Universidad de las Ciencias Informáticas

Facultad 4

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

Título:

Subsistema para la creación de sistemas conversacionales en la suite XEDRO-GESPRO

Autores:

Tania Socarras Navarro

Ariel Acevedo Guevara

Tutores:

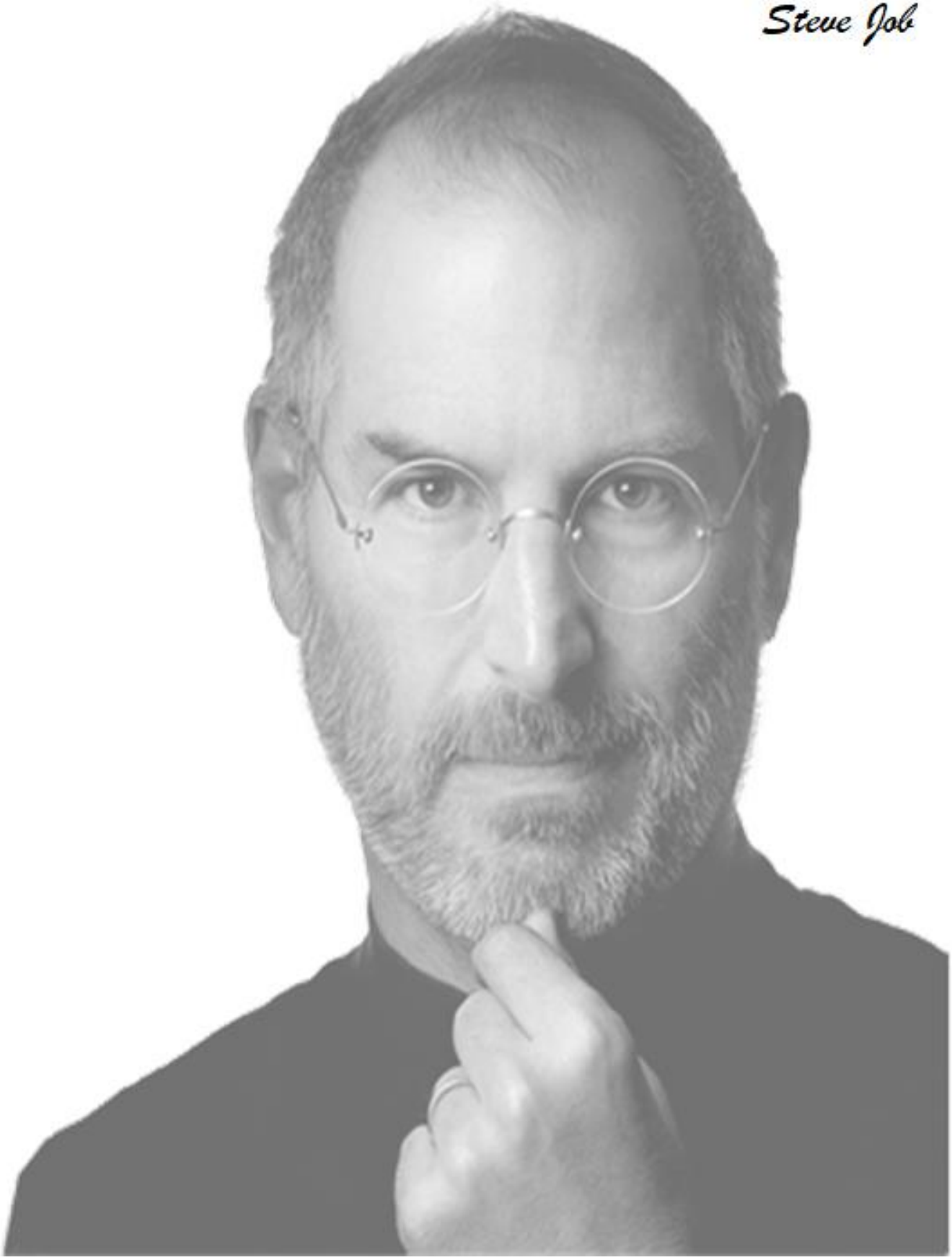
***Ing**. Eliuvis Matos Matos*

***Dr.C** Pedro Yobanis Piñero Pérez*

La Habana, 2019

"Si tú no trabajas por tus sueños, alguien te contratará para que trabajes por los suyos"

Steve Job



Declaración de autoría

Declaramos ser los únicos autores de la presente tesis y reconocemos al Centro de Consultoría y Desarrollo de Arquitecturas Empresariales de la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.

Para que así conste firmamos la presente a los ____ días del mes de _____ del año _____.

Tania Socarras Navarro
Autor

Ariel Acevedo Guevara
Autor

Ing. Eliuvis Matos Matos
Tutor

Dr.C Pedro Yobanis Piñero Pérez
Tutor

Datos de contacto

Tutor: Ing. Eliuvis Matos Matos

Ubicación: Universidad de las Ciencias Informáticas, La Habana, Cuba

Correo Electrónico: emmatos@uci.cu

Tutor: Dr.C Pedro Yobanis Piñero Pérez

Ubicación: Universidad de las Ciencias Informáticas, La Habana, Cuba

Correo Electrónico: ppp@uci.cu

Autora: Tania Socarras Navarro

Ubicación: Universidad de las Ciencias Informáticas, La Habana, Cuba

Correo Electrónico: tsocarras@uci.cu

Autor: Ariel Acevedo Guevara

Ubicación: Universidad de las Ciencias Informáticas, La Habana, Cuba

Correo Electrónico: aacevedo@uci.cu

Dedicatoria

Tania:

Dedico este trabajo a mi mami y papi que fueron el motor impulsor en esta travesía. Ustedes son el tesoro más grande que tengo.

A mi mamá Miriam, donde sea que este que sé que estará orgullosa.

A mi mamá Melba, mi tía Tania y a mi hermano por siempre consentirme y apoyarme en todo momento.

Ariel:

Este logro va dedicado a mi papá, que durante este tiempo ha sido madre y padre a la vez, y que de la noche a la mañana, me ha moldeado y me ha dado el carácter suficiente como para enfrentarme a la vida.

Agradecimientos

Tania:

A mi mami por ser siempre una madre perfecta y única, por brindarme todo tu amor y cariño incondicionalmente. ¡Mami! Te agradezco todo lo que fui, soy y seré, pero sobre todo que seas mi mayor inspiración. Espero que sepas que eres y será siempre mi lucero guía.

A mi ¡Papi!, por siempre ayudarme a cumplir mis metas y darme fuerzas para nunca rendirme ante las adversidades de la vida. Por siempre apoyarme en todo momento, aunque a veces esos momentos fueran de travesuras para mami. Sé que este es una etapa muy feliz para ti porque estás viendo a tu Rucha crecer y volverse la mujer que siempre quisiste. Por todo esto quiero que sepan los dos que el amor que siento por ustedes no tiene un valor calculable, porque los amo con mi vida.

A mis abuelas por darme los regalos más preciados de mi vida ¡A mis Padres!

A mamá Melba por consentirme y siempre estar atentan de mí cuando la necesitaba. A mi mamá Miriam, donde sea que este que sé que estará orgullosa y feliz porque ya su nieta es Ingeniera. Las quiero mucho. A mi hermano que a pesar de sus locuras siempre estuvo ahí para mí, y hoy puedo decirle que soy Ingeniera por los dos. A mi tía Tania que es como una madre.

A mi familia en general, gracias a todos por apoyarme a pesar de la distancia, por sus llamadas durante estos 5 años de la carrera que fueron de mucha ayuda y por siempre tenerme presente.

A mi Cosi, por su amor, cariño y comprensión en los buenos y malos momentos. Fuiste mi amigo, mi compañero y la persona que me ha hecho más feliz en esta aventura.

A mi compañero de tesis Ariel, por siempre estar ahí, por darme apoyo, por los momentos que vividos en estos 5 años de carrera. Por ser un amigo, un hermano y parte de mi familia UCI.

A mis tutores Eliuvis y Pedro que sin su ayuda y sus consejos no hubiera sido posible alcanzar esta meta para mí. Les estoy muy agradecida a los dos por siempre exigirme más para convertirme en una buena profesional. A mi oponente Iliana que siempre estuvo dispuesta a ayudarnos sin importar la hora. A los tres miembros del tribunal que siempre nos dieron buenos consejos y recomendaciones para superarnos.

A los amigos que conocí en estos cinco años de la universidad y a los que se convirtieron en una familia para mí. En especial a: Claudia, Arlette, Lidice, Maday, Yeny, Ewe, Idalis, Sol, Neivis, Carlos y Tito. Gracias por todo y los quiero mucho.

Ariel:

A mi madre que, a pesar de no tenerla a mi lado desde muy pequeño, siempre fue una guía y un estímulo para todos mis logros. Siéntete orgullosa donde quiera que estés, que esa semilla que un día plantaste, ya es un árbol dando sus frutos.

A mi papá, mi héroe, mi Superman, por la confianza que siempre me has tenido. Por brindarme fuerzas en los momentos más bajos y por luchar para darme todo lo que estuvo a tu alcance. Por no rendirte nunca, por enseñarme que con esfuerzo y dedicación todo se logra y que, para lograr un sueño, solo hace falta soñarlo. ¡Te quiero papá!

A mi familia: mi tía Teresa y mis primas Yamisel y Yedelin, que más que primas son como hermanas. Gracias por brindarme todo su cariño y por estar constantemente preocupadas por mí.

A mi suegra Osmaira y mi suegro Boris, por acogerme en su familia como un hijo más. Por quererme y ayudarme en todo lo posible.

A mi novia Arlete por caminar a mi lado en esta universidad. Por los buenos momentos que hemos pasado y por todo lo que hemos logrado juntos. Por nunca perder la fe en mí y por ayudarme a convertirme en una mejor persona.

A mis tutores, a Eliuvis por guiarme y estar disponible en todo momento. Por estar conmigo hasta altas horas de las noches encontrando un simple punto y coma o una ruta mal puesta. Por aparte de tutor, ser amigo incondicional y nunca perder la confianza en mí. Al Doctor Pedro por sus sabias correcciones y por su constante preocupación. Por su típica manera de hablar, que hacía cuestionarse si estábamos en el camino correcto. Gracias a los dos.

A mi compañera de tesis Tania. Que la admiro y la quiero como a una hermana. Gracias por aguantarme estos 5 años, por compartir buenos momentos y tantas alegrías, así como tantas noches realizando esta tesis.

A Frank, mi padrino como cariñosamente le digo. Gracias por tu preocupación y por tus consejos.

A mis amistades: A todos mis compañeros de aula, desde el grupo 19 en la FICI hasta el 4504. A Raidel, Raquel, Juan Pablo, Anisley , Rafael, Rocio , Khaterin , Laurita, Arlet, Eliany , Rachel, Rey y Luis miguel, Eduardo, Sander, Leisdany ,Anabel, Jonathan, Maikel, Yenisse , Alexander, Pabel, Karen, Erick y Ewelyn.

A mi piquete como les digo cariñosamente. A Claudia, Tito, Lidice, Carlos, Maday. Personas que son incondicionales y que puedo contar con ellos para cualquier problema.

Al colectivo de Futbol: Roberley, Yeidel, David, el chino, Aroldo, Efrain, Sabuque, Harol, Osvaldo, Marquito, Ali, Pablo, Lobaina, Ernan, Yasmani, Alvaro.

A mis dos grandes amistades. Pedro y Andy. Dos de las primeras personas con las que compartí las canchas en la uci.

A mis amistades del grupo 5. Milena, Leisy, Alain, Machin, Zuyen que fue, además de mis compañeros de cuarto en primer año, una de las primeras personas que conocí en la universidad.

A todos muchas gracias.

Resumen

En muchas ocasiones la información no está disponible, y, el acceso y la búsqueda de la misma se torna difícil para los usuarios. Por esta razón, muchas empresas o entidades están en la constante búsqueda de herramientas y tecnologías que faciliten que los usuarios conozcan, de manera interactiva, sobre las características de sus productos y servicios. Una de las tecnologías que soluciona esta dificultad es la integración de sistemas conversacionales o *chatbots,* enfocados mayormente en prestar servicios de atención a los clientes de las empresas.

El Centro de Desarrollo de Arquitecturas Empresariales (CDAE) es una de estas entidades que busca brindar información de sus servicios de manera fácil e interactiva. El centro desarrolla la *suite* XEDRO-GESPRO, que es una plataforma que permite gestionar de forma organizada las actividades que se desarrollan en la ejecución de los proyectos. Esta no cuenta con un canal de comunicación que esté disponible a tiempo completo, donde se puedan consultar dudas referentes al uso de la *suite* y a su funcionamiento. Además, la búsqueda de información en su manual de usuario se dificulta debido a que es muy extenso. Por estos motivos se desarrolló un subsistema que permitiera crear sistemas conversacionales para asistir a los usuarios que interactúan con la *suite.*

Dicho subsistema se integra como un módulo más para XEDRO-GESPRO y consta de tres áreas. La primera gestiona toda la información referente a los sistemas conversacionales y sus componentes. La segunda está compuesta por la interfaz de mensajería a través de la cual se establece la comunicación entre usuario y el sistema conversacional. Y la tercera obtiene los datos de la interacción entre el usuario y el sistema*,* que se utilizarán para el entrenamiento del mismo.

Palabras claves: Comunicación, Disponibilidad, Información, Sistema Conversacional.

Abstract

In many occasions the information is not available and access and search for it becomes difficult for users. For this reason, many companies or entities are constantly searching for tools and technologies that facilitate users to know, interactively, about the characteristics of their products and services. One of these technologies that solves these difficulties is the integration of conversational systems or Chabot's, focused mainly on providing services to the clients of the companies.

The Center for the Development of Business Architectures (CDAE) is one of these entities that seeks to provide information about its services easily and interactively. The center develops the XEDRO-GESPRO suite, which is a platform that allows managing in an organized way the activities that are developed in the execution of the projects. This does not have a communication channel that is available full time, where you can consult questions regarding the use of suite and its operation. In addition, the search for information in your user manual is difficult because it is very extensive. For these reasons, a subsystem was developed that allowed the creation of conversational systems to assist users who interact with the suite.

This subsystem is integrated as a module for XEDRO-GESPRO and consists of three areas. The first manages all the information regarding the conversational systems and their components. The second is composed of the messaging interface through which the communication between the user and the conversational system is established. And the third one obtains the data of the interaction between the user and the system, which will be used for the training of the same.

Keywords: Communication, Availability, Information, Conversational System.

Índice de Contenido

Índice de Tabla

Índice de Imagen

Índice de Ecuaciones

Introducción

La comunicación es considerada un pilar fundamental en el desarrollo y evolución de la sociedad. La mayoría de las actividades que realizan los seres humanos tienen como base este proceso, el cual permite transmitir información y conocimiento. Se podría pensar que la comunicación tiene solo una finalidad meramente informativa, pero en realidad se trasmiten elementos más profundos como los sentimientos y pensamientos (Cambría, 2016).

El desarrollo tecnológico alcanzado a lo largo de la historia ha posibilitado que se amplíen y creen nuevas formas de comunicación, entre las más comunes que se utilizan se encuentran: el teléfono, el correo postal y el correo electrónico. Paralelamente a esto, también ha aumentado considerablemente la cantidad de información que se quiere comunicar.

Debido al gran volumen de datos que se generan en el mundo cada segundo y la necesidad que tienen las personas por transmitir y guardar la información, existe la constante búsqueda de nuevas y mejores técnicas que satisfagan las necesidades de comunicarse de las personas. Lo anterior trae como consecuencia que surja lo que se conoce como Tecnologías de la Información y Comunicaciones (TIC). Las TIC es un término plural que denota el amplio espectro de tecnologías vinculadas al procesamiento de información y al envío y la recepción de mensajes (UNESCO Biblioteca Digital, s. f.). Estas se centran en mejorar la disponibilidad de la información, el intercambio de esta y que se realice de forma segura.

Las TIC han permitido además llevar las comunicaciones a niveles más sofisticados pero que a la vez facilitan la accesibilidad de información. Uno de estos avances es que el proceso de comunicarse ya solo no se realiza entre dos o más personas, sino también entre una persona y un programa informático. Ejemplo de esto fue la creación de Eliza, considerado el siglo pasado como uno de los primeros programas informáticos capaz de mantener una conversación coherente con el usuario (Shum, He, & Li, 2018).

Desde el nacimiento de Eliza, hasta nuestros días, se han creado sistemas informáticos capaces de simular conversaciones reales con las personas. Estos sistemas se les conocen como sistemas conversacionales, *chatbots* o *bots* conversacional. Además, se han desarrollado distintas plataformas que permiten la creación de *chatbots* e integrarlos a cualquier compañía o negocio, permitiendo que estas entidades aumenten su estrategia de *marketing* digital para conseguir clientes potenciales. También, utilizan estos medios para

que los servicios de atención al cliente sean más interactivos y para que las consultas de información de los servicios y productos que se brindan estén disponibles a tiempo completo. Está tecnología ha llegado a casi todos los lugares del planeta manifestándose como prototipos de investigación o aplicaciones comerciales; posibilitando al usuario obtener información, conducir transacciones o desarrollar otras tareas orientadas a la solución de algún problema determinado (Tade & García, 2014).

En Cuba estos sistemas son una tecnología novedosa y prácticamente incipiente. Lo cual no imposibilita que se realicen estudios sobre el tema, y que la informática se siga desarrollando con el fin de adquirir conocimientos sobre estas nuevas formas de comunicación.

La Universidad de Ciencias Informáticas (UCI), es una institución donde se vincula la docencia con la producción de *software*, además desempeña una labor fundamental en el proceso de informatización del país. Para cumplir su misión, esta entidad cuenta con varios centros que desarrollan actividades de investigación y producción, encargados de las aplicaciones informáticas, el desarrollo tecnológico y las investigaciones asociadas («Centros de Desarrollo», s. f.). Dichos proyectos se realizan tanto para clientes nacionales como internacionales.

Uno de los centros de este campus universitario es el Centro de Consultoría y Desarrollo de Arquitecturas Empresariales (CDAE). Este cuenta con un departamento que se especializa en la investigación y el desarrollo de aplicaciones orientadas a la gestión de proyectos. En este departamento se trabaja en dos líneas fundamentales: el desarrollo de una *suite* de Gestión de Proyectos denominada XEDRO-GESPRO que permite gestionar de forma organizada las actividades que se desarrollan en la ejecución de los proyectos y la “Maestría en Gestión de Proyectos Informáticos” (MGPI), un programa de postgrado que tiene la misión de fomentar las buenas prácticas en el desarrollo de proyectos y la dirección de equipos de proyecto.

La *suite* XEDRO-GESPRO es constantemente actualizada y se le incorporan nuevas funcionalidades. Las nuevas actualizaciones permiten mejorar las prestaciones de la herramienta, poniendo a disposición de los usuarios nuevas funcionalidades que les ayuden a gestionar la información de sus proyectos. Dicha *suite* es un sistema que se compone de varios módulos o servicios. A causa de lo anterior, el proceso de implantación de esta solución en las determinadas empresas o entidades debe ir acompañado de un proceso de capacitación. Esta capacitación se brinda a las personas que van a interactuar con el

sistema y es donde se explica el funcionamiento para adiestrarlos en el manejo de la herramienta. Simultáneamente al proceso de capacitación, se entrega un manual de usuario que cuenta con más de 600 páginas donde se detallan todos los componentes con los que cuenta la *suite.* Este escenario de desarrollo y evolución está limitado por los siguientes elementos:

- La *suite* XEDRO-GESPRO, está orientada mayormente a decisores y expertos, por tanto, los de pocos años de experiencias en el área están menos familiarizados con los términos y no dominan el flujo de información en los procesos de gestión de proyectos.
- Mientras que no se realice el proceso de capacitación a los usuarios, estos no saben cómo utilizar la *suite*, desconocen sus funcionalidades y los componentes que esta incorpora.
- El manual de usuario es muy extenso por lo que se hace difícil para el usuario la búsqueda de información.
- La *suite* cuenta con un módulo de ayuda que contiene videos que explican cómo trabajar con el sistema, pero no abarca detalles específicos de su funcionamiento.
- Los usuarios no cuentan con un canal de comunicación a tiempo completo donde puedan consultar las dudas referentes al uso de la *suite* XEDRO-GESPRO.

A partir de estos elementos, se plantea el siguiente **problema de investigación**: ¿Cómo contribuir a aumentar la accesibilidad a la información desde el enfoque de modelos conversacionales de los procesos que se gestionan en la *suite* XEDRO-GESPRO?

Teniendo en cuenta el problema planteado se define como **objeto de estudio:** accesibilidad a la información desde el enfoque de modelos conversacionales y como **campo de acción:** Desarrollo de sistemas conversacionales basados en tecnologías de *chatbot*.

En relación con el problema de investigación, se plantea el siguiente **objetivo de investigación:** Desarrollar un subsistema que permita la creación de sistemas conversacionales basado tecnologías de *chatbot* con el propósito de aumentar la accesibilidad a la información de los procesos que modela la herramienta XEDRO-GESPRO.

Para llevar a cabo el desarrollo de la presente investigación y dar cumplimiento al objetivo planteado se desarrollan los siguientes **objetivos específicos:**

1. Caracterizar los sistemas conversaciones a través de la elaboración de un marco teórico para conocer los métodos y herramientas utilizadas en el desarrollo de sistemas conversacionales.
2. Sistematizar el conocimiento sobre las herramientas para el desarrollo de sistemas tanto desde la modelación como de la implementación.
3. Obtención de datos sobre el dominio del problema (conceptos, relaciones, restricciones, reglas, flujo de información) a través del estudio de la bibliografía.
4. Selección y configuración de las herramientas para el desarrollo del sistema.
5. Implementación del subsistema que permitirá la creación de sistemas conversacionales.
6. Validar la investigación a través de un conjunto de pruebas que contemplen su efectividad, eficiencia y validez.

Para resolver el problema dado, se plantea la siguiente **hipótesis:** si se desarrolla un subsistema que permita la creación de sistemas conversacionales basado en tecnologías de *chatbot*, se podrá aumentar la accesibilidad a la información desde el enfoque de modelos conversacionales. A continuación, se definirán las variables independientes y dependientes para poder realizar una correcta validación de la hipótesis planteada:

Tabla 1: Variable dependiente.

	Indicadores	**Unidad de medida**
Aumentar la accesibilidad a la información desde el enfoque de modelos conversacionales.	Nivel de satisfacción del cliente.	__ Sí __ No

Tabla 2: Variable independiente.

	Indicadores	**Unidad de medida**
Subsistema que permita la creación de sistemas conversacionales basado tecnologías de *chatbot.*	Eficiencia	__ Alto __ Medio __ Bajo
	Adecuación funcional	__ Alto __ Medio __ Bajo
	Tiempo de respuesta	__ Alto __ Medio __ Bajo

Para el desarrollo de estos objetivos, se emplearon varios métodos científicos de investigación. El método científico de investigación es la forma de abordar la realidad, de estudiar la naturaleza, la sociedad y el pensamiento, con el propósito de descubrir su

esencia y sus relaciones. Estos métodos se clasifican en teóricos y empíricos (León & González, 2012).

Los métodos teóricos permiten estudiar las características del objeto de investigación que no son observables directamente. Facilitan la construcción de modelos e hipótesis de investigación, contribuyendo al desarrollo de las teorías científicas y para su ejecución se apoyan en el proceso de análisis y síntesis (León & González, 2012); de ellos se emplearon los siguientes:

Métodos de Teóricos:

- ✓ *Histórico-Lógico:* Se emplea para identificar las tendencias actuales de los sistemas conversacionales y la evolución de estos.
- ✓ *Analítico-Sintético:* Para el análisis, evaluación y selección de las herramientas y tecnologías a emplear para el desarrollo de la aplicación. Además, en la identificación de los elementos del marco teórico de la investigación.
- ✓ *Modelación:* Método mediante el cual se crean abstracciones con el objetivo de explicar la realidad (León & González, 2012). Utilizado para modelar mediante diagramas las características de la propuesta, facilitando la compresión de las relaciones y clases presentes en el sistema.

Los métodos empíricos representan un nivel de la investigación cuyo contenido procede de la experiencia y es sometido a cierta elaboración racional (León & González, 2012)de estos se utilizaron los siguientes:

Métodos de Nivel Empírico:

- ✓ *Entrevistas:* Se utilizó para intercambiar información con especialistas y profesores sobre las características que debe cumplir el sistema.

Capítulo 1: Marco teórico referencial

En este capítulo se abordarán los conceptos y definiciones fundamentales en relación con los sistemas conversacionales y las plataformas que permiten crear sistemas conversacionales, así como sus características, ventajas y desventajas. Además, se exponen las principales tecnologías, herramientas y metodología que serán utilizadas para llevar a cabo la implementación de la solución.

1.1. Inteligencia artificial, agente inteligente y sistemas conversacionales

El término Inteligencia Artificial (IA) se define como “la capacidad de las máquinas y/o programas de que actúen y tengan un pensamiento racional similar al de los humanos, sobre problemas poco convencionales y de los cuales no se conoce una respuesta factible o es muy difícil de discernir” (Matos, 2016). Otra definición se refiere a la capacidad de emular las funciones inteligentes del cerebro humano (Badaró, Ibañez, & Agüero, 2013).

En general, estas definiciones indican que la IA es el conjunto de técnicas dedicadas a la creación de hardware y *software* que simulan el pensamiento humano y el comportamiento de los sistemas naturales. Su principal objetivo es llevar a la computadora capacidades para resolver problemas para los cuales no hay algoritmos o que los que existen tienen complejidad no polinomial. Y su fin no es reemplazar al hombre, sino proveerlo de herramientas poderosas para asistirlo en su trabajo (Pérez & Pupo, 2019).

Dentro del extenso campo de la IA, se encuentran los Agentes Inteligentes (AI). Estos son entidades capaces de percibir su medioambiente con la ayuda de sensores y actuar en ese medio utilizando elementos que reaccionan a un estímulo realizando una acción (Russell & Norvig, 2010).Es posible clasificar los AI en (Russell & Norvig, 2010):

- ✓ Agentes reactivos simples: responden directamente a las percepciones.
- ✓ Agentes reactivos basados en modelo: mantienen un estado interno que les permite seguir el rastro de aspectos del mundo que no son evidentes según las percepciones actuales.
- ✓ Agentes basados en objetivos: actúan con la intención de alcanzar sus metas.
- ✓ Agentes basados en utilidad: intentan maximizar su “felicidad” deseada.
- ✓ Agentes conversacionales: pueden dialogar directamente con su interlocutor.

Un sistema conversacional, *bot* conversacional o *chatbot* es un agente conversacional. Ya que estos sistemas conversacionales son una herramienta que permite una interacción entre humano y máquina, simulando la experiencia de estar hablando con una persona real

(Campos & Díaz, 2015). Entre estos podemos encontrar los que tienen un diálogo dirigido por la máquina y los que son de respuesta interactivos. Los primeros tenderán a realizarle al usuario una serie de preguntas, mientras que los segundos podrán interactuar con el usuario de una forma más natural (Rojas, 2001).

1.2. Características de los sistemas conversacionales

Los sistemas conversacionales cuentan con varias características que los diferencian de otros sistemas existentes. Además, presentan una arquitectura propia y están estructurados por varios elementos.

1.2.1. Arquitectura de los sistemas conversacionales

La arquitectura básica de los sistemas conversacionales se puede observar en la Imagen 1, y el proceso que esta refleja es el siguiente (Campos & Díaz, 2015):

- El conocimiento del usuario que crea el *chatbot* debe introducirse en la base de conocimiento en función de las plantillas, patrones y reglas.
- El usuario introduce la entrada mediante un campo de texto en un formulario y este lo envía al motor de inferencia.
- El motor de inferencia es el encargado de analizar las situaciones correspondientes a cada entrada y obtener la respuesta correcta de acuerdo a la información guardada en la base de conocimiento, enviándola a la interfaz de usuario.

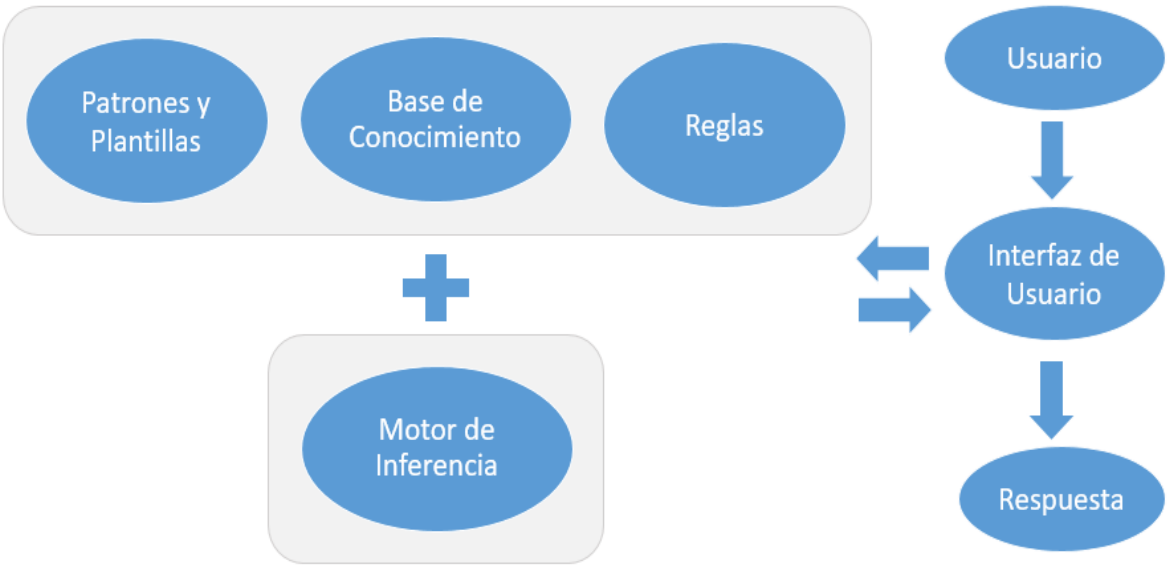


Imagen 1: Arquitectura de los sistemas conversacionales. Fuente:(Campos & Díaz, 2015).

1.2.2. Componentes de los sistemas conversacionales

Generalmente, los *chatbots* están integrados por distintos componentes, estos varían en dependencia de las necesidades y el nivel de sofisticación que se necesite. Estos componentes se obtienen del análisis de varias plataformas porque no existe una regla o patrón que rija su construcción. Entre los conceptos más generales se encuentran:

- **Habilidades:** Es una acción que el *chatbot* sabe hacer, esta se compone de una o varias intenciones que en su conjunto permiten responder sobre un tema en específico. Por ejemplo, una habilidad puede ser “preguntas” y sus intenciones pueden ser “preguntas sobre gestión de proyecto”, “preguntas sobre la maestría de gestión de proyecto” y “preguntas sobre matrícula”.
- **Intenciones:** Corresponde a una acción que el usuario desea realizar, y se expresa mediante un conjunto de expresiones. Por ejemplo, una intención puede ser “saludar” y estaría compuesta por las expresiones “hola”, “buenos días”.
- **Expresiones:** Son las frases, preguntas o el texto que el usuario utiliza para comunicarse con el *chatbot* a través de la interfaz de mensajería.
- **Reglas:** Permiten una vez identificada la intención del usuario, brindarle una respuesta.
- **Entidades**: Son un conjunto de palabras que comparten una característica o están asociadas por un concepto específico.

1.2.3. Ventajas y desventajas del uso de los sistemas conversacionales

Los sistemas conversacionales ofrecen grandes ventajas a cualquier compañía que los integren, a continuación se muestran algunas de ellas (Novicell, 2019):

- **Disponibilidad**: Los *chatbots* pueden estar activos durante todo el día por un periodo largo de tiempo, el cual estará definido por el usuario que lo creó.
- **Costos**: Las empresas que utilizan esta tecnología para interactuar con sus clientes ahorran dinero en contratación de personal pues sustituyen a las personas por *chatbots* para brindar servicios de atención al cliente.
- **Gran volumen de solicitudes:** Pueden atender muchas peticiones de varios usuarios a la misma vez y responder a todas las preguntas que se le realizan.
- **Analítica:** Registran datos, tendencias y métricas para posteriormente monitorear las interacciones, ajustar sus procesos y respuestas en consecuencia.

- **Inteligencia artificial:** Los *chatbot*s acumulan el conocimiento que reciben y se vuelven más inteligentes y flexibles después de cada conversación con el cliente.

Estos sistema presentan ciertas desventajas en su uso, tales como (Novicell, 2019):

- **Inflexible:** El *chatbot* creado no permitirá lidiar con algunas preguntas porque no estarán en su base de datos. Si el *chatbot* no puede ofrecer una respuesta y no hay presencia humana, el usuario que interactúe con él, puede sentirse frustrado e insatisfecho.
- **Pérdida de datos:** Como todo sistema informático, están expuestos a fallas y si no se tiene una copia con los datos que se han generado de las interacciones entre usuarios y *chatbots*, esta información podría perderse.
- **No aceptación por los usuarios:** Muchas de los usuarios prefieren hablar con una persona real y no con *chatbots*.

1.3. Plataformas conversacionales

Las plataformas conversacionales son herramientas que permiten la interacción humano/máquina para comunicarse con aplicaciones, sitios *web* y dispositivos utilizando un lenguaje natural a través de voz o texto (Ochoa, 2018). Estas permiten crear sistemas conversacionales con características que difieren en dependencias de la plataforma que se utilice.

1.3.1. Características de las plataformas conversacionales

Las plataformas conversacionales tienen como objetivo crear interfaces de usuario de conversación, *chatbots* y asistentes virtuales. Ofrecen integración en interfaces de *chat* como plataformas de mensajería, redes sociales, servicio de mensajes simples (SMS), *chat* de sitios *web* o similares. La mayoría de estas tienen una *API*[1] de desarrollador para que terceros puedan ampliar la plataforma con sus propias personalizaciones y adiciones (Revang, Baker, Manusama, & Mullen, 2018).

Estas están siendo empleadas por las empresas para construir aplicaciones que responden preguntas y pueden dar soporte en múltiples servicios sin necesitar supervisión humana (Ochoa, 2018). Se encuentran disponibles en cualquier momento y muchas cuentan con una estructura sencilla en el proceso de construcción de un sistema conversacional.

[1] Una interfaz de programación de aplicaciones (por sus siglas en inglés API) es un conjunto de herramientas, definiciones y protocolos para crear e integrar software de aplicaciones (Red Hat, 2019).

Inicialmente, muchas son gratuitas durante un corto periodo de tiempo, en el cual la compañía o el usuario que la utilice prueban sus funcionalidades y servicios.

Muchos usuarios y compradores de las empresas interactúan con ellas a través de mensajes, utilizando *Facebook, Messenger, Telegram* u otras plataformas de mensajerías. Los *chatbots* creados por las plataformas conversacionales están orientados a permitir que las empresas y compañías lleguen a sus consumidores y clientes de una manera más interactiva y en su entorno.

1.3.2. Ventajas y desventajas del uso de plataformas conversacionales

Las plataformas conversacionales ofrecen grandes beneficios a cualquier empresa que las utilice como son (García, 2018):

- **Servicio personalizado:** Las empresas utilizan las plataformas para crear sistemas conversacionales personalizados teniendo en cuenta el cliente que la utilice.
- **Integración:** Es la capacidad de las plataformas de interconectarse con redes sociales de manera rápida y de realizar trámites de todo tipo sin tener que cambiar de canal. Estas se pueden usar a través de aplicaciones descargadas en dispositivos móviles en el momento en que el cliente estime.
- **Ahorro de tiempo:** Estas plataformas permiten ahorrar tiempo en la contratación de personal y en el trato con el cliente, permitiendo una mayor eficiencia y facilidad de interacción a los usuarios que la utilicen.

Pero estas plataformas traen consigo ciertas desventajas, tales como (García, 2018):

- **Costos:** Muchas de estas que son utilizadas para crear sistemas conversacionales inicialmente gratuitos y para acceder a funcionalidades más complejas o servicios de soporte, es necesario suscribirse a un plan de pago en un período de tiempo determinado.
- **Manejo de la información**: Crean *chatbots* realizando este proceso para plataformas de mensajerías como *Messenger* o *Telegram*. La información que se extrae de las interacciones con los usuarios y estos sistemas conversacionales, no solo serán gestionados por el usuario o la empresa que los creó, sino también por estas plataformas de mensajería.
- **Inseguridad:** Usan los datos de los usuarios para darles una mejor atención y por esto es importante que tengan una copia de seguridad de estos datos porque al llegar a perderlos puede ser costoso recuperarlos y también es difícil buscar personas que le respondan a los clientes (Guevara, 2019).

1.3.3. Ejemplos de plataformas de sistemas conversacionales

En la actualidad existen muchas plataformas para el desarrollo de *chatbots* que son sencillas y fáciles de usar. En la mayoría de ellas solo hay que definirles el flujo de la conversación y la estructura que estos presentarán. A continuación, se describen unos ejemplos de plataformas que se utilizan para la creación de sistemas conversacionales.

Chatfuel

Esta plataforma fue creada en el 2015 con el objetivo de facilitar la creación de *bots* conversacionales para cualquier usuario. En la actualidad su enfoque está dirigido a que sus usuarios puedan crear *chatbots* en *Facebook Messenger* (Chatfuel, 2019b). Hasta el momento de esta investigación, esta plataforma presenta un plan para la versión libre y otro plan para la versión de pago. La primera, permite un máximo de 1000 usuarios a interactuar con el *chatbot* creado, brindando características básicas y herramientas esenciales para automatizar (Demeter, 2019). Si sobrepasa esta cifra, los usuarios restantes no podrán interactuar con el *chatbot* hasta que no se cambia a la versión de pago. Con esta versión los usuarios obtienen una serie de beneficios para realizar su *chatbot* más robusto y con mejores funcionalidades. Muchas multinacionales como *Adidas*, *MTV*, *British Airways*, *Volkswagen* y otras usan *Chatfuel* para sus *bots* conversacionales de *Facebook Messenger* (Chatfuel, 2019a).

Flow XO

Flow XO es una plataforma completa para crear y administrar *chatbots* para plataformas de mensajería, como *Messenger, Slack y Telegram* (Flow XO, 2018). Esta permite adaptar el *chatbot* y sus flujos de trabajo para satisfacer las necesidades que presente el usuario. Una manera sencilla de usar el *chatbot* creado es como saludo para un sitio *web*. O si el usuario tiene algún negocio y necesita que su *bot* conversacional sea más hábil desde el punto de vista técnico, esta plataforma permitirá configurarlo para que comprenda las consultas y frases comunes asociadas con su producto o servicio que ofrezca (Flow XO, 2019a). Está limitado a ciertos números de conversaciones, luego de lo cual requiere una subscripción para continuar su uso y un plan de pago de tiempo mensual.

Octane.ai

Octane AI es una plataforma creadora de *chatbot* para *Facebook Messenger* fundada en mayo del 2016 (Parr, K-Brooks, Berry, & Schlicht, 2019). Esta tiene como misión mejorar la experiencia de compra para los comerciantes y sus clientes. Es utilizada por los

comerciantes para crear *chatbots* que envían mensajes a los suscriptores y automaticen el *marketing* realizado a través de *Messenger* (Octane AI, 2019). Usando *Octane AI*, los comerciantes pueden hacer lo siguiente con el *marketing* de *Messenger:*

- Enviar automáticamente notificaciones.
- Recomendar productos a los visitantes en función de su historial de navegación.
- Automatizar el servicio al cliente y verificar el estado del pedido de un cliente sin abrir otra pestaña.

Botsify

Esta es una plataforma para crear *chatbots* para *Facebook.* Los *chatbots* creados tienen la capacidad de recopilar los datos escritos por los usuarios a través de las diversas conversaciones y luego construir su propia base de datos, que se utilizará para responder a futuras preguntas. La plataforma viene con algunas plantillas integradas que pueden ser útiles para aplicaciones específicas: plantillas para hospitales, agencias de viajes, restaurantes y preguntas frecuentes en general. Además, cuenta con técnicas de procesamiento del lenguaje natural para que procesar las entradas del usuario (Suja, 2018). En la siguiente tabla se muestra un resumen las características de cada una de las plataformas mencionadas anteriormente:

Tabla 3: Tabla comparativa de sistemas homólogos.

Características / Sistemas	***Chatfuel***	***Botsify***	***Octane.ai***	***Flow XO***
IA	Trabaja mediante un sistema basado en regla (Chatfuel, 2019a).	Procesamiento del lenguaje natural (Suja, 2018).	Crea una base de conocimiento (Octane AI, 2019).	Lógica condicional (Flow XO, 2018).
Cantidad de usuarios que interactúan con el *chatbot*	Para el plan gratuito, el límite de usuarios accesibles es de 1,000 usuarios (Demeter, 2019).	Los *chatbot* creados podrán interactuar con 30,000 usuarios (Noman, 2019).	Podrán interactuar con el *chatbot* una cantidad de 150,000 usuarios (Schlicht et al., 2019).	Esta plataforma tiene un plan gratuito en el que permite hasta 500 interacciones y 5 *chatbots*. El plan estándar permite hasta 5,000 interacciones y 15

				chatbots (Flow XO, 2019b).
Pago	La versión libre de pago no tiene costo, en cambio para la versión de pago y para la versión de pago superior hay que efectuar un costo según la cantidad de usuarios con los que interactúe el *chatbot* (Demeter, 2019).	Es gratis durante los primeros 14 días, luego de este tiempo, se deberá suscribirse al Plan de pago, y realizar un pago de $ 50 al mes (Noman, 2019).	La versión básica permite usarla por 30 días de manera gratuita, luego hay que realizar un pago de $ 9 dólares al mes. La versión de pago usarla por 30 días de manera gratuita y luego hay que realizar un pago de $ 209 al mes (Schlicht et al., 2019).	Tiene un plan gratuito con 2 semanas de registro e incluye soporte y el plan estándar con un costo de $ 19 al mes e incluye 3 meses de registros y soporte prioritario. Se puede añadir 5 *chatbots* por un precio de $ 10 al mes o 25,000 interacciones por un precio de $ 25 (Flow XO, 2019b).

Estas plataformas pueden llegar a convertirse en grandes soluciones debido a la sencillez que presentan en el proceso de creación de sistemas conversacionales, pero muchas de ellas tienen limitantes en cuanto a eficiencia y robustez. Esto es porque el acceso a mejores funcionalidades y asesoramiento para transformar el *chatbot* creado no es posible sin antes haber realizado un pago. Otra limitante que presentan estas plataformas son la cantidad de interacciones que pueden tener los usuarios con el *chatbot*, así como el número de *chatbots* que se les está permitido. Además, muchas de estas plataformas están conectadas en *Facebook Messenger,* lo que implica que la información que genere y guarden los *chatbots* no será gestionada solamente por el usuario creador, como son los ejemplos de *Botsify, Octane AI* y *Chatfuel*.

1.4. Sistemas Basados en Conocimiento

Dentro del campo de la IA existen varias técnicas y métodos, entre ellos los Sistemas Basados en Conocimiento (SBC) que permiten adquirir información para convertirlo en conocimiento. Los SBC son definidos como "un sistema computarizado que usa conocimiento sobre un dominio para arribar a una solución de un problema de ese dominio. Esta solución es esencialmente la misma que la obtenida por una persona experimentada en el dominio del problema cuando se enfrenta al mismo problema" (Matos, 2016).

La información en los SBC varía en dependencia de la forma en que se represente ese conocimiento almacenado. A estas se le conoce como Formas de Representación del Conocimiento (FRC) y dentro de ellas se encuentran: los Sistemas Basados en Reglas (SBR), los Sistemas Basados en *Frames*, los Sistemas Basados en Casos, los Sistemas Basados en Probabilidades, Redes Expertas y los Sistemas Basados en Modelos (Badaró et al., 2013).

Los SBR trabajan mediante la aplicación de reglas, comparación de resultados y aplicación de las nuevas reglas basadas en la situación modificada (Badaró et al., 2013).

También pueden trabajar por inferencia lógica dirigida, bien empezando con una evidencia inicial en una determinada situación y dirigiéndose hacia la obtención de una solución, o bien con hipótesis sobre las posibles soluciones y volviendo hacia atrás para encontrar una evidencia existente (o una deducción de una evidencia existente) que apoya una hipótesis en particular.

1.5. Procesamiento del Lenguaje Natural aplicado a plataformas conversacionales

El procesamiento del lenguaje natural (PLN) emplea técnicas computacionales con el fin de aprender, comprender y producir contenido en lenguaje humano. Los primeros enfoques computacionales para la investigación del lenguaje se centraron en automatizar el análisis de la estructura lingüística del lenguaje y desarrollar tecnologías básicas como la traducción automática, el reconocimiento de voz y la síntesis de la voz (Hirschberg & Manning, 2015).

El PLN se ocupa de la formulación e investigación de mecanismos eficaces computacionalmente para la comunicación entre personas y máquinas por medio de lenguajes naturales (Soto, 2018).

El PLN no trata de la comunicación por medio de lenguajes naturales de una forma abstracta, sino de diseñar mecanismos para comunicarse que sean eficaces computacionalmente y que se puedan realizar por medio de programas que ejecuten o

simulen la comunicación (Soto, 2018). Este campo está muy relacionado con la creación de *bots* conversacionales; pues estos, tienen que comprender las frases, preguntas y textos que el usuario envíe durante su interacción con él.

Muchas veces resulta complejo programar y estructurar un sistema conversacional que sea capaz de reconocer eficientemente el lenguaje natural. Lo anterior está dado porque los usuarios pueden expresar una misma idea de varias formas posibles. Para mitigar este problema, dentro de esta área existen métodos y técnicas que permiten analizar la información escrita u oral. Entre estas tareas se encuentra la similitud textual, que se encarga de comparar textos para conocer el parecido entre ellos (Carmona, 2014).

Dentro de la similitud textual, existen medidas de similitud de palabras y medidas de similitud de textos. Las medidas de similitud de texto se representan a través de una matriz vectorial de alta dimensionalidad que hace una comparación entre los textos. Por otro lado, las medidas de similitud de palabras calculan la semejanza de cada parte de ambos textos para obtener un solo resultado de similitud (Carmona, 2014).

Algunas de las técnicas de medida de similitud de palabras que podemos encontrar son: basadas en caracteres y basadas en términos. Las basadas en caracteres consideran cada cadena de caracteres como una secuencia ininterrumpida de caracteres (Carmona, 2014). Las técnicas basadas en términos consideran cada cadena como un conjunto de subcadenas separadas por caracteres especiales (Amón & Jiménez, 2010).

Estas dos técnicas tienen diversas funciones de similitud que son aplicables para el trabajo con textos cortos. Ejemplos de estas funciones son: distancia de *Jaro-Winkler*, el algoritmo de *Levenshtein* y similitud de coseno *TF-IDF*. Esta última función no es eficaz bajo la presencia de variaciones a nivel de caracteres, como errores ortográficos o variaciones en el orden de los términos. Por tanto, *Cohen et al.* (2003) proponen una variante llamada *Soft TF-IDF* para solucionar este problema (Amón & Jiménez, 2010).

Técnicas basadas en caracteres

- ✓ **Distancia de *Jaro-Winkler***

La función *Jaro* determina la similitud entre dos secuencias de textos sobre la base de un carácter común. Además de calcular el número de transposiciones que puede existir entre estos caracteres comunes, como la comparación fallida del i-ésimo de estos entre las dos cadenas. Esta técnica se utiliza principalmente en el área de detección de duplicados de textos de pequeña longitud (Ruiz, 2017).

Winkler en 1990 propone una variante que asigna puntajes de similitud mayores a cadenas que comparten algún prefijo, basándose en un estudio realizado por *Pollock* y Zamora en 1984. *Cohen et. al.*, en 2003 proponen un modelo basado en distribuciones Gaussianas. *Yancey* (2006) compara la eficacia de la similitud de *Jaro* y algunas de sus variantes. *Winkler*, ha propuesto algunas mejoras de este modelo (Ruiz, 2017).

La distancia *Jaro-Winkler*, sin embargo, utiliza una escala **p** prefijo que emite puntuaciones mucho más adecuadas para las cadenas que coinciden desde el principio para un conjunto **l** longitud del prefijo. Dadas dos cadenas S1 y S2, con estos datos la medida *Jaro-Winkler* **Dw** está definida como la siguiente Ecuación 1 (Ruiz, 2017):

Ecuación 1: Ecuación de similitud de Jaro-Winkler.

$$d_w = d_j + (\ell p(1 - d_j))$$

Donde:

- **dj:** distancia *Jaro* para las cadenas S1 y S2.
- **l:** longitud del prefijo común en el principio de la cadena, hasta un máximo de 4 caracteres.
- **p:** la longitud del prefijo común, p no debe exceder de 0.25, de lo contrario la distancia puede ser mayor que 1. El valor estándar para esta constante en la obra de *Winkler* es p = 0,1 (Ruiz, 2017).

✓ **Algoritmo de *Levenshtein***

Este algoritmo es también conocido como distancia de edición y se basa en permitir número mínimo de operaciones requeridas para transformar una palabra en otra. Se entiende por operaciones de edición, una inserción, eliminación o la sustitución de un carácter (Carmona, 2014). La cuenta final de la distancia se convertirá en porcentaje mediante la Ecuación 2 como se muestra a continuación (Chen, Yong, & Ishak, 2014):

Ecuación 2: Ecuación del algoritmo Levenshtein.

$$Distance = (100 - total\ distance\ count) * 100$$

Donde:

- *total distance count* es el número de caracteres a transformar de la cadena original para obtener la cadena en comparación.

Técnica basada en términos

- ✓ **Similitud de coseno *Sof TF-IDF***

Esta medida tiene en cuenta las parejas de términos ($\propto_i$, β_j) cuya similitud es mayor que cierto umbral mediante alguna función de similitud basada en caracteres (Amón & Jiménez, 2010). El resultado de esta función es el coseno del ángulo entre los dos vectores a comparar; mientras menor sea el ángulo, mayor será la similitud (Carmona, 2014). El valor del ángulo resultante de entre dos vectores se define en la siguiente Ecuación 3:

Ecuación 3: Ecuación de la función de similitud coseno Soft TF-IDF.

$$\theta = \cos^{-1}\left(\frac{\vec{a}\vec{b}}{|\vec{a}||\vec{b}|}\right)$$

A cada una de las medidas mencionadas anteriormente se les realizó un estudio para demostrar su nivel de eficiencia en disímiles situaciones. Y luego determinar cuál de ellas resulta apropiada para emplear en la similitud de textos cortos de los sistemas conversacionales. Dado que estas funciones se emplearán en la comparación con textos que escriban los usuarios. Se debe tener en cuenta que estos textos pueden tener: errores ortográficos, abreviaturas, omisión de términos, prefijo o sufijo sin valor semántico, términos en desorden, espacios en blanco y diferencia de términos escritos en mayúsculas y minúsculas que se deben de validar para lograr una respuesta correcta (Amón & Jiménez, 2010). Basado en estas situaciones mencionadas anteriormente, las funciones de similitud arrojarán resultados de la comparación de dos textos cortos en un intervalo de [0,1], donde: si el resultado es 1 entonces ambos textos son iguales y mientras más cercano sea el resultado a 0 serán diferentes. Teniendo en cuenta todos los aspectos anteriores se elaboró la siguiente tabla comparativa:

Tabla 4: Tabla de comparativa de funciones de similitud de textos cortos. Fuente: (Amón & Jiménez, 2010).

Situación / Técnica	**Ejemplo** “Jorge Eduardo Rodríguez López” vs.	***Levenshtein***	***Jaro-Winkler***	***Soft TF-IDF***
Errores ortográficos	“Jorje Eduadro Rodrigues Lopes”	0.7	0.85	0.89
Abreviatura	“Jorge E Rodríguez L”	0.64	0.81	0.92

Términos faltantes	“Jorge Rodríguez”	0.51	0.75	0.9
Prefijo/ Sufijo sin valor semántico	“PhD Jorge Eduardo Rodríguez López, U Nal”	0.73	0.78	0.93
Términos en desorden	“Rodríguez López Jorge Eduardo”	0.16	0.78	1
Espacios en blanco	“JorgeEduardo Rodríguez López”	0.9	0.87	0.9
Mayúsculas y minúsculas	“EDUARDO” vs. “eduardo”	0	1	0
Promedio		0.52	0.83	0.80

A los valores adquiridos, luego de probar cada una de las funciones con las situaciones anteriores, se les realizó un cálculo de promedio para determinar cuál era mejor para utilizar. De estas funciones se destacaron por su promedio *Jaro-Winkler* y *Soft TF-IDF* siendo la mejor *Jaro-Winkler* pero en cuanto al índice de similitud sobresale *Soft TF-IDF.* Estas diferencias se derivan a que *Soft TF-IDF* tiene seis situaciones con resultados superiores a *Jaro-Winkler,* en cambio, *Jaro-Winkler* solo la supera en una situación con resultados opuestos. En la siguiente tabla se evidencian como serían los resultados de estas dos funciones sin la situación de Mayúsculas y minúsculas.

Tabla 5: Tabla comparativa entre Jaro-Winkler y Soft TF-IDF. Fuente: (Amón & Jiménez, 2010).

	Errores ortográficos	Abreviatura	Términos faltantes	Prefijo/ Sufijo sin valor semántico	Términos en desorden	Espacios en blanco	**Promedio**
Jaro-Winkler	0.85	0.81	0.75	0.78	0.78	0.87	0.8
Soft TF-IDF	0.89	0.92	0.9	0.93	1	0.9	0.92

Por todo lo analizado anteriormente, se puede concluir que por sus buenos resultados las mejores funciones son distancia *Jaro-Winkler* y la variación del coseno *Soft TF-IDF* para aplicar en la solución de la investigación dentro del marco de trabajo de la *suite* XEDRO-GESPRO.

1.6. Marco de trabajo XEDRO-GESPRO

XEDRO GESPRO es una *suite* orientada a la *web* que permite la planificación, seguimiento y control de productos en forma de proyectos. Cuenta con herramientas para el apoyo a la

toma de decisiones a nivel de proyecto, nivel de entidad ejecutora y nivel gerencial. Se presenta en un modelo de negocios basado en servicios que combinan el uso de una solución informática para la dirección integrada de proyectos y un sistema de formación especializada en gestión de proyectos. Esta combinación posibilita no sólo la informatización de las organizaciones sino también la mejora integral de los procesos de planificación, control y seguimiento de proyectos («GESPRO 13.05», s. f.). Esta *suite* cuenta con distintos módulos entre los que se encuentran (Piñero, Torres, Matias, & Garcia, 2013):

- **Módulo de configuración y seguridad**: Incluye facilidades para la configuración y adaptación del sistema para diferentes escenarios.
- **Módulo de planificación de proyectos**: Posibilita la dirección integrada de portafolios de proyectos. Permite la gestión de alcance y tiempo posibilitando la identificación de requisitos y control de la calidad, así como la construcción semiautomática de cronogramas. Facilita, además, la identificación de la línea base de los proyectos y la asignación de recursos materiales y humanos.
- **Módulo de gestión de riesgos:** Posibilita la identificación, análisis y respuesta de los riesgos, compartir riesgos entre diversos proyectos y la generación semiautomática de planes de mitigación y contingencia.

Estos módulos engloban las funcionalidades principales de la *suite*, entre las cuales se destacan («GESPRO 13.05», s. f.):

- Gestión de Tiempo.
- Gestión de Costos.
- Gestión de Contratos.
- Gestión de los Recursos Humanos.
- Gestión de la Calidad.
- Gestión de Riesgos.

1.7. Tecnologías, herramientas y metodología

Para la creación de aplicaciones informáticas es necesario usar tecnologías y herramientas que faciliten definir cuál va a ser su estructura, además de metodologías que permitan regir la forma de trabajo del equipo de desarrollo. A continuación, se exponen la metodología, tecnologías y herramientas a emplear basadas en el marco de trabajo para el desarrollo de la propuesta de solución.

1.7.1. Metodología de desarrollo de *software*

Las metodologías de desarrollo de *software* tienen como objetivo proporcionarle al equipo de desarrollo una guía mediante la cual puedan ordenar las actividades de un equipo, dirigir las tareas de cada miembro del equipo por separado y las del equipo como un todo. También, especifican los artefactos que deben elaborarse y ofrecen criterios para el control, la medición de los productos y actividades del proyecto (Jacobson, Booch, & Rumbaugh, 2011).

Las metodologías de desarrollo se pueden dividir en dos grupos de acuerdo con sus características y los objetivos que persiguen: ágiles y tradicionales o pesadas (Cadavid, Martínez, & Vélez, 2013). Las metodologías ágiles se caracterizan por el desarrollo iterativo e incremental; la simplicidad de la implementación; las entregas frecuentes; la priorización de los requerimientos o características a desarrollar a cargo del cliente; y la cooperación entre desarrolladores y clientes. Las metodologías ágiles dan como un hecho que los requerimientos van a cambiar durante el proceso de desarrollo (Cadavid et al., 2013). Una de las metodologías de desarrollo ágil es la Programación Extrema (XP[2]).

En este trabajo se opta por utilizar esta metodología debido a que XP es una de las metodologías de desarrollo de *software* más exitosas para proyectos de corto plazo, equipos pequeños y donde el cliente es parte del equipo de desarrollo. Además, usa un enfoque orientado a objetos como paradigma de desarrollo, y engloba un conjunto de reglas y prácticas que ocurren en el contexto de cuatro actividades estructurales: planeación, diseño, codificación y pruebas (Pressman, 2010).

Imagen 2: Fases de la metodología de desarrollo XP. Fuente: Elaboración propia.

[2] Proviene de sus siglas en ingles *Extreme Programming.*

En la metodología XP se generan varios artefactos, para el desenlace de la investigación se emplearán los siguientes:

- **Historias de usuario:** Representan una breve descripción del comportamiento del sistema y emplea terminología del cliente sin lenguaje técnico. Se realiza una por cada característica principal del sistema. Difieren de los casos de uso porque son escritas empleando terminología del cliente, son más amigables que los casos de uso formales (Escribano, 2002).
- **Tarjetas CRC:** Estas tarjetas se dividen en tres secciones que contienen la información del nombre de la clase, sus responsabilidades y sus colaboradores. Describen los objetos de la realidad con sus atributos y métodos. Permite la asignación de las responsabilidades entre una clase y su colaboración (Ferigra & Santiago, 2015).
- **Tareas de ingeniería:** Permiten determinar cómo deberían funcionar los requisitos. Para el desarrollador representa la serie de pasos que debe seguir para implementar una historia de usuario, ya que describen lo que debe hacer (Ferigra & Santiago, 2015).

1.7.2. Lenguaje de programación

Un lenguaje de programación es un mecanismo de abstracción que le permite a un programador especificar una operación abstracta y dejar que el programa (usualmente llamado ensamblador, compilador o interprete) implemente la especificación de una forma detallada para la ejecución de esa operación en la computadora o sea que la computadora entienda lo que tiene que hacer para realizar esa operación (Ben-Ari, 2006). Para desarrollar el objetivo de esta investigación, se empleará como lenguaje de programación Ruby porque es sobre el cual está creada la *suite* XEDRO-GESPRO.

Ruby

Ruby es un lenguaje de *script* creado por *Yukihiro Matsumoto*. Además, interpretado para programación orientada a objetos con una filosofía y sintaxis muy limpia lo que hace que programar sea elegante. Cuenta con un sistema de tipo dinámico y automático de gestión de memoria, y, una biblioteca de clases centrales rica y poderosa. Es un lenguaje de programación de alto nivel multiplataforma, es decir trabaja en varios sistemas operativos (Pardo, Tapia, Moreno, & F, 2018).

JavaScript

JavaScript es un lenguaje de programación interpretado, orientado a objetos, basado en prototipos y en funciones de primera clase, más conocido como el lenguaje de *script* de la *web* (Castillo, 2017). Es un lenguaje que no requiere de compilación al ser interpretado directamente por los navegadores. Además, que se puede considerar ligero, ágil y permite crear contenido nuevo y dinámico, controlar archivos de multimedia y crear imágenes animadas.

1.7.3. Biblioteca *JQuery*

JQuery es una biblioteca de *JavaScript* rápida, pequeña y con muchas funciones. Ofrece facilidades para que la manipulación de documentos HTML, el manejo de eventos, la animación y *Ajax*[3] sean mucho más sencillos con una *API* fácil de usar que funciona en una multitud de navegadores (JS Foundation, 2019).

1.7.4. Framework

Un *framework* es un esquema para el desarrollo e implementación de una aplicación, que facilita la programación debido a que muchas funciones vienen implementadas en sus bibliotecas (Aroca, 2018). Para el desarrollo del trabajo se utilizan varios *framework* como:

- ***Rails*:** es un *framework* para el desarrollo de aplicaciones *web* de código abierto, orientado a objeto y escrito en el lenguaje de programación *Ruby*. Siguiendo el paradigma de la arquitectura Modelo-Vista-Controlador (Aroca, 2018).
- ***Highcharts*:** es un *framework* sencillo que consta de una biblioteca escrita en *JavaScript* que permite la creación de gráficos (Highcharts, 2019).

1.7.5. Entorno de Desarrollo Integrado (IDE)

Un IDE es una aplicación informática que proporciona servicios integrales para facilitar el desarrollo de *software* al programador o desarrollador. Un IDE consiste en un editor de código fuente, herramientas de construcción automática y un depurador, además de auto-completado inteligente de código (Fuentes, 2016). *RubyMine* es un entorno de desarrollo dedicado para *Ruby on Rails*. Este IDE ofrece una amplia gama de herramientas esenciales para desarrolladores de *Ruby*, estrechamente integrado en conjunto para crear un ambiente propicio para el desarrollo productivo de *Ruby* y desarrollo *web* con *Ruby on Rails*. Para el desarrollo de la aplicación se emplea *RubyMine* en su versión 2018.1.4 que es gratis.

[3] Asynchronous JavaScript And XML por sus siglas en inglés.

1.7.6. Gestor de Bases de Datos

Un Sistema Gestor de Base de Datos (SGBD), se define como una colección de datos interrelacionados y un conjunto de programas para acceder a dichos datos. El SGBD permite almacenar y recuperar la información de una base de datos (BD) de manera práctica y eficiente («Sistema gestor de base de datos», s. f.). Debido a que en el marco de trabajo del departamento GESPRO se utiliza *PostgresSQL* en su versión 9.2 para la gestión de los datos de la *suite* XEDRO-GESPRO, se empleará este mismo gestor en la investigación.

PostgresSQL

Es un SGBD de código abierto, utiliza un modelo cliente-servidor y usa multi-procesos en lugar de multi-hilos para garantizar la estabilidad del sistema. Un fallo en uno de los procesos no afectará al resto y el sistema continuará funcionando. Para apoyar el trabajo y administración de los datos se utilizará como apoyo la herramienta PgAdmin III (Guerrero, 2019).

1.7.7. Sistema de control de versiones

Es un sistema donde se registran los cambios realizados en un archivo o conjunto de archivos a lo largo del tiempo. Generalmente se utiliza el código fuente del *software* como los archivos que se controlan por versión, aunque en realidad puede hacerse con casi cualquier tipo de archivo en una computadora (GitBook, 2019). Para el presente trabajo se emplea el repositorio *Gitlab* en su versión 11.6.3.

1.7.8. Lenguaje de modelado UML

UML es un lenguaje de modelado visual que se usa para especificar, visualizar, construir y documentar artefactos de un sistema de *software*. Se usa para entender, diseñar, hojear, configurar, mantener, y controlar la información sobre estos sistemas (Rumbaugh, Jacobson, & Booch, 2000).

1.7.9. Herramientas *CASE* (Ingeniería de *Software* Asistida por Computadora)

Las herramientas *CASE*[4] se puede definir a como un conjunto de programas y ayudas que dan asistencia a los analistas, ingenieros de *software* y desarrolladores, durante todos los pasos del ciclo de vida de desarrollo de un *software* (INEI, 1999).

[4] Proviene de sus siglas en inglés, *Computer Aided Software Engineering.*

1.7.10. Herramienta para el modelado

Visual Paradigm para UML es una herramienta *CASE*. La misma propicia un conjunto de ayudas para el desarrollo de programas informáticos, desde la planificación, pasando por el análisis y el diseño, hasta la generación del código fuente de los programas y la documentación. *Visual Paradigm* es una herramienta de *software* diseñada para que los equipos de desarrollo de *software* modelen el sistema de información empresarial y gestionen los procesos de desarrollo. *Visual Paradigm* es compatible con lenguajes y estándares de modelado de la industria clave como el lenguaje de modelado unificado. Ofrece a las empresas de *software* de conjunto de herramientas necesarias para la captura de requisitos, análisis de procesos, diseño de sistemas, diseño de bases de datos (Visual Paradigm, s. f.). La herramienta que se empleará para el diseño de la investigación es *Visual Paradigm* en su versión v-5.0.

1.8. Conclusiones del capítulo

A partir del estudio realizado en el marco teórico referencial a plataformas que existen y permiten la construcción de sistemas conversacionales, fue posible arribar a las siguientes conclusiones:

- Se identificó las principales ventajas y desventajas que tienen las plataformas conversacionales existentes, lo que provoca la necesidad de crear una propia que cumpla con las necesidades planteadas en el problema de investigación.
- Se opta por utilizar las herramientas y tecnologías empleadas en el desarrollo del marco de trabajo en el que se desarrolla la investigación. Las cuales tiene como lenguaje *Ruby* y *JavaScript, frameworks Rails* y *Highcharts. S*e implementará sobre el IDE *Rubymine* y *PostgreSQL* como SGBD.
- El estudio realizado a la metodología de desarrollo XP que permitirá guiar el proceso ingeniería de *software* de la propuesta de solución.

Capítulo 2: Propuesta de solución

En este capítulo se describe la propuesta de solución que permita resolver el problema planteado en la investigación. Se definen los requisitos funcionales y no funcionales del sistema. De las fases que propone la metodología XP se precisan las historias de usuarios, la planificación de entrega de las versiones del producto y el diseño del sistema.

2.1. Propuesta solución

Para darle solución a la problemática planteada se propone un subsistema para la creación de sistemas conversacionales que se integrará a la *suite* XEDRO-GESPRO como un nuevo módulo para esta herramienta. Este subsistema permitirá que los usuarios con determinado rol que interactúen con la *suite* XEDRO-GESPRO puedan crear *chatbots* personalizados. Los usuarios que cumplan con los permisos necesarios encargados de desarrollar estos sistemas conversacionales deberán ser expertos que conozcan sobre la gestión y organización de la información y comprendan el negocio sobre el cual se va a aplicar.

Estos *bots* conversacionales podrán tener la temática que el propio usuario determine. Por tanto, cada *bot* conversacional debe abarcar un tema específico, bien determinado, que lo distinga de los demás para proporcionar una mejor accesibilidad de la información a los usuarios. Además, contarán con una funcionalidad que permita administrar los usuarios que gestionarán la información que estos contengan.

Para explicar claramente la propuesta de solución se expondrán individualmente las tres áreas fundamentales que abarca el subsistema de acuerdo a las funcionalidades que se ofrecen a los usuarios.

Área de gestión de datos:

Esta área permite que los usuarios que cuenten con los permisos necesarios puedan gestionar, administrar y configurar los datos de los *bots* conversacionales y los componentes que los conforman. En esta área se lleva a cabo el proceso fundamental del subsistema porque es donde se entrenan los *chatbots,* se construyen las reglas y se escoge uno de los algoritmos de similitud por el usuario experto.

Para gestionar los datos que componen un *chatbot*, los usuarios deben tener presentes algunas características importantes y la jerarquía que existe entre sus componentes. Esto conllevará a una mejor comprensión cuando se construyan los componentes de la herramienta y mejorará el resultado final. A continuación, se describen las características de cada uno de los componentes y se mostrará en la Imagen 3 la jerarquía entre ellos:

- ***Chatbots*:** Deben estar bien diferenciados por el nombre. Además, para interactuar con ellos han de tener habilitada la propiedad “Público”, logrando que el *chatbot* esté disponible en la lista de *bots* conversacionales con los que se puede establecer una conversación. También el usuario seleccionará una función de similitud y definirá un rango de macheo para comparar el valor que devuelva la función de similitud escogida.
- **Habilidades:** Estarán bien diferenciadas por el nombre y serán de 3 tipos. Las generales, servirán para todos los *chatbots* del subsistema pero su composición variará según el usuario que la utilice. Las de negocio, abarcarán todo lo referente al tema del *chatbot;* y las habilidades por defecto se utilizarán para enviar mensajes a las preguntas que no tienen una respuesta defina en el *bot* conversacional.
- **Intenciones:** No podrán existir dos intenciones con el mismo nombre. Las intenciones que sean sobre un tema deberán ser agrupadas con las habilidades que tengan características en común para lograr una mayor organización.
- **Expresiones:** Estarán formadas por una o más palabras. Las expresiones pueden estar escritas correctas e incorrectamente, simulando los posibles mensajes enviados por los usuarios.
- **Entidades:** El usuario podrá crear tantas entidades como desee. Estarán conformadas por una o varias palabras separadas por coma. Además, son elementos puntuales que permiten diferenciar entre un conjunto de respuestas que contiene una misma intención.
- **Reglas:** Siempre contendrán uno o varios mensajes de respuestas. No se pueden crear reglas vacías y serán tan complejas o sencillas en dependencia de la necesidad del usuario.

Imagen 3: Jerarquía de los componentes de los sistemas conversacionales.

A los mensajes introducidos por los usuarios se les aplican funciones de similitud para textos cortos como distancia de *Jaro-Winkler* o la variación del coseno *Soft TF-IDF,* que estarán definidas en la etapa de construcción de los sistemas conversacionales. Estas funciones permitirán el procesamiento de las distintas preguntas con las expresiones introducidas previamente por la persona que construyó el *chatbot*. De este procesamiento se captura un valor de semejanza que se compara con el rango de macheo. Si este valor es mayor que el rango definido, se obtiene la intención asociada a la expresión que se identificó en la base de conocimiento y de esta intención su respectiva regla.

Si existen varias expresiones que contienen valores de similitud superiores al rango de macheo, se escoge la de mayor valor. Debido a que los usuarios puede introducir un rango de macheo muy alto, se aconseja que el sistema conversacional posea un entrenamiento elevado porque sus expresión tendrán que ser muy similares a las introducidas. En cambio, si el rango de macheo es muy pequeño puede que no existan una respuesta a la pregunta y entonces responda un mensaje que no se corresponda.

Área de chat:

En esta área se muestra una interfaz de cara a los usuarios que podrán interactuar con los *chatbots* disponibles*.* Esta área se compone de dos partes fundamentales: la primera, muestra un listado de los sistemas conversacionales disponibles a los usuarios y la segunda muestra una interfaz de mensajería después de seleccionar un *chatbot,* permitiendo iniciar una conversación con este a través del envio y recibo de mensajes de textos (Anexo 2).

Área de monitoreo:

En esta área se monitorea el funcionamiento del *bot* conversacional. También se comprueba: el comportamiento que está teniendo el *chatbot*, cómo reacciona a una cierta cantidad de usuarios conectados, cuáles son los componentes más usados en las conversaciones, cuántas conversaciones se tuvieron. Todos estos elementos se evidencian en los distintos gráficos que se generan en un periodo de tiempo que el usuario escoja (Ver Anexo 4). Además, registra todos los mensajes escritos por los usuarios, para entrenar mejor el *chatbot,* teniendo en cuenta los mensajes que no recibieron respuestas o mejorar algunas respuestas que no fueron de agrado (Ver Anexo 5).

En la Imagen 4 se muestra la arquitectura de la propuesta de solución. Esta arquitectura permitirá comprender la estructura que conforma el subsistema propuesto, la cual se descomponen de la siguiente manera:

- El usuario con la experiencia necesaria debe introducir las habilidades, intenciones, expresiones, reglas y entidades que conforman la base de conocimiento.
- El usuario introduce un mensaje de texto a través de la interfaz de usuario y este lo envía al motor de interferencia.
- El motor de inferencia es el encargado de analizar las reglas correspondientes a cada intención y obtener la respuesta acorde a la información guardada en la base de conocimiento, enviándola a la interfaz de usuario.

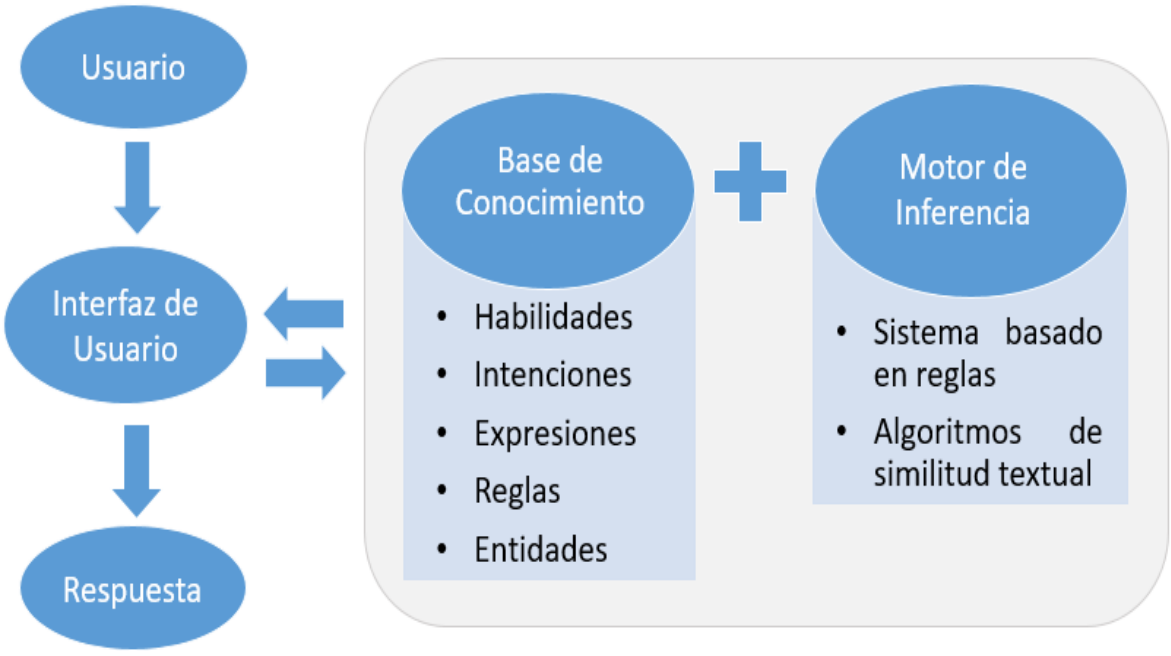


Imagen 4: Arquitectura de la propuesta de solución.

A continuación, en la Imagen 5 se modela como resultaría el proceso desde él envió de mensajes por el usuario hasta que adquiere una respuesta del subsistema. El usuario entra un mensaje que se compara con las expresiones creadas previamente en el subsistema. El motor de inferencia identifica las intenciones correspondientes a la expresión con la que se comparó el mensaje. Luego se aplican las reglas que, dada la intención y las entidades presentes en los términos de la expresión, generan una respuesta a los mensajes entrados por los usuarios.

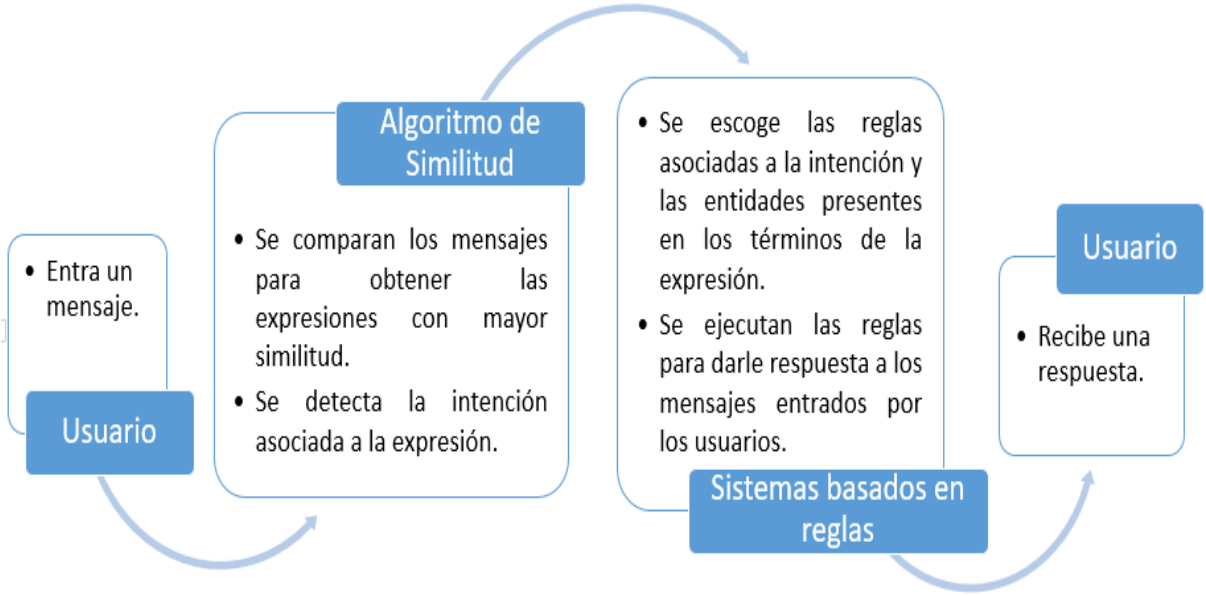


Imagen 5: Flujo del proceso de envió y recibo de mensaje por los usuarios.

2.2. Captura de requisitos

Para poder cumplir con el desarrollo de la propuesta de solución es necesario levantar formalmente los elementos que se van a considerar en el desarrollo del subsistema, esto se conoce como levantamiento de requisitos. Según la *IEEE* (1990), dentro del contexto de la Ingeniería de *Software*, define el concepto de requisito o requerimiento de la siguiente manera: "Un requisito de *software* es la capacidad que debe alcanzar o poseer un sistema o componente de un sistema para satisfacer un contrato, estándar, especificación u otro documento formal" (Moreno & Marciszack, 2013). Por tanto, la captura de requisitos permite definir las características o funcionalidades que debe seguir el subsistema a nivel interno y externo para resolver un problema concreto. Los requisitos deberán quedar plasmado en un documento oficial y escrito tras una entrevista entre el cliente y el equipo de proyecto. Los requisitos se clasifican en dos formas: requisitos funcionales y no funcionales.

2.2.1. Requisito funcionales

Los requisitos funcionales son las particularidades que debe cumplir el subsistema en situaciones determinadas para su buen funcionamiento. Estos requisitos pueden ser de forma general, que detallen las funciones del subsistema o hasta muy específicos como sus entradas y salidas de datos (Sommerville, 2011). A continuación, se muestran los requisitos que se definieron con el cliente:

Tabla 6: Lista de requisitos funcionales.

No.	Descripción
RF1.	Permitir a los usuarios crear, editar, listar y eliminar *bots* conversacionales.
RF2.	Permitir a los usuarios crear, editar, listar y eliminar habilidades.
RF3.	Permitir a los usuarios crear, editar, listar y eliminar intenciones.
RF4.	Permitir a los usuarios crear, listar y eliminar expresiones.
RF5.	Permitir a los usuarios crear, listar y eliminar entidades.
RF6.	Mostar detalles del *bot* conversacional.
RF7.	Asignar usuarios a un *bot* conversacional.
RF8.	Eliminar usuarios asignados a un *bot* conversacional.
RF9.	Mostrar un listado de todos los *bots* conversacionales creados por los usuarios del sistema.
RF10.	Visualizar una interfaz de mensajería para interactuar con los *bots* conversacionales.
RF11.	Permitir dar respuestas a los mensajes de los usuarios del *bot* conversacional.
RF12.	Permitir visualizar los mensajes enviados por los usuarios del *bot* conversacional.
RF13.	Mostrar una memoria de la conversación en curso.
RF14.	Elaborar reglas para dar la respuesta a los mensajes del usuario.
RF15.	Listar las reglas que permiten dar la respuesta a los mensajes del usuario.
RF16.	Permitir que el usuario ingrese una fecha de inicio para filtrar la información a monitorear.
RF17.	Permitir que el usuario ingrese una fecha de fin para para filtrar la información a monitorear.
RF18.	Permitir filtrar la información que se monitorea a partir de un rango de fecha especificado por el usuario.
RF19.	Mostrar un resumen de estadísticas de las habilidades, intenciones, conversaciones, mensajes y los usuarios.
RF20.	Mostar gráfica de línea con la información estadística de las conversaciones.
RF21.	Mostar gráfica de línea con la información estadística de los usuarios.
RF22.	Mostar gráfica de barra con la información estadística de las habilidades.
RF23.	Mostar gráfica de barra con la información estadística de las intenciones.
RF24.	Mostrar un registro de expresiones.
RF25.	Asignarle a una expresión a una entidad del registro de expresiones.
RF26.	Eliminar expresión del registro de expresiones.

2.2.2. Requisitos no funcionales

Los requisitos no funcionales se basan en las propiedades del subsistema como su usabilidad, almacenamiento y tiempo de respuesta. Además, pueden especificar restricciones al subsistema para su implementación, diseño, seguridad y eficiencia (Moreno & Marciszack, 2013). *Ian Sommerville* define que los requerimientos no funcionales suelen ser más críticos que los requerimientos funcionales, dado que la ausencia de uno de los mismos puede inutilizar al subsistema. A continuación, se precisan los siguientes requisitos no funcionales:

RNF 1. Requisitos de *software*

Servidor de Base de Datos y Aplicaciones

- **RNF 1.1.** El subsistema debe funcionar bajo la distribución Ubuntu 18.04 del sistema operativo GNU/Linux.

RNF 2. Requisitos de *hardware*

Servidor de Base de Datos

- **RNF 2.1.** La computadora que se emplee debe contar como mínimo con 8GB de RAM y de almacenamiento 40GB (recomendando 80GB).

Servidor de Aplicaciones

- **RNF 2.2.** La computadora que se emplee debe contar como mínimo con 4GB de RAM y de almacenamiento 20GB (recomendando 40GB).

RNF 3. Requisitos de seguridad

- **RNF 3.1.** El sistema deberá seguir las políticas de seguridad del departamento GESPRO.

RNF 4. Requisitos de apariencia o interfaz externa

- **RNF 4.1.** Se emplearán colores cálidos como azules, grises y blancos muy similares a los de GESPRO.

RNF 5. Requisitos de restricciones del diseño y la implementación

- **RNF 5.1.** El sistema deberá ser desarrollado en el lenguaje de programación *Ruby*.
- **RNF 5.2.** Se empleará como *framework Rails* y como gestor de base de datos *PostgreSQL*.

2.3. Fase I: Planificación

La fase de planeación comienza tras la captura de requisitos para que el equipo de trabajo pueda entender cómo funciona el negocio. En esta fase el cliente especifica las historias de usuarios (HU) que sean necesarias para el desarrollo del subsistema. A cada historia se le

describen sus características y se le otorga una prioridad. Para que luego, el equipo de trabajo evalúe cada historia y le asigne un costo medido en semanas de desarrollo. De ahí que se determine el plan de iteración y el plan de entrega de proyecto que se corregirá tantas veces se modifiquen las HU («57dc2692ee164.pdf», s. f.).

2.3.1. Historias de usuarios

La historia de usuario es la técnica que se emplea en la metodología XP para que se representen los requisitos de *software* que debe cumplir el subsistema. Las historias de usuarios son escritas para ser implementadas en un tiempo de 1 a 3 semanas como máximo. Cuando una historia de usuario esté compuesta por diferentes funcionalidades será dividida en varias historias (Beck & Fowler, 2001). Algunas no tienen un alto grado de complejidad para implementarlas. Entonces, se determinará que su período de desarrollo sea de 1, 2, o 4 días en un equivalente de 0.2, 0.4 y 0.8 en semanas respectivamente, para así calcular el tiempo estimado del proyecto. A continuación, se presentan algunas de las historias de usuarios.

Tabla 7: Historia de usuario 1.

Historia de usuario	
Número: 1	**Usuario:** Administrador
Nombre de HU: Crear *bot* conversacional.	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Medio
Tiempo estimado: 0.4	**Iteración Asignada:** 1
Programador responsable: Ariel Acevedo Guevara	
Descripción: El subsistema debe permitir crear *bots* conversacionales, validando que los datos introducidos por el usuario en los campos Nombre, Descripción, Público, Algoritmo de similitud y Rango de macheo sean correctos; y que los campos obligatorios Nombre y Descripción no queden en blanco.	
Observaciones: El subsistema mostrará los *bots* conversacionales creados en una tabla.	

Prototipo de Interfaz:

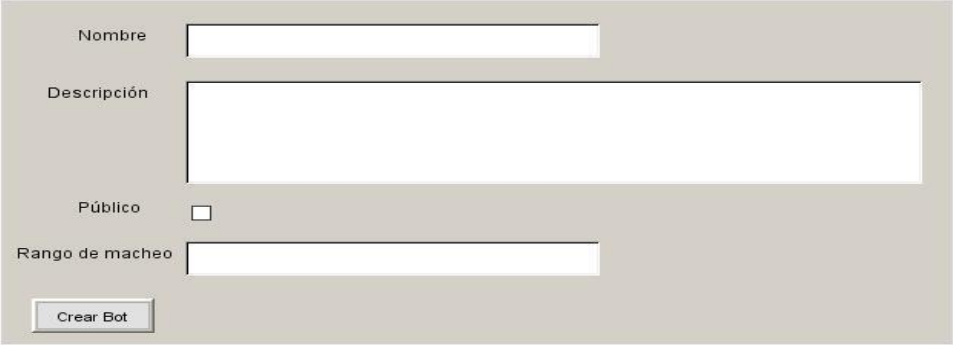


Tabla 8: Historia de usuario 26.

Historia de usuario	
Número: 26	**Usuario:** Administrador
Nombre de HU: Mostrar una memoria de la conversación en curso, que al terminar esta se actualice.	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Alta
Tiempo estimado: 2	**Iteración Asignada:** 2
Programador responsable: Ariel Acevedo Guevara	
Descripción: El subsistema permitirá que mientras el usuario se encuentre activo en él mismo, el *bot* conversacional guarde el historial de la conversación en curso.	
Observaciones: -.	
Prototipo de Interfaz:	

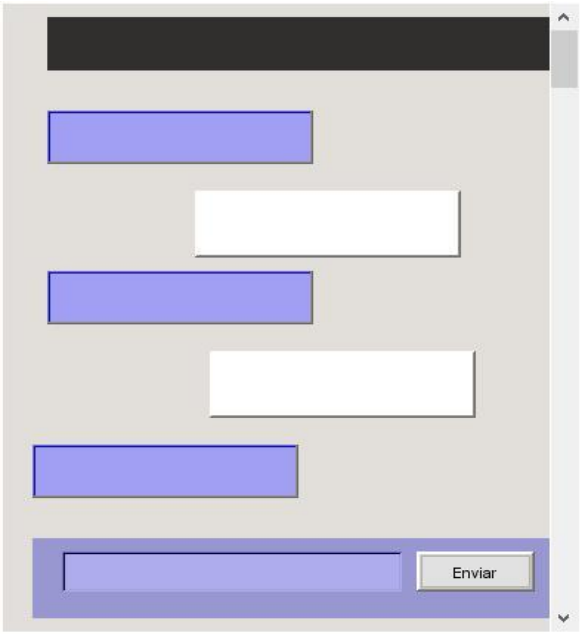


Tabla 9: Historia de usuario 29.

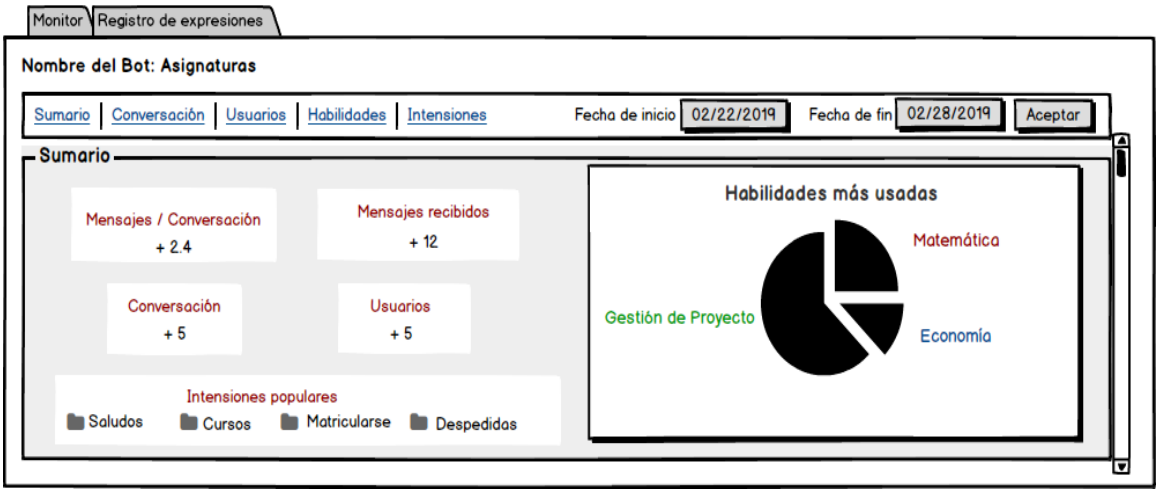
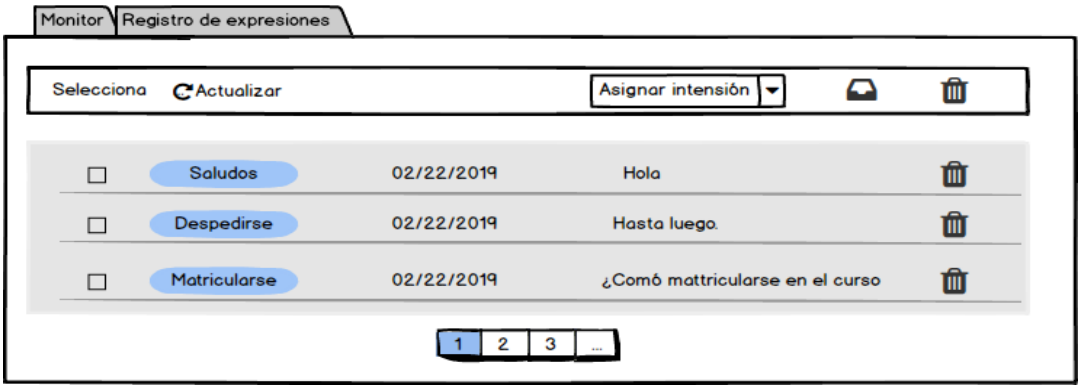
Historia de usuario	
Número: 29	**Usuario:** Administrador
Nombre de HU: Mostrar un resumen de estadísticas de las habilidades, conversaciones, intenciones, mensajes recibidos, promedio de mensajes por conversación y los usuarios.	
Prioridad en negocio: Media	**Riesgo en desarrollo:** Media
Tiempo estimado: 1	**Iteración Asignada:** 3
Programador responsable: Tania Socarras Navarro	
Descripción: El subsistema debe permitir mostrar un resumen estadístico de las conversaciones, habilidades e intenciones más usadas. Además, de los mensajes recibidos, promedio de mensajes por conversación y la cantidad de usuarios que se usan en el *bot* conversacional por día.	
Observaciones: Se deberá elegir un período de fechas del que se quiere la información; de no elegir uno se tomará por defecto el último período de fechas.	
Prototipo de Interfaz: 	

Tabla 10: Historia de usuario 34.

Historia de usuario	
Número: 34	**Usuario:** Administrador
Nombre de HU: Mostrar registro de expresiones.	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Alta
Tiempo estimado: 1.2	**Iteración Asignada:** 3
Programador responsable: Tania Socarras Navarro	
Descripción: Las expresiones deben aparecer listadas en una tabla. Esta aparecerá en la misma interfaz donde se registrarán los datos como: esta pública, la entidad a la que pertenece, la fecha en que fue creada y la expresión.	

Observaciones: En el subsistema debe aparecer la opción filtrar el registro de expresiones por los mensajes que coincidan o no con una intención.
Prototipo de Interfaz: 

2.3.2. Estimación de esfuerzo por historia de usuarios

La estimación del esfuerzo necesario para el desarrollo de la aplicación mediante las HU consiste en la suma del tiempo establecido en cada una de las iteraciones donde se distribuyeron esas HU. La siguiente tabla muestra las HU con los puntos de estimación en semanas.

Tabla 11: Estimación de esfuerzo por HU.

Historia de Usuario		Puntos estimados (semanas)
1	Crear *bots* conversacionales.	1
2	Editar *bots* conversacionales.	
3	Listar *bots* conversacionales.	
4	Eliminar *bots* conversacionales.	
5	Crear habilidades.	1
6	Editar habilidades.	
7	Listar habilidades.	
8	Eliminar habilidades.	
9	Crear intenciones.	1
10	Editar intenciones.	
11	Listar intenciones.	
12	Eliminar intenciones.	
13	Crear expresiones.	0.8
14	Listar expresiones.	
15	Eliminar expresiones.	
16	Crear entidades.	1

17	Editar entidades.	
18	Listar entidades.	
19	Eliminar entidades.	
20	Mostar detalles del *bot* conversacional.	0.2
21	Asignar usuarios a un *bot* conversacional.	0.4
22	Eliminar usuarios asignados a un *bot* conversacional.	0.2
23	Mostrar un listado de todos los *bots* conversacionales creados por los usuarios del sistema.	0.2
24	Visualizar una interfaz de mensajería para cada *bots* conversacionales que se cree.	1
25	Permitir comunicación, entre usuario y *bots* conversacionales, mediante la mensajería instantánea a través de una interfaz visual.	2
26	Mostrar una memoria de la conversación en curso, que al terminar esta se actualice.	2
27	Elaborar reglas para dar la respuesta a los usuarios.	1.2
28	Listar las reglas que dan respuesta a los usuarios.	0.2
29	Mostrar un resumen de estadísticas de las habilidades, conversaciones, intenciones, mensajes recibidos, promedio de mensajes por conversación y los usuarios conectados.	1
30	Calcular estadísticas de las conversaciones.	1
31	Calcular estadísticas de los usuarios.	0.8
32	Calcular estadísticas de las habilidades.	0.8
33	Calcular estadísticas de las intenciones.	1
34	Mostrar un registro de expresiones.	1.2

Al realizar la estimación, el tiempo de desarrollo obtenido fue de 18 semanas siendo equivalente a 4 meses y 2 semanas.

2.3.3. Desarrollo del plan de iteraciones

Luego de estimar el tiempo de desarrollo de las historias de usuarios se realizará el plan de iteraciones, mediante este se seleccionan las HU que se desarrollarán en cada una de las etapas de entregas pendientes. El cálculo del tiempo se realizó teniendo en cuenta que una semana de trabajo consta de 5 días donde se trabaja 6 horas diarias y no se cuentan los

días feriados. La siguiente tabla muestra el plan de iteraciones con el tiempo empleado en cada una de ellas y las HU que las componen.

Tabla 12: Plan de duración de las iteraciones.

Iteraciones	Historia de Usuario		Puntos estimados
Iter. No. 1	1	Crear *bots* conversacionales.	6
	2	Editar *bots* conversacionales.	
	3	Listar *bots* conversacionales.	
	4	Eliminar *bots* conversacionales.	
	5	Crear habilidades.	
	6	Editar habilidades.	
	7	Listar habilidades.	
	8	Eliminar habilidades.	
	9	Crear intenciones.	
	10	Editar intenciones.	
	11	Listar intenciones.	
	12	Eliminar intenciones.	
	13	Crear expresiones.	
	14	Listar expresiones.	
	15	Eliminar expresiones.	
	16	Crear entidades.	
	17	Editar entidades.	
	18	Listar entidades.	
	19	Eliminar entidades.	
	20	Mostar detalles del *bot* conversacional.	
	21	Asignar usuarios a un *bot* conversacional.	
	22	Eliminar usuarios asignados a un *bot* conversacional.	
	23	Mostrar un listado de todos los *bots* conversacionales creados por los usuarios del sistema.	
Iter. No. 2	24	Visualizar una interfaz de mensajería para cada *bots* conversacionales que se cree.	6
	25	Permitir comunicación, entre usuario y *bots* conversacionales, mediante la mensajería instantánea a través de una interfaz visual.	
	26	Mostrar una memoria de la conversación en curso, que al terminar se actualice.	

	27	Elaborar reglas para dar la respuesta a los usuarios.	
	28	Listar reglas para dar la respuesta a los usuarios.	
Iter. No. 3	29	Mostrar un resumen de estadísticas de las habilidades, conversaciones, intenciones, mensajes recibidos, promedio de mensajes por conversación y los usuarios.	6
	30	Calcular estadísticas de las conversaciones.	
	31	Calcular estadísticas de los usuarios.	
	32	Calcular estadísticas de las habilidades.	
	33	Calcular estadísticas de las intenciones.	
	34	Mostrar un registro de expresiones.	
Total			18 semanas

2.3.3. Plan de entregas

El plan de entrega se realiza con el propósito de definir las fechas en que se deberán entregar al cliente cada una de las iteraciones del subsistema, donde cada iteración comienza donde termina la otra como se muestra a continuación.

Tabla 13: Plan de entrega.

Iteración	Fecha de inicio	Fecha de fin
1ra Iteración	17 de diciembre de 2018	30 de enero de 2019
2da Iteración	31 de enero de 2019	15 de marzo de 2019
3ra Iteración	18 de marzo de 2019	3 de mayo de 2019

2.4. Fase II: Diseño del sistema

En el marco de trabajo del proyecto GESPRO se utiliza el lenguaje de programación *Ruby* con el apoyo del *framework Rails*. Este *framework* sigue el paradigma de la arquitectura Modelo-Vista-Controlador (MVC) (Aroca, 2018). El patrón MVC según *Steve Burbeck* (2015) es un patrón de arquitectura de *software* que separa los datos y la lógica de una interfaz de usuario y el módulo encargado de gestionar los eventos y las comunicaciones (Coba, 2016). Este patrón consta de tres componentes como lo específica su nombre:

- ✓ **Modelo:** Es la clase que representa las tablas de la base de datos para manejar los datos y controlar todas sus transformaciones. Además, en el modelo se especifican las relaciones entre clases. Y las migraciones que expresan cambios en la base de datos. Un ejemplo de modelo es la clase *GesprochatBot* que permite gestionar la información de la tabla de BD *gesprochat_bots*.

- ✓ **Controlador:** Es el objeto que proporciona acciones de respuestas cuando ocurra una interacción del usuario con la Vista y responde manipulando los datos representados del Modelo. Un ejemplo de controlador es la clase *BotController* que permite gestionar todos los datos correspondientes a los *bots* conversacionales del subsistema, ya sea crearlo, modificarlo, listarlo o eliminarlo.
- ✓ **Vista:** Es el objeto que maneja la presentación visual de los datos representados por el Modelo. Y creando una interfaz visual se muestran los datos a los usuarios. Un ejemplo de vista es la clase *index* del controlador *bots*, esta muestra un listado con los *bots* conversacionales sobre los cuales tiene permiso el usuario en cuestión y que permite gestionarlos.

La propuesta de solución planteada anteriormente cuenta en su totalidad con 10 vistas, 8 controladores y 9 modelos.

2.4.1. Patrones de diseño

Los patrones de diseño según *Craig Larman*, son los que nos permiten describir fragmentos de diseño y reutilizar ideas de diseño, ayudando a beneficiarse de la experiencia de otros. Los patrones dan nombre y reutilizan código, reglas y buenas prácticas de técnicas orientadas a objetos (Larman, 2000). Además, se encargan de describir las relaciones de las clases y los objetos para dar solución a un problema.

Los patrones Generales de Asignación de Responsabilidades (*GRASP*[5]) son una ayuda en el aprendizaje que permiten al desarrollador entender lo esencial del diseño de objetos y a aplicar el razonamiento de una forma metódica, racional y explicable. Este enfoque se entiende en el uso de los principios del diseño basado en patrones para la asignación de responsabilidades a las clases y objetos de una aplicación (Noorullah, 2012). Los patrones GRASP se desglosan en 5 patrones: experto, creador, controlador, alta cohesión y bajo acoplamiento. A continuación, se mostrarán como se implementaron en el sistema:

- ❖ **Experto:** Son aquellas clases que cuentan con la información necesaria para cumplir con la responsabilidad que se les asigne. Por ejemplo, la clase *GesprochatBot* es la que contiene toda la información sobre los *bots* conversacionales creados por los usuarios porque es la que incorpora todos los atributos para la creación de un objeto de tipo *bot* conversacional.

[5] Proviene de sus siglas en inglés, *General Responsibility Assignment Software Patterns.*

- **Creador:** es aquella clase a la cual se le asigna la responsabilidad de crear una instancia de otra clase en algunos de los siguientes casos (Larman, 2000):
  - Clase B agrega los objetos de A.
  - Clase B contiene los objetos de A.
  - Clase B registra las instancias de los objetos de A.
  - Clase B utiliza específicamente los objetos de A.
  - Clase B tiene los datos de inicialización que serán trasmitidos a A cuando este objeto sea creado.

Un ejemplo de esto es la clase *IntentsController* la cual además de crear instancias de las intenciones, crea y contiene los objetos e instancias de las expresiones.

- **Controlador:** son aquellas clases destinadas para manejar los eventos del sistema. Ejemplo de esto es la clase *BotController* la cual se encarga de mostrar las listas de todos los *bots* conversacionales públicos en el sistema.
- **Bajo Acoplamiento:** Este patrón garantiza que cada una pueda realizar sus funciones por separado, relacionándose con las otras clases cuando sea necesario; y que, si en el futuro si se deseara realizar alguna modificación al sistema, se minimizaran las afectaciones a otros componentes. Esto se evidencia en la clase *EntityController* porque no tiene dependencias con otras clases del subsistema.
- **Alta Cohesión:** Este está muy vinculado al anterior se encarga de que las clases solo realicen funciones estrechamente relacionadas con su contenido, lo que facilita la organización del proyecto y que estas no estén sobrecargadas de funcionalidades ajenas. Ejemplo de esto es la clase *EntityController* porque se encarga solamente de gestionar la información de las entidades.

Los patrones *GoF*[6] se clasifican por sus características en 3 tipos de patrones: estructurales, creacionales y de comportamiento. Los patrones de creación conciernen al proceso de creación de objetos. Los patrones estructurales tratan con la composición de las clases o los objetos. Los patrones de comportamiento caracterizan la vía en que las clases u objetos interactúan y distribuyen las responsabilidades (Gamma, Helm, Johnson, & Vlissides, 1997). A continuación, se mostrarán ejemplo de los patrones *GOF* que se utilizaron en la propuesta de solución:

[6] Proviene de sus siglas en inglés, *Gang of Four.*

Patrones de Creación

- **Patrón de Registro Activo *(ActiveRecord):*** Es un método para acceder a los datos en una base de datos relacionales. Una tabla de base de datos se envuelve en una clase (GitHub, 2015/2019). Ejemplo de esta clase modelo son *GesprochatBot* que inserta las filas de la tabla en la base de datos.
- **Principio de Responsabilidad Única (*Single Responsibility Principle)*:** Establece que cada clase debe tener responsabilidad sobre una parte única de la funcionalidad provista por el *software*, y que la responsabilidad debe estar completamente encapsulada por la clase. Todos sus servicios deben estar estrechamente alineados con esa responsabilidad (GitHub, 2015/2019). Esto se evidencia con la clase *BotController* que es la encargada de todas las funcionalidades relacionadas con la gestión de los datos de crear *bots* conversacionales.

Patrones Estructurales

- **Fachada *(Facade):*** Es un objeto que proporciona una interfaz simplificada a un cuerpo de código más grande, como una biblioteca de clases (GitHub, 2015/2019). Como se puede evidenciar con el uso de la biblioteca de *JQuery* que se usó para controlar el comportamiento de las vistas.

2.4.2. Diagrama de paquetes

El diagrama de paquetes representa de forma estática los componentes del sistema de información que está siendo modelado. Es utilizado para definir los distintos paquetes a nivel lógico que forman parte de la aplicación y la dependencia entre ellos (DiagramasUML, 2019). El diagrama de paquete del sistema permitirá una mejor comprensión de la estructura de la solución propuesta como se puede apreciar en el Anexo 36.

2.4.3. Modelo de datos

Un modelo de datos es un sistema formal y abstracto que permite describir los datos de acuerdo con reglas y convenios predefinidos o se puede decir que es un conjunto de conceptos que permiten describir, a distintos niveles de abstracción, la estructura de una base de dato (Tom, 2010). El modelo de datos de la propuesta de solución se puede apreciar en el Anexo 37.

2.4.4. Tarjetas CRC

Las tarjetas Clase-Responsabilidad-Colaboradores (CRC) se dividen en tres partes que contienen la información del nombre de la clase, sus responsabilidades y sus colaboradores (Ver Tabla 14). Una clase es cualquier evento o interfaz visual. Las responsabilidades son las acciones o métodos asociados a la clase. Y los colaboradores son las clases con las que se relaciona o trabaja esta para poder lograr sus responsabilidades.

Tabla 14: Prototipo de tarjeta CRC.

Clase	
Responsabilidad	**Colaboración**

Tabla 15: Tarjeta CRC 1.

Tarjeta CRC	
Clase: *GesprochatBot*.	
Responsabilidad	**Colaboración**
Representar los datos persistentes en la base de datos correspondiste a los *bots* conversacionales del sistema. Siendo una clase Modelo o conocida también como Entidad. • defaul_range_match (): es la encargada de definir el rango de macheo del *bot* conversacional. Su valor por defecto es de 0,7.	✓ *BotController.* ✓ *ChatbotController.* ✓ *IntentsController.* ✓ *MonitorController.* ✓ *SkillController.* ✓ *RulesController.*

Tabla 16: Tarjeta CRC 8.

Tarjeta CRC	
Clase: *GesprochatConversationLog*.	
Responsabilidad	**Colaboración**
Representar los datos persistentes en la base de datos, correspondiste a las conversaciones que se tienen entre el usuario y *bots* conversacionales. Siendo una clase Modelo o conocida también como Entidad.	✓ *MonitorController*.

Tabla 17: Tarjeta CRC 10.

Tarjeta CRC	
Clase: *BotController*	
Responsabilidad	**Colaboración**

Es la clase encargada de gestionar los datos correspondientes a los *bots* conversacionales del sistema. • *create ():* se crea un nuevo *bot* conversacional. • *update ():* se modifica un *bot* conversacional. • *destroy ():* se elimina un *bot* conversacional. • *add_owner ():* se asigna un usuario de la base de datos a un *bot* conversacional. • *delete_user_admin():* se elimina un usuario asignado a un *bot* conversacional.	✓ *GesprochatBot.* ✓ *User.* ✓ *Gesprochat_admin_of_bot.*

Tabla 18: Tarjeta CRC 15.

Tarjeta CRC	
Clase: *MonitorController*.	
Responsabilidad	**Colaboración**
Es la clase del subsistema que se encarga de: • *index ():* Genera gráficos estadísticos sobre la información del bot correspondiente y respuestas al usuario. Provee la funcionalidad de escoger un intervalo de fecha para mostrar sus datos estadísticos. • *registry ():* Administrar el almacenamiento de expresiones previamente creadas en cada intención.	✓ *GesprochatConversationLog.* ✓ *GesprochatMessages.* ✓ *GesprochaEntities.* ✓ *GesprochatIntents.* ✓ *GesprochatSkill.* ✓ *GesprochatExpressions.*

2.5. Conclusiones del capítulo

En este capítulo se abordaron los aspectos referentes a la planificación de la propuesta de solución y sus principales características funcionales. Esto permitió llegar a las siguientes conclusiones:

- A partir de las definiciones de las HU se pudieron identificar las principales funcionalidades a desarrollar en correspondencia a las áreas de la propuesta de solución.
- La estimación del tiempo para la implantación de las HU definidas, permitió calcular una entrega final de la propuesta en 4 meses y 2 semanas.
- Se identificaron los patrones de diseño *GRASP* para el desarrollo de la solución y los *GOF* que propone el *framework* de desarrollo *Rails*.

Capítulo 3: Implementación y pruebas

En este capítulo se abordarán todos los elementos que propone la metodología XP en sus dos últimas fases: desarrollo y prueba. De la fase de desarrollo se exponen las tareas de ingenierías generadas de cada HU en cada una de sus iteraciones. Y de la fase de prueba se describen las diferentes pruebas que se realizaron al subsistema como son las unitarias, funcionales, rendimiento y operacionales. De los resultados de cada una de estas pruebas se podrá definir si el producto cumple con los requisitos establecidos por el cliente validándolo haciendo uso de la técnica de *Iadov*.

3.1. Fase III: Desarrollo

La fase de desarrollo es una de las más importantes de la metodología XP, porque se realiza en cada una de las iteraciones por las que pasa el proyecto. Estas iteraciones se definen en el plan de iteración y son también utilizadas para medir el progreso del proyecto. Una iteración terminada sin errores es una medida clara de avance (Joskowicz, 2008).

La disponibilidad del cliente es una de las buenas prácticas que se tiene en cuenta en esta fase; ya que el cliente forma parte del equipo de desarrollo (Joskowicz, 2008). Otros elementos que propone la metodología como buenas prácticas son: la programación por parejas, integraciones permanentes, ritmo sostenido y programación dirigidas por las pruebas (*Test-driven programming*) (Joskowicz, 2008).

3.1.1. Tareas de ingeniería

Los usuarios proponen HU que no contienen los detalles necesarios para realizar el desarrollo del código, por lo que es necesario descomponer las HU en tareas de ingeniería. Las tareas de ingeniería o programación permiten organizar la implementación de cada una de las HU. Teniendo en cuenta el plan de iteración se desarrollaron tres iteraciones, con sus respectivas tareas de ingeniería de las que solo se muestran algunas a continuación y las otras se pueden apreciar en los anexos.

Tareas de ingeniería para la iteración I

En la primera iteración se realizaron un total de 28 tareas de ingeniería relacionadas a la parte de gestión de datos. Las más importantes son:

Tabla 19: Tarea de ingeniería 1 de la iteración I.

Tarea de ingeniería	
Número de tarea: 1	**HU:** 1

Nombre de la tarea: Desarrollar vista para crear *bots* conversacionales.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 0.4
Fecha de inicio: 17 de diciembre de 2018	**Fecha de fin:** 18 de diciembre de 2018
Programador responsable: Ariel Acevedo Guevara	
Descripción: Se desarrolla una vista que permita crear *bots* conversacionales definiendo de cada uno los siguientes campos Nombre, Descripción, Público y Rango de macheo.	

Tabla 20: Tarea de ingeniería 2 de la iteración I.

Tarea de ingeniería	
Número de tarea: 2	**HU:** 2
Nombre de la tarea: Desarrollar vista para editar *bots* conversacionales.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 0.2
Fecha de inicio: 19 de diciembre de 2018	**Fecha de fin:** 19 de diciembre de 2018
Programador responsable: Ariel Acevedo Guevara	
Descripción: Se desarrolla una vista que permita editar *bots* conversacionales definiendo de cada uno los siguientes campos Nombre, Descripción, Público, Algoritmo de similitud y Rango de macheo.	

Tareas de ingeniería para la iteración II

En la segunda iteración se realizaron un total de 5 tareas de ingeniería relacionados a la parte del *chat*. Destacándose las siguientes tareas:

Tabla 21: Tarea de ingeniería 2 de la iteración II.

Tarea de ingeniería	
Numero de tarea: 2	**HU:** 25
Nombre de la tarea: Desarrollar una interfaz que permita la comunicación entre usuario y *bots* conversacionales, mediante la mensajería instantánea.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 2
Fecha de inicio: 7 de febrero de 2019	**Fecha de fin:** 20 de febrero de 2019
Programador responsable: Ariel Acevedo Guevara	
Descripción: Se desarrolla una interfaz que permitirá la comunicación entre usuario y *bots* conversacionales mediante la mensajería instantánea.	

Tabla 22: Tarea de ingeniería 3 de la iteración II.

Tarea de ingeniería	
Numero de tarea: 3	**HU:** 26

Nombre de la tarea: Implementar la funcionalidad que permita dar seguimiento a la conversación en curso entre el *bot* conversacional y el usuario.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 2
Fecha de inicio: 21 de febrero de 2019	**Fecha de fin:** 6 de marzo de 2019
Programador responsable: Ariel Acevedo Guevara	
Descripción: Se implementará una funcionalidad que mostrará en la interfaz de mensajería, la conversación en curso entre el *bot* conversacional y el usuario.	

Tareas de ingeniería para la iteración III

En la tercera iteración se realizaron un total de 9 tareas de ingeniería relacionadas a la parte del monitor. A continuación, se mostrarán las más relevantes:

Tabla 23: Tarea de ingeniería 1 de la iteración III.

Tarea de ingeniería	
Número de tarea: 1	**HU:** 29
Nombre de la tarea: Desarrollar vista para generar la gráfica de pastel sobre el uso de las Habilidades.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 0.6
Fecha de inicio: 18 de marzo de 2019	**Fecha de fin:** 20 de marzo de 2019
Programador responsable: Tania Socarras Navarro	
Descripción: Se desarrolla una vista para generar un gráfico de pastel de las habilidades más usadas.	

Tabla 24: Tarea de ingeniería 2 de la iteración III.

Tarea de ingeniería	
Número de tarea: 2	**HU:** 29
Nombre de la tarea: Desarrollar vista para mostrar datos estadísticos del *bot* conversacional.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 0.4
Fecha de inicio: 21 de marzo de 2019	**Fecha de fin:** 22 de marzo de 2019
Programador responsable: Tania Socarras Navarro	
Descripción: Se desarrolla una vista para mostrar un resumen estadístico de las intenciones más usadas, los mensajes recibidos, mensajes por conversación y los usuarios que se conectaron a un *bot* conversacional.	

Tabla 25: Tarea de ingeniería 7 de la iteración III.

Tarea de ingeniería	
Número de tarea: 7	**HU:** 34

Nombre de la tarea: Desarrollar vista para generar un registro de expresiones.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 0.4
Fecha de inicio: 26 de abril de 2019	**Fecha de fin:** 27 de abril de 2019
Programador responsable: Tania Socarras Navarro	
Descripción: Se desarrolla una vista para mostrar un registro de expresiones donde se puedan administrar las mismas.	

3.1.2. Estilos de codificación

Es el conjunto de reglas o normas usadas para escribir código y que incluye una gran gama de aspectos dentro del proceso de codificación. Un buen estilo de programación debe aportar a la eficiencia del proceso de desarrollo, logrando que los programas sean más robustos y comprensible (Perera, 2017). Algunos de los estilos de codificación que fueron definidos por el equipo de desarrollo para lograr una estandarización en la programación del *software* son:

- Los nombres de los métodos y clases estén escritos en inglés.
- Las líneas de código no excedan de 100 caracteres.
- El tamaño de los métodos no excedan las 20 líneas de código, de ser así se dividirá en dos métodos.

En la Imagen 6 se muestra algunos de los estilos de codificación definidos por el equipo de desarrollo empleados en la implementación del subsistema.

```
def create
  @bot = GesprochatBot.new(bot_params)
  @bot.created_by = User.current.id
  if @bot.save
    owner = GesprochatAdminOfBot.new(user_id: User.current.id, bot_id: @bot.id, admin_of_bot: true)
    if owner.save
      redirect_to gesprochat_bots_path, notice: l(:messages_bot_created)
    end
  else
    render :new
  end
end
```

Imagen 6: Ejemplo de uso de estándares de codificación.

3.1.3. Convenciones de codificación

Las convenciones de codificación son un conjunto de normas para un lenguaje de programación específico que recomiendan estilos de programación, buenas prácticas y métodos para mantener el aspecto del código fuente. Estas convenciones incluyen la organización de archivos, la indentación, los comentarios, las declaraciones los espacio en blanco, las llaves de apertura y cerrado (Vega, 2017). Para el desarrollo del subsistema se emplearon los estándares de codificación que propone el lenguaje de programación *Ruby* con el propósito de normalizar su desarrollo. Cada lenguaje de programación tiene su propia convención de codificación para nombrar las variables, métodos, clases y otros elementos del código. Ejemplo de estas convenciones de codificación empleadas son (Marquez, 2016):

- ***CamelCase:*** Se utilizó para escribir los nombres de las clases, siguiendo una estructura de frases compuestas donde la primera letra de cada palabra se escribirá en mayúscula y todas las palabras estarán juntas.
- ***snake_case:*** Se empleó para nombrar métodos, siguiendo una estructura de frases compuestas donde cada palabra se escribe en minúscula y es separado por barras bajas.
- ***SCREAMING_SNAKE_CASE:*** Se utilizó para las constantes y define que todos los elementos se escriben con letra mayúscula separadas por un carácter de subrayado (_) y sin espacios.
- Emplear el signo de # para comentarios simples.

A continuación, se muestran algunos ejemplos de estas convenciones de configuración utilizados para la implementación del subsistema.

```
def delete_user_admin
  user_admin = GesprochatAdminOfBot.where(bot_id: params[:bot_id], user_id: params[:user_id])
  GesprochatAdminOfBot.delete user_admin
  redirect_to action: 'index'
end
```

Imagen 7: Ejemplos de uso de convenciones de codificación CamelCase y snake_case.

```
ALGORITHM_COSINE_DISTANCE = 1
ALGORITHM_JARO_DIATANCE = 2
```

Imagen 8: Convenciones de codificación SCREAMING_SNAKE_CASE.

```
# Add user of the bots
```

Imagen 9: Ejemplos de uso de convenciones de codificación de comentarios.

3.1.4. Estructura de la aplicación

La aplicación posee la misma estructura de los otros módulos que contiene la *suite* XEDRO-GESPRO. Esta se conforma por un conjunto de carpetas en las que está organizado todo el código, siguiendo el paradigma de la arquitectura Modelo-Vista-Controlador. En la Imagen 10 muestra cómo está estructurado el proyecto.

Imagen 10: Representación de los directorios que forman parte de la aplicación.

A continuación, se ofrece una descripción del contenido de cada directorio:

1. *controllers*: Este directorio contiene todos los controladores, estos son los componentes encargados de recibir las peticiones de los usuarios. Dentro del directorio *config/routes.rb* se establece qué controlador recibe qué petición.
2. *helpers*: Se utilizan para disminuir la complejidad de un objeto en concreto. Por ejemplo una vista o un controlador, haciendo más sencilla la lectura y el entendimiento del código.
3. *models*: Contiene todos los modelos, estos son las clases que representan datos que se almacenan dentro de la base de datos.
4. *views*: Está carpeta se conforma de todas las vistas. Las vistas son los archivos donde se define la mayor parte de la representación visual permitiendo mostrar la información de manera dinámica.

5. *assets*: En este directorio se situarán aquí todos los recursos necesarios que permiten darle estilo y dinamismo a la aplicación.

3.2. Fase IV: Pruebas

Las pruebas del *software* son un elemento crítico para la garantía de calidad y representa una revisión final de las especificaciones, del diseño y la implementación (Pressman, Roger S., 2002). Las pruebas intentan demostrar que un programa hace lo que se especificó al inicio del desarrollo de la investigación, así como descubrir defectos en el programa antes de usarlo (Sommerville, 2011).

Una estrategia para las pruebas es descomponiéndolas en niveles como plantea el modelo V (Anexo 85). Para lograr una correcta verificación y validación del sistema se realizaron las pruebas unitarias, funcionales y rendimiento; correspondientes a los niveles de módulos y sistema.

3.2.1. Pruebas unitarias

Las pruebas unitarias son el proceso de probar componentes del programa, como métodos o clases de objetos. Comprueban el correcto funcionamiento de cada componente de código por separado (Sommerville, 2011). Estas pruebas son preferentemente automatizadas, pues esto facilita al programador identificar funciones que no ofrecen una salida acorde con la lógica que se deseaba implementar. A continuación se muestran dos de las funcionalidades a las cuales se les realizaron pruebas unitarias (Imagen 11).

```
def test_only_one_bot_for_name
  attributes = {
      name: 'My bot',
      description: 'My bot description',
      master: true,
      is_public: true,
      created_by: User.find(1),
      range_match: 1.2,
      algorithm: 1
  }
  bot = GesprochatBot.new(attributes)
  bot.save
  new_bot = GesprochatBot.new(attributes)
  assert !new_bot.save
end

def test_default_range_match
  assert_equal( exp 0.7, GesprochatBot.new.range_match)
end
```

Imagen 11: Casos de pruebas unitarias del método Gesprochat_Bot_Test.

Se realizaron un total de 7 pruebas unitarias, de ellas 2 arrojaron no conformidades como se muestra en la Imagen 12 y se corrigieron antes de terminar la iteración. Estas pruebas unitarias fueron basadas en los algoritmos de similitud de texto y en requerimientos que son obligatorios para el funcionamiento de los *chatbots* creados por los usuarios.

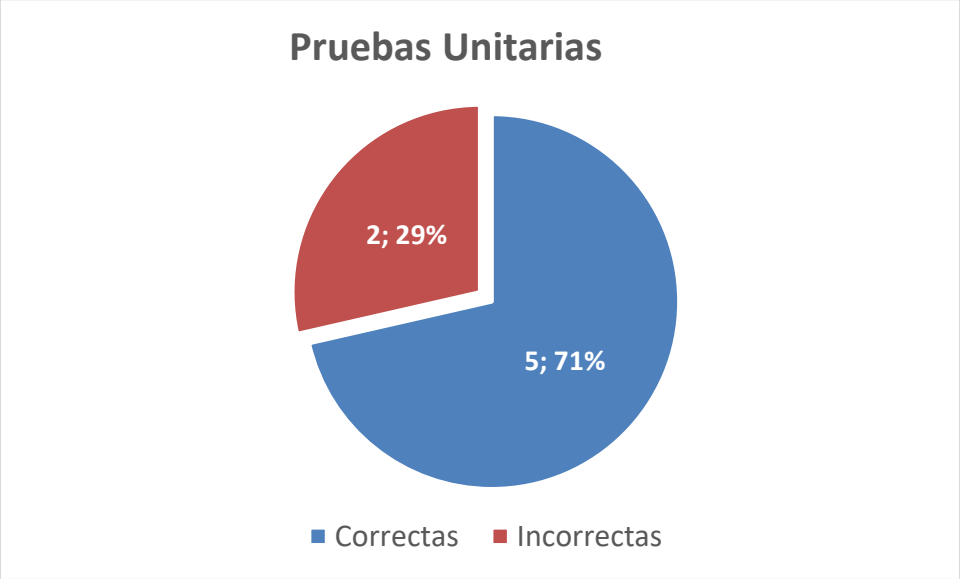


Imagen 12: Resultados de las pruebas unitarias.

3.2.2. Pruebas funcionales

Las pruebas funcionales son consideradas pruebas de caja negra debido a que están dirigidas a comportamiento externo del subsistema. Estas pruebas tienen como objetivo asegurar que el subsistema realiza correctamente todas las funciones que se han detallado en las especificaciones dadas por el cliente. Además, se pueden realizar de forma automatizada para comprobar que nada de lo probado con anterioridad ha dejado de funcionar.

Se realizaron un total de 40 pruebas funcionales entre las tres iteraciones del sistema, obteniendo: 5 no conformidades en la primera iteración, 2 no conformidades en la segunda iteración y en la tercera iteración ninguna no conformidad. Todas las no conformidades se resolvieron antes de ejecutar la siguiente iteración para poder integrarlas. A continuación, se mostrará de forma más detallada los resultados obtenidos para cada una de las iteraciones.

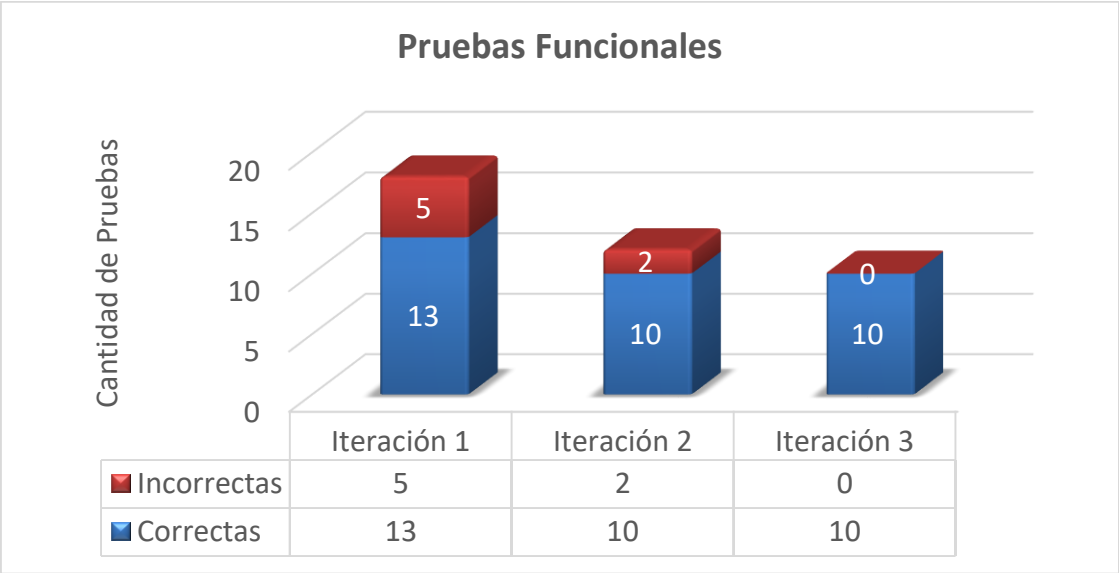


Imagen 13: Resultados de las pruebas funcionales.

3.2.3. Pruebas de rendimiento

Las pruebas de rendimiento deben diseñarse para garantizar que el subsistema procese su carga pretendida. Generalmente, esto implica efectuar una serie de pruebas donde se aumenta la carga, hasta que el rendimiento del sistema se vuelve inaceptable (Sommerville, 2011). Para automatizar las pruebas de rendimiento se empleó la aplicación *Apache JMeter*.

En el sistema implementado se realizan un gran número de transacciones de mensajes por usuarios al cual es necesario ver su comportamiento en casos de carga extrema. Para probar esta funcionalidad se empleó una base de datos de prueba compuesta por 3 *bots* y 500 expresiones. A esta BD se le realizaron distintas peticiones y se registraron los tiempos referidos a la demora en que el subsistema le daba respuesta a los mensajes de los usuarios. El *Bot* 1 cuenta con 50 expresiones, el *Bot* 2 cuenta con 150 expresiones y el *Bot* 3 cuenta con 300 expresiones. Esta prueba consiste simular un número de usuarios realizando peticiones de forma concurrente sobre *bots* conversacionales con esas características. El siguiente gráfico se muestra cómo se comportó la funcionalidad para tres grupos de usuarios de diferentes tamaños.

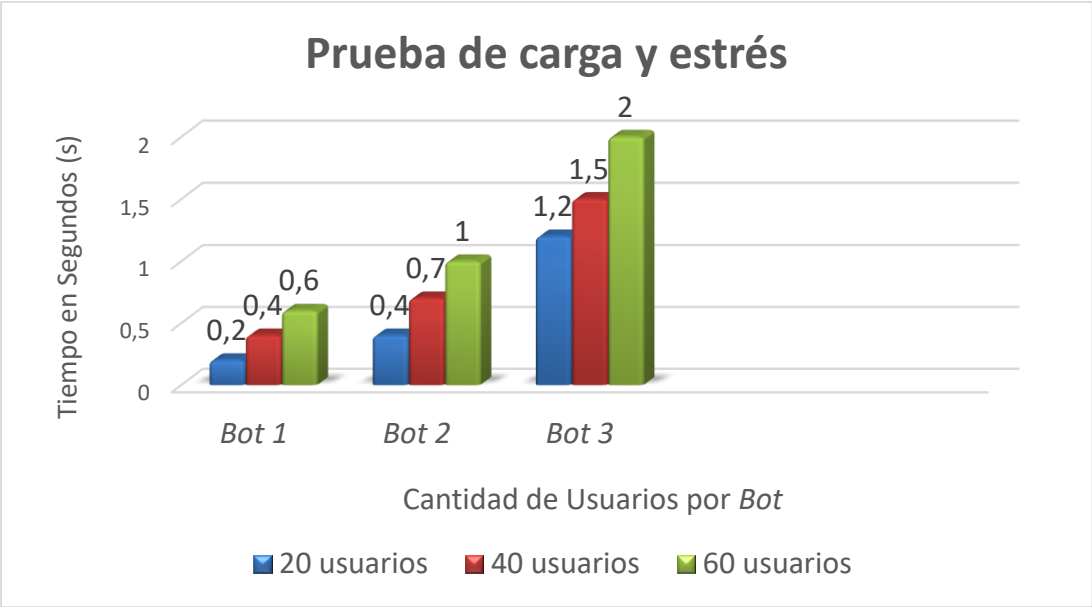


Imagen 14: Resultados de la prueba de carga y estrés.

La gráfica anterior muestra como en los tres *bots* conversacionales hubo un aumento en el tiempo de respuesta en dependencia de la cantidad de peticiones que se recibían de los usuarios conectados. Aunque para los *bots* (1) y (2) los tiempos fueron por debajo de un segundo. Esto se debe a las técnicas de similitud textual que se emplearon para la comparación de los mensajes y a la cantidad de expresiones por las que están compuestos los *bots* conversacionales. A medida que a los *bots* conversacionales se les incrementen la cantidad de expresiones el proceso será más lento porque tendrá más respuestas con las que comparar cada pregunta que el usuario le envíe.

3.3. Validación

El subsistema desarrollado provee una nueva forma de comunicación a los usuarios que interactúan con la *suite* XEDRO-GESPRO. Esta plataforma será un nuevo módulo de la *suite* que permite la creación de sistemas conversacionales. El subsistema tiene como objetivo permitir que los usuarios puedan acceder a la información que los sistemas conversacionales contienen, siempre que la aplicación esté disponible. Además, facilitará la búsqueda de informaciones específicas.

Para validar dicho subsistema, se utilizó la técnica de *Iadov* que constituye una vía indirecta para el estudio de la satisfacción, ya que los criterios que se utilizan se fundamentan en las relaciones que se establecen entre tres preguntas cerradas y dos abiertas que se intercalan dentro de un cuestionario (Orestes Febles, 2014). El cuestionario empleado para determinar

el grado de satisfacción de los usuarios con la propuesta del subsistema cuenta con un total de cinco preguntas, de ella tres cerradas (1, 2 y 5) y dos abiertas (3 y 4), cuya relación ignora el sujeto (Ver Anexo 86). Estas tres preguntas se relacionan a través de lo que se denomina el "Cuadro Lógico de *Iadov*" (Anexo 87).

El número resultante de la interrelación de las tres preguntas nos indica la posición de cada sujeto en la escala de satisfacción, o sea su satisfacción individual. La escala de satisfacción utilizada es la siguiente (Orestes Febles, 2014):

1. Clara satisfacción.
2. Más satisfecho que insatisfecho.
3. No definida.
4. Más insatisfecho que satisfecho.
5. Clara insatisfacción.
6. Contradictoria.

Esta técnica también permite obtener el índice de satisfacción grupal (ISG), para lo cual se trabaja con los diferentes niveles de satisfacción que se expresan en la escala numérica que oscila entre +1 y -1, como se muestra en la Tabla 26:

Tabla 26: Índice para el cálculo de satisfacción grupal. Fuente: (Orestes Febles, 2014).

Escala	Resultado
+1	Máximo de satisfacción
0.5	Más satisfecho que insatisfecho
0	No definido y contradictorio
-0.5	Más insatisfecho que satisfecho
-1	Máxima insatisfacción

La satisfacción grupal se calcula por la siguiente Ecuación 4:

Ecuación 4: Fórmula para determinar la satisfacción grupal. Fuente: (Orestes Febles, 2014).

$$ISG = \frac{A(+1) + B(+0.5) + C(0) + D(-0.5) + E(-1)}{N}$$

Donde:

- A, B, C, D, E, representan el número de sujetos con índice individual 1; 2; 3 o 6; 4; 5.
- N representa el número total de sujetos del grupo.

El índice grupal arroja valores entre + 1 y - 1. Los valores que se encuentran comprendidos entre - 1 y - 0,5 indican insatisfacción; los comprendidos entre - 0,49 y + 0,49 evidencian

contradicción y los que caen entre 0,5 y 1 indican que existe satisfacción (Orestes Febles, 2014).

Se crearon dos sistemas conversacionales con sus respectivos componentes. Uno llamado *GESPRO_Bot* que será integrado a investigaciones del Departamento de Maestría y Doctorado. Y el segundo llamado *Adán_Bot* que fue entrenado con el fin de explicar cómo función y está conformado el subsistema conversacional. Se aplicó la técnica de *Iadov* a 12 trabajadores del departamento de Gestión de Proyectos haciendo uso del *bot* conversacional *Adán_Bot*, obteniendo los siguientes resultados que se muestran en la Imagen 15:

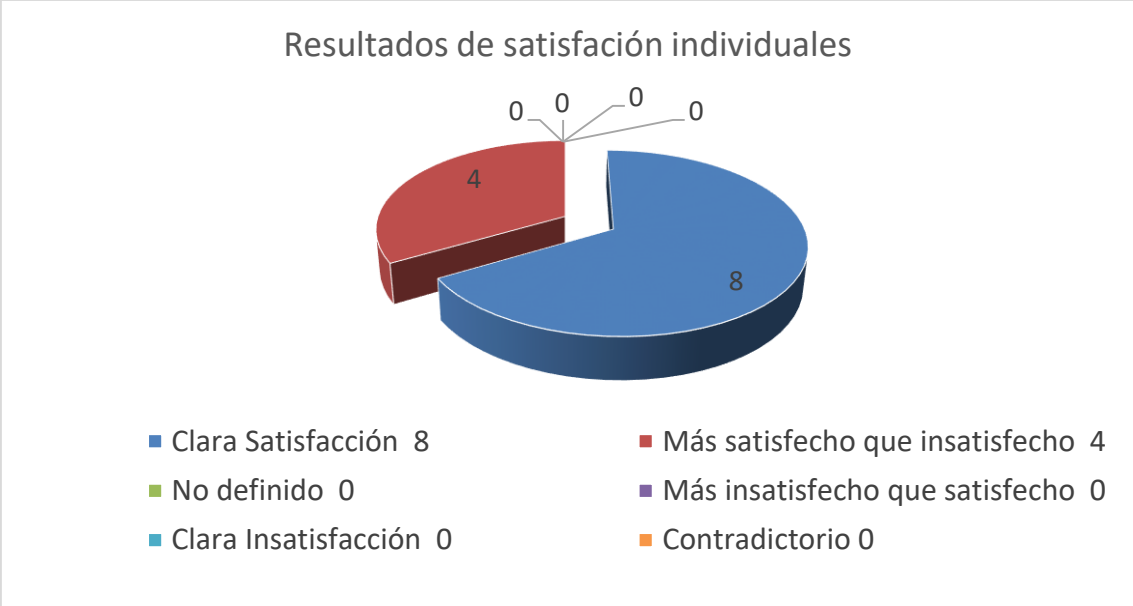


Imagen 15: Resultados de la técnica de Iadov.

Los resultados de la encuesta de satisfacción individual a los trabajadores del departamento de Gestión de Proyecto devolvieron un total de 4 trabajadores más satisfechos que insatisfecho y 8 trabajadores con una clara satisfacción. Estos resultados insatisfacción fueron dados por el poco tiempo de entrenamiento y que el rango de macheo que tenía el *chatbot Adan_Bot* eran muy alto. Provocando que algunas veces no obtuvieran respuestas a sus preguntas o no fueran coherentes las respuestas que daba.

Sin embargo, los otros trabajadores sí tenía una clara satisfacción porque pudieron apreciar las ventajas que le puede ofrecer al subsistema con la creación de sistemas conversacionales a la *suite*. Además, permite una mejor accesibilidad de la información que se gestión en la *suite*, a través del ahorro de tiempo y la facilidad de la búsqueda de

información que se encontraba en el manual de usuario y que ahora está en el *chatbot Gespro_Bot.*

Los resultados que se capturados en la encuesta de satisfacción individual se sustituyeron en la ecuación del ISG. Obteniendo un resultado de ISG de 0,67 aproximadamente, lo que refleja aceptación de la propuesta y un buen grado de satisfacción por parte de los trabajadores del departamento de Gestión de Proyectos.

3.4. Conclusiones del capítulo

En este capítulo se especificaron las dos últimas fases que propone la metodología XP, logrando obtener las siguientes conclusiones:

- El desglose de las HU en tareas de ingenierías permitió especificar a detalle el proceso de implementación del subsistema.
- La realización de las pruebas unitarias, funcionales y de rendimiento permitieron detectar fallas en el subsistema y corregirlas antes de la entrega final del subsistema.
- El empleo de la técnica de *Iadov* consistió en obtener los resultados de satisfacción individual y grupal de algunos usuarios de la *suite* XEDRO_GESPRO logrando ser satisfactorio para un índice de 0.67.

Conclusiones

Con el desarrollo de la presente investigación se concluye que:

- Las desventajas identificadas en el estudio a las plataformas de creación de sistemas conversacionales existentes no permitían cumplir con el objetivo propuesto, por lo que se optó por el desarrollo de una plataforma que supliera estas carencias.
- Las técnicas, herramientas y metodología escogidas resultaron ser muy provechosas y facilitaron el desarrollo de la propuesta de solución.
- La implementación de una herramienta para crear sistemas conversacionales en la *suite* XEDRO-GESPRO permitió dotar a dicha *suite* de una nueva vía para el intercambio de información con sus usuarios.
- Los algoritmos de similitud textual empleados en la construcción del subsistema resultaron ser eficaces para realizar la comparación de los textos introducidos por los usuarios.
- La técnica de validación que se le realizó al subsistema brindó como resultado de satisfacción un índice de 0.67, equivalente a un buen grado de aceptación por parte de los trabajadores del departamento de Gestión de Proyecto.

Recomendaciones

Para darle continuidad a la presente investigación se recomienda:

1. Incorporar otros algoritmos de similitud de textos cortos que permitan determinar la respuesta de manera eficiente a las preguntas de los usuarios.
2. Implementar algoritmos que permitan nuevas formas de comunicación como archivos multimedia (imágenes y videos) y mensajes de voz en las respuestas de los sistemas conversacionales.
3. Crear una API REST que contengan las principales funcionalidades del subsistema para que otras aplicaciones accedan a los servicios de creación y uso de sistemas conversacionales.

Referencias bibliográfica

57dc2692ee164.pdf. (s. f.). Recuperado de http://zera.media.uci.cu/uploads/resources/57dc2692ee164.pdf

Amón, I., & Jiménez, C. (2010). Funciones de similitud sobre cadenas de texto: Una comparación basada en la naturaleza de los datos.

Aroca, D. A. (2018). *Aplicación Web de bases de datos usando el Framework Ruby on Rails* (Tesis de grado). Universidad Politécnica de Valencia, Valencia.

Badaró, S., Ibañez, L. J., & Agüero, M. J. (2013, octubre). Sistemas Expertos: Fundamentos, Metodologías y Aplicaciones. *Ciencia y Tecnología*, *13*, 349-364.

Beck, K., & Fowler, M. (2001). *Planning Extreme Programming*. Addison-Wesley Professional.

Ben-Ari, M. (2006). *Understanding Programming Languages* (John Wiley & Sons, Chichester, 1996.). Weizmann Institute of Science.

Cadavid, A. N., Martínez, J. D. F., & Vélez, J. M. (2013). Revisión de metodologías ágiles para el desarrollo de software. *PROSPECTIVA*, *11*(2), 30-39.

Cambría, A. (2016, mayo 2). La importancia de la comunicación estratégica. Instituto Español de Estudios Estratégicos. Recuperado de ieee.es

Campos, L. M. A., & Díaz, B. O. M. (2015). *Diseño de un sistema web de búsqueda inteligente conversacional para ubicación de empresas y servicios*. Universidad Nacional de Trujillo, Trujillo, Perú.

Carmona, M. Á. Á. (2014). *Detección de similitud semántica en textos cortos* (Tesis de Maestría). Tonantzintla, Puebla.

Castillo, A. A. (2017). *Curso de Programación Web: JavaScript, Ajax y jQuery.* (2da ed.). IT Campus Academy.

Centros de Desarrollo. (s. f.). Recuperado 12 de abril de 2019, de http://www.uci.cu/investigacion-y-desarrollo/centros-de-desarrollo

Chatfuel. (2019a). About Chatfuel. Recuperado 3 de mayo de 2019, de https://chatfuel.com/

Chatfuel. (2019b). Chatfuel. Recuperado 21 de febrero de 2019, de https://chatfuel.com./

Chen, Y. Y., Yong, S.-P., & Ishak, A. (2014). Email hoax detection system using Levenshtein distance method. Recuperado 12 de mayo de 2019, de /paper/Email-Hoax-Detection-System-Using-Levenshtein-Chen-Yong/9eb2070831680b4e64511bf4f404bc4d3c85d7f4

Coba, A. F. P. (2016). *Análisis descriptivo de la tecnología Ruby on Rails para el desarrollo de páginas web*. Quito.

Demeter, A. (2019, abril 28). Pro & Free Plan Changes. Recuperado 28 de abril de 2019, de https://docs.chatfuel.com/configure/chatfuel-pro/pro-free-plan-changes

DiagramasUML. (2019). Diagrama de Paquetes. Recuperado 7 de mayo de 2019, de https://diagramasuml.com/paquetes/

Escribano, G. F. (2002, diciembre 19). Introducción a Extreme Programming. Ingenieria del Software II.

Ferigra, L., & Santiago, G. (2015). Marco de trabajo ágil de desarrollo de software combinando Scrum y XP. Aplicación a un caso de estudio. Recuperado de http://dspace.udla.edu.ec/handle/33000/4364

Flow XO. (2018, noviembre 29). Introduction to Flow XO. Recuperado 28 de abril de 2019, de https://support.flowxo.com/article/132-introduction

Flow XO. (2019a). AI online chatbot software, live chat on websites. Recuperado 21 de febrero de 2019, de https://flowxo.com/

Flow XO. (2019b). Automated chat bot for websites, flexible pricing. Recuperado 6 de mayo de 2019, de https://flowxo.com/pricing/

Fuentes, A. (2016, mayo 10). 10 términos sobre desarrollo de software que debes conocer (II). Recuperado 22 de mayo de 2019, de http://mhp-net.es/10-terminos-sobre-desarrollo-de-software-2/

Gamma, E., Helm, R., Johnson, R., & Vlissides, J. (1997). *Desing patterns: elements of reusable Object-Oriented Software*. Addison Weasley.

García, J. (2018, diciembre 3). Plataformas conversacionales, la tendencia que está cambiando el marketing. Recuperado 14 de mayo de 2019, de https://telcelempresas.com/plataformas-conversacionales-la-tendencia-que-esta-cambiando-el-marketing/

GESPRO 13.05. (s. f.). Recuperado 4 de mayo de 2019, de https://www.uci.cu/investigacion-y-desarrollo/productos/xedro/gespro-1305

GitBook. (2019). Git Book. Recuperado 22 de febrero de 2019, de https://git-scm.com/book/es/v2

GitHub. (2019). *Contribute to hackerschoolmty/rails-patterns development by creating an account on GitHub*. Ruby, Hacker School Monterrey. Recuperado de https://github.com/hackerschoolmty/rails-patterns (Original work published 2015)

Guerrero, R. M. (2019). PostgreSQL-ES. Recuperado 22 de febrero de 2019, de https://e-mc2.net/es/postgresql-es, http://www.postgresql.org.es/sobre_postgresql

Guevara. (2019, abril 17). Plataformas Conversacionales. Recuperado 23 de mayo de 2019, de https://prezi.com/p/amjiwv7vzixl/plataformas-conversacionales/

Highcharts. (2019). Interactive JavaScript charts for your webpage. Recuperado 21 de febrero de 2019, de https://www.highcharts.com/

Hirschberg, J., & Manning, C. D. (2015). Advances in natural language processing. *Science*, *349*(6245), 261–266. https://doi.org/10.1126/science.aaa8685

INEI. (1999). *Herramientas CASE*. Instituto Nacional de Estadística e Informática.

Jacobson, Booch, & Rumbaugh. (2011, marzo 9). El Proceso Unificado de Desarrollo de Software. Recuperado 3 de mayo de 2019, de https://es.scribd.com/doc/50327385/El-Proceso-Unificado-de-Desarrollo-de-Software-Jacobson-Booch-Rumbaugh

Joskowicz, I. J. (2008, febrero 10). Reglas y Prácticas en eXtreme Programming.

JS Foundation. (2019). jQuery. Recuperado 21 de febrero de 2019, de https://jquery.com/

Larman, C. (2000). *UML y Patrones. Una introducción al análisis y diseño orientado a objetos y al proceso unificado.* (2da ed.). Prentice Hall.

León, R. A. H., & González, S. C. (2012). *El Proceso de Investigación Científica* (2da ed.). La Habana, Cuba: La Editorial Universitaria del Ministerio de Educación Superior.

Marquez, J. C. C. (2016, junio 30). Estándares de Ruby. Recuperado 11 de mayo de 2019, de http://aprendiendoingenieria.es/estandares-ruby-on-rails/

Matos, E. M. (2016). *Sistema Basado en el Conocimiento para asistir el diagnóstico y sugerir tratamientos en pacientes hemipléjicos con alteraciones motoras* (Tesis de grado). Universidad de las Ciencias Informáticas, La Habana, Cuba.

Moreno, J. C., & Marciszack, M. M. (2013, noviembre). La usabilidad desde la perspectiva de la validación de requerimientos no funcionales para aplicaciones web. Recuperado 27 de febrero de 2019, de https://www.researchgate.net/publication/278035280_La_Usabilidad_Desde_La_Perspectiva_De_La_Validacion_de_Requerimientos_No_Funcionales_Para_Aplicaciones_Web

Noman, U. (2019, mayo 3). Understand Botsify Pricing. Recuperado 3 de mayo de 2019, de https://help.botsify.com/faqs/understand-botsify-pricing/

Noorullah, R. M. (2012). Grasp and GOF patterns in solving design problems. *International Journal of Engineering and Technology*, *1*(3).

Novicell. (2019, mayo 3). ¿Qué es un chatbot? Ventajas y desventajas. Recuperado 3 de mayo de 2019, de https://www.novicell.es/es/blog/chatbots-empresa/

Ochoa, M. (2018, septiembre 5). Qué son las Plataformas Conversacionales y por qué no hay que perderlas de vista. Recuperado 14 de mayo de 2019, de https://itmastersmag.com/noticias-analisis/que-son-las-plataformas-conversacionales-y-por-que-no-hay-que-perderlas-de-vista/

Octane AI. (2019, abril 28). About US | Octane AI for Shopify Stores. Recuperado 28 de abril de 2019, de https://join.octaneai.com/about

Orestes Febles. (2014, junio). *Gc*. Internet. Recuperado de https://es.slideshare.net/ofebles/gc-37299112

Pardo, M. R. V., Tapia, J. A. H., Moreno, A. S. G., & F, V. S., L. (2018, septiembre 14). Comparación de tendencias tecnológicas en aplicaciones web, *7*(3), 28-49.

Parr, B., K-Brooks, L., Berry, M., & Schlicht, M. (2019, abril 28). Octane AI. Recuperado 28 de abril de 2019, de https://www.crunchbase.com/organization/octane-ai#section-overview

Perera, N. (2017, febrero 4). Usando una guía de estilos en Ruby. Recuperado 22 de mayo de 2019, de https://nicanor.github.io/articles/2017/02/usando-una-guia-de-estilos-en-ruby/

Pérez, D. . P. Y. P., & Pupo, Ms. I. P. (2019, mayo). *Introducción a softcomputing. Sistemas de inferencia borrosos*. Universidad de las Ciencias Informáticas.

Piñero, P. Y., Torres, S., Matias, M. I., & Garcia, J. A. L. (2013, marzo). GESPRO. Paquete para la gestión de proyectos. Recuperado 9 de mayo de 2019, de https://www.researchgate.net/publication/260418890_GESPRO_Paquete_para_la_gestion_de_proyectos

Pressman, R. S. (2010). *Ingeniería de Software un enfoque práctico. Journal of Chemical Information and Modeling* (7ma ed., Vol. 53). Mc Graw Hill.

Pressman, Roger S. (2002). *Ingeniería de Software. Un enfoque práctico* (5ta ed., Vol. 1). F. Varela. Recuperado de http://libreria-universitaria.blogspot.com

Red Hat. (2019). What are APIs? Recuperado 27 de mayo de 2019, de https://www.redhat.com/en/topics/api/what-are-application-programming-interfaces

Revang, M., Baker, V., Manusama, B., & Mullen, A. (2018, junio 20). Market guide for conversational platforms. Recuperado de https://www.gartner .com/doc/reprints?id=1-5AQ8F36&ct=180806&st=sb&utm_campaign=General%2520Nurture&utm_medium=email&_hsenc=…

Rojas, D. P. (2001). *Manejo del dialogo en sistemas de iniciativa mixta* (Tesis de grado). Puebla, México.

Ruiz, I. J. H. (2017). *Procedimiento para la evaluación de competencias laborales basado en funciones de similitud de textos* (Tesis de Maestría). Universidad de las Ciencias Informáticas, La Habana, Cuba.

Rumbaugh, J., Jacobson, I., & Booch, G. (2000). *El Lenguaje Unificado de Modelado. Manual de referencia*. Madrid, España: Addison, Weasley.

Russell, S., & Norvig, P. (2010). *Artificial Intelligence. A Modern Approach* (3ra ed.). PEARSON.

Shum, H., He, X., & Li, D. (2018). From Eliza to XiaoIce: challenges and opportunities with social chatbots. *Frontiers of Information Technology & Electronic Engineering*, *19*(1), 10-26. https://doi.org/10.1631/FITEE.1700826

Sistema gestor de base de datos. (s. f.). Universidad Politécnica de Puebla.

Sommerville, I. (2011). *Ingeniería de Software* (9na ed.). México.

Soto, M. G. (2018, marzo 9). Episodio 1: Procesamiento de lenguaje natural. Recuperado 16 de abril de 2019, de https://planetachatbot.com/1-procesamiento-de-lenguaje-natural-1443ff471ed0

Suja, A. (2018, abril 11). Entrevista con Usama, CEO de Botsify en Chatbots y Messenger Marketing. Recuperado 3 de mayo de 2019, de https://ecommerce-platforms.com/es/interviews/botsify-interview

Tade, A. C. R., & García, I. G. (2014). Robot Virtual en Orientación Vocacional. *Revista Iberoamericana de Producción Académica y Gestión Educativa*, *1*(2).

Tom. (2010, agosto 28). Definición de un Modelo de Datos. Recuperado 7 de mayo de 2019, de https://tombasededatos.wordpress.com/2010/08/28/2-1-definicion-de-un-modelo-de-datos/

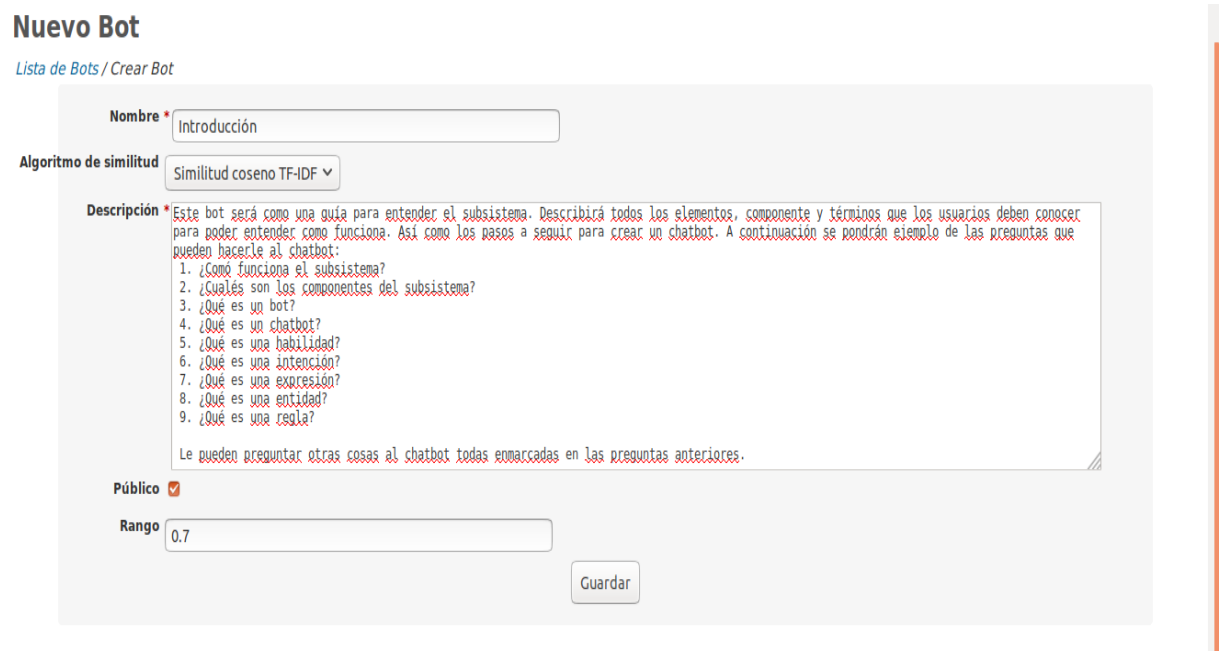
UNESCO Biblioteca Digital. (s. f.). Las Tecnologías de la Información y la Comunicación en la enseñanza: manual para docentes o cómo crear nuevos entornos de aprendizaje abierto por medio de las TIC. Recuperado 22 de mayo de 2019, de https://unesdoc.unesco.org/ark:/48223/pf0000139028_spa

Vega, A. A. (2017, junio 11). Convención de nombres: desde el Camel_Case hasta el kebab_case. Recuperado 11 de mayo de 2019, de https://medium.com/@alonsus91/convenci%C3%B3n-de-nombres-desde-el-camelcase-hasta-el-kebab-case-787e56d6d023

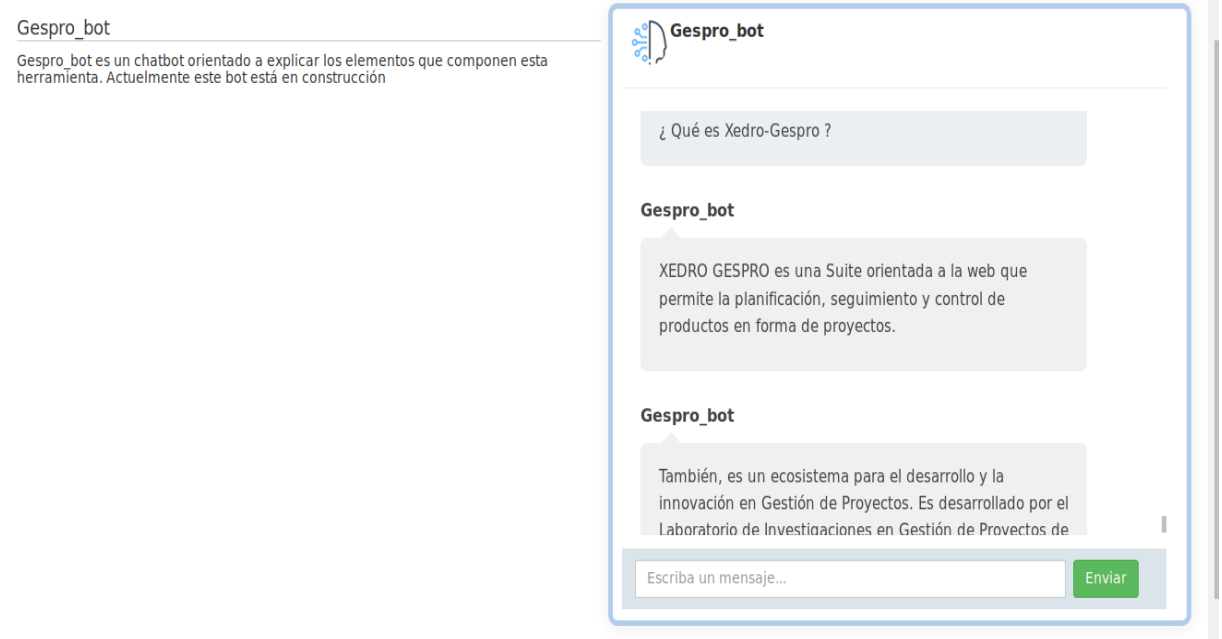
Visual Paradigm. (s. f.). Visual Paradigm Frequently Asked Questions. Recuperado 22 de mayo de 2019, de https://www.visual-paradigm.com/support/faq.jsp

Anexo

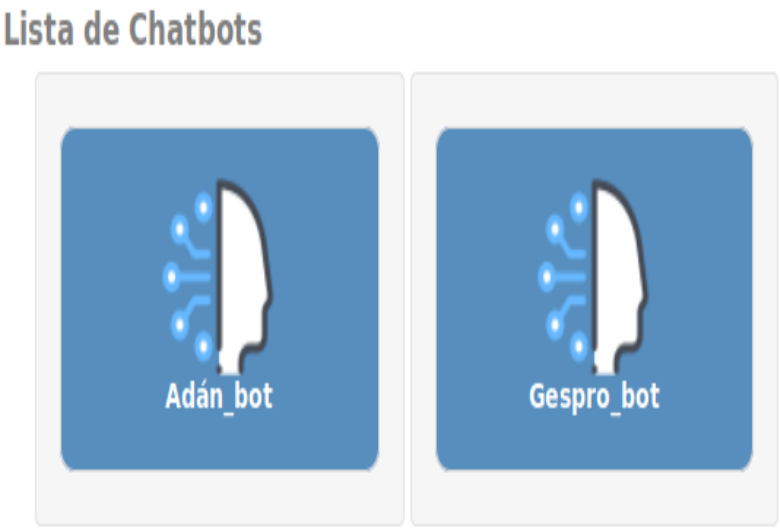
Imágenes de la propuesta de solución:



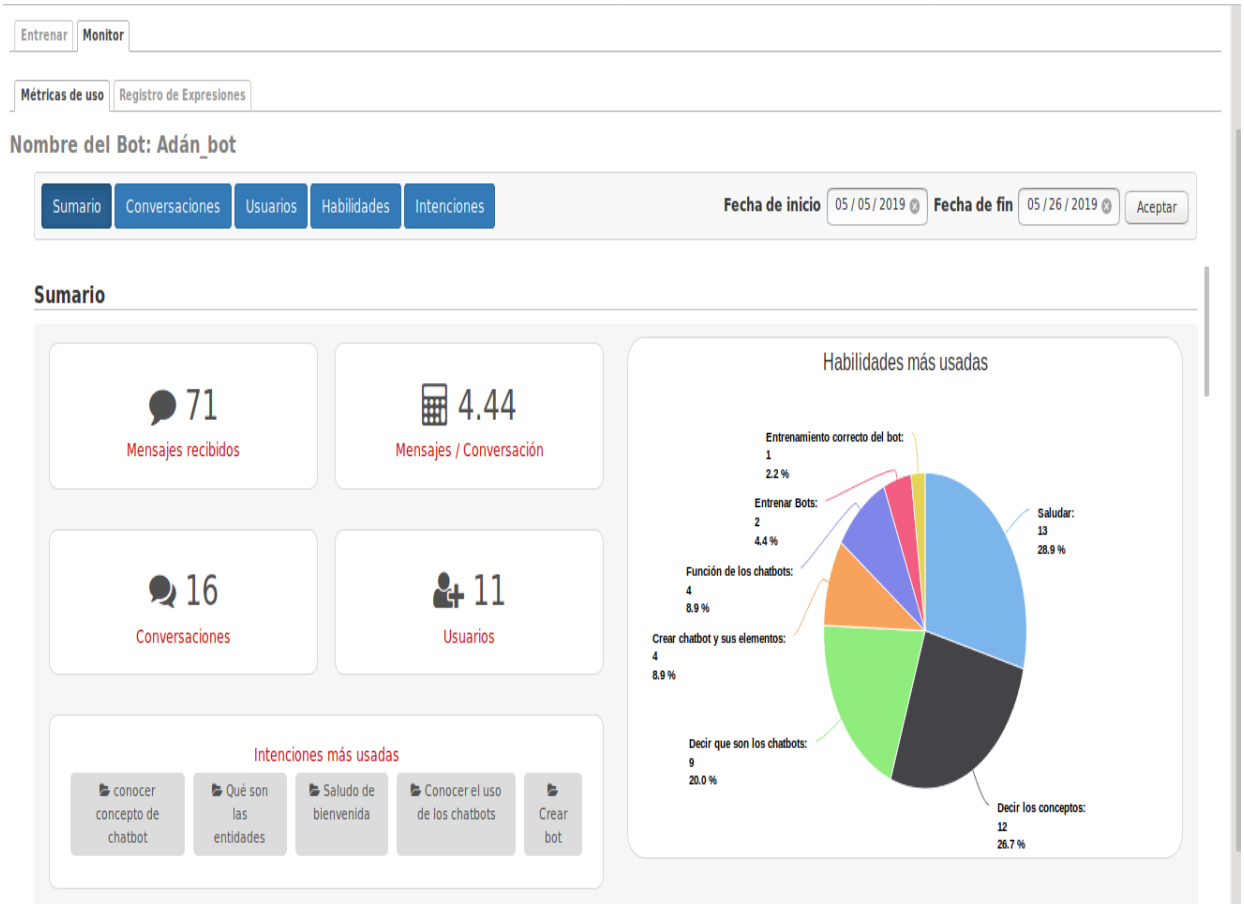
Anexo 1: Crear bots conversacional (Área de gestión de datos).



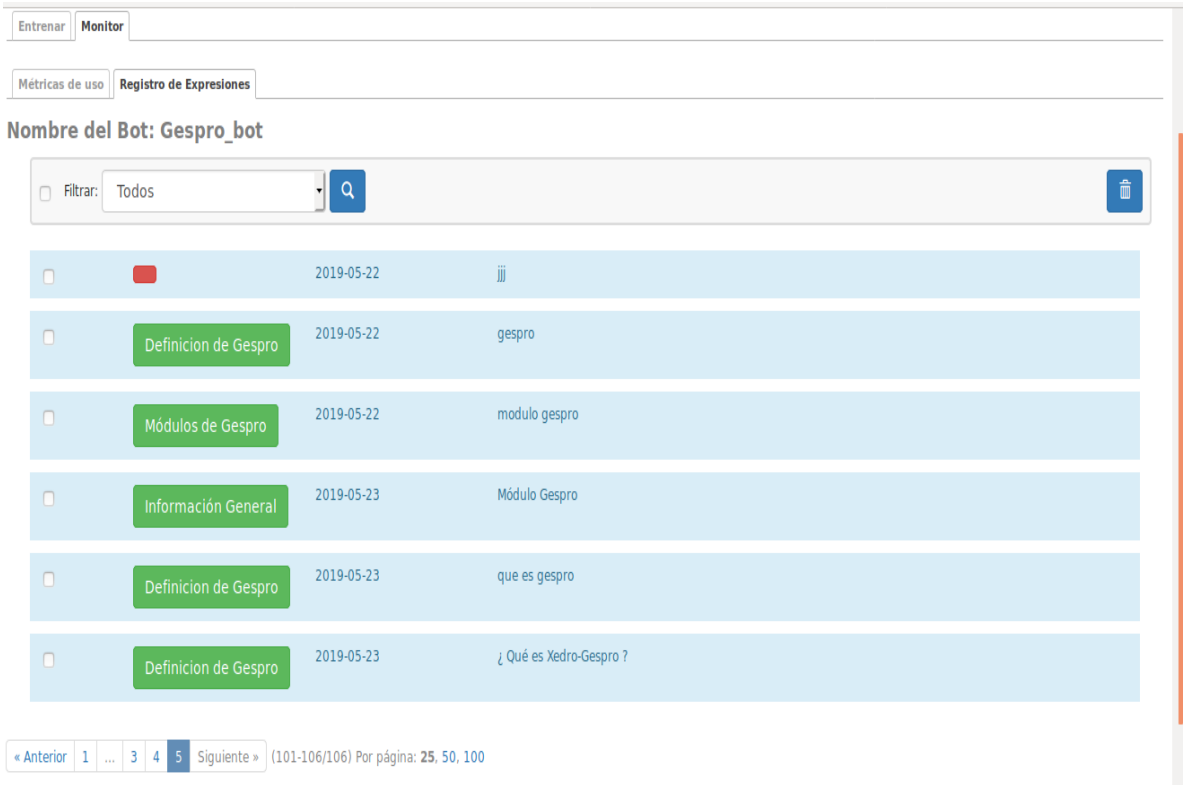
Anexo 2: Interfaz de mensajería (Área de chat).



Anexo 3: Lista de chatbots disponibles (Área de chat).



Anexo 4: Ejemplo 1 del área de monitoreo (Métricas de uso).



Anexo 5: Ejemplo del área de monitoreo (Registro de expresiones).

Historias de usuarios:

Anexo 6: Historia de usuario 2.

Historia de usuario	
Número: 2	**Usuario:** Administrador
Nombre de HU: Editar *bot* conversacional.	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Medio
Tiempo estimado: 0.2	**Iteración Asignada:** 1
Programador responsable: Ariel Acevedo Guevara	
Descripción: El sistema debe permitir editar los campos Nombre, Descripción, Público y Rango de macheo, validando que los nuevos datos introducidos sean correctos; y que los campos obligatorios Nombre y Descripción no queden en blanco.	
Observaciones: En la tabla donde se listarán los *bots* conversacionales, el sistema permitirá editar los valores de los campos del *bot* conversacional mediante la funcionalidad Editar.	
Prototipo de Interfaz:	

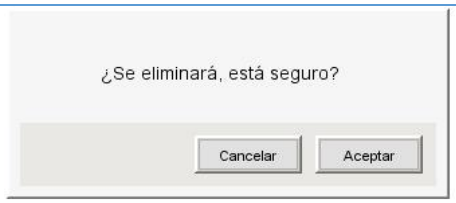
Anexo 7: Historia de usuario 3.

Historia de usuario	
Número: 3	**Usuario:** Administrador
Nombre de HU: Listar *bot.*	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Medio
Tiempo estimado: 0.2	**Iteración Asignada:** 1
Programador responsable: Ariel Acevedo Guevara	
Descripción: El sistema debe permitir listar los *bots* conversacionales creados. Estos se mostrarán en una tabla, uno a continuación de otro.	
Observaciones: -.	
Prototipo de Interfaz:	

#	Nombre	Descripción	Público	Rango de macheo			
1	Ingeniería	Conceptos fundamentales	Si	0.8	Detalles	Modificar	Eliminar
2	Gestion de proyectos	Conceptos sobre gestion de proyectos	No	0.7	Detalles	Modificar	Eliminar

Anexo 8: Historia de usuario 4.

Historia de usuario	
Número: 4	**Usuario:** Administrador
Nombre de HU: Eliminar *bot* conversacional.	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Medio
Tiempo estimado: 0.2	**Iteración Asignada:** 1
Programador responsable: Ariel Acevedo Guevara	
Descripción: El sistema debe permitir eliminar los *bots* conversacionales creados.	
Observaciones: En la tabla donde se listarán los *bots* conversacionales, el sistema permitirá eliminar los *bots* conversacionales mediante la funcionalidad Eliminar. Se reafirmará a través de un mensaje de notificación si es el *bot* conversacional que se quiere borrar.	
Prototipo de Interfaz:	

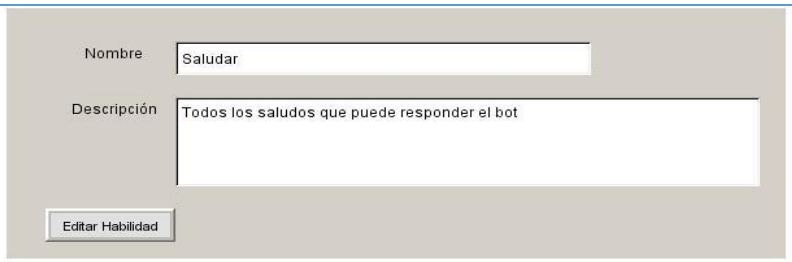


Anexo 9: Historia de usuario 5.

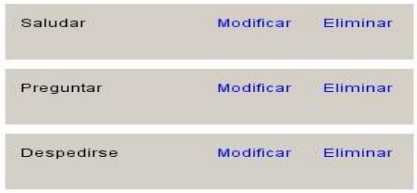
Historia de usuario	
Número: 5	**Usuario:** Administrador
Nombre de HU: Crear habilidades.	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Medio
Tiempo estimado: 0.4	**Iteración Asignada:** 1
Programador responsable: Ariel Acevedo Guevara	
Descripción: El sistema debe permitir crear habilidades, validando que los campos Nombre y Descripción no queden en blanco; y que los datos introducidos sean correctos.	
Observaciones: Para crear una habilidad, el usuario previamente deberá crear un *bot* conversacional.	
Prototipo de Interfaz: 	

Anexo 10: Historia de usuario 6.

Historia de usuario	
Número: 6	**Usuario:** Administrador
Nombre de HU: Editar habilidades.	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Medio
Tiempo estimado: 0.2	**Iteración Asignada:** 1
Programador responsable: Ariel Acevedo Guevara	
Descripción: El sistema debe permitir editar los campos Nombre y Descripción de las habilidades creadas, validando que los datos introducidos sean correctos y que no hallan campos en blanco.	
Observaciones: En la interfaz donde se listarán las habilidades, el sistema permitirá editar los valores de los campos de las habilidades mediante la funcionalidad Editar.	
Prototipo de Interfaz:	

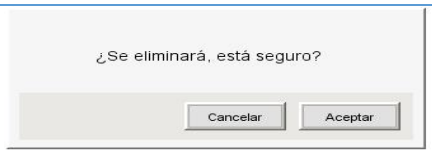


Anexo 11: Historia de usuario 7.

Historia de usuario	
Número: 7	**Usuario:** Administrador
Nombre de HU: Listar habilidades.	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Medio
Tiempo estimado: 0.2	**Iteración Asignada:** 1
Programador responsable: Ariel Acevedo Guevara	
Descripción: El sistema debe permitir listar las habilidades creadas. Estas se mostrarán en una interfaz, una a continuación de otra.	
Observaciones: -.	
Prototipo de Interfaz: 	

Anexo 12: Historia de usuario 8.

Historia de usuario	
Número: 8	**Usuario:** Administrador
Nombre de HU: Eliminar habilidades.	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Medio
Tiempo estimado: 0.2	**Iteración Asignada:** 1
Programador responsable: Ariel Acevedo Guevara	
Descripción: El sistema debe permitir eliminar las habilidades creadas.	
Observaciones: En la interfaz donde se listarán las habilidades, el sistema permitirá eliminarlas mediante la funcionalidad Editar. Se reafirmará a través de un mensaje de notificación si es la habilidad que se quiere borrar.	
Prototipo de Interfaz:	

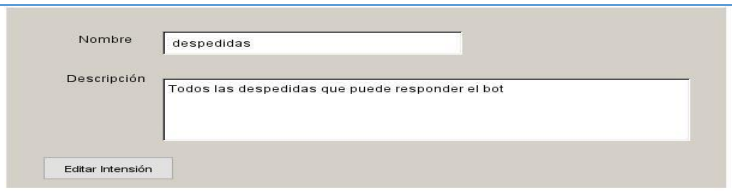


Anexo 13: Historia de usuario 9.

Historia de usuario	
Número: 9	**Usuario:** Administrador
Nombre de HU: Crear intención.	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Medio
Tiempo estimado: 0.4	**Iteración Asignada:** 1
Programador responsable: Ariel Acevedo Guevara	
Descripción: El sistema debe permitir crear intenciones, validando que los campos Nombre y Descripción no queden en blanco; y que los datos introducidos sean correctos.	
Observaciones: Las intenciones creadas aparecerán listadas en una tabla. Para crear una intención, el usuario primeramente deberá crear un *bot* conversacional.	
Prototipo de Interfaz: 	

Anexo 14: Historia de usuario 10.

Historia de usuario	
Número: 10	**Usuario:** Administrador
Nombre de HU: Editar intención.	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Medio
Tiempo estimado: 0.2	**Iteración Asignada:** 1
Programador responsable: Ariel Acevedo Guevara	
Descripción: El sistema debe permitir editar los campos Nombre y Descripción de las intenciones creadas, validando que los datos introducidos sean correctos y que no hallan campos en blanco.	
Observaciones: En la tabla donde se listarán las intenciones, el sistema permitirá editar los valores de los campos de las intenciones mediante la funcionalidad Editar.	
Prototipo de Interfaz:	



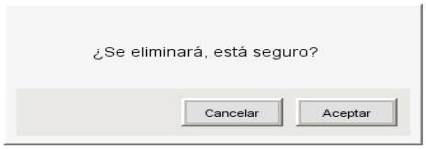
Anexo 15: Historia de usuario 11.

Historia de usuario	
Número: 11	**Usuario:** Administrador
Nombre de HU: Listar intención.	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Medio
Tiempo estimado: 0.2	**Iteración Asignada:** 1
Programador responsable: Ariel Acevedo Guevara	
Descripción: El sistema debe permitir listar las intenciones creadas. Estas se mostrarán en una tabla, una a continuación de otra.	
Observaciones: -.	
Prototipo de Interfaz:	

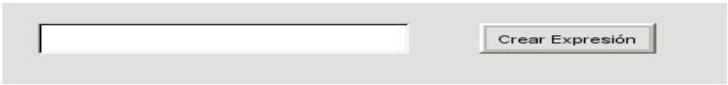
#	Nombre	Descripción		
1	Conceptos	Preguntas sobre los conceptos	Modificar	Eliminar
2	Diagramas	Preguntas sobre los diagramas	Modificar	Eliminar

Anexo 16: Historia de usuario 12.

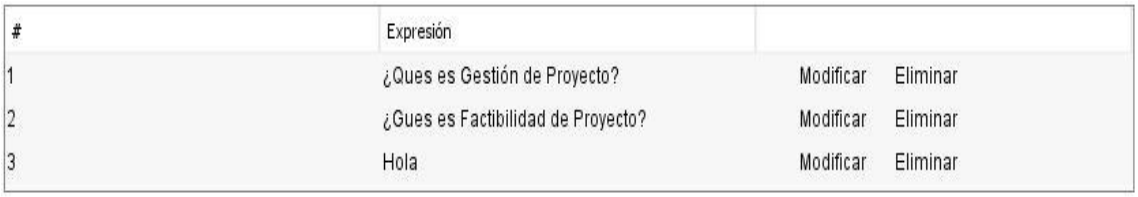
Historia de usuario	
Número: 12	**Usuario:** Administrador
Nombre de HU: Eliminar intención.	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Medio
Tiempo estimado: 0.2	**Iteración Asignada:** 1
Programador responsable: Ariel Acevedo Guevara	
Descripción: El sistema debe permitir eliminar las intenciones creadas.	
Observaciones: En la tabla donde se listarán las intenciones, el sistema permitirá eliminar las intenciones mediante la funcionalidad Eliminar. Se reafirmará a través de un mensaje de notificación si es la intención que se quiere borrar.	
Prototipo de Interfaz:	



Anexo 17: Historia de usuario 13.

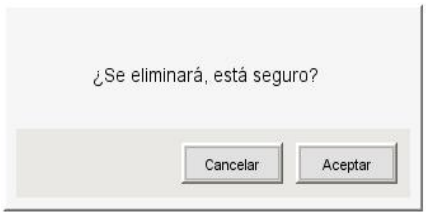
Historia de usuario	
Número: 13	**Usuario:** Administrador
Nombre de HU: Crear expresión.	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Medio
Tiempo estimado: 0.4	**Iteración Asignada:** 1
Programador responsable: Ariel Acevedo Guevara	
Descripción: El sistema debe permitir crear expresiones validando que no queden campos en blanco y que los datos introducidos sean correctos.	
Observaciones: Las expresiones deben aparecen listadas en una tabla. Esta aparecerá en la misma interfaz donde se registrará el dato para una nueva expresión, el cual es Expresión.	
Prototipo de Interfaz: 	

Anexo 18: Historia de usuario 14.

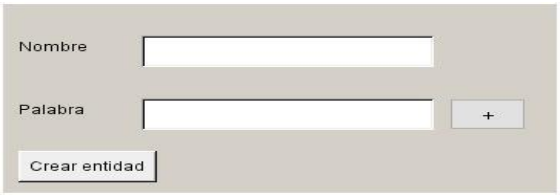
Historia de usuario	
Número: 14	**Usuario:** Administrador
Nombre de HU: Listar expresión.	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Medio
Tiempo estimado: 0.2	**Iteración Asignada:** 1
Programador responsable: Ariel Acevedo Guevara	
Descripción: El sistema debe permitir mostrar las expresiones creadas.	
Observaciones: Las expresiones deben aparecen listadas en una tabla una a continuación de la otra.	
Prototipo de Interfaz: 	

Anexo 19: Historia de usuario 15.

Historia de usuario	
Número: 15	**Usuario:** Administrador
Nombre de HU: Eliminar expresión.	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Medio

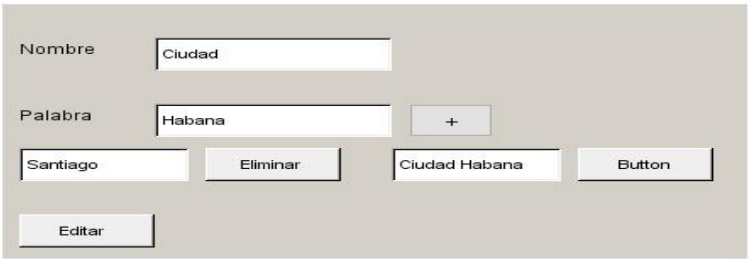
Tiempo estimado: 0.2	**Iteración Asignada:** 1
Programador responsable: Ariel Acevedo Guevara	
Descripción: El sistema debe permitir eliminar las intenciones creadas.	
Observaciones: En la tabla donde se listarán las intenciones, el sistema permitirá eliminar las intenciones mediante la funcionalidad Eliminar. Se reafirmará a través de un mensaje de notificación si es la intención que se quiere borrar.	
Prototipo de Interfaz: 	

Anexo 20: Historia de usuario 16.

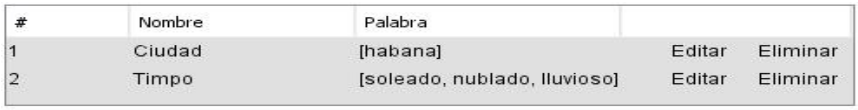
Historia de usuario	
Número: 16	**Usuario:** Administrador
Nombre de HU: Crear entidad.	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Medio
Tiempo estimado: 0.4	**Iteración Asignada:** 1
Programador responsable: Ariel Acevedo Guevara	
Descripción: El sistema debe permitir Crear entidades, validando que los datos introducidos por el usuario en los campos Nombre y Palabra sean correctos y que el campo Nombre no quede en blanco.	
Observaciones: El sistema mostrará las entidades creados en una tabla. Para crear una entidad, el usuario podrá asignarle una o varias palabras mediante un símbolo de más (+), el cual al ser presionado añadirá un nuevo campo de texto.	
Prototipo de Interfaz: 	

Anexo 21: Historia de usuario 17.

Historia de usuario	
Número: 17	**Usuario:** Administrador
Nombre de HU: Editar entidad.	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Medio

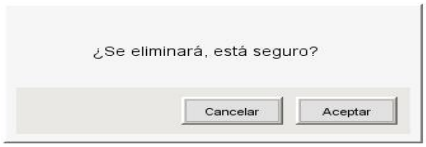
Tiempo estimado: 0.2	**Iteración Asignada:** 1
Programador responsable: Ariel Acevedo Guevara	
Descripción: El sistema debe permitir listar las entidades creadas. Estas se mostrarán en una tabla, una a continuación de otra.	
Observaciones: En la interfaz donde se listarán las habilidades, el sistema permitirá eliminarlas mediante la funcionalidad Editar.	
Prototipo de Interfaz: 	

Anexo 22: Historia de usuario 18.

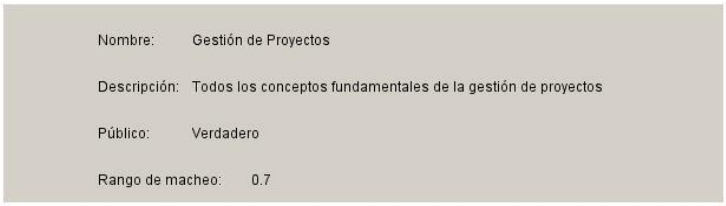
Historia de usuario	
Número: 18	**Usuario:** Administrador
Nombre de HU: Listar entidades.	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Medio
Tiempo estimado: 0.2	**Iteración Asignada:** 1
Programador responsable: Ariel Acevedo Guevara	
Descripción: El sistema debe permitir listar las entidades creadas. Estas se mostrarán en una tabla, una a continuación de otra.	
Observaciones: -.	
Prototipo de Interfaz: 	

Anexo 23: Historia de usuario 19.

Historia de usuario	
Número: 19	**Usuario:** Administrador
Nombre de HU: Eliminar entidades.	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Medio
Tiempo estimado: 0.2	**Iteración Asignada:** 1
Programador responsable: Ariel Acevedo Guevara	
Descripción: El sistema debe permitir eliminar las entidades creadas.	

Observaciones: En la interfaz donde se listarán las habilidades, el sistema permitirá eliminarlas mediante la funcionalidad Eliminar. Se reafirmará a través de un mensaje de notificación si es la entidad que se quiere borrar.
Prototipo de Interfaz: 

Anexo 24: Historia de usuario 20.

Historia de usuario	
Número: 20	**Usuario:** Administrador
Nombre de HU: Mostar detalles del *bot* conversacional.	
Prioridad en negocio: Baja	**Riesgo en desarrollo:** Medio
Tiempo estimado: 0.2	**Iteración Asignada:** 1
Programador responsable: Ariel Acevedo Guevara	
Descripción: El sistema debe mostrar los datos del *bot* conversacional *de los campos* Nombre, Descripción, Público y Rango de macheo.	
Observaciones: El sistema mostrará la información referente al *bot* conversacional seleccionado en la tabla donde se permitirá editar los datos introducidos anteriormente por el usuario.	
Prototipo de Interfaz: 	

Anexo 25: Historia de usuario 21.

Historia de usuario	
Número: 21	**Usuario:** Administrador
Nombre de HU: Asignar usuarios a un *bot* conversacional.	
Prioridad en negocio: Media	**Riesgo en desarrollo:** Medio
Tiempo estimado: 0.4	**Iteración Asignada:** 1
Programador responsable: Ariel Acevedo Guevara	
Descripción: El sistema debe permitir asignar un usuario a un *bot* conversacional. El usuario asignado podrá modificar y eliminar el *bot* conversacional al cual fue asignado.	

Observaciones: Cuando el usuario acceda a la interfaz para editar un *bot* conversacional, el sistema mostrará un formulario donde se editará los campos del *bot* conversacional y una tabla donde se mostrarán los usuarios asociados a ese *bot* conversacional. Además, aparecerá un campo donde se podrán seleccionar los usuarios que se quieran asignar al *bot* conversacional.
Prototipo de Interfaz: 

Anexo 26: Historia de usuario 22.

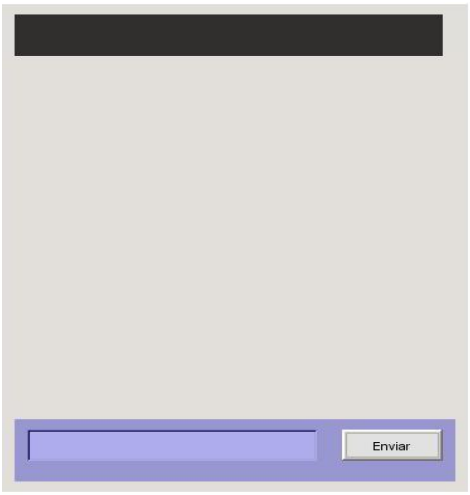
Historia de usuario	
Número: 22	**Usuario:** Administrador
Nombre de HU: Eliminar usuarios asignados a un *bot* conversacional.	
Prioridad en negocio: Media	**Riesgo en desarrollo:** Medio
Tiempo estimado: 0.2	**Iteración Asignada:** 1
Programador responsable: Ariel Acevedo Guevara	
Descripción: Cuando el usuario acceda a la interfaz para editar un *bot* conversacional, el sistema mostrará un formulario donde se editará los campos del *bot* conversacional y una tabla donde se mostrarán los usuarios asociados a ese *bot* conversacional. En cada fila de la tabla, se mostrará la funcionalidad Eliminar, la cual eliminará el usuario del *bot* correspondiente a cada fila.	
Observaciones: Se reafirmará a través de un mensaje de notificación si es el *bot* conversacional que se quiere borrar.	
Prototipo de Interfaz: 	

Anexo 27: Historia de usuario 23.

Historia de usuario	
Número: 23	**Usuario:** Administrador
Nombre de HU: Mostrar un listado de todos los *bots* conversacionales creados por los usuarios del sistema.	
Prioridad en negocio: Media	**Riesgo en desarrollo:** Medio
Tiempo estimado: 0.2	**Iteración Asignada:** 1
Programador responsable: Ariel Acevedo Guevara	

Descripción: El sistema debe permitir mostrar todos los *bots* conversacionales creados por todos los usuarios de sistema.
Observaciones: -.
Prototipo de Interfaz: 

Anexo 28: Historia de usuario 24.

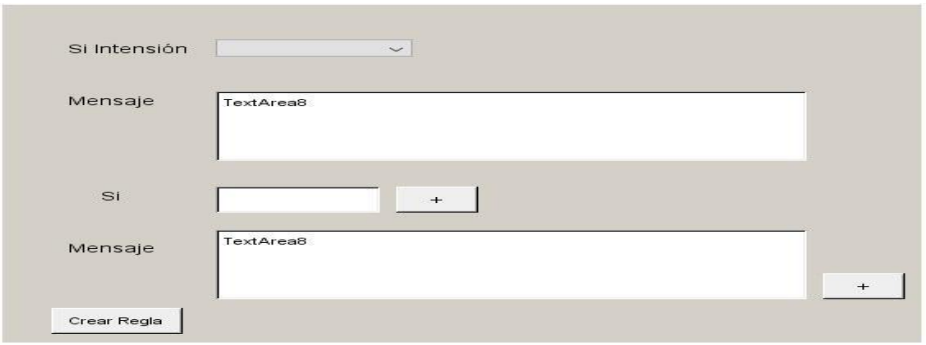
Historia de usuario	
Número: 24	**Usuario:** Administrador
Nombre de HU: Visualizar una interfaz de mensajería para cada *bot* conversacional que se cree.	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Medio
Tiempo estimado: 1	**Iteración Asignada:** 2
Programador responsable: Ariel Acevedo Guevara	
Descripción: Para cada *bot* conversacional creado, se mostrará una interfaz de mensajería mediante la cual el usuario podrá interactuar.	
Observaciones: -.	
Prototipo de Interfaz: 	

Anexo 29: Historia de usuario 25.

Historia de usuario	
Número: 25	**Usuario:** Administrador
Nombre de HU: Permitir la comunicación, entre usuario y *bots* conversacionales, mediante la mensajería instantánea a través de una interfaz visual.	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Medio
Tiempo estimado: 2	**Iteración Asignada:** 2

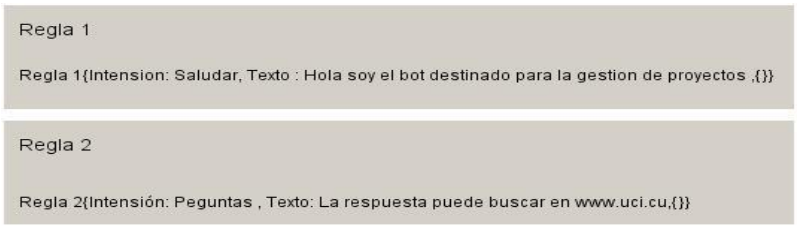
Programador responsable: Ariel Acevedo Guevara
Descripción: A través de una interfaz de *chat*, el usuario podrá interactuar con el *bot* conversacional seleccionado mediante el envío y recibo de mensajes.
Observaciones: -.
Prototipo de Interfaz: 

Anexo 30: Historia de usuario 27.

Historia de usuario	
Número: 27	**Usuario:** Administrador
Nombre de HU: Elaborar reglas para brindar respuestas a los mensajes del usuario.	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Alta
Tiempo estimado: 1.2	**Iteración Asignada:** 2
Programador responsable: Ariel Acevedo Guevara	
Descripción: El sistema mostrará una interfaz donde el usuario podrá crear las distintas reglas para brindar respuestas a los mensajes enviados por el usuario que interactúe con un *bot* conversacional.	
Observaciones: -.	
Prototipo de Interfaz: 	

Anexo 31: Historia de usuario 28.

Historia de usuario	
Número: 28	**Usuario:** Administrador
Nombre de HU: Listar reglas para brindar respuestas a los mensajes del usuario.	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Alta
Tiempo estimado: 0.2	**Iteración Asignada:** 2
Programador responsable: Ariel Acevedo Guevara	
Descripción:	

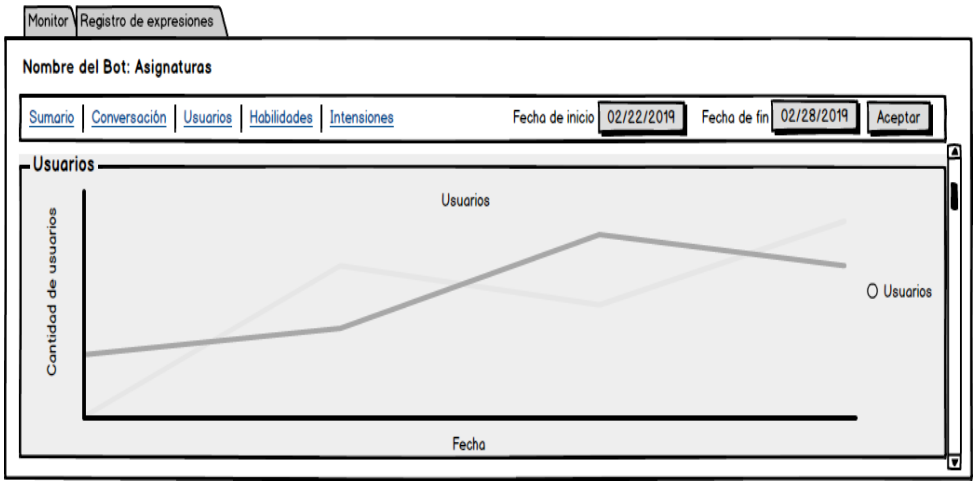
Observaciones: -.
Prototipo de Interfaz: 

Anexo 32: Historia de usuario 30.

Historia de usuario	
Número: 30	**Usuario:** Administrador
Nombre de HU: Calcular estadísticas de las conversaciones.	
Prioridad en negocio: Media	**Riesgo en desarrollo:** Medio
Tiempo estimado: 1	**Iteración Asignada:** 3
Programador responsable: Tania Socarras Navarro	
Descripción: El sistema debe permitir calcular estadísticas de las conversaciones que se tienen por día entre el *chatbot* y los usuarios.	
Observaciones: Se deberá elegir un período de fechas del que se quiere la información; de no elegir uno se tomará por defecto el último período de fechas.	
Prototipo de Interfaz: 	

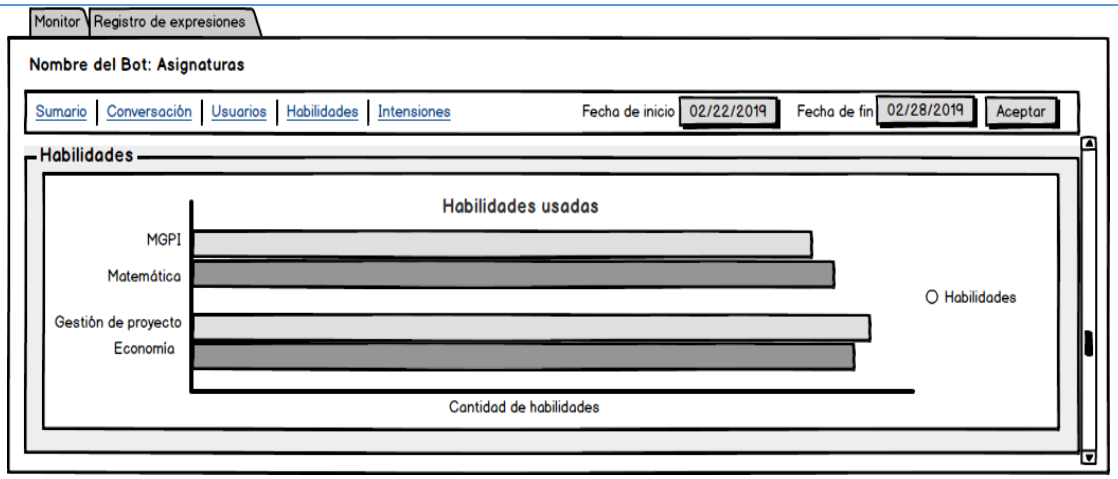
Anexo 33: Historia de usuario 31.

Historia de usuario	
Número: 31	**Usuario:** Administrador
Nombre de HU: Calcular estadísticas de los usuarios.	
Prioridad en negocio: Media	**Riesgo en desarrollo:** Medio
Tiempo estimado: 0.8	**Iteración Asignada:** 3

Programador responsable: Tania Socarras Navarro
Descripción: El sistema debe permitir calcular estadísticas de los usuarios que se conecten por día en el *bot* conversacional.
Observaciones: Se deberá elegir un período de fechas del que se quiere la información; de no elegir uno se tomará por defecto el último período de fechas.
Prototipo de Interfaz: 

Anexo 34: Historia de usuario 32.

Historia de usuario	
Número: 32	**Usuario:** Administrador
Nombre de HU: Calcular estadísticas de las habilidades.	
Prioridad en negocio: Media	**Riesgo en desarrollo:** Medio
Tiempo estimado: 0.8	**Iteración Asignada:** 3
Programador responsable: Tania Socarras Navarro	
Descripción: El sistema debe permitir calcular las habilidades usadas en una conversación con el *bot* conversacional por día de un período de fecha previamente escogido.	
Observaciones: Se deberá elegir un período de fechas del que se quiere la información; de no elegir uno se tomará por defecto el último período de fechas.	
Prototipo de Interfaz:	



Anexo 35: Historia de usuario 33.

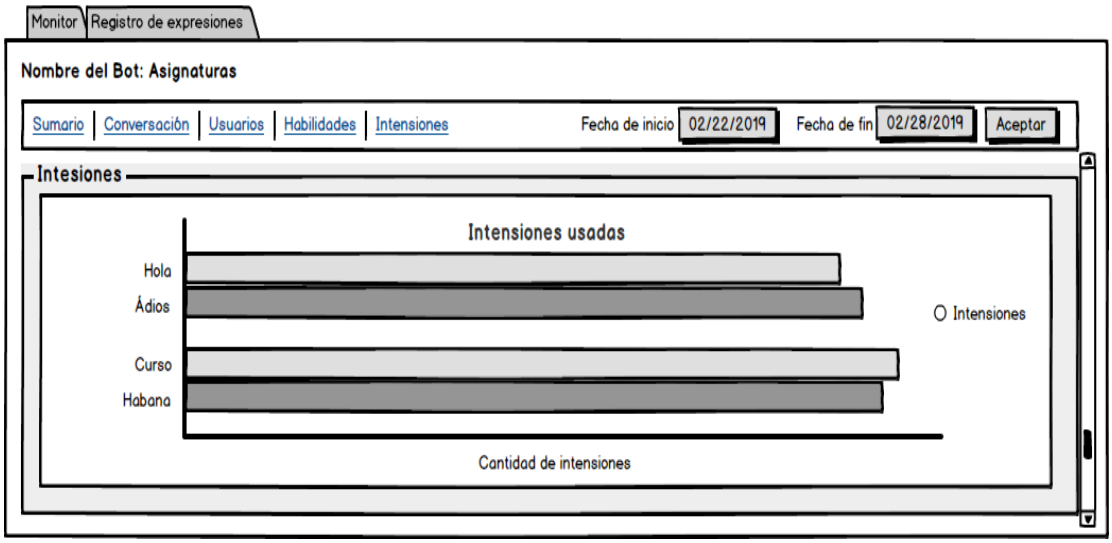
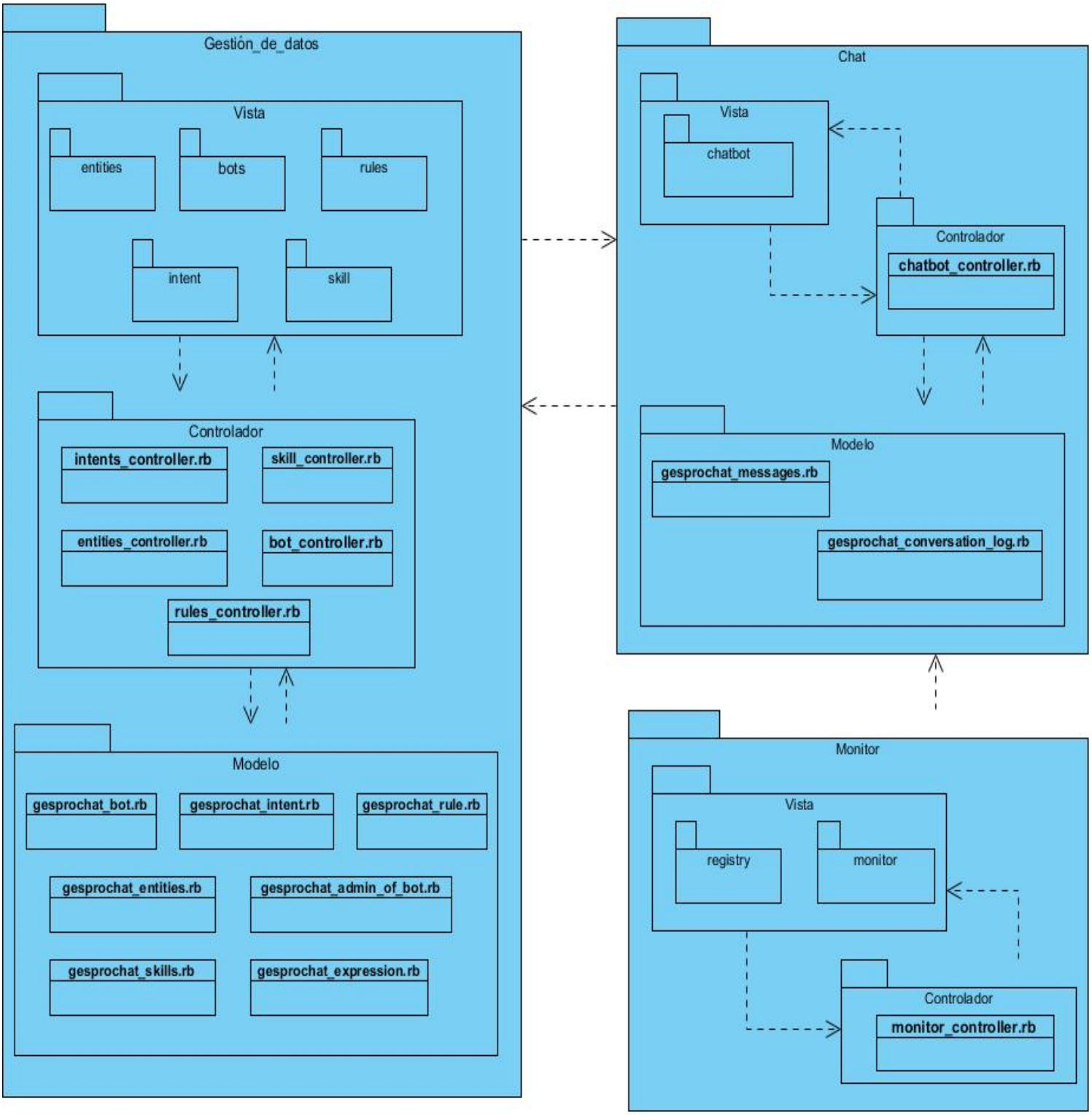
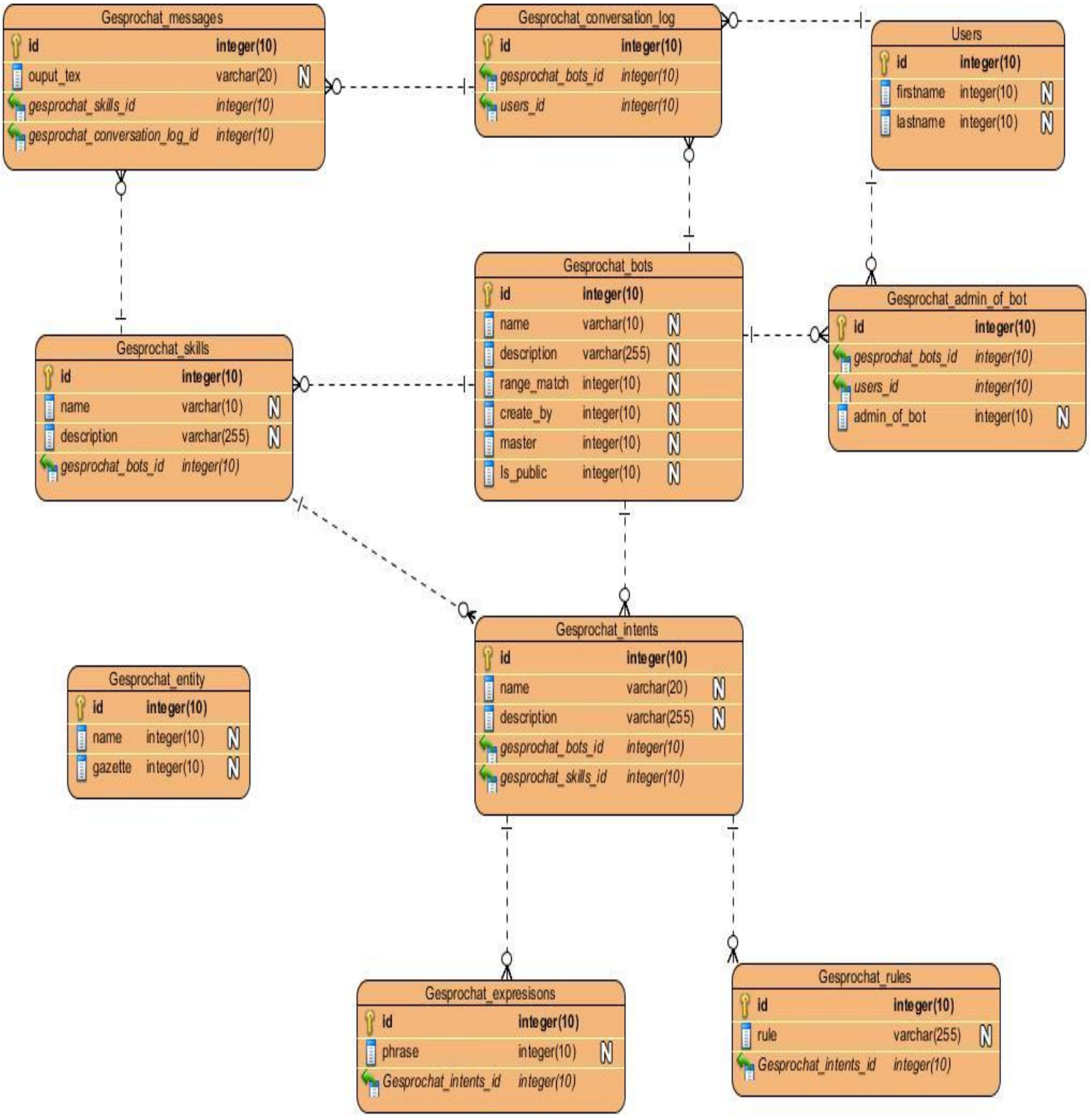
Historia de usuario	
Número: 33	**Usuario: Administrador**
Nombre de HU: Calcular estadísticas de las intenciones.	
Prioridad en negocio: Media	**Riesgo en desarrollo:** Media
Tiempo estimado: 1	**Iteración Asignada:** 3
Programador responsable: Tania Socarras Navarro	
Descripción: El sistema debe permitir calcular las intenciones empleadas en una conversación con el *bot* conversacional por día en el período de fecha elegido.	
Observaciones: Se deberá elegir un período de fechas del que se quiere la información; de no elegir uno se tomará por defecto el último período de fechas.	
Prototipo de Interfaz: 	

Diagrama de paquete del sistema:



Anexo 36: Diagrama de paquete del sistema.

Diagrama de modelo de datos:



Anexo 37: Modelo de datos.

Tarjetas CRC:

Anexo 38: Tarjeta CRC 2.

Tarjeta CRC	
Clase: *GesprochatSkills*.	
Responsabilidad	**Colaboración**
Representar los datos persistentes en la base de datos, correspondientes a las habilidades de los *bots* conversacionales. Siendo una clase Modelo o conocida también como Entidad.	✓ *IntentsController.* ✓ *MonitorController.* ✓ *SkillController.*

Anexo 39: Tarjeta CRC 3.

Tarjeta CRC	
Clase: *GesprochatIntents*.	
Responsabilidad	**Colaboración**
Representar los datos persistentes en la base de datos, correspondiste a las intenciones de las habilidades. Siendo una clase Modelo o conocida también como Entidad.	✓ *ChatbotController.* ✓ *IntentsController.* ✓ *MonitorController.* ✓ *SkillController.* ✓ *RulesController.*

Anexo 40: Tarjeta CRC 4.

Tarjeta CRC	
Clase: *GesprochatExpression.*	
Responsabilidad	**Colaboración**
Representar los datos persistentes en la base de datos, correspondiste a las expresiones de las intenciones. Siendo una clase Modelo o conocida también como Entidad.	✓ *IntentsController.* ✓ *MonitorController.*

Anexo 41: Tarjeta CRC 5.

Tarjeta CRC	
Clase: *GesprochatAdminOfBot*.	
Responsabilidad	**Colaboración**
Representar los datos persistentes en la base de datos, correspondiste a los usuarios que tienen *bots* conversacionales asociados. Siendo una clase Modelo o conocida también como Entidad.	✓ *BotController*.

Anexo 42: Tarjeta CRC 6.

Tarjeta CRC	
Clase: *GesprochatEntities*.	
Responsabilidad	**Colaboración**
Representar los datos persistentes en la base de datos, correspondiste a las entidades. Siendo una clase Modelo o conocida también como Entidad.	✓ *EntitiesController*.

Anexo 43: Tarjeta CRC 7.

Tarjeta CRC	
Clase: *GesprochatRule*.	
Responsabilidad	**Colaboración**
Representar los datos persistentes en la base de datos, correspondiste a las reglas. Siendo una clase Modelo o conocida también como Entidad.	✓ *RuleController*.

Anexo 44: Tarjeta CRC 9.

Tarjeta CRC	
Clase: *GesprochatMessages*.	
Responsabilidad	**Colaboración**
Representar los datos persistentes en la base de datos, correspondiste a los mensajes que se tienen entre el usuario y *bots* conversacionales. Siendo una clase Modelo o conocida también como Entidad.	✓ *MonitorController*.

Anexo 45: Tarjeta CRC 12.

Tarjeta CRC	
Clase: *SkillController*.	
Responsabilidad	**Colaboración**
Es la clase encargada de gestionar los datos correspondientes a las habilidades del sistema. • *create ():* se crea una nueva habilidad. • *update ():* se modifica una habilidad. • *destroy ():* se elimina una habilidad.	✓ *GesprochatSkills.* ✓ *GesprochatBot.*

Anexo 46: Tarjeta CRC 13.

Tarjeta CRC	
Clase: *IntentsController*.	
Responsabilidad	**Colaboración**

Es la clase encargada de gestionar los datos correspondientes a las intenciones del sistema y de las expresiones. • *create ():* se crea una nueva intención. • *update ():* se modifica una intención. • *destroy ():* se elimina una intención. • *create_expression ():* se crea una nueva expresión. • *update_expression ():* se modifica una expresión. • *destraoy_expresion ():* se elimina una expresión.	✓ *GesprochatIntents.* ✓ *GesprochatSkill.* ✓ *GesprochatBot.* ✓ *GesprochatExpression.*

Anexo 47: Tarjeta CRC 14.

Tarjeta CRC	
Clase: *ChatbotController*.	
Responsabilidad	**Colaboración**
Es la clase encargada de gestionar los datos correspondientes a las conversaciones realizadas entre el usuario y los *bots* conversacionales del sistema mediante una interfaz para *chat*. • *load_chat ():* es la encargada de mostrar la interfaz de *chat* que utilizara el usuario para interactuar con el *bot* conversacional. • *messages_algorith ():* procesa los mensajes enviados por el usuario y brindar una respuesta a estos.	✓ *GesprochatBot.* ✓ *GesprochatIntents.* ✓ *GesprochatEntities*.

Anexo 48: Tarjeta CRC 15.

Tarjeta CRC	
Clase: *EntitiesController*	
Responsabilidad	**Colaboración**
Es la clase encargada de gestionar los datos correspondientes a las entidades del sistema. • *buil ():* muestra las entidades creadas. • *create ():* crea una nueva entidad. • *update ():* se modifica una intención. • *destroy ():* se elimina una intención.	✓ *GesprochatEntities*.

Tareas de ingenierías de la iteración I:

Anexo 49: Tarea de Ingeniería 3 de la Iteración I.

Tarea de ingeniería	
Número de tarea: 3	**HU:** 3

Nombre de la tarea: Desarrollar vista para listar *Bots* conversacionales.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 0.2
Fecha de inicio: 20 de diciembre de 2018	**Fecha de fin:** 20 de diciembre de 2018
Programador responsable: Ariel Acevedo Guevara	
Descripción: Se desarrolla una vista que permita listar todos los *bots* conversacionales creados.	

Anexo 50: Tarea de ingeniería 4 de la iteración I.

Tarea de ingeniería	
Número de tarea: 4	**HU:** 4
Nombre de la tarea: Implementar la eliminación *bots* conversacionales.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 0.1
Fecha de inicio: 21 de diciembre de 2018	**Fecha de fin:** 21 de diciembre de 2018
Programador responsable: Ariel Acevedo Guevara	
Descripción: Se implementa una funcionalidad que permita eliminar un *bots* conversacionales.	

Anexo 51: Tarea de ingeniería 5 de la iteración I.

Tarea de ingeniería	
Número de tarea: 5	**HU:** 4
Nombre de la tarea: Implementar la notificación eliminación *bots* conversacionales.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 0.1
Fecha de inicio: 21 de diciembre de 2018	**Fecha de fin:** 21 de diciembre de 2018
Programador responsable: Ariel Acevedo Guevara	
Descripción: Se implementa una funcionalidad que permita mostrar un mensaje para reafirmar si se quiere eliminar el *bots* conversacionales.	

Anexo 52: Tarea de ingeniería 6 de la iteración I.

Tarea de ingeniería	
Número de tarea: 6	**HU:** 5
Nombre de la tarea: Desarrollar vista para crear habilidades.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 0.4
Fecha de inicio: 24 de diciembre de 2019	**Fecha de fin:** 26 de diciembre de 2019
Programador responsable: Ariel Acevedo Guevara	
Descripción: Se desarrolla una vista que permita crear habilidades definiendo de cada una los siguientes campos Nombre y Descripción.	

Anexo 53: Tarea de ingeniería 7 de la iteración I.

Tarea de ingeniería	
Número de tarea: 7	**HU:** 6
Nombre de la tarea: Desarrollar vista para editar habilidades.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 0.2
Fecha de inicio: 27 de diciembre de 2019	**Fecha de fin:** 27 de diciembre de 2019
Programador responsable: Ariel Acevedo Guevara	
Descripción: Se desarrolla una vista que permita editar habilidades definiendo de cada una los siguientes campos Nombre y Descripción.	

Anexo 54: Tarea de ingeniería 8 de la iteración I.

Tarea de ingeniería	
Número de tarea: 8	**HU:** 7
Nombre de la tarea: Desarrollar vista para listar habilidades.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 0.2
Fecha de inicio: 28 de diciembre de 2019	**Fecha de fin:** 28 de diciembre de 2019
Programador responsable: Ariel Acevedo Guevara	
Descripción: Se desarrolla una vista que permita listar todas las habilidades creadas.	

Anexo 55: Tarea de ingeniería 9 de la iteración I.

Tarea de ingeniería	
Número de tarea: 9	**HU:** 8
Nombre de la tarea: Implementar la eliminación habilidades.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 0.1
Fecha de inicio: 3 de enero de 2019	**Fecha de fin:** 3 de enero de 2019
Programador responsable: Ariel Acevedo Guevara	
Descripción: Se implementa una funcionalidad que permita eliminar una habilidad.	

Anexo 56: Tarea de ingeniería 10 de la iteración I.

Tarea de ingeniería	
Número de tarea: 10	**HU:** 8
Nombre de la tarea: Implementar la notificación eliminación habilidad.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 0.1
Fecha de inicio: 3 de enero de 2019	**Fecha de fin:** 3 de enero de 2019
Programador responsable: Ariel Acevedo Guevara	
Descripción: Se implementa una funcionalidad que permita mostrar un mensaje para reafirmar si se quiere eliminar la habilidad.	

Anexo 57: Tarea de ingeniería 11 de la iteración I.

Tarea de ingeniería	
Número de tarea: 11	**HU:** 9
Nombre de la tarea: Desarrollar vista para crear intención.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 0.4
Fecha de inicio: 4 de enero de 2018	**Fecha de fin:** 7 de enero de 2018
Programador responsable: Ariel Acevedo Guevara	
Descripción: Se desarrolla una vista que permita crear intención definiendo de cada uno los siguientes campos Nombre y Descripción.	

Anexo 58: Tarea de ingeniería 12 de la iteración I.

Tarea de ingeniería	
Número de tarea: 12	**HU:** 10
Nombre de la tarea: Desarrollar vista para editar intención.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 0.2
Fecha de inicio: 8 de enero de 2018	**Fecha de fin:** 8 de enero de 2018
Programador responsable: Ariel Acevedo Guevara	
Descripción: Se desarrolla una vista que permita editar intención definiendo de cada una los siguientes campos Nombre y Descripción.	

Anexo 59: Tarea de ingeniería 13 de la iteración I.

Tarea de ingeniería	
Número de tarea: 13	**HU:** 11
Nombre de la tarea: Desarrollar vista para listar intención.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 0.2
Fecha de inicio: 9 de enero de 2018	**Fecha de fin:** 9 de enero de 2018
Programador responsable: Ariel Acevedo Guevara	
Descripción: Se desarrolla una vista que permita listar todas las intenciones creadas.	

Anexo 60: Tarea de ingeniería 14 de la iteración I.

Tarea de ingeniería	
Número de tarea: 14	**HU:** 12
Nombre de la tarea: Implementar la eliminación intención.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 0.1
Fecha de inicio: 10 de enero de 2019	**Fecha de fin:** 10 de enero de 2019
Programador responsable: Ariel Acevedo Guevara	
Descripción: Se implementa una funcionalidad que permita eliminar una intención.	

Anexo 61: Tarea de ingeniería 15 de la iteración I.

Tarea de ingeniería	
Número de tarea: 15	**HU:** 12
Nombre de la tarea: Implementar la notificación eliminación intención.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 0.1
Fecha de inicio: 10 de enero de 2019	**Fecha de fin:** 10 de enero de 2019
Programador responsable: Ariel Acevedo Guevara	
Descripción: Se implementa una funcionalidad que permita mostrar un mensaje para reafirmar si se quiere eliminar la intención.	

Anexo 62: Tarea de ingeniería 16 de la iteración I.

Tarea de ingeniería	
Número de tarea: 16	**HU:** 13
Nombre de la tarea: Desarrollar vista para crear expresión.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 0.4
Fecha de inicio: 11 de enero de 2019	**Fecha de fin:** 14 de enero de 2019
Programador responsable: Ariel Acevedo Guevara	
Descripción: Se desarrolla una vista que permita crear expresión.	

Anexo 63: Tarea de ingeniería 17 de la iteración I.

Tarea de ingeniería	
Número de tarea: 17	**HU:** 14
Nombre de la tarea: Desarrollar vista para listar expresión.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 0.2
Fecha de inicio: 15 de enero de 2019	**Fecha de fin:** 15 de enero de 2019
Programador responsable: Ariel Acevedo Guevara	
Descripción: Se desarrolla una vista que permita listar todas las expresiones creadas.	

Anexo 64: Tarea de ingeniería 18 de la iteración I.

Tarea de ingeniería	
Número de tarea: 18	**HU:** 15
Nombre de la tarea: Implementar la eliminación expresión.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 0.1
Fecha de inicio: 16 de enero de 2019	**Fecha de fin:** 16 de enero de 2019
Programador responsable: Ariel Acevedo Guevara	
Descripción: Se implementa una funcionalidad que permita eliminar una expresión.	

Anexo 65: Tarea de ingeniería 19 de la iteración I.

Tarea de ingeniería	
Número de tarea: 19	**HU:** 15
Nombre de la tarea: Implementar la notificación eliminación expresión.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 0.1
Fecha de inicio: 16 de enero de 2019	**Fecha de fin:** 16 de enero de 2019
Programador responsable: Ariel Acevedo Guevara	
Descripción: Se implementa una funcionalidad que permita mostrar un mensaje para reafirmar si se quiere eliminar la expresión.	

Anexo 66: Tarea de ingeniería 20 de la iteración I.

Tarea de ingeniería	
Número de tarea: 20	**HU:** 16
Nombre de la tarea: Desarrollar vista para crear entidades.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 0.4
Fecha de inicio: 17 de enero de 2019	**Fecha de fin:** 18 de enero de 2019
Programador responsable: Ariel Acevedo Guevara	
Descripción: Se desarrolla una vista que permita crear entidades definiendo de cada una los siguientes campos Nombre y Palabra. El campo palabra puede ser creado dinámicamente.	

Anexo 67: Tarea de ingeniería 21 de la iteración I.

Tarea de ingeniería	
Número de tarea: 21	**HU:** 17
Nombre de la tarea: Desarrollar vista para editar entidades.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 0.2
Fecha de inicio: 21 de enero de 2019	**Fecha de fin:** 21 de enero de 2019
Programador responsable: Ariel Acevedo Guevara	
Descripción: Se desarrolla una vista que permita editar entidades definiendo de cada una los siguientes campos Nombre y Palabra.	

Anexo 68: Tarea de ingeniería 21 de la iteración I.

Tarea de ingeniería	
Número de tarea: 21	**HU:** 18
Nombre de la tarea: Desarrollar vista para listar entidades.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 0.2
Fecha de inicio: 22 de enero de 2019	**Fecha de fin:** 22 de enero de 2019

Programador responsable: Ariel Acevedo Guevara
Descripción: Se desarrolla una vista que permita listar todas las entidades creadas.

Anexo 69: Tarea de ingeniería 22 de la iteración I.

Tarea de ingeniería	
Número de tarea: 22	**HU:** 19
Nombre de la tarea: Implementar la eliminación entidades.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 0.1
Fecha de inicio: 23 de enero de 2019	**Fecha de fin:** 23 de enero de 2019
Programador responsable: Ariel Acevedo Guevara	
Descripción: Se implementa una funcionalidad que permita eliminar una entidad.	

Anexo 70: Tarea de ingeniería 23 de la iteración I.

Tarea de ingeniería	
Número de tarea: 23	**HU:** 19
Nombre de la tarea: Implementar la notificación eliminación entidad.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 0.1
Fecha de inicio: 23 de enero de 2019	**Fecha de fin:** 23 de enero de 2019
Programador responsable: Ariel Acevedo Guevara	
Descripción: Se implementa una funcionalidad que permita mostrar un mensaje para reafirmar si se quiere eliminar la entidad.	

Anexo 71: Tarea de ingeniería 24 de la iteración I.

Tarea de ingeniería	
Numero de tarea: 24	**HU:** 20
Nombre de la tarea: Desarrollar vista para mostrar los detalles de un *bot* conversacional.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 0.2
Fecha de inicio: 24 de enero de 2019	**Fecha de fin:** 24 de enero de 2019
Programador responsable: Ariel Acevedo Guevara	
Descripción: Se desarrolla una vista que muestre todos los campos del *bot* conversacional con sus respectivos valores asignados.	

Anexo 72: Tarea de ingeniería 25 de la iteración I.

Tarea de ingeniería	
Numero de tarea: 25	**HU:** 21
Nombre de la tarea: Implementar la funcionalidad que permita asignar usuarios a un *bot* conversacional.	

Tipo de tarea: Desarrollo	**Puntos estimados:** 0.4
Fecha de inicio: 25 de enero de 2019	**Fecha de fin:** 28 de febrero de 2019
Programador responsable: Ariel Acevedo Guevara	
Descripción: Se desarrolla una funcionalidad que permite asignar usuarios del sistema a un *bot* conversacional.	

Anexo 73: Tarea de ingeniería 26 de la iteración I.

Tarea de ingeniería	
Numero de tarea: 26	**HU:** 22
Nombre de la tarea: Implementar la funcionalidad que permita eliminar usuarios asignados a un *bot* conversacional.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 0.1
Fecha de inicio: 29 de enero de 2019	**Fecha de fin:** 29 de enero de 2019
Programador responsable: Ariel Acevedo Guevara	
Descripción: Se desarrolla una funcionalidad que permite eliminar los usuarios que estén asignados un *bot* conversacional.	

Anexo 74: Tarea de ingeniería 27 de la iteración I.

Tarea de ingeniería	
Numero de tarea: 27	**HU:** 22
Nombre de la tarea: Implementar la notificación de eliminación de usuarios asignados a un *bot* conversacional.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 0.1
Fecha de inicio: 29 de enero de 2019	**Fecha de fin:** 29 de enero de 2019
Programador responsable: Ariel Acevedo Guevara	
Descripción: Se implementa una funcionalidad que permita mostrar un mensaje para reafirmar si se quiere eliminar un usuario asignado a un *bot* conversacional.	

Anexo 75: Tarea de ingeniería 28 de la iteración I.

Tarea de ingeniería	
Numero de tarea: 28	**HU:** 23
Nombre de la tarea: Desarrollar una vista que permita mostrar un listado con todos los *bots* creados por los usuarios del sistema.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 0.2
Fecha de inicio: 30 de enero de 2019	**Fecha de fin:** 30 de enero de 2019
Programador responsable: Ariel Acevedo Guevara	

Descripción: Se desarrolla una vista que permitirá mostrar un listado con todos los *bots* creados por los usuarios del sistema.

Tareas de ingenierías de la iteración II:

Anexo 76: Tarea de ingeniería 1 de la iteración II.

Tarea de ingeniería	
Numero de tarea: 1	**HU:** 24
Nombre de la tarea: Desarrollar una vista para visualizar una interfaz de mensajería para cada *bot* conversacional público creado en el sistema.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 1
Fecha de inicio: 31 de enero de 2019	**Fecha de fin:** 6 de febrero de 2019
Programador responsable: Ariel Acevedo Guevara	
Descripción: Se desarrolla una vista que permitirá mostrar una interfaz de mensajería para cada *bot* conversacional público creado en el sistema.	

Anexo 77: Tarea de ingeniería 4 de la iteración II.

Tarea de ingeniería	
Número de tarea: 4	**HU:** 27
Nombre de la tarea: Desarrollar vista para crear reglas.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 1.2
Fecha de inicio: 7 de marzo de 2019	**Fecha de fin:** 14 de marzo de 2019
Programador responsable: Ariel Acevedo Guevara	
Descripción: Se desarrolla una vista que permita crear entidades definiendo de cada una los siguientes campos Nombre y Palabra.	

Anexo 78: Tarea de ingeniería 5 de la iteración II.

Tarea de ingeniería	
Número de tarea: 5	**HU:** 28
Nombre de la tarea: Desarrollar vista para listar regla.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 0.2
Fecha de inicio: 15 de marzo de 2019	**Fecha de fin:** 15 de marzo de 2019
Programador responsable: Ariel Acevedo Guevara	
Descripción: Se desarrolla una vista que permita listar todas las reglas creadas.	

Tareas de ingenierías de la iteración III:

Anexo 79: Tarea de ingeniería 3 de la iteración III.

Tarea de ingeniería	
Número de tarea: 3	**HU:** 30
Nombre de la tarea: Desarrollar vista para generar la gráfica de Conversación.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 1
Fecha de inicio: 25 de marzo de 2019	**Fecha de fin:** 29 de marzo de 2019
Programador responsable: Tania Socarras Navarro	
Descripción: Se desarrolla una vista para generar un gráfico de línea de las conversaciones que se tienen con el *bot* conversacional.	

Anexo 80: Tarea de ingeniería 3 de la iteración III.

Tarea de ingeniería	
Número de tarea: 4	**HU:** 31
Nombre de la tarea: Desarrollar vista para generar la gráfica de Usuarios.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 0.8
Fecha de inicio: 1 de abril de 2019	**Fecha de fin:** 4 de abril de 2019
Programador responsable: Tania Socarras Navarro	
Descripción: Se desarrolla una vista para generar un gráfico de línea de los usuarios que se conectaron en el *bot* conversacional.	

Anexo 81: Tarea de ingeniería 5 de la iteración III.

Tarea de ingeniería	
Número de tarea: 5	**HU:** 32
Nombre de la tarea: Desarrollar vista para generar la gráfica de barra horizontal de Habilidades.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 0.8
Fecha de inicio: 5 de abril de 2019	**Fecha de fin:** 10 de abril de 2019
Programador responsable: Tania Socarras Navarro	
Descripción: Se desarrolla una vista para generar un gráfico de barra horizontal de habilidades que se usaron en el *bot*	

Anexo 82: Tarea de ingeniería 6 de la iteración III.

Tarea de ingeniería	
Número de tarea: 6	**HU:** 33
Nombre de la tarea: Desarrollar vista para generar la gráfica de Intenciones.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 1
Fecha de inicio: 11 de abril de 2019	**Fecha de fin:** 24 de abril de 2019

Programador responsable: Tania Socarras Navarro
Descripción: Se desarrolla una vista para generar un gráfico de barra horizontal de intenciones que se usaron en el *bot* conversacional.

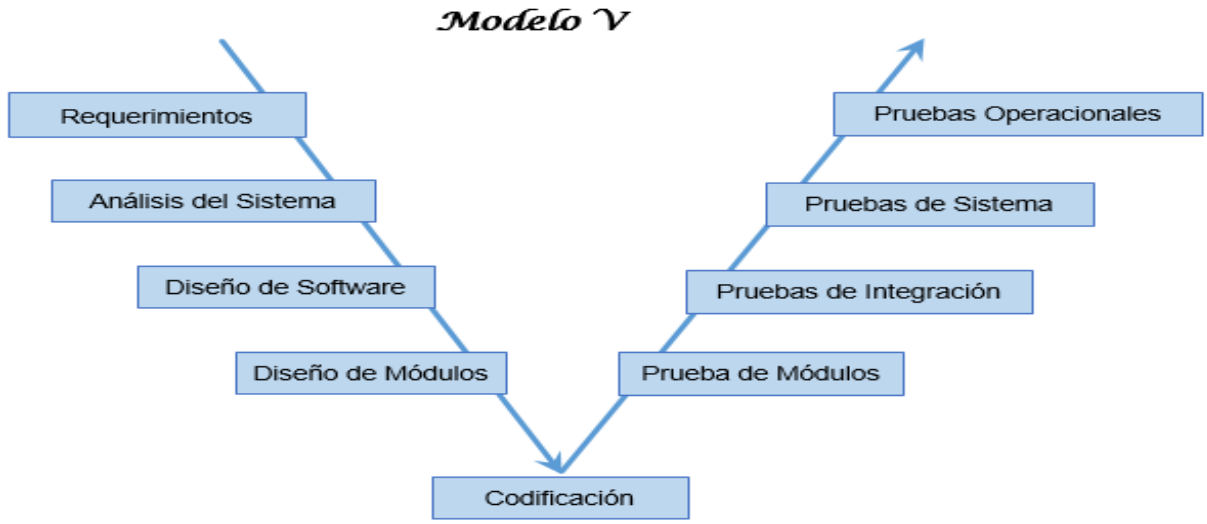
Anexo 83: Tarea de ingeniería 8 de la iteración III.

Tarea de ingeniería	
Número de tarea: 8	**HU:** 34
Nombre de la tarea: Implementar filtro de los mensajes por intenciones.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 0.8
Fecha de inicio: 27 de abril de 2019	**Fecha de fin:** 30 de abril de 2019
Programador responsable: Tania Socarras Navarro	
Descripción: Se implementa una funcionalidad que permita filtrar las expresiones que tengan o no intenciones asociadas.	

Anexo 84: Tarea de ingeniería 9 de la iteración III.

Tarea de ingeniería	
Número de tarea: 9	**HU:** 34
Nombre de la tarea: Implementar la eliminación de mensajes del registro de expresiones.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 0.2
Fecha de inicio: 3 de mayo de 2019	**Fecha de fin:** 3 de mayo de 2019
Programador responsable: Tania Socarras Navarro	
Descripción: Se implementa una funcionalidad que permita eliminar un mensaje del registro de expresiones.	

Modelo V:



Anexo 85: Modelo V. Fuente: (Pérez, 2013).

Encuesta de satisfacción:

Lee con cuidado cada pregunta antes de responder. Te agradecemos tu participación y franqueza al decimos lo que piensas de la plataforma para la creación de sistemas conversacionales que se integrará a la Suite XEDRO-GESPRO.

1. ¿Considera usted que este nuevo módulo es útil para la accesibilidad de la información de los procesos que se gestionan en la *suite* XEDRO-GESPRO?

   ____ Sí ____ No ____ No sé

2. ¿Encuentras difícil la interacción con los sistemas conversacionales que ofrece la plataforma?

   ____ Sí ____ No ____ No sé

3. Luego de haber interactuado con la plataforma, a su consideración ¿Cuáles son las deficiencias que esta presentó en su interacción?

   _____

   _____

   _____

4. A su consideración, ¿Qué otras formas de respuesta que no sea a través de mensajes de textos considera que deba ofrecer la plataforma para darle respuestas a sus peticiones?

   _____

   _____

   _____

5. ¿Se siente satisfecho con la información que obtuvo al interactuar con el *chatbot Adán Bot*?

   ___ Me satisface mucho.

   ___ No me satisface tanto.

   ___ Me da lo mismo.

   ___ Me disgusta más de lo que me satisface.

   ___ No me satisface nada.

   ___ No sé qué decir.

Anexo 86: Encuesta de satisfacción.

Anexo 87: Cuadro lógico de Iadov. Fuente: Elaboración propia.

	1. ¿Siente usted que este nuevo módulo es útil para la accesibilidad de la información de los procesos que se gestionan en la *suite* XEDRO-GESPRO?								
5. ¿Se siente satisfecho con la información que obtuvo al interactuar con el *chatbot Adán Bot*?	No			No sé			Sí		
	2. ¿Encuentras difícil la interacción con los sistemas conversacionales que ofrece la plataforma?								
	Sí	No sé	No	Sí	No sé	No	Sí	No sé	No
Me satisface mucho.	1	2	6	2	2	6	6	6	6
No me satisface tanto.	2	2	3	2	3	3	6	3	6
Me da lo mismo.	3	3	3	3	3	3	3	3	3
Me disgusta más de lo que me satisface.	6	3	6	3	4	4	3	4	4
No me satisface nada.	6	6	6	6	4	4	6	4	5
No sé qué decir.	2	3	6	3	3	3	6	3	4