web debe ser amigable para el usuario y debe estar disponible 99% del tiempo. • El servicio debe tener la capacidad de dar soporte a todos los usuarios que usen el chat a la vez, sin afectar su rendimiento o funcionalidad. • El servicio debe ser ejecutado desde cualquier navegador que usen JavaScript, entre los cuales están Google Chrome y Mozilla Firefox, así como también en tablets o móviles.
Observaciones:

Fuente: Autor
Elaboración: Autor

Tabla 12. Historia de usuario 6: Ayuda para uso de chatbot

Historia de Usuario	
Usuario: Administrador, Funcionario	
Nombre historia: Ayuda para uso de chatbot	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Medio
Puntos estimados: 5	**Iteración asignada:** 3
Programador responsable: Alcides Toledo	
Descripción: • El chatbot debe ofrecer una funcionalidad de ayuda con una explicación de cómo un usuario puede interactuar con él. • Si el usuario desea contactarse de otra forma, el chatbot debe proporcionar información alterna para ponerse en contacto con los funcionarios encargados sobre temas de becas.	
Observaciones:	

Fuente: Autor
Elaboración: Autor

Tabla 13. Historia de usuario 7: Validación de la información

Historia de Usuario	
Usuario: Administrador, Funcionario	
Nombre historia: Validación de la información	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Medio
Puntos estimados: 3	**Iteración asignada:** 3
Programador responsable: Alcides Toledo	
Descripción: • Es necesario validar la información que servirá para crear los diálogos del chatbot y debe ser obtenida de fuentes confiables, como el Departamento de Bienestar Estudiantil o el Balcón de Servicios Estudiantiles.	
Observaciones:	

Fuente: Autor
Elaboración: Autor

3.2. Diseño

En esta sección se realiza el diseño de chatbot donde se utiliza las tarjetas CRC como recomienda la metodología de Programación Extrema, además se incluyen otros artefactos como la arquitectura de la solución.

Tabla 14. Tarjeta CRC Servidor

Nombre de la clase: Servidor	
Responsabilidades	**Colaboradores**
Se encarga de comprobar y de realizar la conexión con el servicio de Watson Developer Cloud	
Gestiona la conexión a la BD y almacena las conversaciones	
Activa el servicio para que pueda ser consumido a través de una petición HTTP POST	

Fuente: Autor
Elaboración: Autor

Tabla 15. Tarjeta CRC Cliente

Nombre de la clase: Cliente	
Responsabilidades	**Colaboradores**
Realiza la llamada al servidor cuando se realiza una consulta sobre becas	Servidor
Envía la respuesta chatbot al servicio de conversión de texto a voz de Watson Developer Cloud	
Se encarga de presentar la interfaz del chat y de cargar todas las funciones del chat.	

Fuente: Autor
Elaboración: Autor

3.2.1. Arquitectura de la solución

En la Figura 14 se muestra todos componentes que intervienen, su funcionalidad y la relación que existe con cada uno de ellos a nivel general.

En el diagrama la aplicación se compone de 3 capas que son la capa de presentación, capa de negocio y la capa de datos, los mismos que se comunican por protocolos HTTP, además los lenguajes de programación que se utilizan son JavaScript tanto para el lado del cliente como para el lado del servidor. El Servidor se conecta con la base de datos PostgreSQL para almacenar cada conversación iniciada y también se conecta con los servicios de IBM Watson llamando así el chatbot.

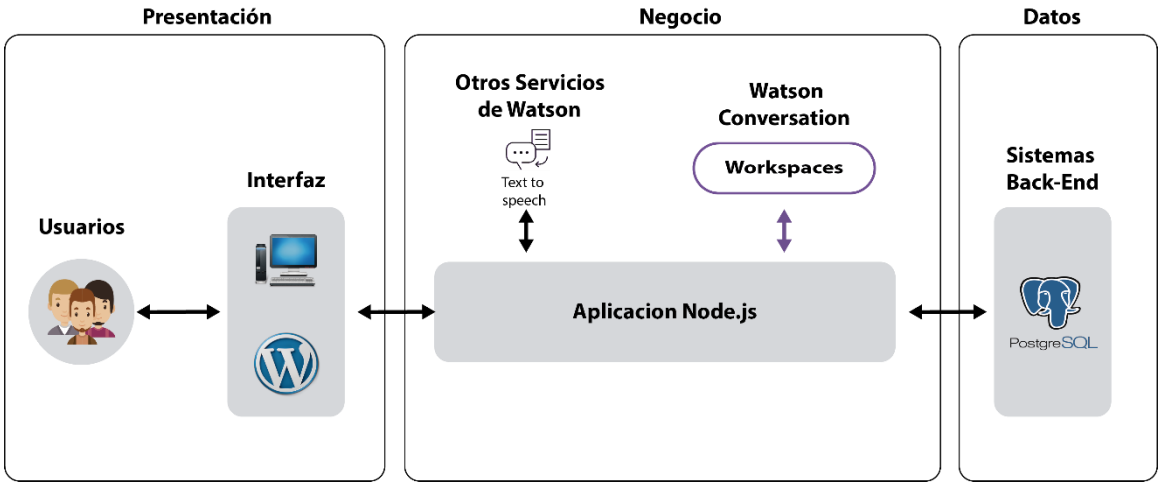


Figura 14. Arquitectura de la solución
Fuente: Autor
Elaboración: Autor

Lo usuarios son las personas que se conectan a la interfaz, los usuarios pueden ingresar cualquier cantidad de consultas. Los mensajes del usuario se representan con la letra “U” y

los mensajes del chatbot se representan con el logo de la Universidad Técnica Particular de Loja, de esta manera será más fácil para el usuario identificar los mensajes que ingrese.

La interfaz es el módulo que interactúa con el usuario que se conecte a sitio y necesite usar el chatbot. Está compuesto por un área donde se ven los mensajes que escribe el usuario y el chatbot, un campo de texto donde el usuario ingresa sus consultas, además de un botón para activar y desactivar el conversor de texto a voz, un botón de ayuda que retorna un mensaje de los temas que puede contestar el chatbot. Además de contar con una página externa donde se despliega el chatbot, también se cuenta con un plugin desarrollado para Wordpress.

El módulo de servidor Node.js se encarga de conectar la interfaz de usuario con el resto de componentes. Es llamado desde la interfaz e ira generando las respuestas que debe mostrar al usuario tanto en forma de texto como de audio.

El módulo de Watson Conversation contiene los *workspaces* o espacios de trabajo, en cada *workspace* se encuentra un chatbot en concreto, donde se encuentra definido las intenciones, entidades y el dialogo que identifica la entrada del usuario y responde la información que este ingresada en el chatbot.

El módulo de Servicios de Watson se encarga de gestionar las entradas relacionadas del usuario, cuando el usuario realiza una consulta, el módulo de chatbot es el primero en ser llamado, de ahí cuando retorna la respuesta al servidor este se encarga de llamar al servicio de *text to speech* (TTS) para realizar la conversión y retornar ambas respuestas al usuario.

La base de datos usa una API que proporciona Bluemix para conectar a ElephantSQL que es un servicio de almacenamiento en bases de datos Postgres. La base de datos consta de dos tablas en la que se almacenan los siguientes campos:

Tabla conversación

En la tabla conversación se almacena campos que identifican la entrada de un usuario al sistema y todas las consultas realizadas por dicho usuario.

- conversation_id: Identificador de la sesión y de la conversación.
- mensaje: Es la consulta del usuario que realiza al chatbot.
- respuesta: La respuesta que da el chatbot a una determinada consulta del usuario.
- fecha_hora: almacena la fecha y hora del mensaje de un usuario en formato UTC.
- ip: Dirección lógica que identifica a un usuario.
- hora: Hora que el usuario envió una consulta.
- fecha: Fecha en que el usuario envió una consulta.

- country: Nombre del país del usuario.
- región: Nombre de la provincia del usuario.
- city: Nombre de la ciudad del usuario.

Tabla calificación

En la tabla calificación se almacena la calificación que da el usuario al sistema.

- id: Identificador del registro.
- conversation_id: Identificador de la sesión y de la conversación.
- calificación: valor que da el usuario al servicio, varía entre 1 y 5.

3.3. Codificación

El presente trabajo de titulación se contempla el desarrollo de la solución para una página web usando la metodología de desarrollo ágil XP y la plataforma Bluemix como herramientas para codificar un chatbot responda preguntas frecuentes referentes a becas en la UTPL.

3.3.1. Codificación de la arquitectura

A continuación, se codifica la solución web propuesta en la Figura 14, que es utilizada por los administradores los cuales gestionan la interfaz del chat, así como también el servicio que activa su funcionamiento.

Para el desarrollo de la solución se realiza con la arquitectura basada en capas. La capa de presentación muestra la interfaz de usuario, la capa de negocio está a cargo de la comunicación entre capas y la capa de Datos contiene funciones para almacenar las interacciones de los usuarios. Además de la utilización del *framework* Bootstrap para el diseño de la interfaz, la librería JQuery para controlar los eventos en la interfaz, varias dependencias como Pg para la capa de datos y Watson Developer Cloud para la comunicación con servicios de *Conversation* y *text-to-speech*, entre otras herramientas para el correcto funcionamiento de la aplicación.

- **Arquitectura por capas**

A continuación, se describe cada una de las capas donde se ha trabajado para el desarrollo del proyecto.

Capa 1 (Presentación).- La presentación se encarga de visualizar la información hacia el usuario final. Generalmente el código que se visualiza en esta capa no con clases sino archivos HTML. En la figura 15 se muestra parte del código

```
17  <body>
18  <div class="container">
19      <div class="row">
20          <div class="chat_window">
21              <div class="top_menu">
22                  <div class="title">
23                          <i class="fa fa fa-volume-up fa-pull-left fa-border playing" aria-hidden="true" id='vol'></i>
24                      <font><b>Asistente Virtual de becas UTPL</b></font>
25                      <i class="fa fa-question-circle fa-border fa-pull-right" aria-hidden="true" id='help'></i>
26                  </div>
27              </div>
28              <audio id="speech"></audio>
29              <ul class="messages"></ul>
30              <div class="clearfix">
31                  <div class="input-group">
32                      <input type="text" class="form-control" placeholder="Ingrese aqui su pregunta" id="chat" required>
33                          <span class="enviar_mensaje input-group-btn">
34                              <button type="button" class="btn btn-default" aria-label="Help">
35                                  <span class="fa fa-paper-plane"></span>
36                              </button>
37                          </span>
38                  </div>
39                  <small id="arrow"></small>
40              </div>
41          </div>
42
43              </div>
44          </div>
```

Figura 15. Código de la vista del chat
Fuente: Autor
Elaboración: Autor

Capa 2 (Negocio).- La capa de negocio actúa como intermediario entre la capa de datos y la capa de presentación y viceversa, todas la consultas que se hagan son atendidas por el controlador para que se comunique con el modelo y este almacene la información. En la Figura 16 se resalta lo más importante del código en la capa de negocio que es la conexión con el servicio de *Conversation* y la base de datos.

```
77  app.post('/converse', function(req, res, next) {
78    var geo = ip2loc.IP2Location_get_all(getClientIP(req));
79    try{
80      var d = new Date(req.body.context.dateC);
81    }
82    catch(e){
83      var d = new Date();
84    }
85    var payload = {
86      workspace_id: workspace,
87      context: {},
88      input: {}
89    };
90
91    if (req.body) {
92        if ( req.body.input ) {
93          payload.input = {text: req.body.input};
94        }
95        if (req.body.context) {
96          payload.context = req.body.context;
97        }
98    }else{
99        payload = {};
100     }
101
102   conversation.message(payload, function(err, data){
103     if ( err ) {
104       console.log(err);
105     }else{
106       insertBD(data.context.conversation_id, data.input.text, data.output.text['0'], d, getClientIP(req), geo.country_long, geo.region, geo.city);
107       return res.json(data);
108     }
109   });
110
111 });
```

Figura 16. Código de la capa controlador
Fuente: Autor
Elaboración: Autor

Capa 3 (Datos).- En esta capa se trabaja con los datos por lo tanto contiene funciones de insertar sobre una base de datos, se plantea una función donde ejecuta instrucciones para almacenar la información para lo cual se utiliza JavaScript como lenguaje de programación y

la dependencia “pg” como driver para establecer la conexión a PostgreSQL. En la figura 17 se muestra el código de la capa de Datos.

```
145  var insertBD = function(conversation_id, mensaje, respuesta, da, ip, country, region, city){
146      var conString = "postgres://goorltcp:y4St6yt1kweA7DPQpcLdsMcX9qq67LOF@hanno.db.elephantsql.com:5432/goorltcp";
147      var client = new pg.client(conString);
148      client.connect(function(err) {
149              if(err) {
150                  return console.error('could not connect to postgres', err);
151              }else{
152                return console.log(" connection succes to bd");
153              }
154          });
155      client.query('INSERT into conversacion (conversation_id, mensaje, respuesta, fecha_hora, ip, hora, fecha,
           country, region, city) values($1,$2,$3,$4,$5,$6,$7,$8,$9,$10)',
156          [conversation_id, mensaje, respuesta, da, ip, (da.getHours()+":"+da.getMinutes()+":"+da.getSeconds()), da,
             country, region, city],function(err, result) {
157      if(err) {
158        return console.error('error running query', err);
159      }
160
161        client.end();
162      });
163
164  }
165
```

Figura 17. Código de la capa de datos
Fuente: Autor
Elaboración: Autor

3.3.2. Herramientas de Desarrollo

3.3.2.1. Plataforma de chatbot

La plataforma para la creación y desarrollo del servicio de chatbot es sobre IBM Bluemix, que facilita la creación de servicios cognitivos de IBM Watson. Hay que tener en cuenta el diseño del flujo conversacional basado en las intenciones y entidades reconocidas para que exista un dialogo fluido.

En el anexo A y B se describen la creación del servicio de chatbot y el proceso para diseñar las intenciones, entidades, y el flujo conversacional.

3.3.2.2. Frameworks

Un framework es una estructura tecnológica definida la cual contiene una serie de artefactos o módulos que sirven de base para el desarrollo de software. Dentro del desarrollo de aplicaciones web se encuentran una gran cantidad de frameworks, particularmente de acuerdo con los requerimientos del sistema se ha definido utilizar los siguientes:

- **Bootstrap.-** Es un framework de código abierto para desarrollar con HTML, CSS y JavaScript. Se puede crear rápidamente prototipos con interfaces responsivas que se adapten a cualquier dispositivo.
- **Qunit.-** es una herramienta para realizar pruebas unitarias al código y complementos de la aplicación capaz de probar cualquier código JavaScript incluyendo a JQuery.

- **Mocha.-** está desarrollado en JavaScript que se ejecuta en Node.js y se lo utiliza como herramienta para realizar pruebas unitarias en el servidor. Las pruebas de Mocha se ejecutan en serie, lo que permite informes flexibles y precisos de la aplicación.

3.3.2.3. Librería

Una librería es una serie de funciones y métodos que son utilizados en un programa para poder facilitar la implementación de dicho programa de la cuales utilizó la siguiente:

- **JQuery.-** esta librería permite facilitar la manera de interactuar con los documentos HTML, manejar eventos, desarrollar animaciones y agregar interacción a páginas web. Ofrece una serie de funcionalidades basadas en JavaScript permitiendo lograr grandes resultados en menos tiempo y espacio.

- **Chai.**- librería que proporciona utilidades para la ejecución y reporte de test unitarios. Se puede integrar con cualquier framework de pruebas en JavaScript. Ofrece varias interfaces como: **assert, expect** y **should**.

3.3.2.4. Dependencias

Las dependencias son aplicaciones o biblioteca requerida por el sistema para poder funcionar correctamente, al faltar alguna de ellas el sistema no funciona de la manera esperada. A continuación, detallan las más importantes:

- **Watson Developer Cloud.-** Esta dependencia es un conjunto de funciones que permiten conectarse a través de un servidor node.js a los servicios cognitivos que proporciona Watson de IBM, como Conversation y Text-to-Speech
- **Pg.-** Es una colección de funciones para node.js para interactuar con bases de datos PostgreSQL. Permitiendo realizar las conexiones y realizar consultas de crear, leer, actualizar y borrar.

3.3.2.5. Servidor de la aplicación

- **Cloud Foundry.-** Es una plataforma como servicio de código abierto (PaaS) que le permite crear y desplegar aplicaciones en la nube ofreciendo marcos de desarrollo para varios lenguajes de desarrollo.

- **Node.js.-** Es una plataforma JavaScript del lado del servidor, que se basa en eventos permitiendo una conexión asíncrona con el cliente permitiendo una conexión más rápida.

3.3.2.6. Lenguaje de programación

Para el desarrollo de la aplicación web se ha utilizado como lenguaje de programación JavaScript tanto de lado del servidor para realizar en enlace a los servicios del chatbot y a la base de datos, de la misma forma del lado del cliente para interactuar el usuario con el chatbot.

3.3.2.7. Diseño y modelado de la interfaz

La interfaz cumple un papel importante en cuanto a la funcionalidad de la aplicación ya que a partir de estas el dependerá que el usuario le sea más fácil manejo del sistema, mientras que el diseño sea sencillo será mucho mejor.

Para el diseño de la interfaz del chatbot se ha utilizado el framework Bootstrap el cual permite que el chat se adapte a diferentes tamaños de pantalla, es decir de manera responsiva, además de un diseño amigable y funcional a la vez.

Para el diseño de la interfaz del plugin en Wordpress se ha utilizado JQuery y CSS junto HTML sin el uso del framework Bootstrap para que no interfiera con los estilos y estructura del tema asignado de Wordpress.

En el anexo H se encuentran imágenes de los prototipos.

3.4. Pruebas

La última actividad dentro de la metodología de desarrollo Programación Extrema corresponde a las pruebas, las cuales miden el nivel de calidad de software, el nivel de satisfacción de los usuarios y confirman el correcto funcionamiento del sistema, así como también los verifica los pasos necesarios que se deben seguir para la puesta en producción del software.

El presente trabajo de titulación nace de la iniciativa de contar una herramienta complementaria que proporcione información referente a las becas, motivo por el cual los requisitos han sido planteados en base a las necesidades que existen dentro del balcón de servicios, del departamento de bienestar estudiantil y de los estudiantes de pregrado, por lo que las pruebas que se han realizado en esta etapa han sido validadas por el director de trabajo de titulación, el balcón de servicios estudiantiles y el departamento de bienestar estudiantil, explotando las diferentes funcionalidades con las que cuenta el sistema.

Dentro de los que se definió en el capítulo dos acerca de la metodología de desarrollo ágil XP manifiesta que, con el objetivo de verificar el buen funcionamiento del sistema, las pruebas deben ser realizadas por los desarrolladores junto con el cliente, en este caso el cliente es el balcón de servicios, el departamento de bienestar estudiantil y junto al director se coordinó para la validación de las pruebas respectivas.

3.4.1. Pruebas de aceptación

La utilización de las pruebas de aceptación garantiza que el desarrollo cumple con los requisitos del cliente. Las pruebas de aceptación se realizan al finalizar cada iteración con el objetivo de validar el correcto funcionamiento del sistema de esta manera se logra determinar en forma acertada los errores y cambios que se deben realizar. En el trabajo de titulación se considera al balcón de servicios, al departamento de bienestar estudiantil y al director del presente trabajo como clientes finales para realizar las validaciones respectivas. En la tabla 16 se detalla el formato a utilizar.

Tabla 16. Formato de prueba de aceptación

Prueba de aceptación	
Número:	**N° de historia de usuario**
Historia de usuario	
Condiciones de ejecución	
Pasos de ejecución:	
Resultados esperados:	
Evaluación de la prueba:	

Fuente: (Pressman & Troya, 2007)
Elaboración: Alcides Toledo

Detalle

- **Número:** Número de la prueba de aceptación.
- **Numero historia de usuario:** Número de la historia de usuario a la que se hace referencia.
- **Descripción:** Resumen de la historia de usuario a aprobarse.
- **Condiciones de ejecución:** Condiciones previas requeridas.
- **Pasos de ejecución:** Pasos que el usuario deberá hacer para ejecutar la acción.
- **Resultados esperados:** Respuesta obtenida del sistema posterior a la ejecución de la acción.
- **Evaluación de la prueba:** nivel de satisfacción con respecto al resultado obtenido del sistema.

3.4.2. Pruebas unitarias

Es necesario realizar las pruebas unitarias las cuales consisten en probar cada uno de los métodos y verificar que todo funcione de forma correcta. Para ejecutar las pruebas unitarias se utiliza la herramienta Mocha.js, el cual es un framework que prueba JavaScript y que se puede ejecutar en un servidor Node.js y el navegador, haciendo que los test sean simples

dando informes flexibles y precisos, al mismo tiempo asigna las excepciones no detectadas a los casos de prueba correctos. Los resultados están disponibles en el anexo F, donde muestra el correcto funcionamiento de cada método de cada clase.

3.4.3. Pruebas de calidad de software

La calidad de software es un conjunto de cualidades que determinan aspectos y características de utilidad, la calidad es sinónimo de eficiencia, flexibilidad, corrección, confiabilidad, mantenibilidad, portabilidad, usabilidad, seguridad e integridad. Para realizar estas pruebas se utiliza SonarQube.

SonarQube es una plataforma de código abierto la utilizan los equipos de desarrollo para controlar la calidad del código fuente de una aplicación. Fue desarrollado con el principal objetivo de hacer accesible la administración de la calidad del código con un mínimo esfuerzo, ofreciendo distintas funcionalidades como: un analizador de código, una herramienta de reportes, un módulo que detecta defectos y una función para regresar los cambios realizados en el código. Los resultados de las pruebas realizadas están disponibles en el anexo G.

3.4.4. Pruebas de usabilidad

Estas pruebas las realizó junto a los interesados, los cuales permiten validar en un ambiente real que el sistema funcione correctamente y de esta manera verificar que el prototipo cumple con los objetivos propuestos por el director del proyecto. Las validaciones de este tipo de prueba se detallan en las estadísticas que proporciona Watson Conversation y las encuestas realizadas a diferentes funcionarios del balcón de servicios y el departamento de bienestar estudiantil, también se realizó la encuesta a estudiantes de la UTPL.

La encuesta que se realizó es un cuestionario de SUS (System Usability Scale) (Brooke, 2013) que ayuda determinar la precisión, funcionalidad, efectividad y satisfacción de la interfaz de la aplicación.

SUS es una encuesta que se utiliza para probar la usabilidad de los sistemas informáticos. En este caso se utilizó para evaluar para evaluar la interfaz del chatbot de becas. La encuesta se aplicó a los funcionarios del Balcón de Servicios Estudiantiles, al departamento de Bienestar Estudiantil y a los estudiantes de la materia de Inteligencia Artificial Avanzada de la titulación de Sistemas Informáticos y Computación de la UTPL. A continuación, en la tabla 17 se encuentran el formato de las preguntas de la encuesta SUS y los resultados obtenidos al aplicarla.

Tabla 17. Cuestionario SUS

Opinión	Grado de Acuerdo				
	Muy de acuerdo	De acuerdo	Ni acuerdo ni desacuerdo	En desacuerdo	Muy en desacuerdo
1. ¿Usaría con frecuencia el chatbot?	23%	58%	19%	0%	0%
2. ¿El chat es complejo?	4%	15%	8%	58%	15%
3. ¿Fue fácil utilizar el chatbot?	46%	50%	4%	0%	0%
4. ¿Consideraría que necesita el apoyo de un experto para utilizar el chatbot?	0%	4%	8%	34%	54%
5. ¿Las diferentes opciones del chatbot están bien integradas?	15%	46%	31%	8%	0%
6. ¿Hubo demasiada inconsistencia (diferentes colores y no se podía leer adecuadamente) en el chatbot?	4%	8%	38%	35%	15%
7. ¿Considera que el chatbot es fácil de usar?	38%	58%	4%	0%	0%
8. ¿El chatbot es muy difícil de usar?	7%	4%	4%	31%	54%
9. ¿Tuvo confianza al utilizar la página web de chatbot?	35%	38%	19%	8%	0%
10. ¿Cree usted qué necesitaría tener conocimientos previos para usar el chatbot?	0%	4%	15%	42%	39%

Fuente: (Brooke, 2013)
Elaboración: Alcides Toledo

Las preguntas que se presentan en la Tabla 18 están destinadas a conocer el nivel de satisfacción con el servicio proporcionado por el chatbot. Mediante esto se quiere conocer lo que el usuario piensa sobre este servicio.

Tabla 18. Preguntas de satisfacción al cliente

Opinión	Grado de Acuerdo				
	Muy satisfecho	Satisfecho	Neutro	Insatisfecho	Muy insatisfecho
¿Considera que el chatbot responde correctamente la consulta que realizó?	11%	58%	23%	8%	0%

Opinión	Grado de Acuerdo				
	Muy satisfecho	Satisfecho	Neutro	Insatisfecho	Muy insatisfecho
¿Considera que el chatbot cuenta con la información necesaria acerca de las becas?	23%	46%	23%	8%	0%
¿Considera que el chatbot intenta mantener una conversación con el usuario como si fuera humano?	11%	50%	31%	8%	0%

Fuente: Alcides Toledo
Elaboración: Alcides Toledo

3.4.5. Interpretación de los resultados

En esta sección se detallan los resultados obtenidos de aplicar el cuestionario SUS, que permite evaluar funcionalidad, efectividad y satisfacción del usuario. La encuesta está compuesta por 10 preguntas que tienen como finalidad determinar la satisfacción del usuario con respecto a la interfaz de la aplicación como son: Complejidad, facilidad de uso, consistencia, confiabilidad e integración.

La medición de la escala SUS genera un único número representando una medida compuesta de usabilidad del sistema global sometido a estudio. Las puntuaciones independientes de cada pregunta no son significativas por sí mismas.

Para calcular la puntuación del SUS, hay que sumar primero las contribuciones de cada punto. La contribución de cada punto valdrá entre 0 y 4. Para los puntos 1, 3, 5, 7 y 9, la contribución será la posición de la escala menos 1. Para los puntos 2, 4, 6, 8 y 10, la contribución será 5 menos la posición en la escala. Se multiplica la suma de los resultados por 2.5 para obtener el valor global del SUS. El resultado estará entre 0 y 100 y que correlacionan con un listado de adjetivos en función a un intervalo de dicha puntuación que se presentan en la Figura 18.

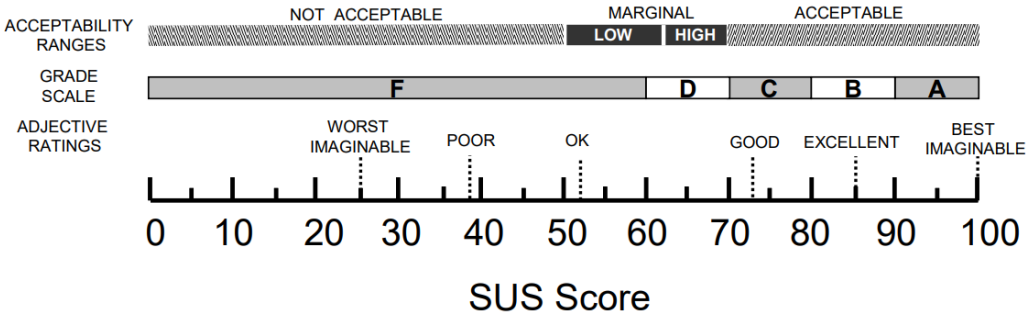


Figura 18. Puntuación SUS.
Fuente: (Bangor, Kortum, & Miller, 2009)
Elaboración: (Bangor et al., 2009)

Los estudiantes de la Institución, el Balcón de Servicios Estudiantiles y el Área de Bienestar Estudiantil valoran como Muy Buena a la aplicación dando una puntuación de 76 habiendo sido aceptada por los encuestados calificándola entre buena y excelente, esto permite tener en cuenta los aspectos que el usuario a detectado ciertas dificultades de utilización de la aplicación, y realizar modificaciones de una forma más rápida.

Las siguientes preguntas se las realizó con el objetivo de conocer la satisfacción del usuario con respecto al chatbot y si las respuestas obtenidas eran congruentes con la entrada del usuario.

1. ¿Considera que el chatbot responde correctamente la consulta que realizó?

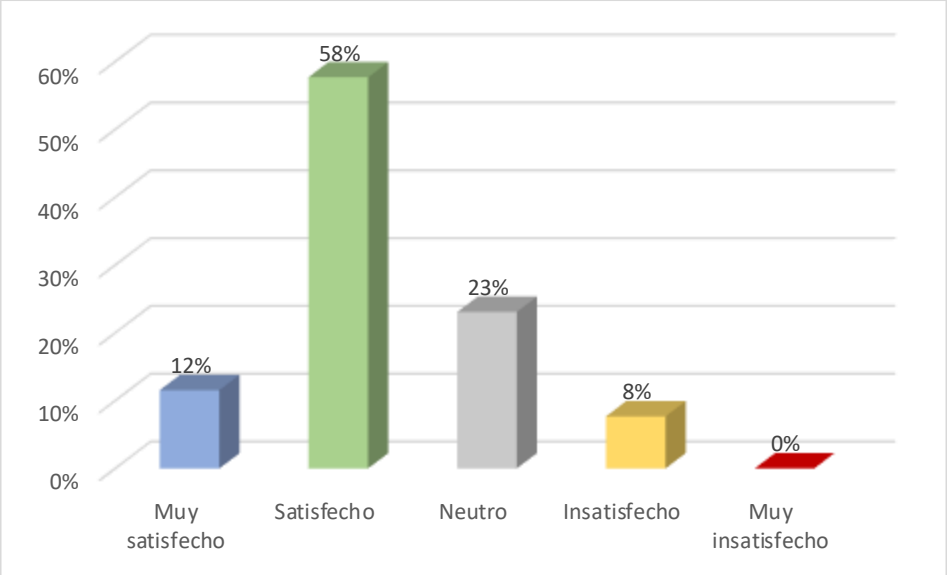


Figura 19. ¿Considera que el chatbot responde correctamente la consulta que realizó?
Fuente: Autor
Elaboración: Autor

La figura 19 muestra que cerca del 70% de personas encuestadas están de acuerdo en que el chatbot responde correctamente a las preguntas planteadas. Además, el 8% estuvo

insatisfecho con las respuestas que obtuvo, esto es debido a que las consultas que hizo el usuario en ese momento no se encontraban en la base de conocimientos o no tenía relación con el tema de becas. Comprobando así la congruencia entre la consulta del usuario y la respuesta que retorna el chatbot.

2. ¿Considera que el chatbot cuenta con la información necesaria acerca de las becas?

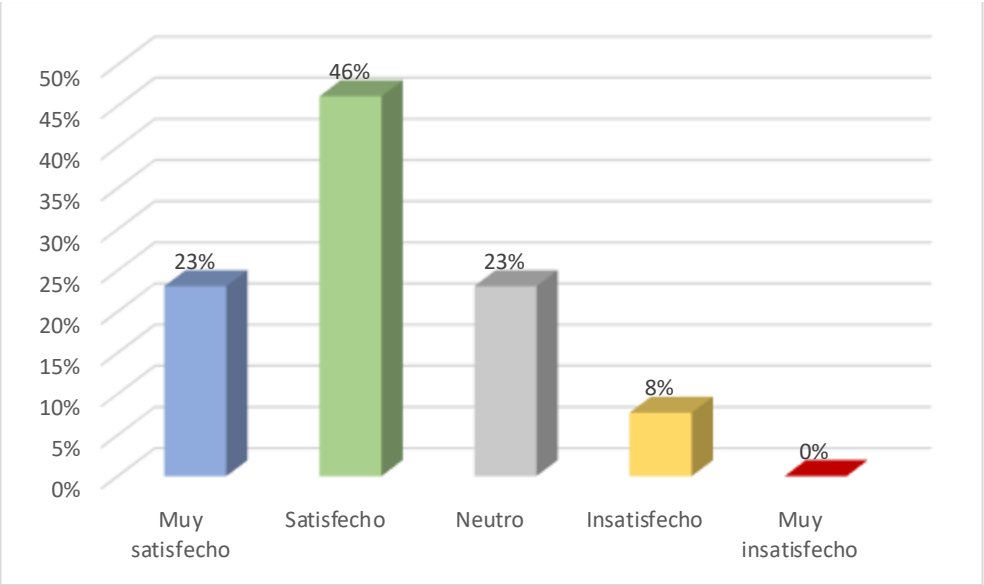


Figura 20. ¿Considera que el chatbot cuenta con la información necesaria acerca de las becas?
Fuente: Autor
Elaboración: Autor

En la Figura 20 se puede observar la opinión de los usuarios encuestados. El 69% de los usuarios están de acuerdo en que el chatbot contiene la información necesaria acerca de las becas en pregrado. Siendo el 23% muy satisfecho, el 46% satisfecho, el 23% neutro y finalmente el 8% insatisfecho.

3. **¿Considera que el chatbot intenta mantener una conversación con el usuario como si fuera humano?**

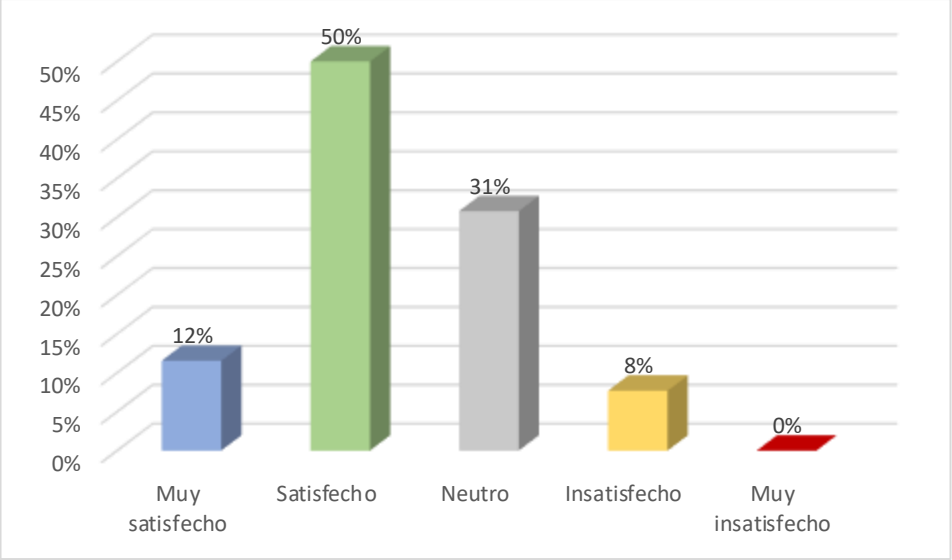


Figura 21. ¿Considera que el chatbot intenta mantener una conversación con el usuario como si fuera humano?
Fuente: Autor
Elaboración: Autor

En la Figura 21 se presenta los resultados, los encuestados consideran que el chatbot si mantiene una conversación con ellos generando interactividad. El 62% de los encuestados aceptaron el chatbot. Siendo, el 12% está muy satisfecho, el 50% satisfecho, el 31% es neutro y por último el 8% insatisfecho.

3.5. Implementación

Una vez realizadas las pruebas se ha hizo la implementación de la aplicación web y el plugin de chat en WordPress. La URL de la página web es http://becasutpl2.mybluemix.net/, y las URL's de los servicios web son:

- IBM Watson Conversation: http://becasutpl2.mybluemix.net/converse.
- IBM Watson text-to-speech: http://becasutpl2.mybluemix.net/audio.

La implementación de la aplicación se ha realizado siguiendo los pasos detallados en el anexo C. A continuación, se detalla los pasos respectivos:

- Tener cuenta activa en Bluemix (https://console.bluemix.net/).
- Tener creado un servicio de Watson Conversation.
- Tener instalado herramienta de líneas de comando (CLI) de Cloud Foundry.
- Cambiar las credenciales del servicio de Conversation en el código.

- Subir los archivos con CLI de Cloud Foundry

Para la instalación del plugin para la página de becas de la institución se detalla en el anexo H. Que de manera general se siguen los pasos a continuación:

- Ingresar al panel de administración del sitio.
- Ingresar a Plugins > Añadir nuevo
- Hacer clic en “Subir”, luego seleccionar el archivo en formato **.zip** y después hacer clic en “Instalar Ahora”.
- Activar el plugin y la instalación estará completada.
- Hacer clic en “Configuración” para editar los servicios web.

TRABAJOS FUTUROS

A continuación, se describen las mejoras al trabajo presentado:

- Aunque se chatbot se ha desarrollado de manera específica para responder un determinado tema, es posible utilizarlo como base para diseñar otros asistentes en distintas áreas que se requiera.
- Mejorar el chatbot incorporando un conversor de voz a texto aumentando la accesibilidad del chatbot a personas en situación de discapacidad.
- Mejorar la calidad del servicio incorporando otros idiomas que el chatbot pueda reconocer.

CONCLUSIONES

El desarrollo de un prototipo de Chatbot para el área de Becas de la UTPL, permitió el cumplimiento del objetivo principal del trabajo de titulación, facilitando así una comunicación interactiva entre los usuarios y la institución, logrando además llegar a las siguientes conclusiones:

- La utilización de la metodología de desarrollo ágil XP, permite la adaptación a los cambios recurrentes, reduciendo incongruencias de las respuestas y posibles defectos de integración.
- El servicio de IBM Watson Conversation, facilita el reconocimiento de consultas de los usuarios y poder responder adecuadamente a cada petición, permitiendo el entrenamiento de intenciones y entidades.
- Cuanto mayor cantidad de información se use para el entrenamiento del chatbot, mayor será el nivel de eficacia y precisión a las preguntas formuladas. Por este motivo se integró una base de datos al chatbot para almacenar las preguntas de los usuarios.
- La continua actualización e incremento de nuevas funcionalidades de la plataforma de Watson Conversation, vuelve imprescindible el constante estudio de la documentación oficial, así como la constante actualización de intenciones, entidades y diseño del diálogo.
- Los análisis de resultados de pruebas de usabilidad mediante la encuesta SUS, permiten evidenciar el alto nivel de aceptación de la herramienta en los usuarios.
- Con el uso de la encuesta de satisfacción al cliente, se pudo comprobar que el chatbot si responde correctamente a las preguntas hechas por los usuarios.
- La utilización de nuevas tecnologías en la implementación de servicios de IBM Watson, permiten su integración de forma modular, logrando una mejor reutilización de cada uno de sus elementos.

RECOMENDACIONES

En base al trabajo de titulación, se propone las siguientes recomendaciones:

- Mantener el chatbot actualizado usando su base de datos integrada; en caso de ser necesario crear nuevas intenciones, entidades y diálogos en la plataforma, para así ofrecer un servicio de calidad.
- Incentivar a la comunidad universitaria a la implementación de este tipo de chatbots en otros ámbitos como el balcón de servicios estudiantiles, en materias de la modalidad abierta y a distancia, entre otros.
- Agregar más respuestas o mejorar las existentes para permitir que el chatbot ofrezca una sensación más humana.
- Desarrollar un módulo para el CMS Drupal que le permita ser integrado con este u otros CMS's de la actualidad.
- Diseñar un esquema de reportes y estadísticas de las interacciones con el chatbot para evaluar el estatus actual de visitas

BIBLIOGRAFÍA

Bangor, A., Kortum, P., & Miller, J. (2009). Determining What Individual SUS Scores Mean: Adding an Adjective Rating Scale. *Journal of Usability Studies*, *4*(3), 114–123. Retrieved from http://www.usabilityprofessionals.org.

Bhatia, P., Gavaldà, M., & Einolghozati, A. (2017). soc2seq: Social Embedding meets Conversation Model. Retrieved from https://arxiv.org/pdf/1702.05512.pdf

Bhilegaonkar, A. (2016). *Machine Learning and Cognitive Computing: A Proposed Framework to Navigate the Opportunities*. Massachusetts Institute of Technology. Retrieved from https://dspace.mit.edu/bitstream/handle/1721.1/107589/974706790-MIT.pdf?sequence=1

Brooke, J. (2013). SUS: A Retrospective. *Journal of Usability Studies*. Retrieved from http://uxpajournal.org/wp-content/uploads/pdf/JUS_Brooke_February_2013.pdf

Canós, J., Letelier, P., & Penadés, M. (2003). *Metodologías Ágiles en el Desarrollo de Software*.

Cardoso, A. C., Bini, A., & Pérez Abelleira, M. A. (2015). Diseño de un sistema de búsqueda de respuestas para diversos tipos de preguntas. Retrieved from http://sedici.unlp.edu.ar/bitstream/handle/10915/52018/Documento_completo.pdf-PDFA.pdf?sequence=1

Cobos T, J. C. (2013). *Integración de un chatbot como habilidad de un robot social con gestor de dialogos*. Retrieved from http://repositorio.educacionsuperior.gob.ec/bitstream/28000/1201/1/T-SENESCYT-000332.pdf

Davydova, O. (2017). 25 Chatbot Platforms: A Comparative Table – Chatbots Journal. Retrieved May 29, 2017, from https://chatbotsjournal.com/25-chatbot-platforms-a-comparative-table-aeefc932eaff

Dialogflow. (2017a). Agents | Dialogflow. Retrieved August 2, 2017, from https://dialogflow.com/docs/agents

Dialogflow. (2017b). Introduction to Dialogflow. Retrieved May 17, 2017, from https://dialogflow.com/docs/getting-started/basics

Ferrer, J. (2002). Programación eXtrema y Software Libre.

Giugni, M., Vera, M., Díaz, A., & Cattafi, R. (2007). *Sistema hipermedia adaptativo para contenidos educativos, basado en tecnología de agentes de software*. Retrieved from

https://www.researchgate.net/profile/Ricardo_Cattafi/publication/237263550_Sistema_hipermedia_adaptativo_para_contenidos_educativos_basado_en_tecnologia_de_agentes_de_software/links/53f34da60cf256ab87b08988.pdf

Hernández, J. (2014). *Análisis y Desarrollo Web*. Retrieved from https://books.google.com.ec/books?id=nYDVBQAAQBAJ&printsec=frontcover#v=onepage&q&f=false

Hertzum, M. (2016). A usability test is not an interview., *23*, 82–84.

IBM. (1998). Rational Unified Process Best Practices for Software Development Teams.

IBM. (2017). Conversation | IBM Watson Developer Cloud. Retrieved May 16, 2017, from https://www.ibm.com/watson/developercloud/doc/conversation/index.html

ICONIX. (2016). ICONIX - Better Agile Methodology and Project Management. Retrieved December 15, 2016, from http://www.iconixsw.com/index.shtml

Inglada, V. (2002). *RT-Message: Desarrollo de Sistemas Multiagentes de Tiempo Real*. Retrieved from http://users.dsic.upv.es/grupos/ia/sma/thesis/pdf/TesisVicenteJ.pdf

Inteco. (2009). METODOLOGÍAS Y CICLOS DE VIDA. Retrieved from http://datateca.unad.edu.co/contenidos/301569/guia_de_ingenieria_del_software.pdf

Kelly III, J. E. (2015). Computing, cognition and the future of knowing. Retrieved from https://www.research.ibm.com/software/IBMResearch/multimedia/Computing_Cognition_WhitePaper.pdf

Letelier, P., & Penadés, M. C. (2006). Métodologías ágiles para el desarrollo de software: eXtreme Programming (XP). Retrieved from http://www.cyta.com.ar/ta0502/v5n2a1.htm

Microsoft. (2017). Language Understanding Intelligent Service (LUIS) in Azure | Microsoft Docs. Retrieved May 17, 2017, from https://docs.microsoft.com/en-us/azure/cognitive-services/luis/home

Morales-Rodríguez, M. L., & Domínguez-Martínez, J. R. (2011). *Agentes Conversacionales como un Sistema de Diálogo*. Retrieved from http://s3.amazonaws.com/academia.edu.documents/7349121/Agentes Conversacionales como un Sistema de Dialogo DEPI2011 JESUS.pdf?AWSAccessKeyId=AKIAIWOWYYGZ2Y53UL3A&Expires=1494515379&Signature=cxi1w78YsqBPxfBFKprziGVslCQ%3D&response-content-

Osuna, J., & Usero, A. (2004). *Proyecto de Desarrollo Software*. Retrieved from

http://users.dsic.upv.es/asignaturas/facultad/lsi/ejemploxp/Gestion_Proyecto.html

Pathak, S., & Mishra, N. (2016). Context Aware Restricted Tourism Domain Question Answering System. *2nd International Conference on Next Generation Computing Technologies (NGCT-2016)*, (October), 534–539.

Pressman, R., & Troya, J. (2007). *Ingeniería del software*. *CITEG Revista Arbitrada* (Vol. 1). https://doi.org/http://zeus.inf.ucv.cl/~bcrawford/Modelado%20UML/Ingenieria%20del%20Software%207ma.%20Ed.%20-%20Ian%20Sommerville.pdf

Pudaruth, S., Boodhoo, K., & Goolbudun, L. (2016). An intelligent question answering system for ICT. *International Conference on Electrical, Electronics, and Optimization Techniques, ICEEOT 2016*, 2895–2899. https://doi.org/10.1109/ICEEOT.2016.7755228

RAE. (2017a). Cognitivo. Retrieved June 14, 2017, from http://dle.rae.es/srv/search?w=cognitivo

RAE. (2017b). Computación. Retrieved June 14, 2017, from http://dle.rae.es/srv/search?m=30&w=computación

Ruiz, A. (2009). *Sistema Inteligente Conversacional para la Orientación Vocacional*. Universidad de Colima. Retrieved from http://digeset.ucol.mx/tesis_posgrado/Pdf/Ana_Claudia_Ruiz_Tadeo.pdf

Sommerville, I. (2011). *Software Engineering*. *Software Engineering*. https://doi.org/10.1111/j.1365-2362.2005.01463.x

Turner, M. S. V. (2006). *Microsoft solutions framework essentials : building successful technology solutions*. Microsoft Press.

Van Woudenberg, A. F. (2014). A Chatbot Dialogue Manager Chatbots and Dialogue Systems: A Hybrid Approach. Retrieved from http://dspace.ou.nl/bitstream/1820/5390/1/INF_20140617_Woudenberg.pdf

Vargas, L. (2015). *METODOLOGÍA DE DESARROLLO DE SOFTWARE DIRIGIDA A EQUIPOS DE TRABAJO REDUCIDOS PARA SU APLICACIÓN EN LOS PROYECTOS*.

Wantroba, E. J., & Romero, R. A. F. (2016). An Interactive Question-Answer System with Dialogue for a Receptionist Avatar. *Proceedings - 12th LARS Latin American Robotics Symposium and 3rd SBR Brazilian Robotics Symposium, LARS-SBR 2015 - Part of the Robotics Conferences 2015*, 360–365. https://doi.org/10.1109/LARS-SBR.2015.54

Wells, D. (2013a). Acceptance Tests. Retrieved February 6, 2018, from

http://www.extremeprogramming.org/rules/functionaltests.html

Wells, D. (2013b). Iteration Planning. Retrieved February 6, 2018, from http://www.extremeprogramming.org/rules/iterationplanning.html

Wells, D. (2013c). Spike solution. Retrieved February 6, 2018, from http://www.extremeprogramming.org/rules/spike.html

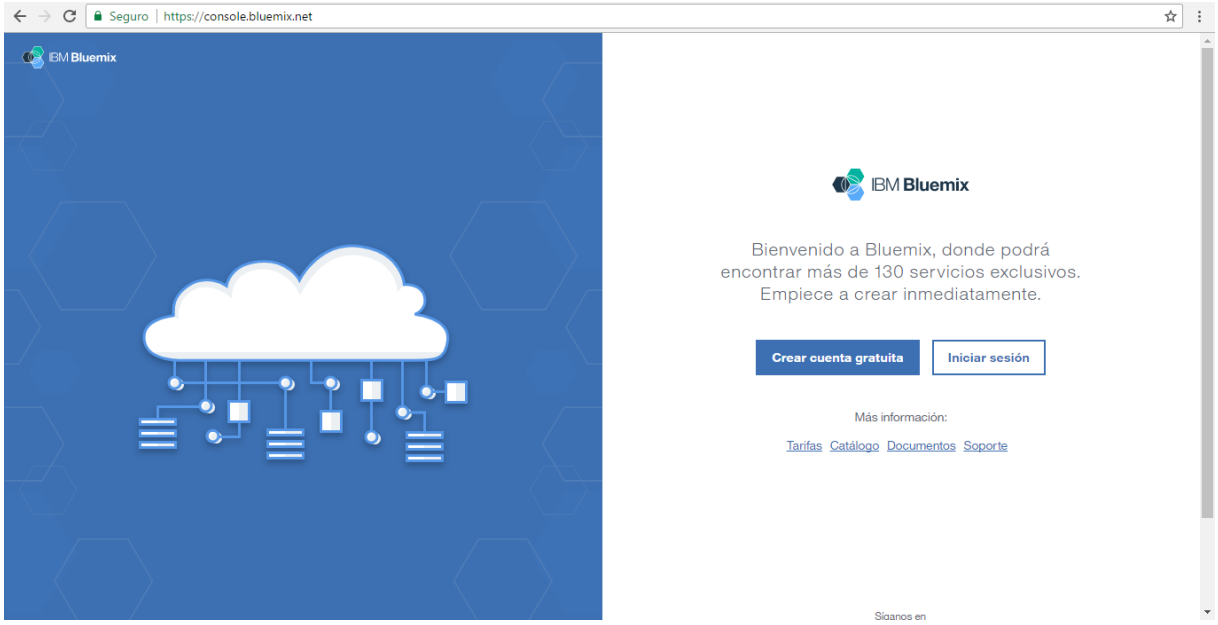
Wells, D. (2013d). Unit Tests. Retrieved February 6, 2018, from http://www.extremeprogramming.org/rules/unittests.html

WordReference. (2017). Conversacional.

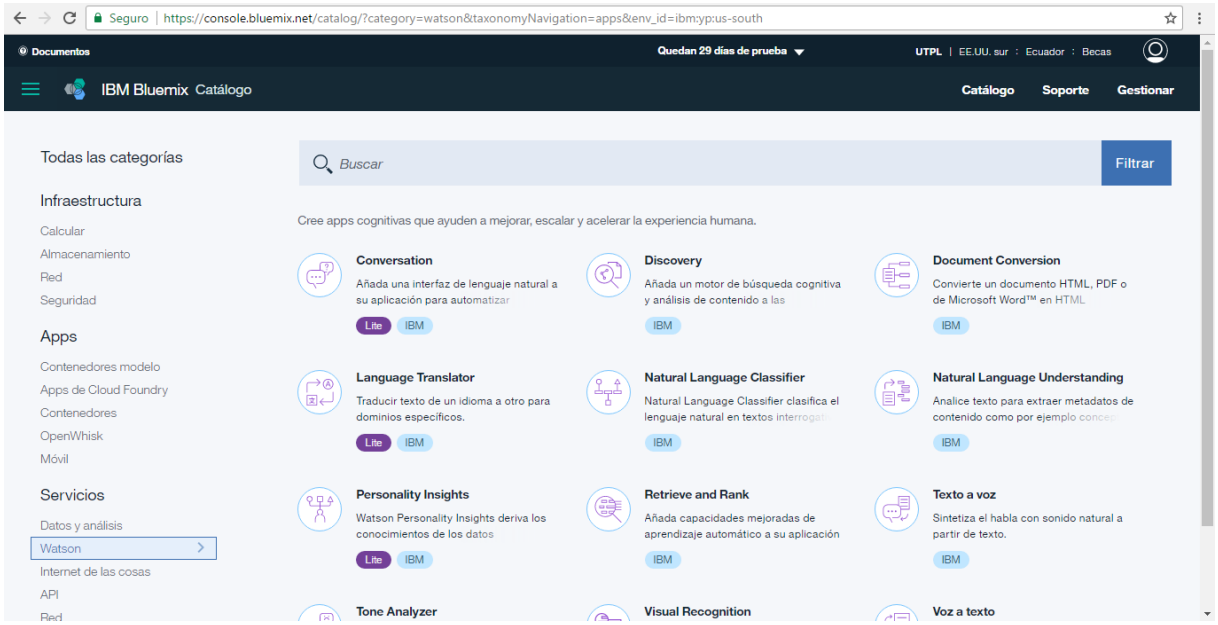
ANEXOS

Anexo A Creación del Servicio

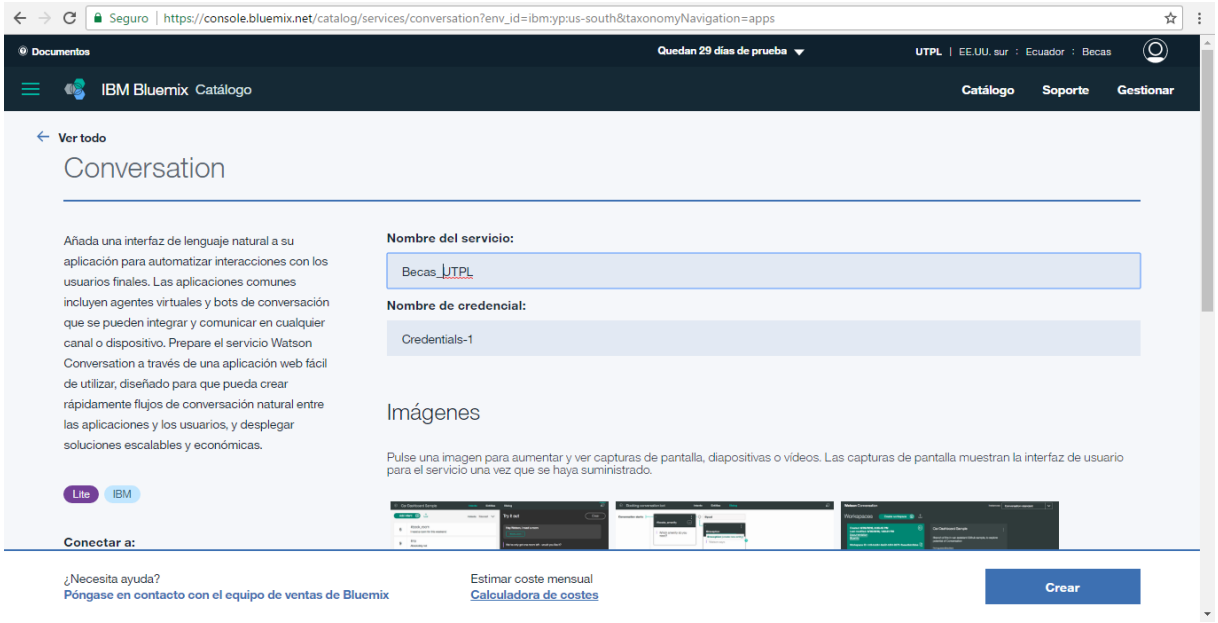
Para poder trabajar con IBM Watson y todos sus servicios se debe registra una cuenta en Bluemix para ello ingresamos en https://console.bluemix.net/



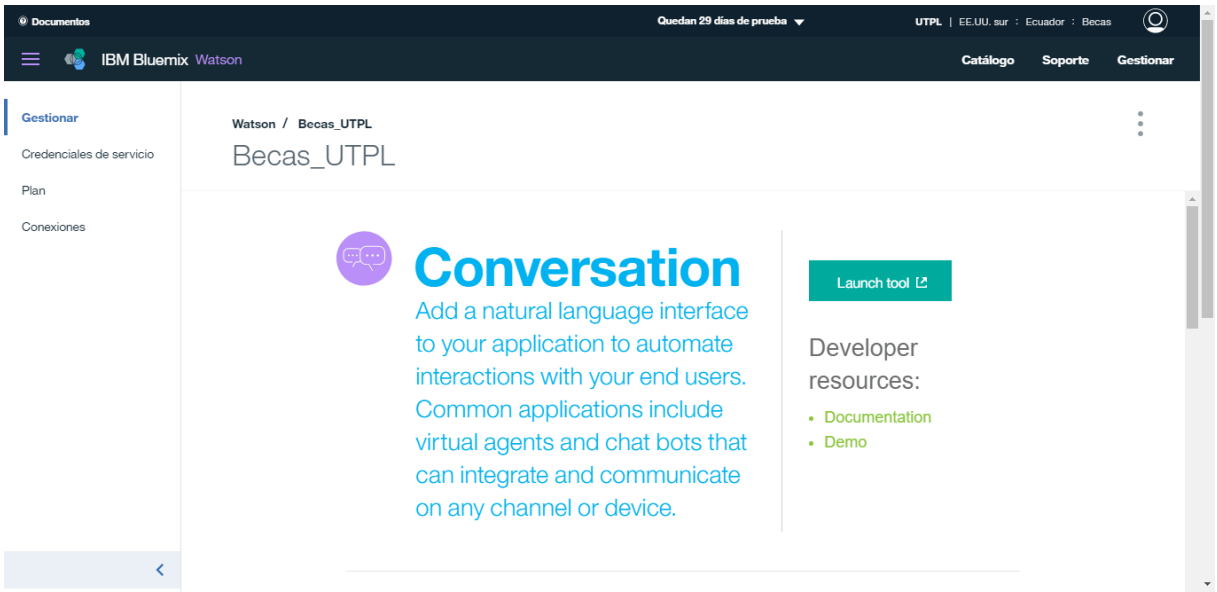
Luego de haber creado el usuario e ingresar a la plataforma se debe crear un espacio de trabajo que ayudaran a gestionar accesos y permisos para un conjunto de recursos, y se correlacionan fácilmente en los estados de desarrollo como desarrollo, prueba y producción. Se puede crear más de un espacio de trabajo. Después de esto se le pedirá que de clic en **Crear app** que direccionara al panel de inicio de Bluemix donde se podrá seleccionar el servicio que se requiera, en este caso seleccionamos en el panel izquierdo el servicio de **Watson** y después se da clic en **Conversation.**



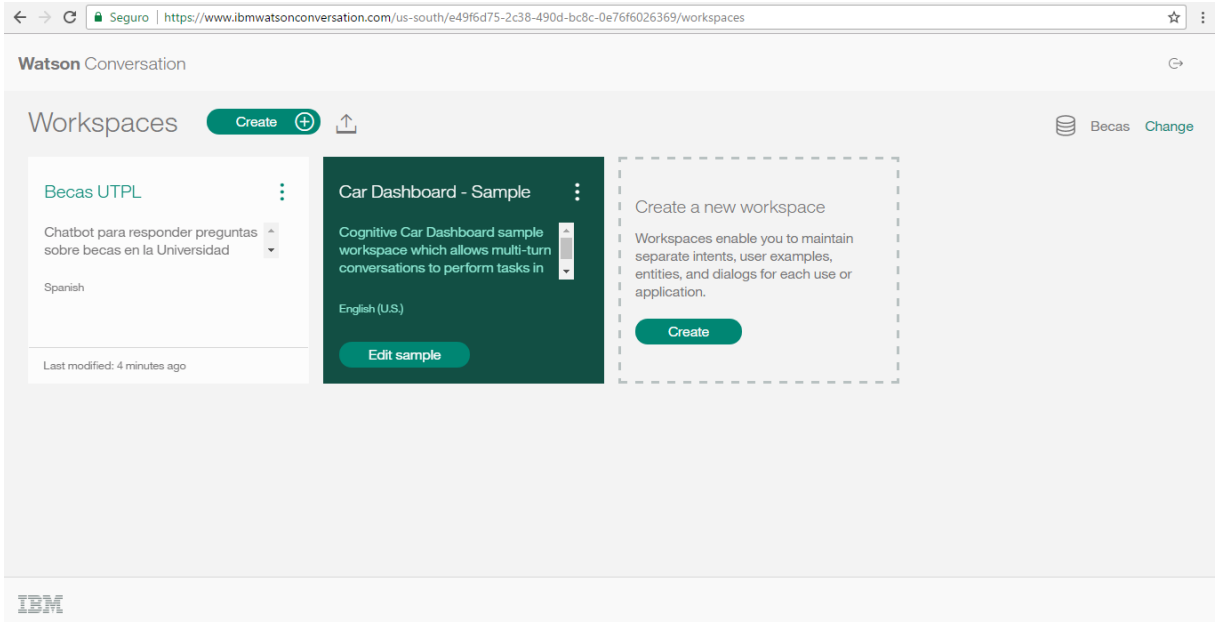
Siguiente se elige el nombre del servicio que se desee.



Se direccionará a la aplicación creada y se despliega el servicio dando clic en **Launch tool** para configurar el workspace.

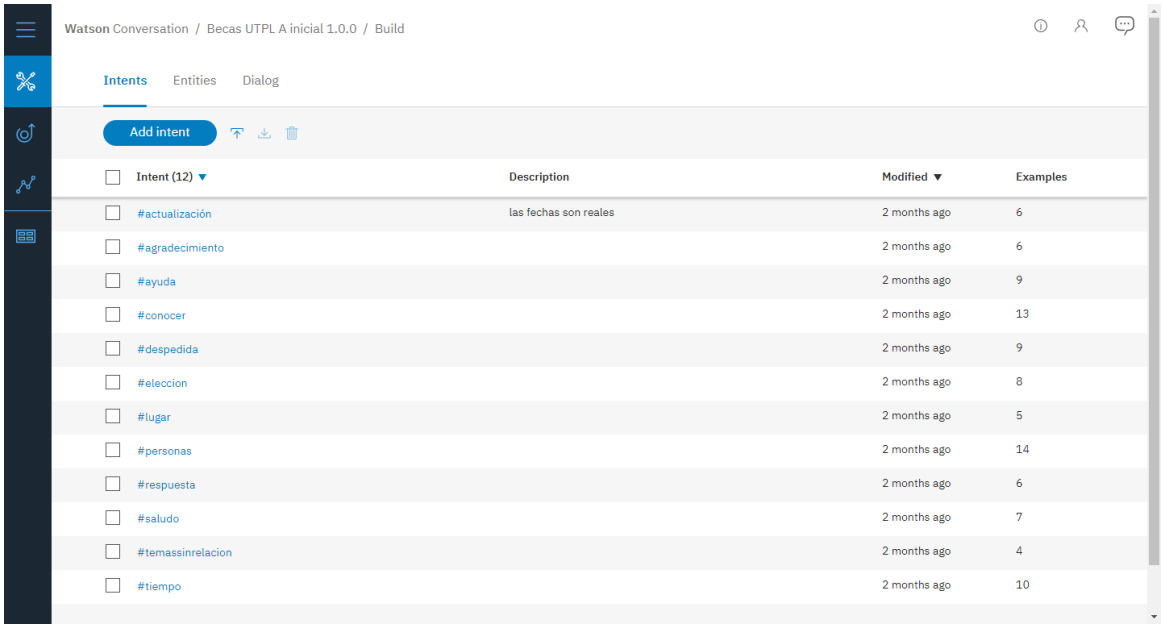


Creamos un nuevo workspace, con un nombre, una descripción y el lenguaje del Chatbot


Seguro | https://www.ibmwatsonconversation.com/us-south/e49f6d75-2c38-490d-bc8c-0e76f6026369/workspaces
Watson Conversation
Workspaces
Create
Becas
Change
Becas UTPL
Chatbot para responder preguntas sobre becas en la Universidad
Spanish
Last modified: 4 minutes ago
Car Dashboard - Sample
Cognitive Car Dashboard sample workspace which allows multi-turn conversations to perform tasks in
English (U.S.)
Edit sample
Create a new workspace
Workspaces enable you to maintain separate intents, user examples, entities, and dialogs for each use or application.
Create
IBM

Anexo B Entrenamiento del Chatbot

Dentro del workspace nos aparecerá la pantalla de **Intents** en donde se puede crear las posibles intenciones donde estarán representadas por #<*intención*>, como, por ejemplo: #saludo que representa todas aquellas palabras que un posible usuario ingresa para iniciar la conversación como un hola, que tal, buenos días, etc.



De forma general se mencionan las siguientes intenciones que se utilizó en el presente trabajo, cada una de ellas cuenta con ejemplos de preguntas específicas:

#saludo

#lugar

#tiempo

#conocer

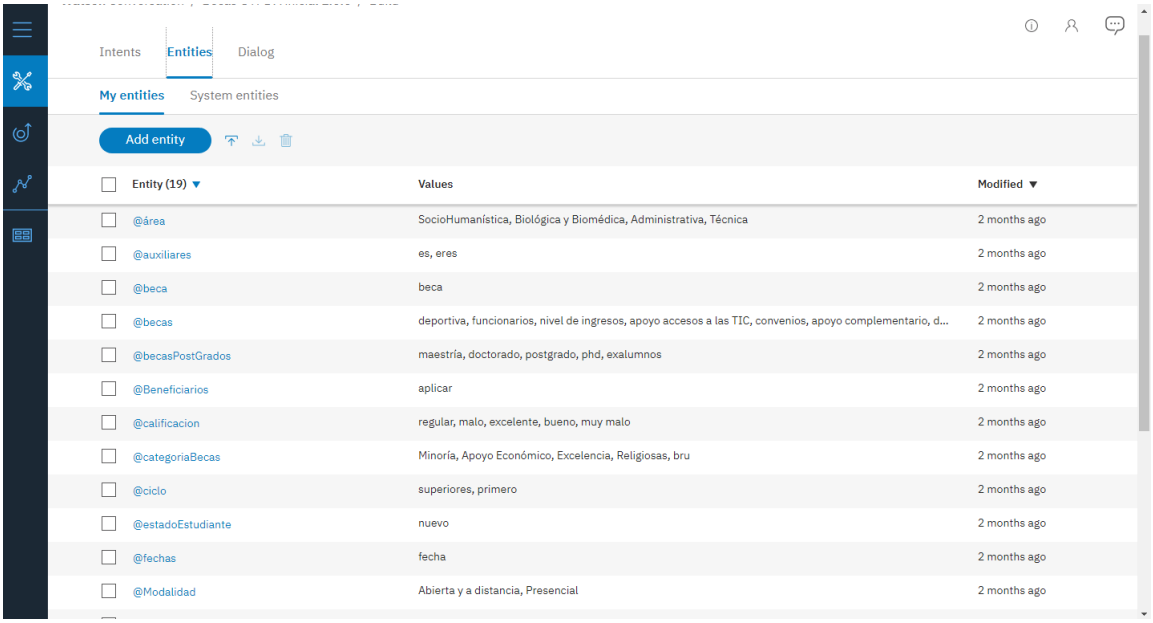
#ayuda

#despedida

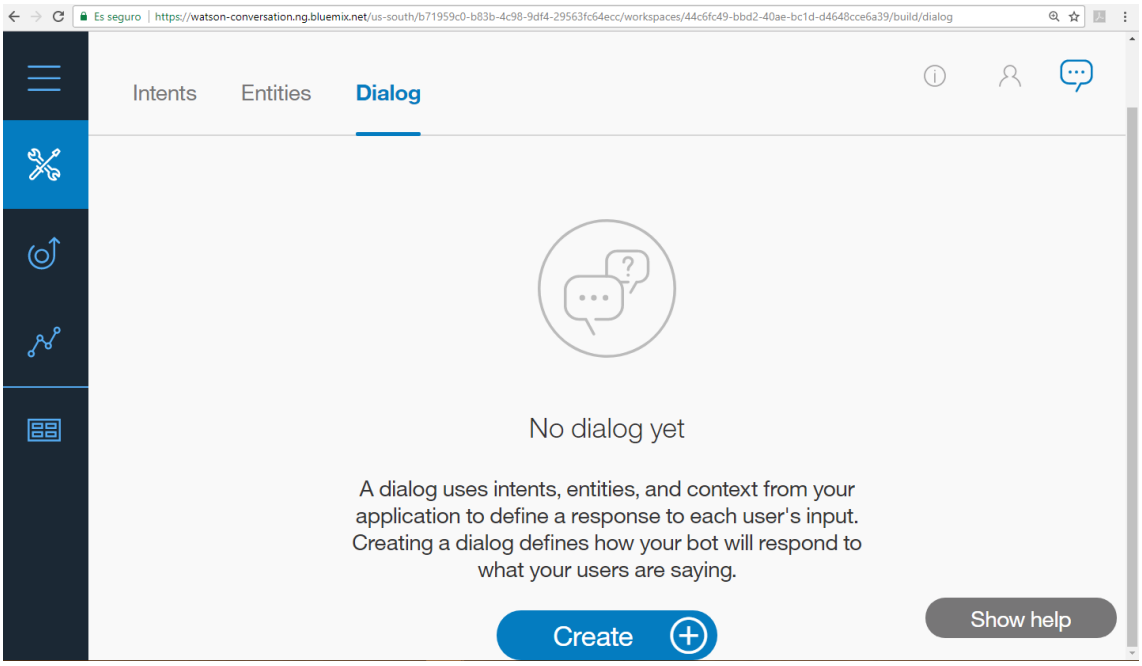
#personas

#respuesta

En la pantalla de **Entities**, donde se definirán entidades que representaran datos relevantes para el usuario dentro del dialogo, el Chatbot al reconocer estas entidades podrá elegir las acciones específicas para poder cumplir las intenciones.

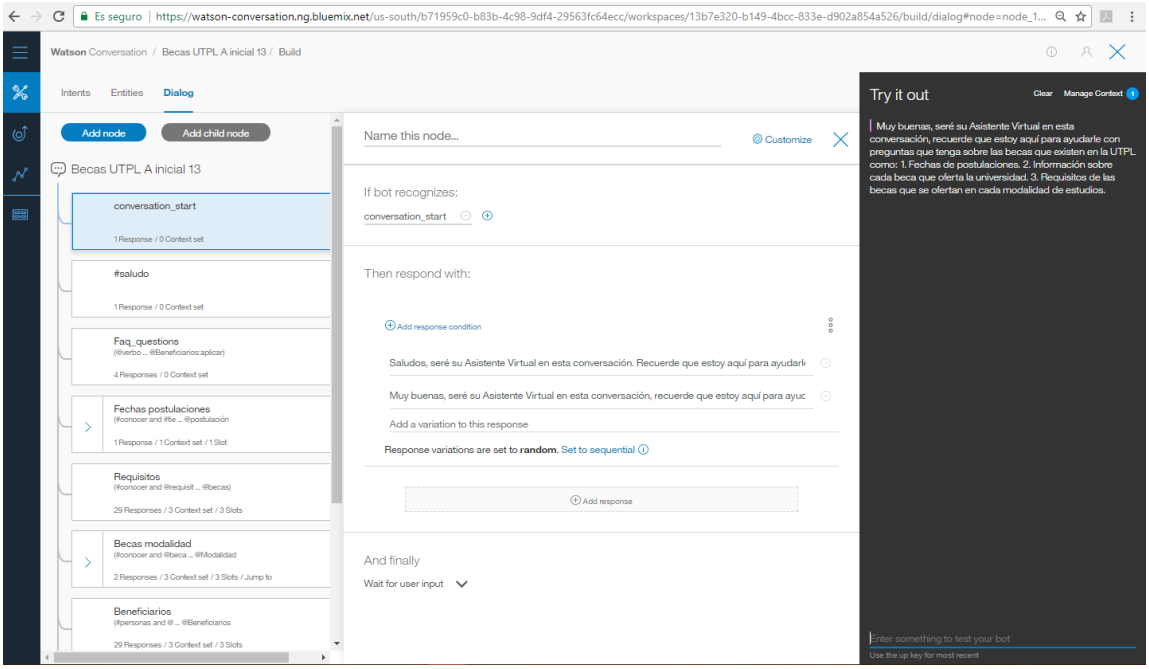


Una vez se tenga las intenciones y entidades definidas, se procede a desarrollar el flujo de dialogo, Se coloca las intenciones que representan las entradas de usuario y se coloca la respuesta que diría el Chatbot, y así poco a poco se debe ir entrenando al chatbot.



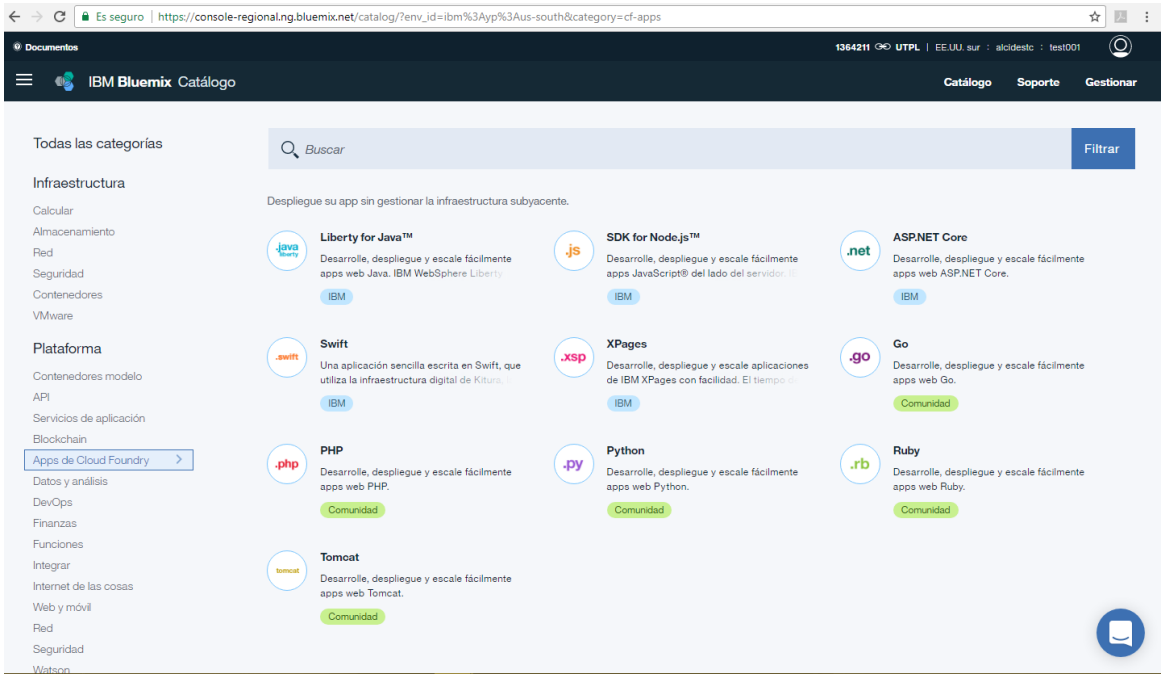
Para dar inicio a la conversación del chatbot se recomienda utilizar la condición **conversation_start** para que automáticamente se muestre como el primer mensaje dando inicio a la conversación.

En la siguiente imagen se muestra la creación del nodo inicial con la instrucción ya mencionada, en la parte derecha se aprecia como el chat de prueba envía un mensaje de saludo inicial.

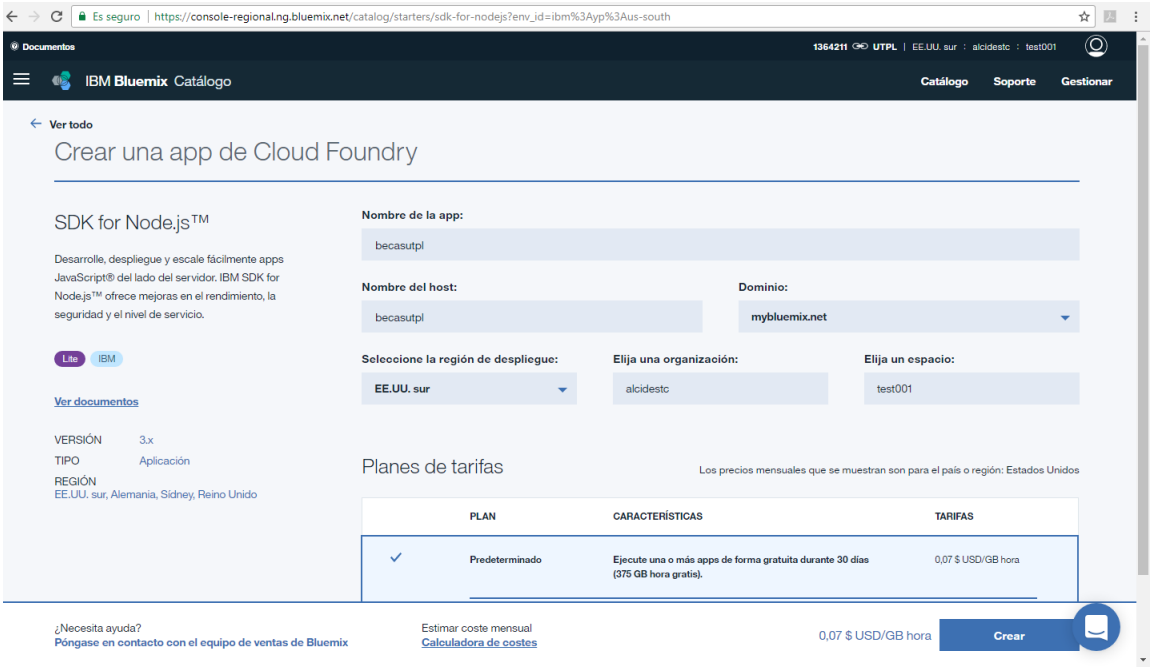

Es seguro
https://watson-conversation.ng.bluemix.net/us-south/b71959c0-b83b-4c98-9df4-29563fc64ecc/workspaces/13b7e320-b149-4bcc-833e-d902a854a526/build/dialog#node=node_1...
Watson Conversation / Becas UTPL A inicial 13 / Build
Intents
Entities
Dialog
Add node
Add child node
Becas UTPL A inicial 13
conversation_start
1 Response / 0 Context set
#saludo
1 Response / 0 Context set
Faq_questions
(@verbo ... @Beneficiarios:aplicar)
4 Responses / 0 Context set
Fechas postulaciones
(#conocer and #fe ... @postulación
1 Response / 1 Context set / 1 Slot
Requisitos
(#conocer and @requisit ... @becas)
29 Responses / 3 Context set / 3 Slots
Becas modalidad
(#conocer and @beca ... @Modalidad
2 Responses / 3 Context set / 3 Slots / Jump to
Beneficiarios
(#personas and @ ... @Beneficiarios
29 Responses / 3 Context set / 3 Slots
Name this node...
Customize
If bot recognizes:
conversation_start
Then respond with:
Add response condition
Saludos, seré su Asistente Virtual en esta conversación. Recuerde que estoy aquí para ayudarle
Muy buenas, seré su Asistente Virtual en esta conversación, recuerde que estoy aquí para ayuc
Add a variation to this response
Response variations are set to random. Set to sequential
Add response
And finally
Wait for user input
Try it out
Clear
Manage Context
Muy buenas, seré su Asistente Virtual en esta conversación, recuerde que estoy aquí para ayudarle con preguntas que tenga sobre las becas que existen en la UTPL como: 1. Fechas de postulaciones. 2. Información sobre cada beca que oferta la universidad. 3. Requisitos de las becas que se ofertan en cada modalidad de estudios.
Enter something to test your bot
Use the up key for most recent

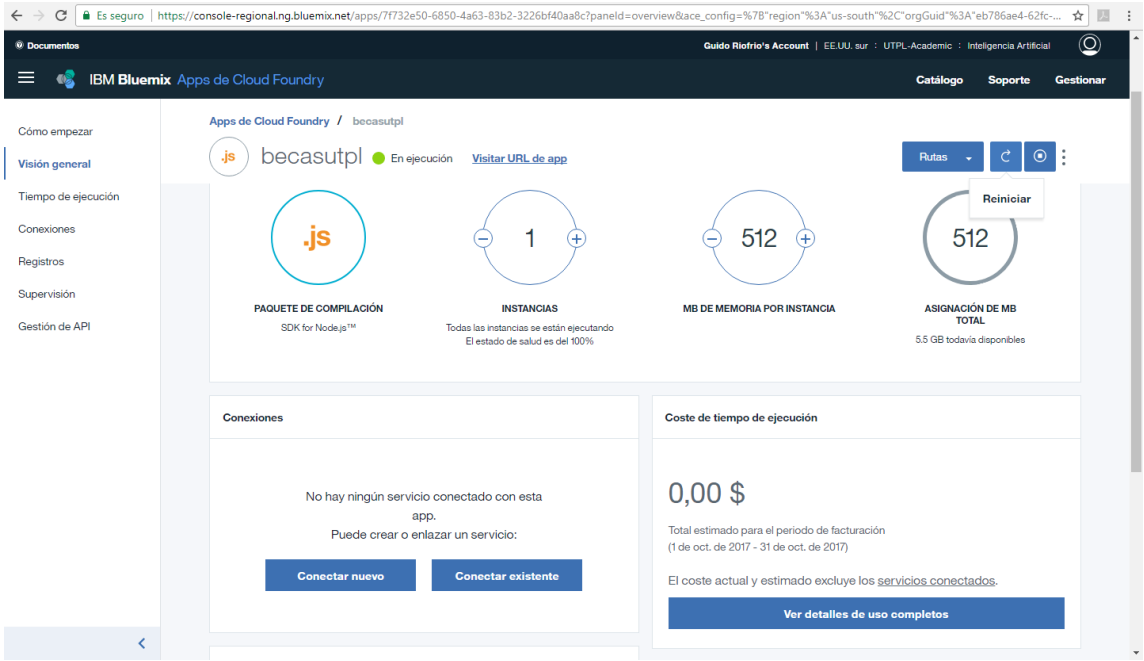
Anexo C Despliegue del Chatbot en Cloud Foundry

Para desplegar la aplicación se debe ingresar a la cuenta de IBM Bluemix, se elige el servicio de **apps de Cloud Foundry**, se selecciona **SDK for Node.js**, y estará listo la plataforma para levantar la aplicación.



Se completa los campos necesarios como el **nombre de la app**, este será la URL para ingresar al chat, después se da clic en **crear.**




Es seguro | https://console-regional.ng.bluemix.net/apps/7f732e50-6850-4a63-83b2-3226bf40aa8c?paneId=overview&ace_config=%7B"region"%3A"us-south"%2C"orgGuid"%3A"eb786ae4-62fc-...
Documentos
Guido Riofrio's Account | EE.UU. sur : UTPL-Academic : Inteligencia Artificial
IBM Bluemix Apps de Cloud Foundry
Catálogo
Soporte
Gestionar
Cómo empezar
Visión general
Tiempo de ejecución
Conexiones
Registros
Supervisión
Gestión de API
Apps de Cloud Foundry / becasutpl
becasutpl
En ejecución
Visitar URL de app
Rutas
Reiniciar
PAQUETE DE COMPILACIÓN
SDK for Node.js™
1
INSTANCIAS
Todas las instancias se están ejecutando
El estado de salud es del 100%
512
MB DE MEMORIA POR INSTANCIA
512
ASIGNACIÓN DE MB TOTAL
5.5 GB todavía disponibles
Conexiones
No hay ningún servicio conectado con esta app.
Puede crear o enlazar un servicio:
Conectar nuevo
Conectar existente
Coste de tiempo de ejecución
0,00 $
Total estimado para el periodo de facturación
(1 de oct. de 2017 - 31 de oct. de 2017)
El coste actual y estimado excluye los servicios conectados.
Ver detalles de uso completos

Anexo D Codificación del servicio

Programación del servidor

Para codificar el presente proyecto se utiliza la dependencia de **watson-developer-cloud** la cual permite conectar con varios servicios de Watson, a continuación, se presenta la función que conecta el servidor en node.js al servicio de Watson Conversation.

```
53   app.get('/', routes.index);
54
55   //Credenciales de acceso al servicio de Watson Conversation
56
57   var conversation = new Conversation({
58     // If unspecified here, the CONVERSATION_USERNAME and CONVERSATION_PASSWORD env properties will be checked
59     // After that, the SDK will fall back to the bluemix-provided VCAP_SERVICES environment property
60     username: '[redacted]',
61     password: '[redacted]',
62     url: 'https://gateway.watsonplatform.net/conversation/api',
63     version_date: '2017-07-11'
64   });
65
66   //ID de usuario que se cambia por su conversación workspace
67   var workspace = '7725549a-cfd1-4902-969d-ef418f787e32';
68
69   app.post('/converse', function(req, res, next) {
70     var geo = ip2loc.IP2Location_get_all(getClientIP(req));
71     try{
72       var d = new Date(req.body.context.dateC);
73     }
74     catch(e){
75       var d = new Date();
76     }
77     var payload = {
78       workspace_id: workspace,
79       context: {},
80       input: {}
81     };
82
83     if (req.body) {
84         if ( req.body.input ) {
85           payload.input = {text: req.body.input};
86         }
87         if (req.body.context) {
88           payload.context = req.body.context;
89         }
90     }else{
91         payload = {};
92       }
93
94     conversation.message(payload, function(err, data){
95       if ( err ) {
96         console.log(err);
97       }else{
98         insertBD(data.context.conversation_id, data.input.text, data.output.text['0'], d, getClientIP(req), geo.
           country_long, geo.region, geo.city);
99         return res.json(data);
100      }
101    });
102
103  });
```

Programación de la conexión a la base de datos

Para la conexión a la base de datos se usa el siguiente código en el cual usa la dependencia "pg", se ingresa a la función el **id** de la conversación, **mensaje** ingresado por el usuario y la **respuesta** del chatbot.

```
5   var express = require('express'),
6   routes = require('./routes'),
7   user = require('./routes/user'),
8   http = require('http'),
9   path = require('path'),
10  fs = require('fs');
11  var Conversation = require('watson-developer-cloud/conversation/v1'); // watson sdk
12  var TextToSpeechV1 = require('watson-developer-cloud/text-to-speech/v1');
13  var pg = require('pg');
14
15
16  //Connect to database
17  var insertBD = function(conversation_id, mensaje, respuesta, da, ip, country, region, city){
18      var conString = "postgres://goorltcp:y4St6yt1kweA7DPQpcLdsMcX9qq67LOF@hanno.db.elephantsql.com:5432/goorltcp";
19      var client = new pg.Client(conString);
20      client.connect(function(err) {
21              if(err) {
22                  return console.error('could not connect to postgres', err);
23              }else{
24                return console.log(" connection succes to bd");
25              }
26          });
27      client.query('INSERT into conversacion (conversation_id, mensaje, respuesta, fecha_hora, ip, hora, fecha, country,
          region, city) values($1,$2,$3,$4,$5,$6,$7,$8,$9,$10)',
28          [conversation_id, mensaje, respuesta, da, ip, (da.getHours()+":"+da.getMinutes()+":"+da.getSeconds()), da,
            country, region, city],function(err, result) {
29      if(err) {
30        return console.error('error running query', err);
31      }
32
33        client.end();
34      });
35
36  }
```

Programación de la interfaz

La interfaz está estructurada en HTML que permitirá visualizar los mensajes del chatbot y a través de CSS y Bootstrap se define el diseño de la interfaz.

```
4  <title>Becas UTPL</title>
5      <meta http-equiv="Content-Type" content="text/html; charset=utf-8">
6      <meta name="viewport" content="width=device-width,initial-scale=1,maximum-scale=1,minimum-scale=1,user-scalable=no"
       />
7      <meta name="apple-mobile-web-app-capable" content="yes" />
8      <link href="css/bootstrap.min.css" rel="stylesheet">
9      <link href="css/bootstrap-theme.min.css" rel="stylesheet">
10     <!--JQuery-->
11     <script src="https://ajax.googleapis.com/ajax/libs/jquery/1.11.3/jquery.min.js"></script>
12     <script type="text/javascript" src="scripts/index.js"></script>
13     <link href="https://maxcdn.bootstrapcdn.com/font-awesome/4.7.0/css/font-awesome.min.css" rel="stylesheet" integrity
       ="sha384-wvfXpqpZZVQGK6TAh5PVlGOfQNHSoD2xbE+QkPxCAFlNEevoEH3Sl0sibVcOQVnN" crossorigin="anonymous">
14     <link rel="stylesheet" href="/style/style.css" />
15
16   </head>
17   <body>
18 <div class="container">
19     <div class="row">
20         <div class="chat_window">
21             <div class="top_menu">
22                 <div class="title">
23                         <i class="fa fa fa-volume-up fa-pull-left fa-border playing" aria-hidden="true" id='vol'></i>
24                     <font><b>Asistente Virtual de becas UTPL</b></font>
25                     <i class="fa fa-question-circle fa-border fa-pull-right" aria-hidden="true" id='help'></i>
26                 </div>
27             </div>
28             <audio id="speech"></audio>
29             <ul class="messages"></ul>
30             <div class="clearfix">
31                 <div class="input-group">
32                     <input type="text" class="form-control" placeholder="Ingrese aqui su pregunta" id="chat" required>
33                         <span class="enviar_mensaje input-group-btn">
34                             <button type="button" class="btn btn-default" aria-label="Help">
35                                 <span class="fa fa-paper-plane"></span>
36                             </button>
37                         </span>
38                 </div>
39                 <small id="arrow"></small>
40             </div>
41         </div>
42
43             </div>
44         </div>
45     <div class="message_template">
46         <li class="message">
47             <div class="avatar"></div>
48             <div class="text_wrapper">
49                 <div class="text"></div>
50             </div>
51         </li>
52     </div>
53
54   </body>
55   </html>
```

Para iniciar una conversación se llama a la función **converse(),** sin ningún parámetro, de esta manera el servidor interpreta la entrada como inicio del diálogo.

Cada vez que el usuario ingresa una pregunta se llama a la función **converse(params)**, este almacena la entrada y la envía al servidor y pueda ser respondida.

```
45  var converse = function(userText, context, guarda) {
46          // check if the user typed text or not
47      if (typeof(userText) != undefined && $.trim(userText) !== '')
48        sendMessage(userText);
49
50      // build the conversation parameters
51
52      var myDate = new Date();
53      console.log(myDate);
54
55
56      params = { input : userText };
57
58      if (paramsConversation) {
59        params.context = paramsConversation.context;
60        //console.log(paramsConversation);
61      }
62      //console.log(Intl.DateTimeFormat().resolvedOptions().timeZone);
63      params.context.timezone = Intl.DateTimeFormat().resolvedOptions().timeZone;
64      console.log(params.context.timezone);
65      params.context.dateC= myDate;
66
67      //WEB SERVICE DE NODE.JS
68      $.post('/converse', params)
69      .done(function onSucess(answers){
70        var intent;
71          console.log("Respuesta del servidor");
72          console.log(answers);
73
74
75            //$chatInput.val(''); // clear the text input
76
77            if (paramsConversation){
78              paramsConversation = answers;
79              //console.log('paramsConversation');
80            }
81
82              sendMessage(answers.output.text,'W');
83
84        })
85      .fail(function(response) {
86        //console.log( response );
87        //sendMessage("El servicio no está en línea por por el momento ",'W');
88      })
89
90  }
```

La función de **sendMessage()** muestra tanto los mensajes del usuario como del servidor identificándolos con una letra para que se mas fácil de saber a quién pertenece cada mensaje.

```
160 sendMessage = function (text, letter) {
161             //console.log(text);
162             var $messages, message;
163
164             $('.message_input').val('');
165             $messages = $('.messages');
166             if(letter === 'W'){
167               message_side = 'left';
168               audio(text);
169             }else{
170               message_side = 'right';
171             }
172             if (text.length >= 2 && message_side == 'left'){
173               for (var i = 0; i<text.length ; i++) {
174                 message = new Message({
175                 text: text[i],
176                 message_side: message_side,
177                 letter: letter
178                 });
179                 message.draw();
180               }
181             }else{
182               message = new Message({
183                 text: text,
184                 message_side: message_side,
185                 letter: letter
186               });
187               message.draw();
188             }
189             return $messages.animate({ scrollTop: $messages.prop('scrollHeight') }, 1000);
190 };
```

Anexo E Pruebas de aceptación

Prueba de aceptación	
Número: 1	**N° de historia de usuario:** 1
Historia de usuario: Inicio de Chatbot	
Condiciones de ejecución: • Ingresar a la URL del chat vía web.	
Pasos de ejecución: • Ingresar la URL del sitio en el navegador. • Confirmar mensaje de bienvenida de Chatbot. • Confirmar mensaje de bienvenida en forma de un audio.	
Resultados esperados: • Al ingresar por primera vez a la aplicación web esta deberá retornar un mensaje de bienvenida, tanto en formato de texto. • Opcionalmente si se cuenta con periféricos de audio el chatbot retornara el mismo mensaje de texto en formato de voz.	
Evaluación de la prueba: Satisfactoria	

Prueba de aceptación	
Número: 2	**N° de historia de usuario:** 2
Historia de usuario: Realizar consulta	
Condiciones de ejecución: • Ingresar a la URL del chat vía web. • Escribir una consulta y enviar al chatbot.	
Pasos de ejecución: • Ingresar la URL del sitio en el navegador. • Esperar mensaje de bienvenida del chatbot. • Enviar consulta presionando la tecla “Enter” o hacer clic en el botón de “Enviar Mensaje”.	
Resultados esperados: • Al ingresar por primera vez a la aplicación web esta deberá retornar un mensaje de bienvenida, tanto en formato de texto. • Una vez recibido en mensaje de bienvenida, los usuarios pueden enviar su consulta al chatbot sobre becas, el mensaje debe aparecer el área de mensajes enviados identificados con la letra U o un icono de usuario.	
Evaluación de la prueba: Satisfactoria	

Prueba de aceptación	
Número: 3	**N° de historia de usuario:** 3
Historia de usuario: Respuesta del Chatbot	
Condiciones de ejecución: • Ingresar a la URL del chat vía web. • Haber enviado una consulta la chatbot.	
Pasos de ejecución: • Ingresar la URL del sitio en el navegador. • Confirmar mensaje de bienvenida de Chatbot. • Enviar consulta al chatbot.	
Resultados esperados: • El chatbot debe responder de forma congruente a la consulta del usuario. • El mensaje del chatbot debe aparecer en el área de mensajes identificándolo con el logo de la Universidad Técnica Particular de Loja.	
Evaluación de la prueba: Satisfactoria	

Prueba de aceptación	
Número: 4	**N° de historia de usuario:** 4
Historia de usuario: Almacenar dialogo	
Condiciones de ejecución: • Ingresar a la URL del chat vía web. • Haber enviado una consulta la chatbot	
Pasos de ejecución: • Ingresar la URL del sitio en el navegador. • Confirmar mensaje de bienvenida de Chatbot. • Enviar consulta al chatbot.	
Resultados esperados: • Se espera que los mensajes de los usuarios se almacenen en una base de datos en forma de logs para llevar un registro de actividad del chat.	
Evaluación de la prueba: Satisfactoria	

Prueba de aceptación	
Número: 5	**N° de historia de usuario:** 5
Historia de usuario: Consumo del servicio web	

Condiciones de ejecución: • Ingresar a los servicios web del chat y conversor de texto a voz. • Haber enviado una consulta la chatbot.
Pasos de ejecución: • Ingresar al servicio de conversación. • Ingresar al servicio de conversor de texto a voz. • Enviar consultas a los dos servicios web.
Resultados esperados: • Se espera que el servicio web de conversación retorne un mensaje en formato JSON. • Se espera que el servicio web retorne un archivo de audio con el mensaje de texto enviado previamente.
Evaluación de la prueba: Satisfactoria

Prueba de aceptación	
Número: 6	**N° de historia de usuario:** 6
Historia de usuario: Ayuda para uso de chatbot	
Condiciones de ejecución: • Ingresar a la URL del chat vía web.	
Pasos de ejecución: • Ingresar al servicio de conversación. • Dar clic en el icono de la parte superior derecha de la página.	
Resultados esperados: • Se espera que al dar clic el icono de ayuda este envíe un mensaje al chatbot pidiendo ayuda, el mensaje que retorne el chatbot debe contener un diálogo que contenga los temas que puede contestar.	
Evaluación de la prueba: Satisfactoria	

Prueba de aceptación	
Número: 7	**N° de historia de usuario:** 7
Historia de usuario: Validación de la información.	
Condiciones de ejecución:	

• Se debe entregar a los funcionarios del Balcón de Servicios Estudiantiles y al Departamento de Bienestar Estudiantil un documento con las preguntas y respuestas que puede contestar el chatbot.
Pasos de ejecución: • El autor del TFT junto a los funcionarios del Balcón de Servicios Estudiantiles y al Departamento de Bienestar Estudiantil deben revisar las preguntas y respuestas que el chat contesta.
Resultados esperados: • Validar la información que el chatbot retorna en cada consulta sobre temas que tienen que ver con las becas de la Universidad Técnica Particular de Loja.
Evaluación de la prueba: Satisfactoria

Anexo F Pruebas unitarias

Para realizar las pruebas unitarias a la aplicación y la comunicación con la base de datos, mediante Mocha y Chai para el servidor Node.js los cuales se muestran en consola.

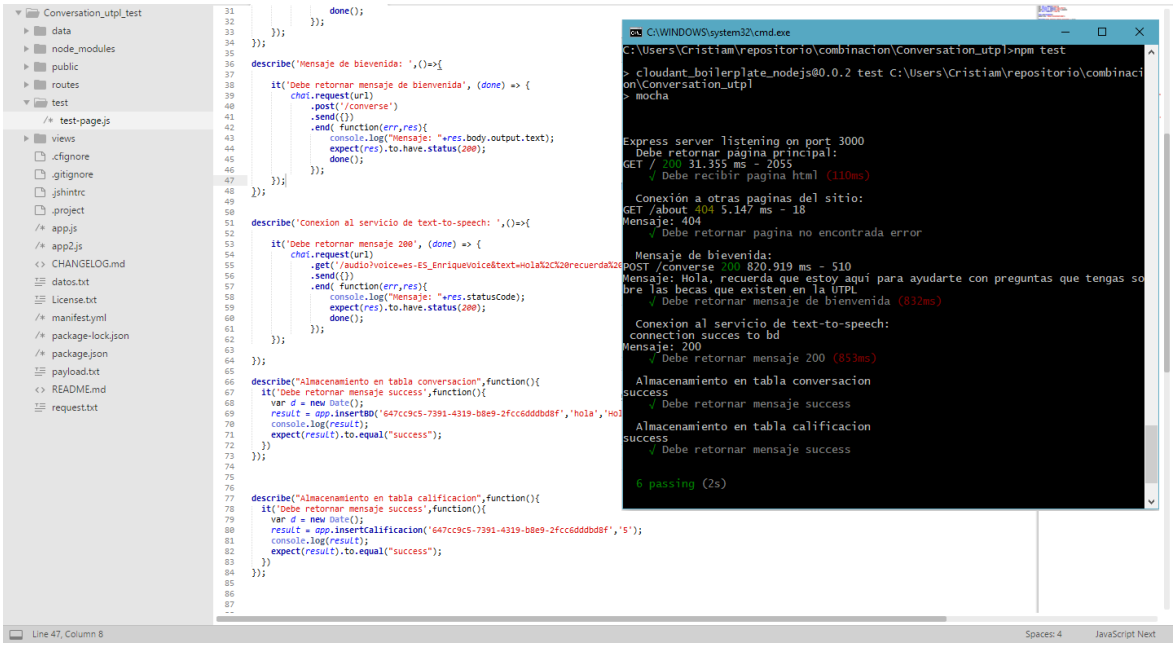
Chai se integra con Mocha para obtener los reportes de cada prueba unitaria y comprueba que la función retorna los resultados esperados, caso contrario, se presenta un mensaje de fallo en la comprobación.

La siguiente captura se presenta las pruebas unitarias que se realizó a la base de datos, confirmando la comunicación y el almacenamiento de las conversaciones.

```
var app = require("../app");

chai.use(chaiHttp);
const url= 'http://localhost:3000';

describe('Debe retornar página principal: ',()=>{

    it('Debe recibir pagina html', (done) => {
        chai.request(url)
            .get('/')
            .end( function(err,res){
                //console.log(res.text);
                expect(res).to.have.status(200);
                done();
            });
    });
});


describe('Conexión a otras paginas del sitio: ',()=>{

    it('Debe retornar pagina no encontrada error', (done) => {
        chai.request(url)
            .get('/about')
            .end( function(err,res){
                console.log("Mensaje: "+res.statusCode);
                expect(res).to.have.status(404);
                done();
            });
    });
});

describe('Mensaje de bievenida: ',()=>{

    it('Debe retornar mensaje de bienvenida', (done) => {
        chai.request(url)
            .post('/converse')
            .send({})
            .end( function(err,res){
                console.log("Mensaje: "+res.body.output.text);
                expect(res).to.have.status(200);
                done();
            });
    });
});


describe('Conexion al servicio de text-to-speech: ',()=>{

    it('Debe retornar mensaje 200', (done) => {
        chai.request(url)
            .get('/audio?voice=es-ES_EnriqueVoice&text=Hola%2C%20recuerda%20que%20estoy%20aqu%C3%A0%20para%20ayudarte%20con%20preguntas%20que%20tengas%20sobre%20
            .send({})
            .end( function(err,res){
                console.log("Mensaje: "+res.statusCode);
                expect(res).to.have.status(200);
                done();
```

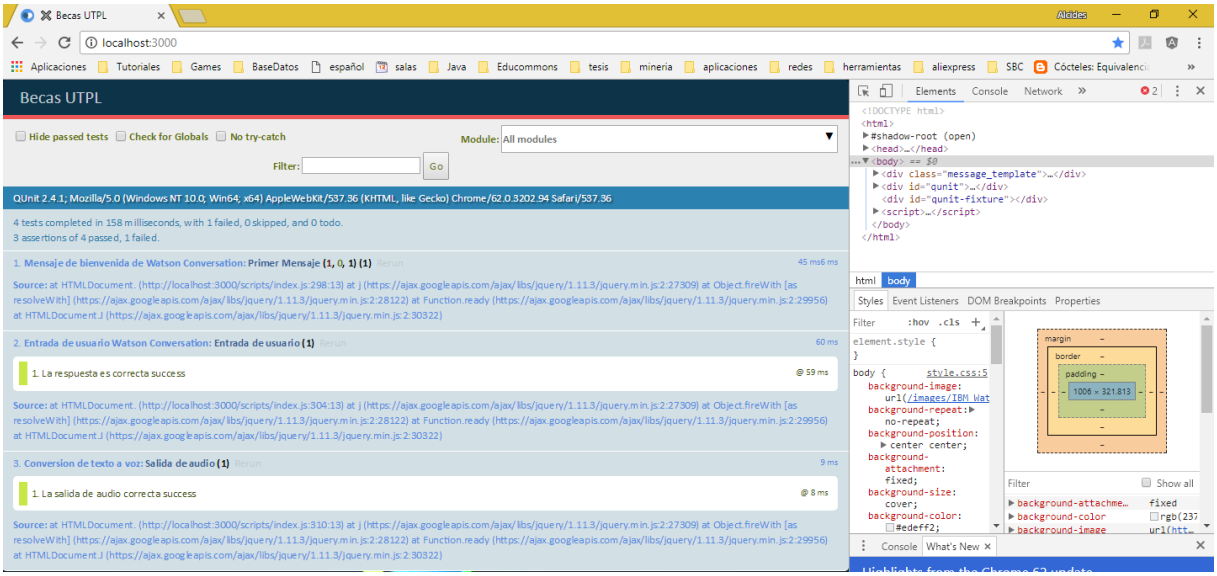
```
C:\Users\Cristiam\repositorio\combinacion\Conversation_utpl>npm test

> cloudant_boilerplate_nodejs@0.0.7 test C:\Users\Cristiam\repositorio\combinaci
on\Conversation_utpl
> mocha


  Debe retornar página principal:
    - Debe recibir pagina html

  Conexión a otras paginas del sitio:
      Debe retornar pagina no encontrada error

  Mensaje de bievenida:
    - Debe retornar mensaje de bienvenida

  Conexion al servicio de text-to-speech:
    - Debe retornar mensaje 200

  Almacenamiento en tabla conversacion
success
    √ Debe retornar mensaje succes

  Almacenamiento en tabla calificacion
success
    √ Debe retornar mensaje succes


  2 passing (97ms)
  4 pending

C:\Users\Cristiam\repositorio\combinacion\Conversation_utpl>
```

A continuación, se muestra el test completo realizado a la aplicación y su comunicación a los servicios de Watson Conversation y Text-to-speech.



En los reportes que retorna Mocha del proyecto se puede observar que la aplicación funciona al 100% sin errores en las funciones y conexiones.

Para la parte de frontend se utilizó el framework Qunit que permite realizar pruebas unitarias para el lenguaje programación JavaScript. En los que se realizó los diferentes test para probar el funcionamiento de cada módulo. A continuación, se presenta la captura de pantalla del test.



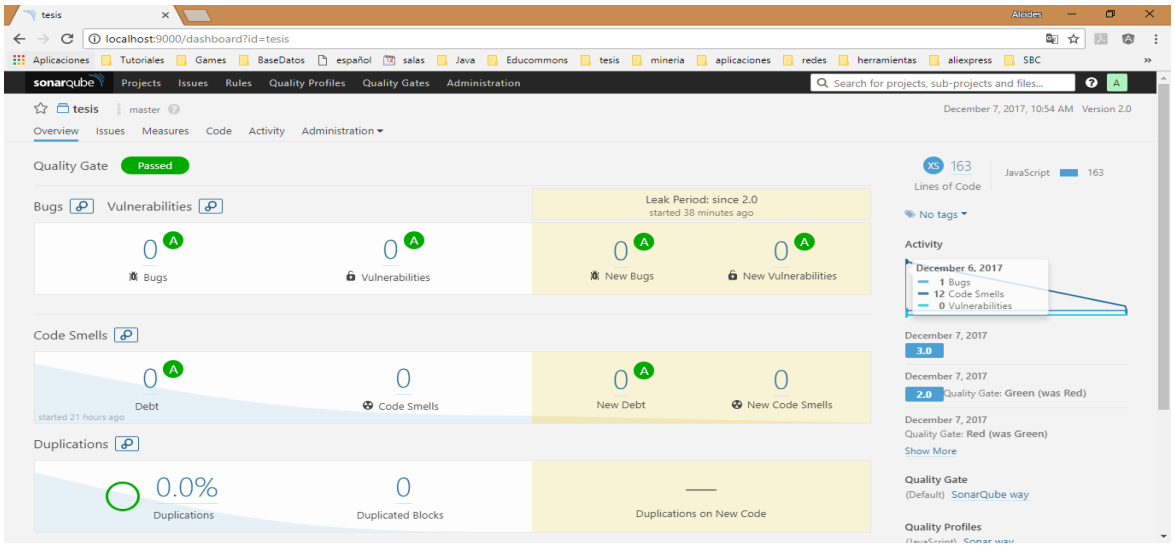
El reporte muestra que la aplicación funciona al 100% sin errores en todos los métodos y funciones del frontend.

Anexo G Pruebas de calidad de software

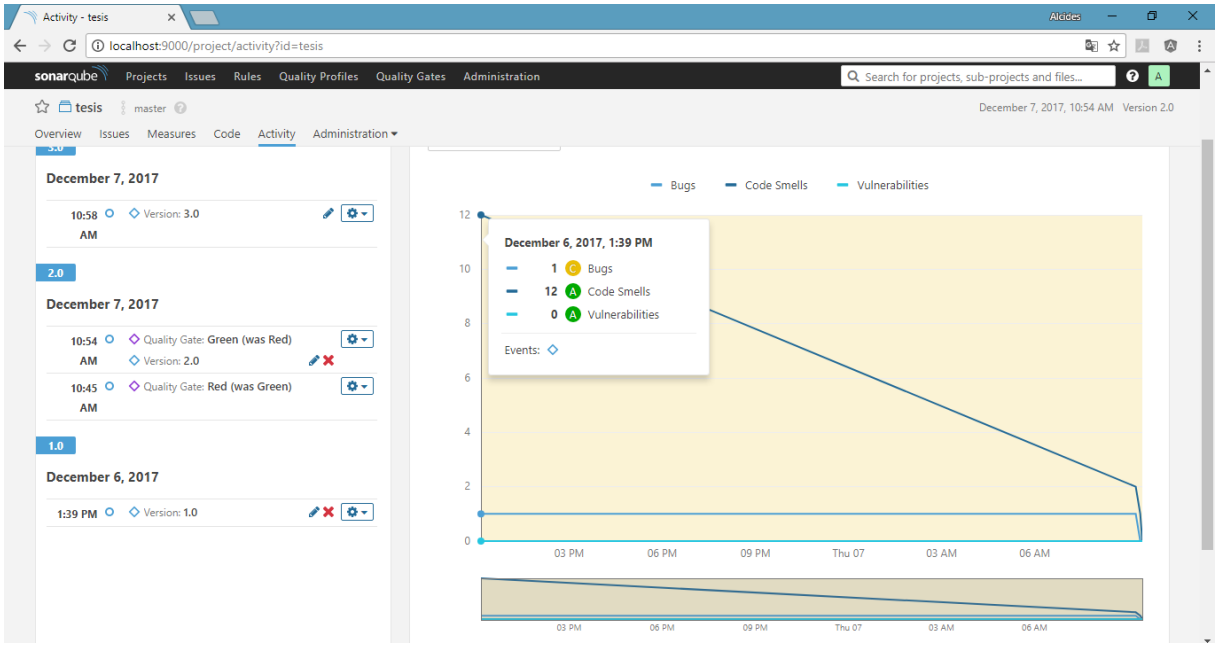
Este test de realizó utilizando el software SonarQube que permite controlar la calidad del código con un mínimo de esfuerzo, validando así la seguridad, portabilidad, mantenibilidad, entre otros.

Las siguientes capturas de pantalla muestran los resultados al evaluar la aplicación que se encuentra desarrollada en JavaScript tanto el lado del servidor como el cliente.

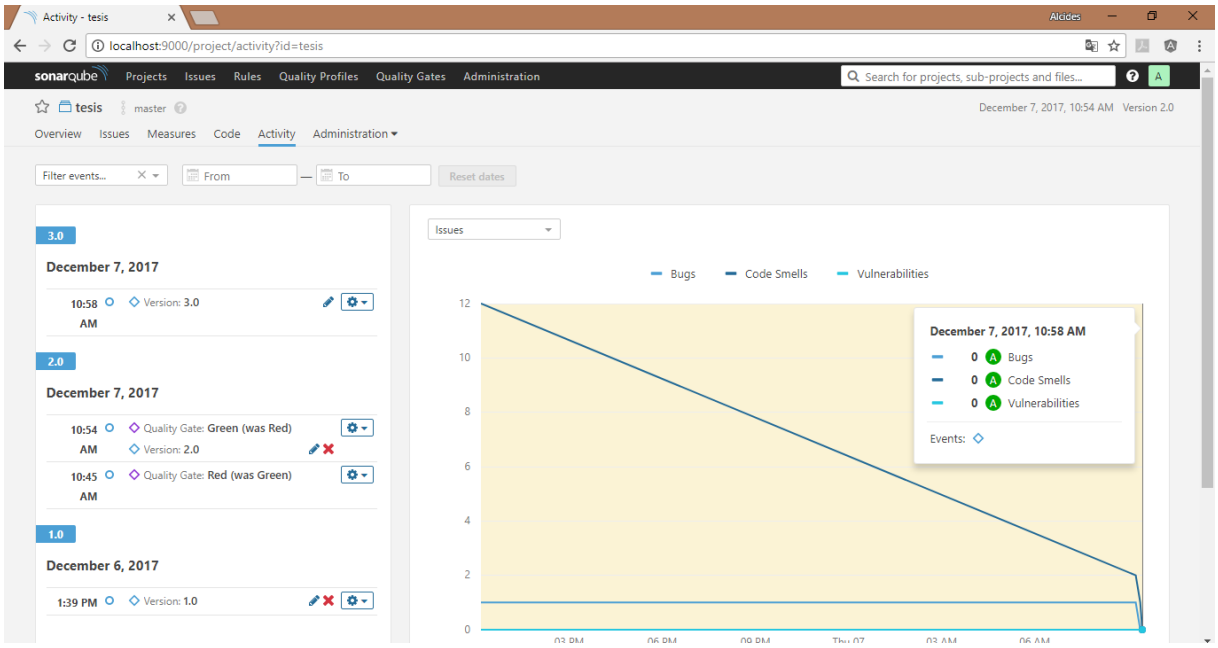
La captura muestra algunas variables como: el número de líneas de código, cantidad de bugs, porcentaje de código duplicado, etc. También muestra un mensaje que indica la calidad del software dando una calificación de A, es decir “aprobado”.



La siguiente Figura es una captura de pantalla de la primera prueba a la aplicación, la cual arroja una calificación de C para bugs en la aplicación y A para code smells o código basura, A para vulnerabilidades del sistema. Esto nos permite corregir el código en cuanto a sintaxis de programación.

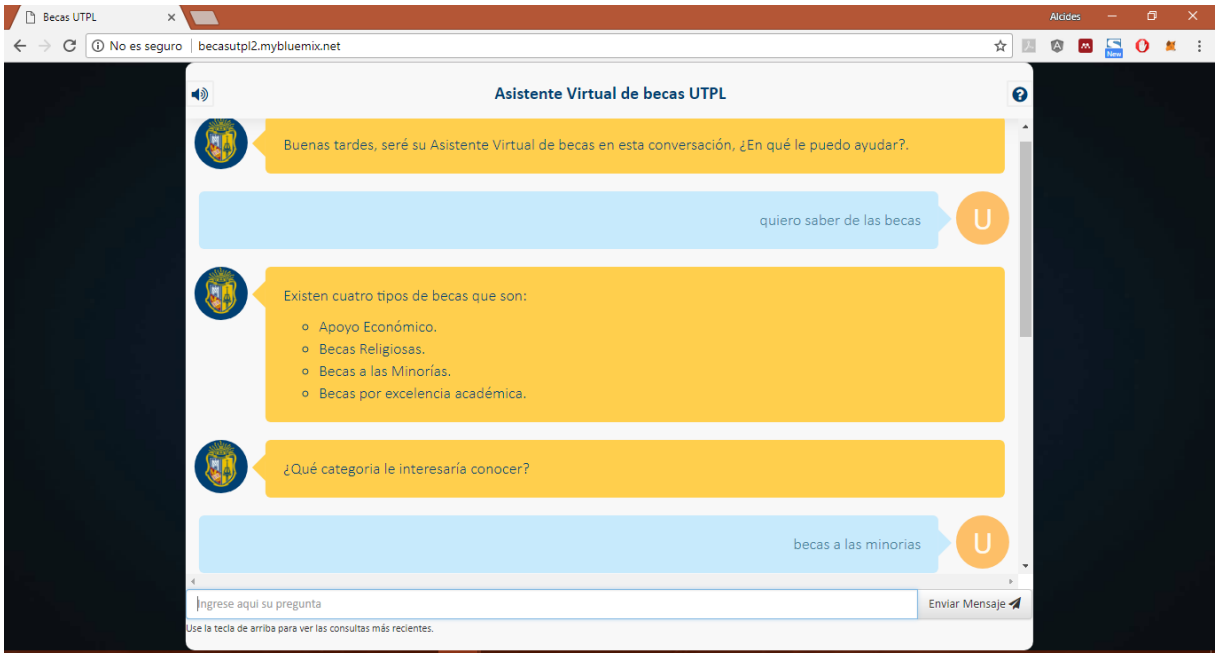


La continuación, se muestra el test final que se realizó a la aplicación, corrigiendo todos los bugs y código basura mostrados en los test anteriores, aumentando la calificación del test y evitando así errores en el funcionamiento del sistema.



Anexo H Manual de usuario

Una vez realizados los pasos que se encuentran en los anexos del A al C, al tener desplegado una aplicación en CloudFoundry de Bluemix, automáticamente se tiene un sitio web del chatbot de becas.



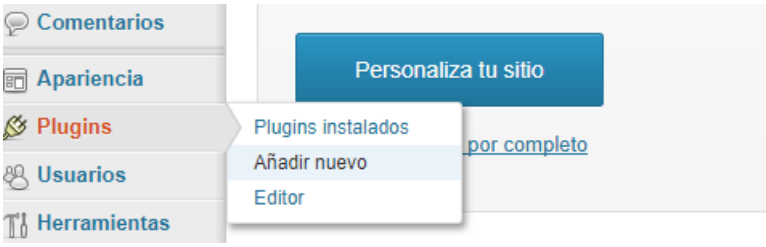
En la figura anterior se muestra la interfaz de chatbot donde los usuarios ingresaran sus consultas que tengan que ver con las becas de la institución. En la parte inferior se encuentra un area los usuarios ingresaran su consulta. Si en la conversacion ya ha enviado mensajes anteriormente puede presionar la tecla de dirección hacia arriba para mostrar las consultas previas.

Una vez enviada la consulta al chat, estas inmediatamente se mostrarán en un contenedor identificandolos con la letra U de usuario. Los mensajes que responda el chat se representan con el logo de la Institución. Si el usuario cuenta con perifericos de audio podra escuchar las respuestas del chat.

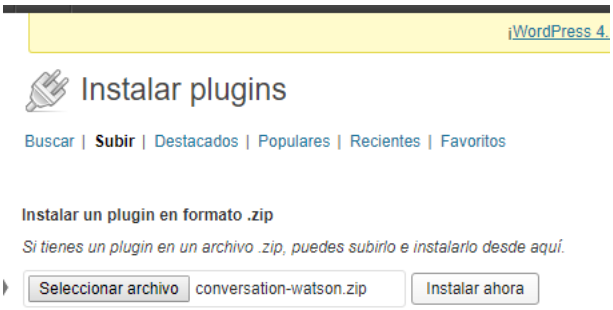
Plugin Wordpress

Para la instalación del plugin en wordpress y la respectiva integración con los servicios de Watson Conversation y text-to-speech se deben seguir los siguientes pasos:

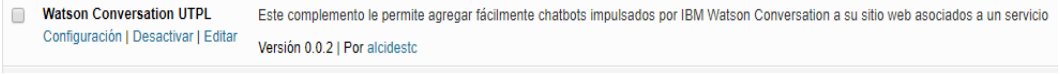
1. Ingresar al panel de administración del sitio.
2. Ingresar a Plugins > Añadir nuevo



3. Hacer clic en “Subir”, luego seleccionar el archivo en formato **.zip** y después hacer clic en “Instalar Ahora”.



4. Activar el plugin y la instalación estará completada.
5. Hacer clic en “Configuración” para editar los servicios web.



6. En el primer campo del formulario se debe ingresar la URL del servicio de conversación. En el segundo campo se debe ingresar el servicio de text-to-speech. Con esta configuración se puede dar clic en guardar cambios.

La sección de Gestión de uso total permite configurar la cantidad total de solicitudes al servicio de conversación para un periodo de tiempo específico que puede ser: por mes, por semana, por día y por hora.

Gestión de uso total

Esta sección le permite evitar el uso excesivo de sus credenciales al limitar el uso del bot de chat.

Si tiene un plan pago para Watson Conversation, entonces el monto que debe pagar está directamente relacionado con la cantidad de solicitudes API realizadas. La cantidad de solicitudes API es igual a la cantidad de mensajes enviados por los usuarios de su bot de chat, además del saludo inicial del chatbot.

Por ejemplo, el plan estándar cobra $ 0.0025 por llamada API. Eso significa que si los visitantes de su sitio envían un total de 1000 mensajes en un mes, se le cobrará ($ 0.0025 por llamada API) x (1000 llamadas) = $ 2.50. Si desea limitar los costos incurridos por este chatbot, puede poner un límite en la cantidad total de solicitudes API para un período de tiempo específico aquí.

Limite total peticiones a API	(•) Si ( ) No
Número máximo de peticiones totales	0 Por Mes ▾

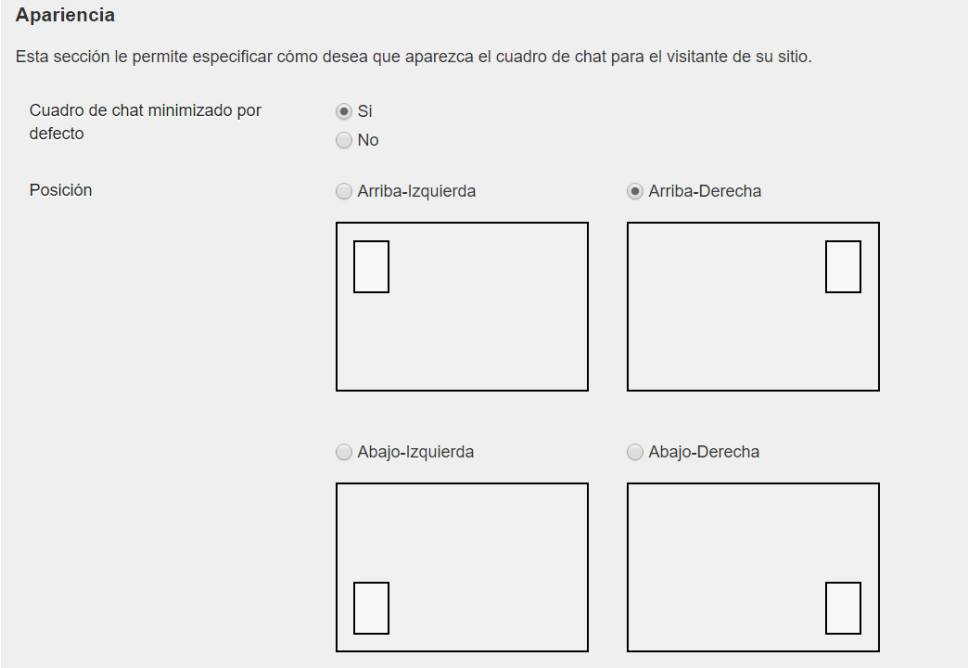
La sección de Comportamiento permite personalizar el comportamiento de chat como el retraso para mostrar el icono del chat, así como elegir si se desea presentar el chat el todo el sitio o solo en páginas específicas del sitio.

Comportamiento

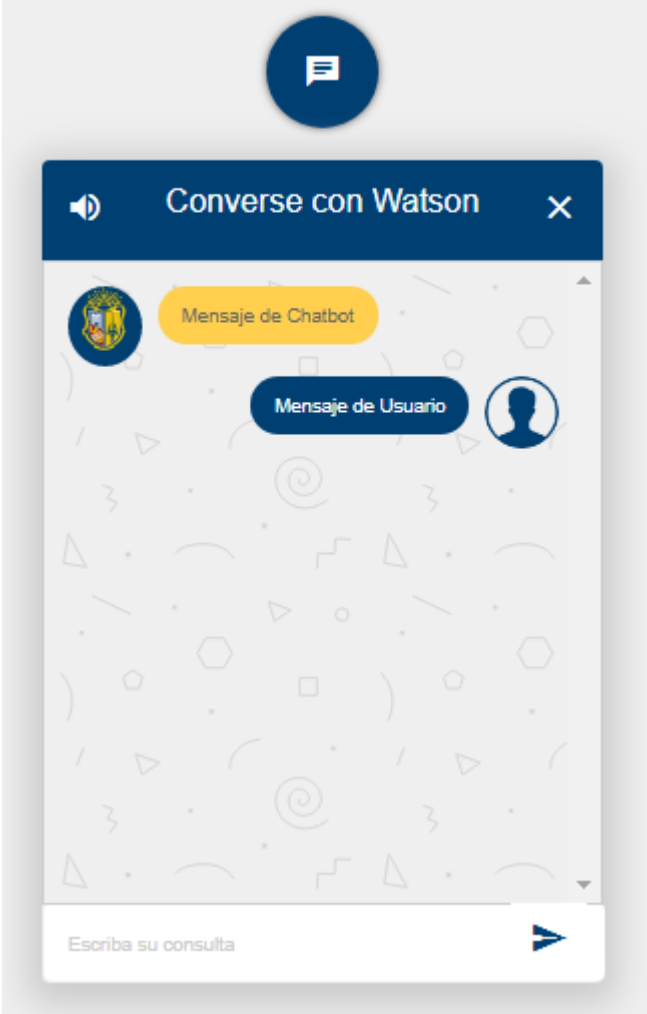
Esta sección le permite personalizar cómo desea que se comporte la caja de chat.

Retraso para mostrar icon chat	0 seconds
Mostrar cuadro de chat	(•) Todas las páginas excepto las siguientes ( ) Solo las siguientes páginas
Página de inicio	[ ] Página de Inicio
Páginas	Seleccione las páginas: [ ] Página de ejemplo 2017-10-20 17:50:14
Posts	Selecciones los posts: [ ] ¡Hola mundo! 2017-10-20 17:50:14
Categorías	Seleccione las categorías: [ ] Sin categoría

La sección de Apariencia permite especificar donde desea que aparezca el cuadro de chat para los usuarios del chat, así como también si desea que el chat aparezca minimizado por defecto.



Se puede configurar el título del chat y al final se presenta una vista previa de como se ve el chat, tanto el icono minimizado como también la caja del chatbot.



En la siguiente figura se presenta el chatbot implementado en la página de becas de la Universidad Técnica Particular de Loja.