UNIVERSIDAD POLITÉCNICA SALESIANA

SEDE QUITO

CARRERA:

INGENIERÍA DE SISTEMAS

Trabajo de titulación previo a la obtención del título de:

Ingenieros de Sistemas

TEMA:

DESARROLLO DE UN PROTOTIPO DE ASISTENTE VIRTUAL PARA LA GESTIÓN DEL CONOCIMIENTO DE UNA ORGANIZACIÓN.

AUTORES:

CHRISTIAN PAUL ABATA QUINCHUQUI

PEDRO DAVID RAMÍREZ SORIA

TUTOR:

FRANKLIN EDMUNDO HURTADO LARREA

Quito, agosto del 2018

CESIÓN DE DERECHOS DE AUTOR

Nosotros, CHRISTIAN PAUL ABATA QUINCHUQUI, con documento de identificación N° 1720339041, y PEDRO DAVID RAMÍREZ SORIA, con documento de identificación N° 1724248032, manifestamos nuestra voluntad y cedemos a la Universidad Politécnica Salesiana la titularidad sobre los patrimoniales en virtud de que somos autores del trabajo de titulación con el tema: "DESARROLLO DE UN PROTOTIPO DE ASISTENTE VIRTUAL PARA LA GESTIÓN DEL CONOCIMIENTO DE UNA ORGANIZACIÓN", el mismo que ha sido desarrollado para optar por el título de INGENIEROS DE SISTEMAS en la Universidad Politécnica Salesiana, quedando la Universidad facultada para ejercer plenamente los derechos cedidos anteriormente.

En la aplicación a lo determinado en la Ley de Propiedad Intelectual, en nuestra condición de autores nos reservamos los derechos morales de la obra antes citada. En concordancia, suscribimos este documento en el momento que hacemos entrega del trabajo final en formato impreso y digital a la Biblioteca de la Universidad Politécnica Salesiana.

CHRISTIAN PAUL	PEDRO DAVID
ABATA QUINCHUQUI	RAMÍREZ SORIA
CI: 1720339041	CI: 1724248032

Quito, agosto del 2018

DECLARATORIA DE COAUTORÍA DEL TUTOR

Yo declaro que bajo mi dirección y asesoría fue desarrollado el Proyecto Técnico, con el tema: "DESARROLLO DE UN PROTOTIPO DE ASISTENTE VIRTUAL PARA LA GESTIÓN DEL CONOCIMIENTO DE UNA ORGANIZACIÓN" realizado por los estudiantes CHRISTIAN PAUL ABATA QUINCHUQUI y PEDRO DAVID RAMÍREZ SORIA, obteniendo un producto que cumple con todos los requisitos estipulados por la Universidad Politécnica Salesiana, para ser consideradas trabajo final de titulación.

Quito, agosto del 2018

.....

FRANKLIN EDMUNDO HURTADO LARREA

CI: 1720339041

Dedicatoria

El tiempo, esfuerzo y sacrificio requerido para este trabajo se lo dedico a mi madre, su ejemplo ha sido la guía necesaria para alcanzar todas mis metas, desde luego incluyendo esta experiencia.

A mis hermanas, quienes dispusieron su tiempo para tenderme una mano de apoyo y comprensión, a mi pareja quién fue parte de este proceso el cual nunca me dejó descuidar, a mi tutor el que con sus sabios consejos me supo apoyar y encaminar al éxito, a Dios desde luego por cuidarme y darme salud y a mi padre que supo inculcarme los valores necesarios para formarme como hombre.

Christian Paul Abata Quinchuqui

A mis padres, Willian y Teresa, así como a mi hermano Ricardo, por ser ejemplos de perseverancia y constancia, por el apoyo incondicional, la guía, los consejos y la motivación constante que siempre me ha permitido crecer personal y profesionalmente. Pero sobre todo por su amor.

A ese ser supremo dador de vida, que día a día ilumina mi existir encaminándome al hombre y al profesional en el que mañana me convertiré.

Pedro David Ramírez Soria

Agradecimiento

En primer lugar, a nuestros padres por apoyarnos en el camino de esta carrera, brindándonos sus sabios consejos y su amor desprendido, extendiendo sus manos generosas suplieron nuestras necesidades físicas, morales y espirituales para que finalmente culminemos con éxito este período y a la vez empecemos una nueva etapa en nuestras vidas.

A nuestras familias, que siempre estuvieron presentes y pendientes en este arduo camino, porque con sus palabras de aliento en los momentos difíciles fueron un pilar importante en este devenir.

A nuestro director de tesis, Ing. Franklin Hurtado, por su dedicación, amabilidad, comprensión y guía en todo aspecto académico, aportando a nuestro desarrollo personal, pues sin lugar a duda, sus palabras y consejos marcaron nuestro desempeño, por ello también tenemos el honor de considerarlo un amigo.

A nuestros profesores, que, a lo largo de esta trayectoria universitaria, entregaron sus valiosos conocimientos, convirtiéndose en puntales importantes para nuestra formación y para finalmente ser acreedores de nuevas experiencias y oportunidades.

A nuestros compañeros que, a lo largo de este viaje maravilloso, se convirtieron en buenos amigos y que en esta travesía se marcaron tantas experiencias halagüeñas.

Por último, pero el más importante, a Dios, pues sin su bendición nada de esto hubiese sido posible, guiando cada uno de nuestros pasos, para que nos convirtamos en seres humanos útiles, responsables y con altos valores.

Christian Paul Abata Quinchuqui

Pedro David Ramírez Soria

ÍNDICE

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

Resumen

El proyecto se enfoca en la creación de un prototipo de asistente virtual para la gestión del conocimiento empresarial, creado con tecnologías existentes, el cual busca apoyar a los usuarios en sus requerimientos de conocimiento procurando que la información presentada sea oportuna.

Para iniciar la construcción del prototipo, el primer paso fue proponer una arquitectura con la que las tecnologías fueron probadas y a partir de la cual se demostró la viabilidad de su desarrollo apegándose a metodología XP.

En la primera iteración se obtienen datos por medio de un API libre, integrando Bot Framework y Luis usando a Node Js como leguaje de programación, adicionalmente se investigan algoritmos de clasificación que servirán para poder discernir la información y conocimiento más importante para un usuario.

La segunda iteración inicia con la creación de un API personalizado con datos simulados, apegados a un escenario médico. Aunque la extracción de datos es correcta, existe un limitante apegado al nivel de madurez de la empresa por lo que se cambia la forma de consumir datos de un API a SQL Server, por otro lado, se encuentra SVM como un algoritmo alterno a los vistos en la primera iteración que supone ser el óptimo para la clasificación de términos.

En la tercera iteración se integran todos los componentes y se prueba el asistente con los métodos de aprendizaje propuestos en la segunda iteración, logrando como resultados la comunicación, extracción y aprendizajes exitosos, se finaliza subiendo el prototipo a un ambiente web.

Abstract

The project focuses on the creation of a virtual assistant prototype for the management of business knowledge, created with existing technologies, which seeks to support users in their knowledge requirements by ensuring that the information presented is timely.

To begin the construction of the prototype, the first step was to propose an architecture with which the technologies were tested and from which the viability of its development was demonstrated adhering to XP methodology.

In the first iteration data is obtained by means of a free API, integrating Bot Framework and Luis using Node Js as a programming language, additionally classification algorithms are investigated that will serve to discern the most important information and knowledge for a user.

The second iteration begins with the creation of a personalized API with simulated data, attached to a medical scenario. Although the extraction of data is correct, there is a limitation attached to the level of maturity of the company so it changes the way of consuming data from an API to SQL Server, on the other hand, is SVM as an alternative algorithm to those seen in the first iteration that supposes to be the optimum for the classification of terms.

In the third iteration, all the components are integrated and the assistant is tested with the learning methods proposed in the second iteration, achieving successful communication results, extraction and learning, and finally uploading the prototype to a web environment.

INTRODUCCIÓN

En la actualidad la mayoría de las empresas utilizan la información y conocimiento no estructurado para la toma de decisiones, la analizan empíricamente, esto dificulta formalizar un proceso y por ende compromete gravemente a la decisión tomada (Schneider, 2016). Una parte considerable del conocimiento reside en las personas en un nivel individual y colectivo, esto implica la posibilidad de una pérdida de conocimiento significativo de la empresa cuando uno o varios empleados salen de ella por cualquier motivo (despido, jubilación, renuncia, etc.); de tal manera que se perdería una continuidad sobre los activos de los procesos de la organización (Becerra Fernandez & Sabherwal, 2015).

Antecedentes

En una organización no necesariamente existe una figura que maneje todo el conocimiento y se encuentre disponible para los usuarios de forma inmediata, por ello, un usuario se ve obligado a encontrarla realizando preguntas en su entorno, El mismo puede ser muy variado ofreciendo un excelente ambiente laboral y bienestar emocional hasta uno en donde no existe una buena comunicación interna, existe falta de compromiso, baja motivación del equipo, escasos incentivos, mala planificación, etc. De tal manera, la comunicación interna puede quedar expuesta por los diversos factores mencionados, y de no ser efectiva eventualmente puede generar un sesgo de la información. Así el conocimiento recibido por un colaborador no será necesariamente el que buscaba, en consecuencia, se incrementa la posibilidad de cometer errores y no alcanzar los objetivos de la empresa.

Problema

Cuando un nuevo colaborador ingresa en la organización necesita una rápida familiarización con las atribuciones, procesos, procedimientos y responsabilidades de su

cargo. Siendo principal responsabilidad de la empresa auxiliar y brindar el conocimiento requerido por el nuevo integrante (Corral Mendivil, Gil Palomares, Velasco Cepeda, & Serrano Cornejo, 2011). Kyle Silvestro, CEO de SyTrue, dice "Una empresa que no maneje bien sus datos inevitablemente en un futuro carecerá de ideas". Aproximadamente el 80% de la información no se encuentra estructurada y al menos el 20% se debería estructurar para satisfacer las necesidades mínimas de presentación de información, para la toma de decisiones en una organización (SyTrue, 2015).

Finalmente, la presentación de la información y el conocimiento debe ser de fácil comprensión, con el fin de que una vez validado el conocimiento se integre a los procesos, sistemas, etc. (Pérez Rodríguez & Coutín Domínguez, 2005).

Justificación

Las organizaciones en la actualidad confrontan entornos cambiantes y dinámicos, en donde la información juega un papel relevante para la toma de decisiones “solucionar problemas y aprovechar oportunidades” (Rodríguez Cruz & Pinto, 2017). Así también la búsqueda de información y conocimiento sobre un proceso o varios procesos realizados dentro de una empresa puede resultar una actividad muy engorrosa, siendo necesaria existencia de una persona capacitada en el tema o en el mejor de los casos los dos, un documento actualizado que indique cómo se debe realizar y una persona. En resumen, el encontrar a esa persona o documento representa un tiempo considerable que afectaría negativamente en la productividad del equipo de trabajo y la organización por lo que el concentrar la información por medio de un canal de comunicación se puede considerar dentro de una de las soluciones para aportar con la productividad de la empresa.

Actualmente las Tecnologías de la Información (TI) presentan una amplia gama de oportunidades limitadas solo por la imaginación. De esta manera implementar IA para

realizar análisis de sentimiento, conversión de audio a texto (y viceversa), entre otros; es relativamente sencillo, en su mayor parte estas soluciones de IA pueden ser utilizadas como servicios, dejando de lado la preocupación del procesamiento computacional necesario para realizar dicha actividad. Sin embargo, para realizar una inferencia de los resultados obtenidos en una búsqueda interna o externa, se puede utilizan un modelo matemático más elaborado que deberá ser implementado en el ambiente de solución del módulo de interacción e inferencia.

En la era del conocimiento y la información el tener acceso a las tecnologías no es una tarea complicada y dentro de ellas, conceptos como movilidad, interconexión, elevados parámetros de calidad de imagen y sonido, diversidad, innovación, etc. (Cabero J, 2007) ayudan a una aplicación para diferenciarse de aplicaciones antiguamente creadas.

Objetivo General

Desarrollar un prototipo de asistente virtual que entregue conocimiento de una empresa de forma más organizada y precisa posible a un miembro de la misma.

Objetivos Específicos

Determinar en base a las investigaciones, aquellas tecnologías existentes que pueden contribuir con la elaboración de un prototipo de asistente virtual para la gestión del conocimiento empresarial.

Diseñar un prototipo de asistente virtual que posea una interfaz amigable para el usuario.

Implementar el prototipo en el que se pueda entregar de manera eficaz información y conocimiento de calidad a las consultas realizadas por el usuario, dentro de un ambiente de pruebas.

Realizar pruebas de las funcionalidades del prototipo en las que se pueda mostrar su eficacia al entregar la información solicitada por el usuario.

Marco Metodológico

De acuerdo con la metodología XP aplicada en este proyecto, existen características identificadas en el marco teórico las cuales han sido adaptadas, dadas las limitantes y necesidades presentadas, como por ejemplo la inexistencia de un cliente. XP habla sobre iteraciones que se deben trabajar junto al cliente, al iterar varias veces la única certeza existente es la creación de escenarios de prueba para validar el prototipo, éste debe ser probado por un usuario dentro del ámbito de cada escenario.

Otra característica de XP obedece al trabajo de elaboración del proyecto en el que la carga de trabajo se distribuye mediante perfiles como un encargado de pruebas, uno de seguimiento, un entrenador, un consultor y un gestor (López Yolanda, 2016). Para el caso de este proyecto todos esos roles se repartieron entre los dos integrantes.

Con respecto a la simplicidad, las pruebas, el desarrollo incremental iterativo y la corrección de errores. Estas características durante el desarrollo del proyecto, en cada una de las iteraciones, fueron reflejándose, evidenciando la correcta selección de XP como metodología.

Las historias de usuario para la aplicación de XP son instrumentos evidenciados desde la primera iteración como las reuniones, apuntes, fotos de propuestas arquitectónicas, sugerencias, etc.

Conforme el proyecto avanza XP se convierte en la metodología más efectiva al acoplarse a toda adaptación tal como se ha visto reflejado.

Finalmente, los diagramas de casos de uso de UML se implementan para describir la interacción entre el usuario y el prototipo de asistente virtual. Y BPMN se adopta para analizar la interacción que surge entre los componentes de software.

CAPÍTULO 1

Marco Teórico

1.1. Conocimiento

La Real Academia Española define conocimiento como la "noción, saber o noticia elemental de algo" (Real Academia Española, 2018), autores como Turri, Walton y Zhang lo expresan como una creencia que debe ser validada y justificada (Valladares & Olivé, 2015). En el contexto empresarial, el conocimiento, se produce cuando un individuo utiliza la información que dispone y lo que sabe para resolver un problema (Almuiñas Rivero, Passailaigue Baquerizo, & Galarza López, 2015), de esta manera se lo clasifica en declarativo (saber el qué), procedimental (saber hacer) y aptitudinal (saber ser) (Latorre Ariño, 2017). Zapata, Villada y Arias abordan al conocimiento como una síntesis de información aplicada en escenarios reales, considerando a la información como un conjunto de datos procesado para tomar decisiones (Zapata Restrepo, Villada Arango, & Arias Chaverra, 2017). En definitiva, se entiende por conocimiento al "hecho de ser consciente " (Rodríguez Rodríguez, Hernández Cabrera, & Plácido Castro, 2006).

1.1.1. Gestión del conocimiento en una empresa

La información en una empresa está destinada a resolver problemas específicos altamente vinculados con el negocio, por ello es indispensable su disponibilidad, accesibilidad, fácil comprensión, precisión, relevancia y flexibilidad; sirviendo para el

desarrollo de cada miembro y de toda la organización (Pérez Rodríguez & Coutín Domínguez, 2005).

Michael Porter, en su teoría Ventaja Competitiva Sostenible, subraya la importancia del conocimiento como un recurso estratégico (Palacios, 2008). Así también, Yadira Nieves y Magda León exponen que "el conocimiento del que disponen las personas en la organización y su recopilación no sólo genera valor económico para la empresa, sino que actualmente es el activo para marcar la diferencia" (Nieves Lahaba & León Santos, 2001). Según Senge "aprender en las organizaciones significa estar sometido de forma continua a la experiencia y transformar esa experiencia en un conocimiento accesible para toda la organización y pertinente a su propósito central" (CIDEC, 2004).

Nofal Nagles expone "la gestión de conocimiento es un proceso, lógico, organizado y sistemático para producir, transferir y aplicar en situaciones concretas" (Nagles G., 2007), sin embargo, generalmente el conocimiento de una organización no posee una estructura que facilite su compresión y peor aún su utilización efectiva. Por ello Nagles propone un modelo gestión del conocimiento en la organización, que distingue las barreras afrontadas en su integración como la dispersión del conocimiento, su proceso de transferencia y sus elementos, y finalmente las acciones para gestión efectiva como la organización y distribución del conocimiento relevante. Apreciado con mayor detalle en la Figura 1.

Gestión del conocimiento en las organizaciones

Figura 1. Gestión del conocimiento en las organizaciones
Elaborado por: Nofal Nagles García

La adquisición del conocimiento, como un elemento fundamental del proceso, comprende la transferencia y transformación del conocimiento de un experto a una persona interesada en dicho conocimiento, proceso que en distintos contextos y escenarios enfrenta barreras como la escasa disponibilidad de tiempo de los involucrados, mala gestión de transferencia de conocimiento, cultura, implicación del personal, etc. (Cárcel Carrasco & Roldán Porta, 2013).

La comunicación empresarial juega un papel importante para la distribución del conocimiento. Dependiendo de la madurez de la empresa, este proceso puede ser formalizado o no, llegando a ser un eje focal que permite optimizar las operaciones y utilización de recursos. En este proceso la participación de los colaboradores es fundamental, son ellos quienes dinamizan la transferencia de conocimiento (Fontalvo Herrera, Quejada, & Puello Payares, 2011). Un mecanismo formal para la transferencia

de conocimiento, almacena el conocimiento en un medio digital como una base de datos o un archivo plano (Cárcel Carrasco & Roldán Porta, 2013).

Nelson Quintanilla reconoce las TI como una herramienta que facilita la gestión del conocimiento en una organización, persigue objetivos como hacer explícito el conocimiento tácito con una participación activa de las personas para su generación y difusión. Entre las herramientas sugeridas para una adecuada gestión del conocimiento se encuentran buscadores, gestores de contenidos, bases de datos, herramientas de ofimática y de Inteligencia de Negocio, ERP's, simuladores, entre otros (Quintanilla Juárez, 2014).

1.2. Madurez de las TI

Existe más de un modelo madurez, pero destacan los modelos de COBIT y Gibson y Nolan, por ello en la Tabla 1 se presenta una comparativa entre los distintos niveles de cada uno basándose en los conceptos presentados en el "Gobierno de las TI para universidades" (Fernández Martínez & Llorens Largo, 2011).

Tabla 1. Modelos de madurez de COBIT y Gibson y Nolan.

COBIT	Gibson y Nolan
Inexistente: no se reconoce ningún proceso.	**Iniciación:** automatización de procesos repetitivos con TI.
Inicial: se identifican los problemas, no hay procesos estandarizados, pero sí métodos ad hoc.	**Contagio:** veloz crecimiento debido a peticiones de usuarios.
Repetible: los procesos se repiten y se depende de los conocimientos de las personas.	**Control:** definición de planes y estándares de desarrollo.
Definida: procedimientos documentados y estandarizados. Pero aún no son sofisticados	**Integración:** intención de integrar los sistemas con bases de datos corporativas.
Administrada: se realiza un seguimiento de los procesos para una mejora continua. Automatización y uso de herramientas limitados.	**Gestión de datos:** explotación de las bases de datos para la obtención de información.
Optimizada: procesos refinados considerados como mejor práctica. TI integrada para automatizaciones.	**Madurez:** alineación entre el plan estratégico de la organización y las TI.

Implementación de herramientas de calidad, La empresa se adapta con rapidez.	

Nota: Esta tabla establece una comparación entre los modelos de madurez de COBIT y Gibson y Nolan.

1.3. Inteligencia Artificial IA

La IA es una ciencia que sintetiza, automatiza tareas intelectuales, acoge y se apoya de una gran variedad de disciplinas como filosofía, matemáticas, economía, neurociencia, psicología, ingeniería, etc. Por ello varios autores proponen diferentes conceptos categorizados desde cuatro enfoques diferentes visualizados en la Figura 2 (Russell & Norvig, 2008).

En la actualidad la IA tiene varias aplicaciones en campos como la comprensión del lenguaje natural, sistemas expertos, sistemas basados en conocimiento, robótica, planificación, aprendizaje automático, etc. (Casali, 2015).

Algunas definiciones de IA, organizadas en cuatro categorías

Sistemas que piensan como humanos	**Sistemas que piensan racionalmente**
«El nuevo y excitante esfuerzo de hacer que los computadores piensen… máquinas con mentes, en el más amplio sentido literal». (Haugeland, 1985) «[La automatización de] actividades que vinculamos con procesos de pensamiento humano, actividades como la toma de decisiones, resolución de problemas, aprendizaje…» (Bellman, 1978)	«El estudio de las facultades mentales mediante el uso de modelos computacionales». (Charniak y McDermott, 1985) «El estudio de los cálculos que hacen posible percibir, razonar y actuar». (Winston, 1992)
Sistemas que actúan como humanos	**Sistemas que actúan racionalmente**
«El arte de desarrollar máquinas con capacidad para realizar funciones que cuando son realizadas por personas requieren de inteligencia». (Kurzweil, 1990) «El estudio de cómo lograr que los computadores realicen tareas que, por el momento, los humanos hacen mejor». (Rich y Knight, 1991)	«La Inteligencia Computacional es el estudio del diseño de agentes inteligentes». (Poole *et al.*, 1998) «IA… está relacionada con conductas inteligentes en artefactos». (Nilsson, 1998)

Figura 2. Algunas definiciones de IA, organizadas en cuatro categorías.
Elaborado por: Rusell y Norvig

1.3.1. Comprensión del lenguaje natural

Ana Casali lo define como un modelo de lenguaje, como una herramienta de comunicación entre el ordenador y las personas que enfrenta retos como la riqueza del lenguaje, ambigüedad de expresiones y significados, descripciones incompletas y la dependencia del idioma (Casali, 2015).

1.3.2. Sistemas expertos

Un sistema experto nace del estudio de la IA, simula el resultado de extraer las reglas que una persona experta en algún tema haya acumulado en el transcurso de su vida, como Sócrates quién caminaba por Atenas buscando expertos con el objetivo de probar sus reglas (DREYFUS & DREYFUS, 2017), para de esta forma ser programadas y presentadas incluso con la expectativa de que dicho resultado sea mejor comparado con el de la persona experta.

1.3.3. Sistemas basados en conocimiento

Un sistema basado en conocimiento se define como la evolución de la forma tradicional de acceder a la información y mostrarla al usuario. En un principio la información se tomaba de ficheros, luego con la aparición de las bases de datos se logra una mejor gestión de la información, para así dar paso a los sistemas basados en conocimiento cuyos datos los produce un usuario experto en el dominio. A diferencia de los sistemas tradicionales en los es necesario definir cómo el usuario interacciona con la aplicación y establecer una secuencia de instrucciones el sistema basado en conocimiento se integra directamente con la aplicación (Villena Román, Crespo García, & García Rueda, 2012).

1.3.4. Aprendizaje automático

El aprendizaje interviene en todas las actividades de inteligencia y por ello es considerado una característica esencial para las aplicaciones de IA. Entre sus técnicas

más conocidas se encuentran: los árboles de decisión, métodos inductivos, redes neuronales artificiales, etc. (Casali, 2015).

1.3.4.1. Aprendizaje

La Real Academia de la Lengua define aprender como la "adquisición del conocimiento de algo por medio del estudio o de la experiencia…" (Real Academia Española, 2018), en IA esta puede ser clasificada en aprendizaje supervisado cuando se utilizan datos clasificados previamente para obtener un modelo que posteriormente permita la predicción de un nueva entidad; y no supervisado cuando sólo se poseen los datos entrada y el algoritmo es el encargado de agruparlos mediante patrones para definir una estructura de dominio de la entidad (González Herrero, 2017).

1.3.4.2. Red Neuronal Artificial RNA

Una RNA es un sistema conformado por un gran número de elementos de procesamiento llamados neuronas, conectados entre sí emulando el funcionamiento de las neuronas del cerebro (Gámez Albán, Orejuela Cabrera, Salas Achipiz, & Bravo Bastidas, 2016).

Un solo modelo matemático no puede acoplarse a todas las necesidades que presenta un sistema con IA, a lo largo de la historia se han establecido algunos modelos, como el de Brain State in a Box (para la extracción de conocimiento de bases de datos), Kohonen (mapas auto organizativos), perceptron (reconocimiento de caracteres), entre otros (de Moya Anegón, Herrero Solana, & Guerrero Bote).

Perceptrón multicapa

El perceptrón multicapa solventa el problema de no linealidad del perceptrón simple, es la red neuronal más usada, es capaz de actuar como un aproximador universal de funciones. La utilización de un algoritmo de retro-propagación le permite aprender la

relación existente entre un conjunto de patrones de entradas y sus salidas, esto a su vez le permitirá realizar la clasificación de nuevos ejemplos (Vidal González, 2015).

Mapas auto-organizativos de Kohonen

SOM por sus siglas en inglés (Self Organizing Maps) son un tipo de red neuronal de aprendizaje no supervisado, utiliza dos capas (entrada y salida) y agrupa en clústeres los registros de manera que en cada grupo son similares entre sí (IBM, 2018). Como se aprecia en la Figura 3 SOM es un modelo de dos capas, donde, la primera recoge y canaliza la información y la segunda se encarga del procesamiento (Gámez Albán, Orejuela Cabrera, Salas Achipiz, & Bravo Bastidas, 2016).

Mapa auto-organizado de Kohonen

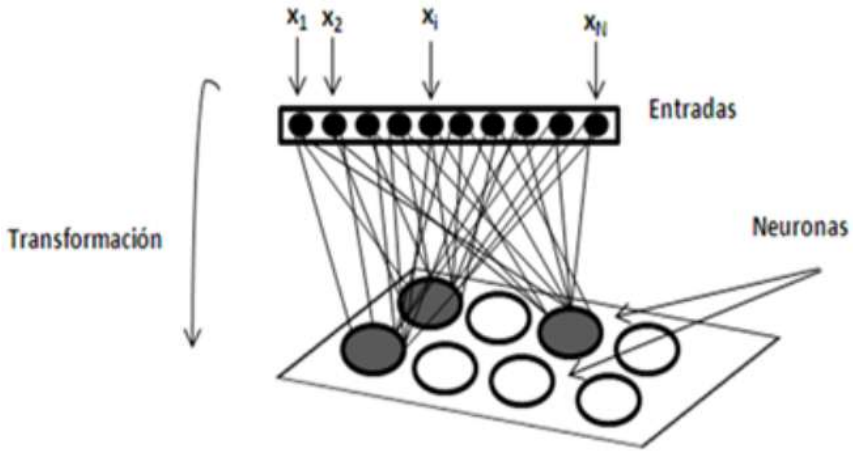

Figura 3. Mapa auto-organizado de Kohonen
Elaborado por: Harol Gámez, Juan Orejuela, Oscar Salas, Juan Bravo

Su función de aprendizaje es:

$$Sw_r = a(t)h_{rs}(t)(x - w_r)$$

En donde:

- Sw_r es la variación del vector de pesos para la neurona r-esima.
- $a(t)$ es la tasa de aprendizaje.

- $h_{rs}(t)$ es la función de vecindad.
- x es el vector de entrada.
- w_r es el vector de pesos de la neurona r-esima.
- t es el índice de la iteración.

SOM ha sido implementado en algunos estudios para la clasificación de términos, como Vicente Guerrero, Cristina López, Cristina Faba, María Reyes, Felipe Zapico y Félix de Moya que en su publicación "Clasificación de términos mediante el algoritmo de Kohonen" proponen un método para clasificar términos de una base de datos de prueba que halle relaciones contextuales entre los mismos (Guerrero Bote, y otros, 2002). Para ello crea vectores referentes a documentos en donde se calcula el peso de los documentos con la siguiente fórmula:

$$a_{ij} = t_{ij} log \frac{D}{ND_i}$$

En donde:

- a_{ij} es el peso asignado al documento D_i en el término t_j.
- t_{ij} es el número de veces que aparece el término t_j en el documento D_i.
- ND_i es el número de términos que aparecen en el documento D_i.
- D es el número total de términos de toda la base.

María Reyes, Vicente Guerrero y Felipe Zapico proponen la implementación Kohonen para clasificar revistas científicas, implementan Kohonen como un modelo de RNA para la clasificación. Justifican su implementación por la posibilidad de clasificación en una rejilla bidimensional (Reyes Barragán, Guerrero Bote, & Zapico Alonso).

1.3.4.3. Máquinas de soporte de vectores

SVM por sus siglas en inglés (Support Vector Machine) es un algoritmo de IA supervisado para problemas de clasificación y regresión (Sánchez Anzola, 2015), a diferencia de otros métodos basados en la minimización de los errores cometidos en el entrenamiento, SVM se basa en la construcción de hiperplanos óptimos de separación que equidista a los ejemplos más cercanos de las clases (Carmon Suárez, 2016). Es decir, busca "definir un margen entre dos clases por la separación máxima entre ellas" como se puede ver en la Figura 4 entre ejemplos linealmente separables existe más de un hiperplano de separación, SVM busca el óptimo (Nievas Lio, 2016).

Hiperplanos de separación e hiperplano óptimo

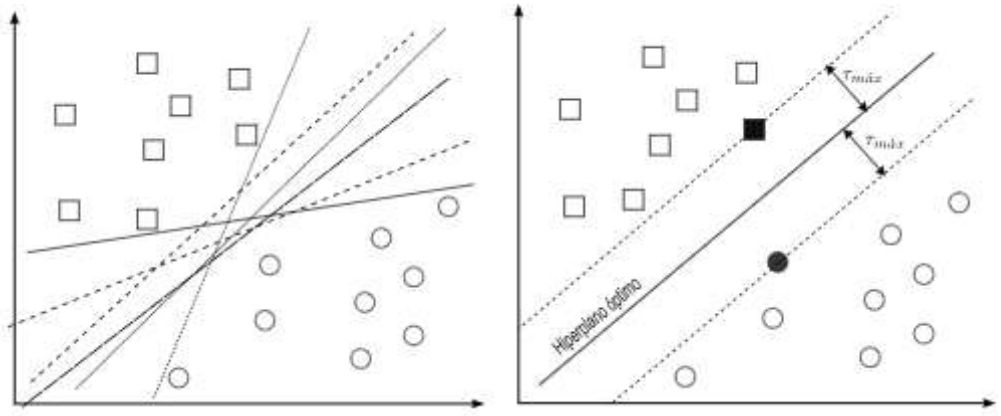

Figura 4. Hiperplanos de sepación e hiperplano óptimo
Elaborado por: Harol Gámez, Juan Orejuela, Oscar Salas, Juan Bravo

Entrenamiento con dos y tres dimensiones

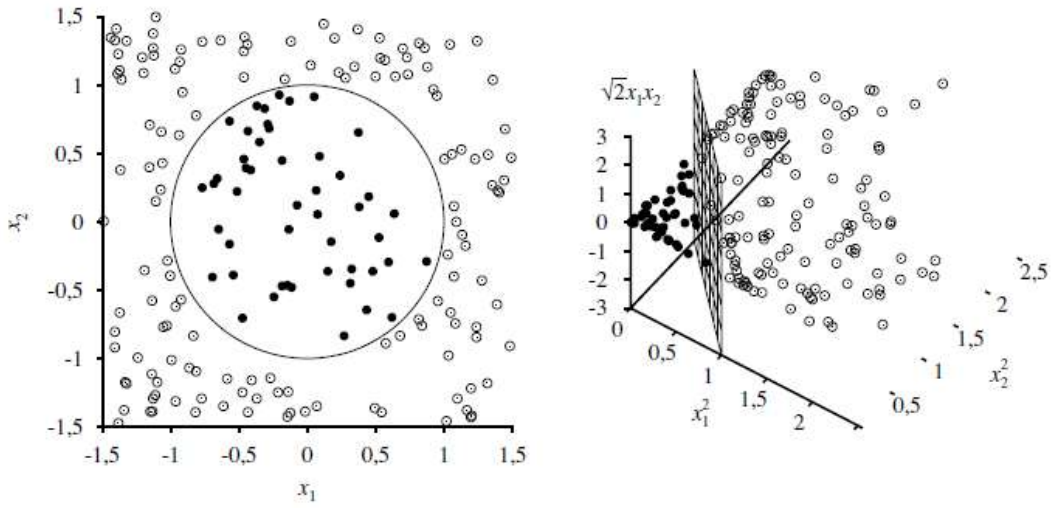

Figura 5. Entrenamiento con dos y tres dimensiones
Elaborado por: Rusell y Norvig

SVM se caracteriza principalmente por su capacidad de computar características adicionales incrementando el número de dimensiones, es decir, si los datos son representados con un número suficientemente alto de características siempre podrán ser separados linealmente los ejemplos. Ver Figura 5.

Adicionalmente SVM posee un parámetro esencial denominado *kernel*, éste permite cambiar la división que genera un hiperplano lineal a una función de proyección más eficaz como por ejemplo una polinomial como se puede ver en la Figura 7. Los tipos de *kernel* más comunes son: lineal, polinomial, sigmoidal y base radial (Benítez, Escudero, & Kanaan, 2013).

Conjunto linealmente no separable

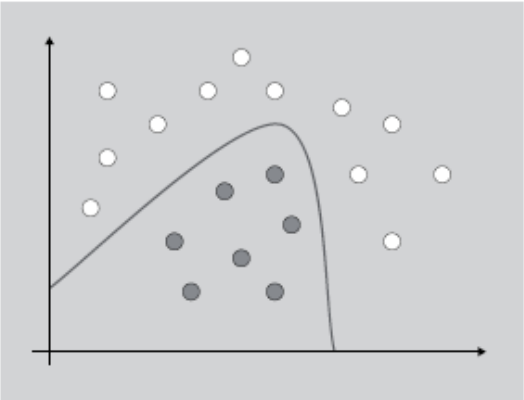

Figura 6. Conjunto linealmente no separable
Elaborado por: Raúl Benitez, Gerard Escudero, Samir Kanaan

Entre sus principales ventajas se destaca ser eficiente en escenarios en donde el número de ejemplos es menor al de dimensiones (Godoy Viera, 2017).

RNA y SVM presentan una capacidad equiparable al realizar clasificación de datos (Sánchez Anzola, 2015). Así, Luis Guarda y Joaquín Carrasco demuestran en su estudio "Comparación entre SVM y RNA para clasificación de sujetos con ojos abiertos o cerrados mediante EEG" conseguir resultados cercanos obteniendo 85 ±13% con SVM y 80 ±13% con RNA (Guarda & Carrasco, 2017). Mientras Nicolás Sánchez obtiene en sus resultados como mejor precisión de pronóstico de 63.9% con RNA y de 63% con SVM (Sánchez Anzola, 2015).

Clasificación

En IA la clasificación se encarga del reconocimiento de patrones para etiquetar a una entidad. La clasificación contiene esencialmente tres fases: la definición de clases, representación de información como características y el aprendizaje. La categorización

de texto es uno de los problemas que intenta resolver la clasificación (Benítez, Escudero, & Kanaan, 2013).

1.3.5. Agente virtual

Las brechas de evolución desde la aparición del lenguaje hace 30.000 años a.C, el alfabeto hace 1600 años a.C, la imprenta 1455 a.C, no fueron tan cortas como después de la aparición del teléfono en el año 1876, rápidamente la radio nace en 1895, la televisión en 1936, el internet en 1961 y cada vez más rápido después de 1999 con la web 2.0 hasta hoy en día con la web 3.0. Esto comunica lo ampliamente difundida que se encuentra la información. Alan Turing manifestó que para el año 2000 existirán "las máquinas que piensan", Deep Blue fue un cerebro desarrollado por IMB qué ganó en ajedrez al maestro Garry Kasparov, WATSON y SIRI son tecnologías integradas al concepto de agente virtual, incluyendo a Eliza que se define como "un programa informático con capacidad de interpretar el lenguaje humano" inventado por Joseph Weizenbaum (Mazza, 2012).

En la actualidad se pretende usar a los agentes virtuales como tutores que acompañen en el proceso de aprendizaje de alumnos en el campo de la educación, entregando información de calidad, para lo cual se definen patrones para identificar tablas las cuales contienen la información, tal como lo plantean Celia G. Róspide y Cristina Puente en la investigación titulada "Agente Virtual Inteligente Aplicado a un Entorno Educativo", como parte de la investigación, la interfaz gráfica se presenta como un chat acompañado de cuatro imágenes que hacen referencia a intenciones preestablecidas. Una de sus conclusiones afirma que el análisis de los patrones es el mejor método para emplear en un *chatbot* (Róspide & Puente, 2012).

1.3.6. Servicios cognitivos

Voz

Se encuentra enfocado a la utilización de un servicio, por medio de algoritmos de procesamiento de voz, para ayudar a moderar automáticamente contenido auditivo, y de esta forma lograr la humanización de una aplicación apegándose a la explicación de Jansari donde menciona "esto sugiere que el cerebro utiliza un nivel más alto de procesamiento para la memoria facial que para otras tareas" (Venkataramanan, 2015).

Imagen

Se presenta como un servicio cognitivo con la capacidad de entregar información sobre el contenido visual encontrado en una imagen, se basa en modelos de identificación de objetos, para etiquetarlos y presentar la característica encontrada por medio de un mensaje JSON (Azure Microsoft, 2018).

1.4. Azure

Azure se presenta como un conjunto de soluciones, éstas soportan varios escenarios de trabajo que aportan al desarrollo del área de TI.

> Azure es una plataforma flexible e interoperable, que permite múltiples protocolos de Internet como HTTP, REST, SOAP y XML. Los entornos de desarrollo soportados pueden ser Microsoft Visual Studio, .Net Framework y Eclipse. Los lenguajes con los que se pueden escribir las aplicaciones son .Net (C# y Visual Basic), C++; PHP, Ruby, Python y Java (Murazzo, Millán, Rodríguez, Segura, & Daniela, 2012).

1.5. Bot Framework

Es un conjunto de herramientas diseñado para interactuar entre sí, permite al usuario en primera instancia utilizar SDK’s código libre para los lenguajes Node Js, .Net y REST, para una simple creación de diálogos y comandos los cuales el programador podrá usar a conveniencia para lograr un ambiente de conversación que pueden verse representados

en imágenes, texto, voz, etc. En segunda instancia un portal de desarrollo en el que se puede conectar, registrar y publicar un *Bot* para posteriormente acoplar a varios medios de comunicación social como GroupMe, Facebook Messenger, Kik, Skype, Slack, Microsoft Teams, Telegram, Text/Sms, Twilio, Cortana, Skype for Business o embeberlo en una aplicación web. Por último, se puede liberar un *Bot* hacia la tienda de BOTS Microsoft (Microsoft, 2018).

Influencia de los medios sociales en una persona

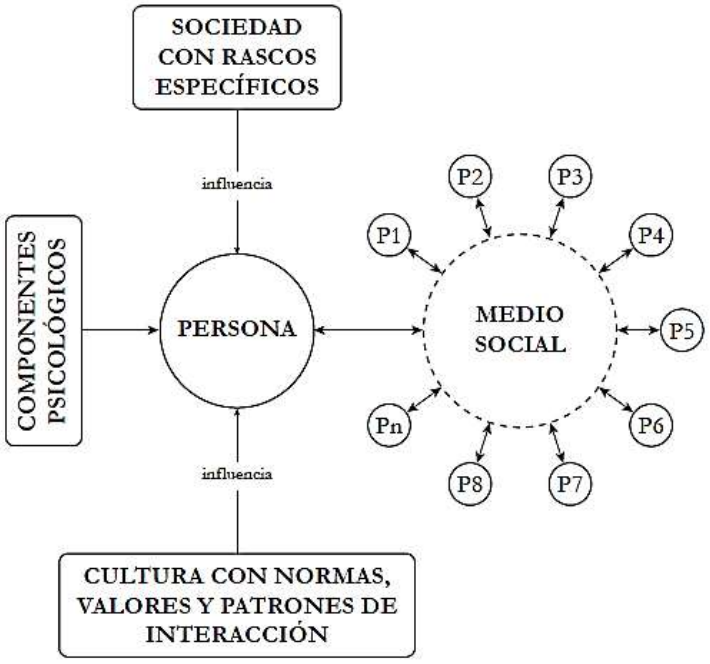

Figura 7. Influencia de los medios sociales en una persona
Elaborado por: Amaro La Rosa

Amaro La Rosa explica que la evolución de las comunicaciones ha posicionado a varias redes sociales como un medio preferencial, ocasionado por la influencia de rasgos específicos, así como también de cultura, normas, valores y otros entes sociales como se evidencia en la Figura 7 (Amaro, 2012).

1.6. LUIS

Sus siglas se refieren a Language Understanding Intelligent Services, y se traducen como Servicios Inteligentes de Compresión de Lenguaje. LUIS es un intérprete de lenguaje natural y basa su desempeño en dos enfoques, el primero es entender cuál es la intención del usuario con las palabras que usa cuando éste se encuentra conversando naturalmente con otro usuario, siempre existe una intención por detrás de cada palabra, un usuario puede pedir una cosa de formas diferentes. Además, captura las entidades se presentadas en la conversación de un usuario, por ejemplo, si se pregunta sobre un lugar, dicho lugar es una entidad dentro de una intención de pregunta.

LUIS puede entender qué es lo que el usuario pregunta para entrenarse, las intenciones son capturadas sin importar la manera de formular la pregunta para así identificar en una futura pregunta que refiera a una misma intención respuesta será la que el usuario necesite siempre (Felman, 2017).

1.7. Servicios web

El esquema de un servicio web básicamente contempla una petición iniciando con un usuario, a través de internet, éste a su vez consume el servicio para ser usado en un proceso (Redbooks, 2014), se puede definir también a servicios web como varios componentes de software publicados en la web, todo esto comunicado por el protocolo web más común usado que es HTTP, encargado de llevar las peticiones y las respuestas (Gurugé).

1.8. Buscadores

Un buscador se encarga de recopilar información, existen empresas con un notable ejemplo de oportunidad de negocio generada por el desarrollo de las TI. Estas empresas se dedican a seleccionar y ordenar información para ponerla a disposición de quien

quiera visualizarla (Pablos Heredero, López Hermoso Agius, Romo Romero, & Salgado, 2012).

1.9. JSON

Diseñado para identificar objetos, sus siglas significan JavaScript Object Notation, haciendo referencia al lenguaje de programación JavaScript, nace como multiplataforma, y por ello se puede usar en varios escenarios de desarrollo. Ha probado ser más rápido y eficiente que XML (Afsari, Eastman, & Castro-Lacouture, 2017).

1.10. Metodología XP

Una metodología ágil es fundamental en la creación de un proyecto de software, la metodología XP o programación extrema se basa en conceptos como simplicidad, comunicación y reutilización de código (Chicaiza Paredes, 2016). De acuerdo con la naturaleza de cambio a la que se apega un proyecto tecnológico, como es el caso del presente trabajo, resulta ser la más efectiva.

XP se soporta adaptaciones a las necesidades del proyecto con una capacidad de respuesta a imprevistos eficaz, es perfecta para grupos de trabajo pequeños (Cruz Martínez & Alarcón Armenteros, 2016), interacción con el cliente, diseño simple, integración continua y test (Fernández González, 2013). Sus reglas y prácticas se encuentran definidas en planificación o análisis, diseño, desarrollo y pruebas (Joskowicz, 2008).

1.11. UML

El Lenguaje de Modelado Unificado (UML por sus siglas en inglés) es un “estándar para el modelado de sistemas de software” (Bonaparte, 2012), se compone de varios diagramas entre los que se destacan el diagrama de clases para describir la interacción de las clases, de casos de uso para describir acciones que el usuario realiza en el sistema

y diagramas de secuencias para describir la interacción de los objetos en el tiempo (Bonaparte, 2012).

1.12. BPMN

Es una notación para describir la secuencia lógica de procesos de negocios para graficar Diagramas de Procesos de Negocio (BPD por sus siglas en inglés). Es considerado un estándar que permite "disminuir la brecha entre los procesos de negocio y su implementación" (Bizagi Suite, 2014).

CAPÍTULO 2

Escenario

El prototipo de asistente virtual, de aquí en adelante se lo tratará como asistente, pretende ser implementado en una organización con un mediano nivel de madurez de TI, la curva de Nolan ubicaría a la organización en una etapa de control, en donde se estandariza el desarrollo para las aplicaciones, el departamento tiene una planeación de sistemas de información, se formalizan los recursos informáticos para hacer más eficiente lo invertido. Mientras COBIT en un nivel de madurez definido donde los procesos se encuentran estandarizados y documentados, es decir, se encuentran formalizados los procesos y no necesariamente son optimizados y refinados.

De esta forma se establece el supuesto de una empresa que posee infraestructura de TI con la capacidad de sostener aplicaciones con arquitectura cliente servidor y bases de datos en al menos dos servidores físicos o virtuales (con la posibilidad de tener activos de TI en nube).

Cabe subrayar, la implementación de los diferentes componentes del prototipo se utiliza una suscripción gratuita de Azure (Dreamspark), enfocada en la academia, que permite publicar aplicaciones y bases de datos, facilitando su construcción y puesta a producción en un ambiente de pruebas.

Primera Iteración

2.1. Requisitos y tecnologías

El proyecto inicia con la consideración de varios aspectos de los cuales se derivan algunas teorías no sujetas a una guía existente, pero sí parte de una clara concepción sobre un sistema experto o un sistema basado en conocimiento, como se menciona en el marco teórico. Se pretende ir más allá de lo existente y crear un asistente virtual con la

capacidad de aprender sobre lo que el usuario está preguntando, identificando la intención y las entidades que pudieran existir en dicha pregunta, para ello se requiere una investigación la cual deriva en la profundización de preguntas como: ¿qué medios virtuales existen para la comunicación con el usuario?, ¿cómo y de dónde se podrá consultar la información necesaria para el usuario? ¿de qué forma el agente podrá interpretar el lenguaje natural del usuario? y finalmente ¿de qué forma aprenderá el agente sobre lo que el usuario pregunte? (IA), tales requisitos, así como los orígenes de información e interfaces se encuentran representadas en la Tabla 2.

Tabla 2. Requisitos, tecnologías y criterios de la primera iteración.

Requisitos	**Tecnologías evaluadas**	**Tecnologías Escogidas**	**Criterios**
Orígenes de información y conocimiento.	Bases de datos API's Gestores de Contenido Archivos planos	API's	Información pública y disponible de forma inmediata para pruebas.
Interfaz tipo chat lo más amigable posible.	Bot Framework Watson	Bot Framework	Vía de comunicación con el usuario que integre sus servicios con aplicaciones de mensajería más usados. Presentación del aplicativo con un mínimo de usabilidad media. Vía de comunicación con el usuario que incorpore un motor de pruebas local.
Servicio cognitivo de búsqueda, comunicación con el usuario, interpretación de lenguaje natural	Recast.ai Api.ai wit.ai Microsoft LUIS	Microsoft LUIS	Interpretación de lenguaje natural del usuario. Interpretación de la intención sobre las preguntas del usuario. Interpretación de entidades sobre las preguntas del usuario. Tiempo de madurez en el mercado.
Captura de conocimiento e inferencia de los resultados.	Perceptrón Kohonen/SOM	Kohonen/SOM	Algoritmo de IA con buena precisión de clasificación de términos. Preferiblemente de autoaprendizaje.

Nota: Esta tabla contiene los requisitos, tecnologías y criterios considerados para la primera iteración.

Dada la naturaleza del proyecto, varias de las tecnologías seleccionadas cambiaron a medida que aparecieron las necesidades de éste, este comportamiento responde a la naturaleza de un proyecto de desarrollo de software caracterizado por ser iterativo, repetitivo haciendo necesaria la prueba y error para notar un descubrimiento. Hasta el término de esta investigación no se encontró, un asistente virtual que aprenda sobre lo que el usuario pregunta en un contexto empresarial.

Es necesario enfatizar en el concepto de asistente, definido en el marco teórico, para efectos de este proyecto será el encargado de entregar información importante y de calidad. En el escenario planteado se pueden encontrar aplicativos BOTS que entregan información estática previamente analizada y filtrada por una persona la cual debe hacer el trabajo de forma manual.

Para saber cómo y de dónde se podrá obtener información para el agente virtual, la primera consideración a tomar en cuenta es que esta información tiene que ser libre y estar limitada a un escenario de trabajo con el que el asistente pueda trabajar y se pueda probar, entonces al verificar de qué forma se podrá integrar esta extracción de datos con Bot Framework la mejor opción es el consumo de información por medio de API REST ya que esta información puede ser actualizada de forma dinámica, a comparación con la extracción de CSV en la que la información se deberá actualizar por medio de un agente humano. Con respecto a realizar una conexión directa a la base de datos, el único impedimento sería el obtener las credenciales de acceso a la misma, con API REST no sería necesario, básicamente es una réplica de la información puesta en una base de datos accedida por medio de una URL.

Al momento de seleccionar entre Bot Framework y Watson las que más se apegan a la necesidad del proyecto, la que se integra con más cantidad de medios sociales como

Skype, Facebook, Telegram (considerado el segundo aplicativo de mensajería social más usado) fue Bot Framework (Diarlu, 2017).

Una interrogante más a resolver es, de qué forma el asistente entenderá lo que el usuario quiere saber, y para esto existen algunas soluciones como: Recast.ai, Api.ai, Wit.ai y Microsoft LUIS, de las cuales se descartan Api, y Wit ya que no detectan la intención de usuario sino más bien necesitan un repositorio de datos de tipo Q&A (preguntas y respuestas) el cual deberá estar apegado a la pregunta del usuario caso contrario no se encontrará una respuesta. Recast como LUIS detectan la intención del usuario junto con las entidades en la redacción del usuario, pero Recast es relativamente nuevo en el mercado con respeto a LUIS. La última mencionada lleva un tiempo de desarrollo y pruebas mayor que Recast y por ende es mucho más madura.

Por último, de acuerdo con la propuesta del prototipo de asistente virtual se necesita tratar la información y conocimiento, y por tanto requiere una inferencia que permita discernir la información realmente oportuna y/o valiosa para el usuario. Para cumplir este propósito la implementación de un algoritmo de IA parece ser el camino más evidente por seguir, entre su amplio espectro de soluciones la IA permite el aprendizaje de máquina. Éste a su vez generalmente resuelve problemas de clasificación y regresión, el primero determina si un ejemplo pertenece o no a una clase, y el segundo busca predecir un valor continuo en base a las características del ejemplo.

Por el supuesto del mediano nivel de madurez de TI que debe poseer la empresa se entiende que tanto la información y el conocimiento se encuentran documentados y registrados en algún medio digital como una base de datos o fichero. En adición debido a la intención de entregar sólo la información que el usuario considera oportuna se identifica el problema de aprendizaje como uno de clasificación, en donde las clases son identificadas como oportuna e inoportuna.

El siguiente paso es el reconocimiento de un algoritmo para resolver el problema y que se adapte a las características específicas del escenario. Así, a pesar de que el prototipo se plantea debe funcionar bajo cualquier ámbito empresarial el escenario general es la clasificación terminológica de las respuestas posibles bajo una pregunta emitida por el usuario. Kohonen es una red neuronal como el perceptrón multicapa, sin embargo, su creador lo diseñó con la intención de clasificar términos, por lo cual ha sido implementado en varias investigaciones con este propósito, en algunas de las cuales se ha demostrado su potencial a diferencia de otras soluciones de IA para problemas de clasificación (de términos). Es un algoritmo de entrenamiento supervisado y no supervisado a diferencia del perceptrón que es solamente supervisado y su propósito es general.

Al finalizar la investigación las tecnologías propuestas en el marco de la solución informática planteada son; LUIS como intérprete de lenguaje natural, Bot Framework para la comunicación y la presentación de los datos del agente hacia el usuario, el consumo de API's libres para la búsqueda de información, y Kohonen para la inteligencia del agente.

Una vez identificadas las herramientas y tecnologías que podrían apoyar a la construcción del asistente, se inicia el planteamiento de una arquitectura posiblemente apropiada, teniendo en cuenta a los actores en todo el proceso. El primer actor es el usuario quién va a interactuar con el asistente directamente, el segundo actor sería el asistente como tal, quién se encarga de obtener información y presentársela al usuario de forma filtrada, en este proceso se logran identificar dos tipos de búsqueda, la primera, una búsqueda interna en donde el asistente encontrará toda la información que la empresa tiene dependiendo su nivel de madurez tecnológico y la segunda, una búsqueda externa en donde el asistente encontrará información que no se encuentre

dentro de la empresa. Cabe recalcar que esta investigación no se enfocará en los métodos existentes para consultar información interna y externa.

Primera arquitectura propuesta

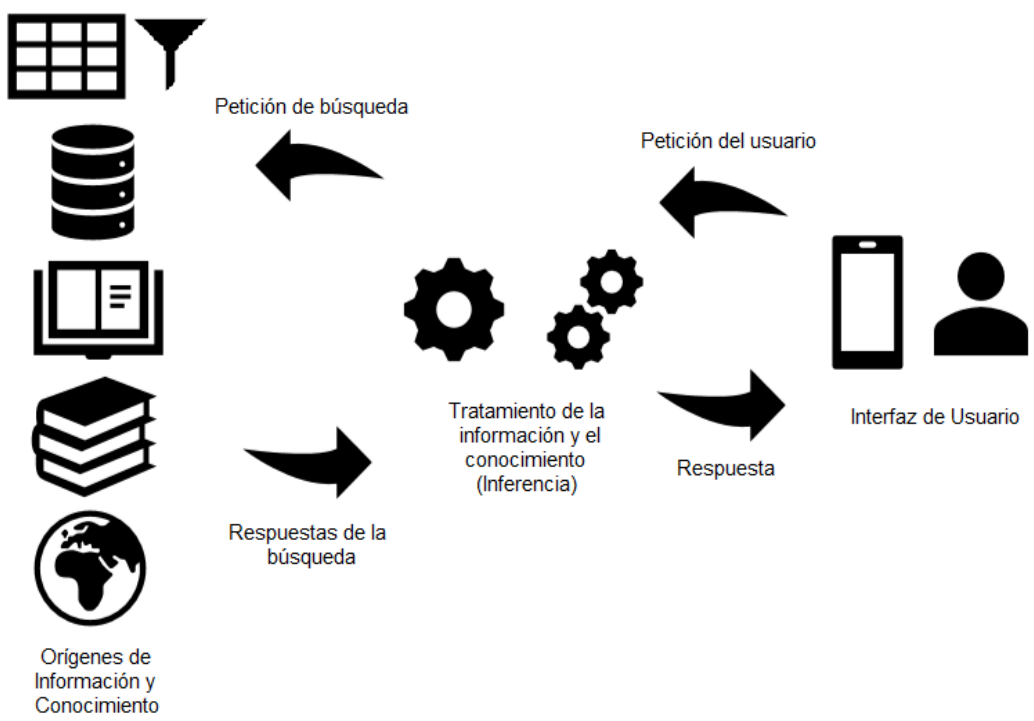

Figura 8. Arquitectura propuesta para la primera iteración
Elaborado por: Christian Abata y Pedro Ramírez

Como se aprecia en la Figura 8, durante el proceso de entrega de información nace la necesidad de un filtro, éste captura la información y la filtrarla de tal forma que sólo se presente información de calidad al usuario.

Es necesario conocer que un diálogo se define por tener un inicio, un contenido el cual muestra la conversación y un fin. El asistente reaccionará por medio de un estímulo del usuario quién será el que inicie la conversación haciendo cualquier pregunta, si la pregunta del usuario está apegada a la intención configurada en LUIS, el asistente entrega una respuesta afirmativa a su pregunta, caso contrario dará una respuesta sin resultados al usuario. Además, pueden plantearse intenciones que terminen con un diálogo o lo inicien.

2.2. Análisis

El análisis realizado en esta iteración tiene como propósito validar la propuesta arquitectónica, por lo cual en la Tabla 3 se describen interacciones entre actor-componente y componente-componente.

Tabla 3. Interacción usuario y asistente.

Escenario	Descripción	Descripción del sub-escenario
Interacción Usuario – Asistente	El usuario realiza una pregunta y el asistente entrega la información más relevante y oportuna de todo el contenido disponible.	En este nivel de interacción, no se posee un sub-escenario.
Interacción: Usuario/interfaz - componentes del prototipo de asistente virtual.	El usuario por medio de la interfaz realiza una pregunta, que pasa por el componente de tratamiento de información, en donde: - Se realiza una comprensión de la petición (pregunta) del usuario. - Se realiza la consulta de la información en los orígenes de datos competentes. - Se analiza las respuestas obtenidas y se discierne las más oportunas para el usuario. Y la interfaz finalmente presenta la información al usuario. Ver Figura 11.	**Interacción: Componente de tratamiento de información - Origen de datos** El componente de tratamiento de información traduce la pregunta del usuario en una petición o consulta de acuerdo con el tipo de origen de datos. El origen de datos provee toda la información que posee.
		Interacción: Servicio Cognitivo de búsqueda y comprensión de lenguaje natural -Subcomponente de Inferencia LUIS comprende la petición del usuario, traduce la pregunta a una consulta sobre el origen de datos, al obtener las respuestas envía las mismas al componente de inferencia quien infiere las respuestas más relevantes para ser presentadas al usuario, devuelve las mismas y LUIS envía al usuario final las respuestas.

Nota: Esta tabla refleja la interacción que existe entre el usuario y el asistente.

El primer escenario puede verse representado mediante el siguiente caso de uso en la Tabla 4, junto con su respectivo diagrama en la Figura 9.

Tabla 4. Caso de uso del usuario con el asistente virtual.

Caso de uso	Interación usuario - asistente
Actores y Componentes	Usuario final Prototipo de asistente virtual
Camino principal	1. Realizar una pregunta informativa al asistente.

Caso de uso	Interación usuario - asistente
	2. Indicar la opción más relevante de las respuestas presentadas.
Camino secundario	1. Rechazar las respuestas presentadas. 2. Reformular la pregunta.
Precondiciones	El usuario accedió al asistente.
Postcondiciones	Salir del asistente.

Nota: Caso de uso de la interacción del usuario con el asistente virtual.

Diagrama de caso de uso: Interacción usuario – asistente

Caso de uso usuario/prototipo

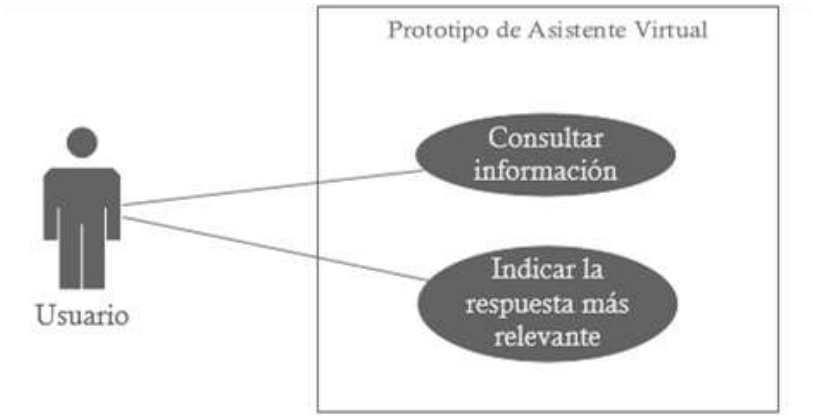

Figura 9. Caso de uso para mostrar la interacción del usuario con el prototipo
Elaborado por: Christian Abata y Pedro Ramírez

El resto de los escenarios no expresan funcionalidad, así un diagrama de casos de uso no es apropiado, pero permiten comprender las funciones realizadas por cada uno de los componentes de la solución y en el transcurso del análisis se determinó que tenían un comportamiento secuencial, tal como en los escenarios de procesos, por lo que se pueden apreciar apropiadamente en diagramas de procesos, para lo cual se adoptó a BPMN como notación. Sin embargo, se realizaron adaptaciones a la nomenclatura como la utilización de componentes en lugar de actores, puesto que el fin de los diagramas no es la automatización de los procesos, sino representar claramente los roles que cumplen los componentes, su interacción, y realizar un análisis estructurado, como se aprecia en la Figura 10. Los diagramas fueron validados mediante las reglas de BPMN 2.0 de Visio 2016.

Diagrama de procesos: Interacción usuario – componentes del prototipo de asistente virtual

Procesos de los componentes del asistente virtual

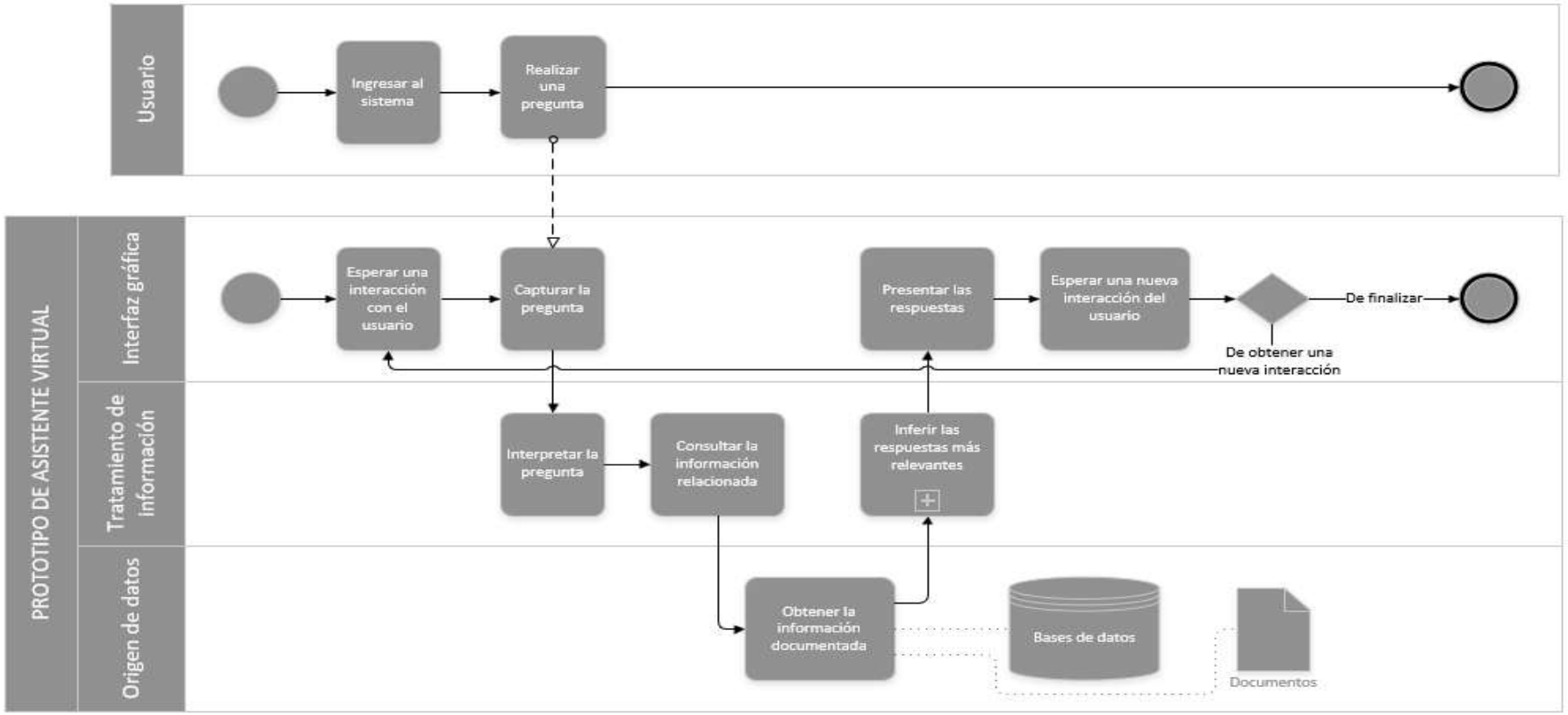

Figura 10. Diagrama de procesos para entender el funcionamiento e interacción de los componentes en la primera iteración.
Elaborado por: Christian Abata y Pedro Ramírez

2.3. Diseño

La arquitectura previamente presentada responde a una arquitectura de distribución de software en donde se distinguen claramente los componentes visualizados en la Tabla 5.

Tabla 5. Funcionamiento de componentes.

Componente	Descripción
Orígenes de información y conocimiento	Es el encargado suministrar la información necesaria para brindar una respuesta al usuario. Dicha información debe encontrarse documentada y/o registrada en un medio digital.
Inferencia	Componente a cargo de tratar la información, así como interpretar las peticiones del usuario y con la capacidad de discernir las respuestas más relevantes u oportunas para ser presentadas al usuario.
Interfaz de usuario	Permite la comunicación entre el usuario y el asistente, registrando las preguntas del usuario y brindando las respuestas que son oportunas de acuerdo con el componente de inferencia.

Nota: La tabla describe cómo funciona cada componente.

Sin embargo, desde una perspectiva modular el asistente puede ser mejor interpretado mediante el PBS presentado en la Figura 11.

PBS Asistente Virtual

Figura 11. PBS del proyecto para la elaboración del asistente virtual
Elaborado por: Christian Abata y Pedro Ramírez

En donde:

- Comunicación con el usuario: interfaz tipo chat, en donde, el usuario puede realizar sus consultas y recibir la información y conocimiento que requiere.

- La interfaz de comunicación: es la encargada de realizar el intercambio de peticiones realizadas desde el aplicativo del usuario final, con las respuestas obtenidas en la búsqueda en los orígenes de datos y el módulo de inferencia.
- Módulo de inferencia: discierne los resultados obtenidos bajo una priorización inteligente, aprendiendo en base a las consultas previas realizadas por el usuario.

2.4. Codificación

El primer paso para iniciar con el prototipo es la configuración de LUIS en el portal, donde se guardan las intenciones y se definen las entidades que el asistente va a reconocer cuando el usuario empiece su interacción escribiendo sobre la interfaz. Paso seguido es el desarrollo llevado a cabo con Node Js, la primera instancia es la conexión hacia LUIS y la creación de los diálogos que el usuario podrá ver, la segunda instancia es codificación y llamado a un servicio API REST por medio de una URL, el cual ya tendrá parametrizado el objeto que el usuario va a buscar.

Dentro de esta codificación las librerías necesarias para la interoperabilidad de los servicios son; Botbuilder conocido como un framework que puede utilizarse para trabajar en Node Js o en .Net, depurándolo con su emulador propio para BOTS, otra librería necesaria es Restify el cual habilita la conexión para el consumo de servicios REST, HTTP habilita un puerto de conexión basado en el protocolo que lleva su mismo nombre y por último Dotenv encargado de almacenar la cadena de conexión que incluye una llave de autenticación para el consumo de los servicios de LUIS.

Una vez terminada toda esta configuración y codificación se procede a realizar las pruebas a nivel local con la herramienta Bot Framework Emulator. Para la primera iteración, el consumir datos de un API no construido a medida para el proyecto requiere de una lectura profunda sobre la documentación que la empresa libera en su portal, por

lo que al probar la extracción de datos se presentaron varios errores los cuales poco a poco se fueron corrigiendo. Para la identificación de las intenciones LUIS resultó ser muy efectivo permitiendo a las pruebas con respecto a lo que el usuario necesitaba sean satisfactorias.

Una vez extraídos los datos del API y la identificación de la intención del usuario con preguntas como: busca el perfil de prayanime, buscar a prayanime, dime los datos de prayanime, si la búsqueda fue afirmativa se muestra un detalle de la entidad prayanime, presentado en forma de tarjeta de presentación.

Antes de iniciar la codificación es necesario configurar previamente las intenciones y entidades en el portal de LUIS ingresando a la página web https://www.luis.ai, para luego crear una aplicación desde el botón create new app, como se aprecia en la Figura 12.

Interfaz para la creación de aplicaciones

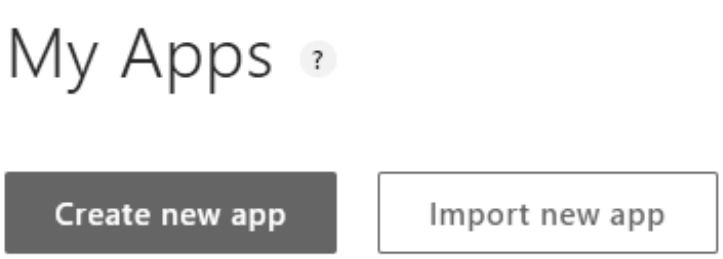

Figura 12. Interfaz para creación de aplicaciones en LUIS
Elaborado por: Christian Abata y Pedro Ramírez

Los datos necesarios serán nombre, cultura y una descripción, que para efectos del proyecto serán llenados con los datos de Tesis Bot, español y Tesis Bot respectivamente. Una vez creada la aplicación en el menú lateral izquierdo se encuentran los accesos para crear las intenciones y las entidades, el primer paso será

crear las entidades dando clic en el botón entities y después dando clic en el botón create new entity, para así agregar la entidad que será guardada como persona de tipo Simple, como se aprecia en la Figura 13.

Interfaz de entidades LUIS

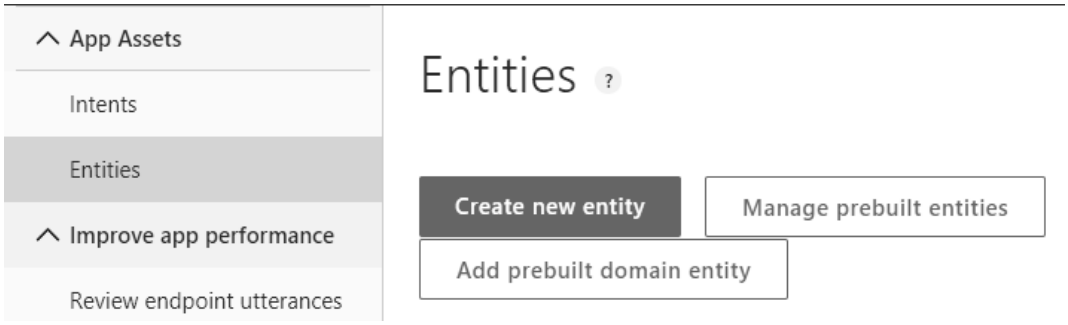

Figura 13. Interfaz para la conFiguración de entidades en LUIS
Elaborado por: Christian Abata y Pedro Ramírez

Una vez creada la entidad, es necesario crear una intención que representará lo que el usuario quiere comunicar, para esto es necesario dar clic en el botón intents y después dando clic en el botón create new intent, como se aprecia en la Figura 14.

Interfaz de entidades LUIS

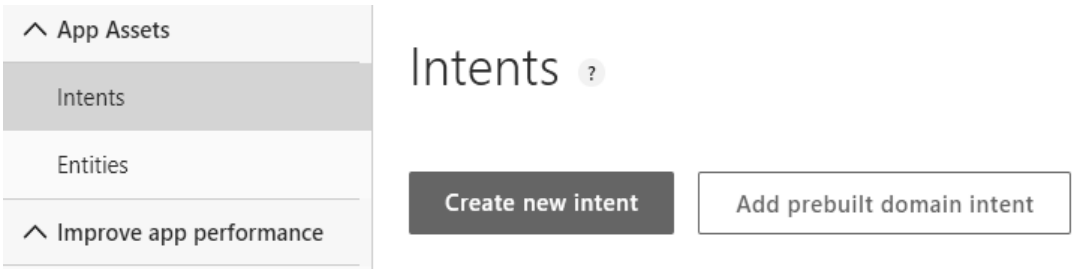

Figura 14. Interfaz para la creación de entidades en LUIS
Elaborado por: Christian Abata y Pedro Ramírez

La intención para pruebas será llamada Buscar**,** en ella se agregan varios ejemplos apegados al lenguaje natural y contengan una intención de búsqueda, así como una entidad como, por ejemplo: "busca el perfil de prayanime", "buscar a prayanime", "dime

los datos de prayanime". Donde prayanime será nuestra entidad persona, como se aprecia en la Figura 15.

Interfaz de entidades LUIS

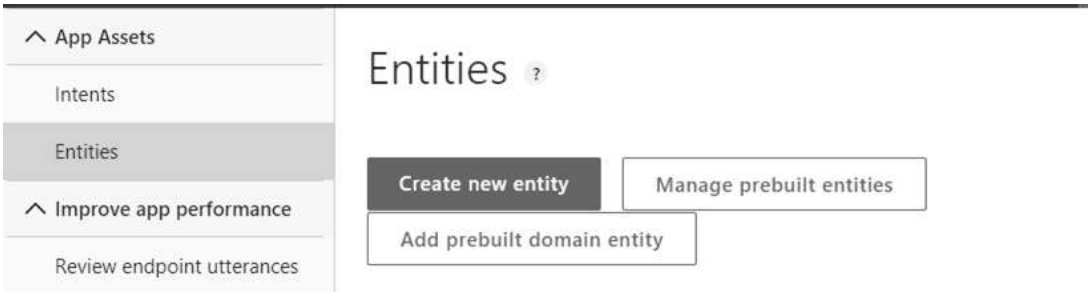

Figura 15. Interfaz para la creación de entidades en LUIS
Elaborado por: Christian Abata y Pedro Ramírez

Una vez agregadas las intenciones es necesario entrenar la aplicación dando clic en el botón Train. Paso seguido, se publicará dando clic en la opción del menú superior publish y después clic en el botón publish to production slot, para poder obtener un URL que será en enlace de comunicación del asistente con LUIS, como se aprecia en la Figura 16.

Interfaz para entregamiento y publicación

Figura 16. Interfaz para el entrenamiento y publicación de la aplicación
Elaborado por: Christian Abata y Pedro Ramírez

Con la configuración de LUIS terminada, el siguiente paso es crear la estructura del proyecto en el IDE de desarrollo Visual Studio Code con el cual se podrá codificar con el lenguaje de programación Node Js.

Los archivos que serán creados se llamarán:

package.JSON, que contendrá las instancias de las librerías necesarias para que el proyecto se pueda ejecutar, estas librerías son: Restify, Dotenv, HTTP, Botbuilder, querystring mismas que se instalan desde la consola integrada de Visual Studio Code con los comandos visualizados en la Figura 17.

Comandos para instalar librerías

```
npm install -s botbuilder restify
npm install --save-dev @types/restify
npm install dotenv --save
```

Figura 17. Comandos de instalación para librerías Node Js
Elaborado por: Christian Abata y Pedro Ramírez

conexion.js, que contendrá el código necesario para iniciar el servidor de pruebas locales junto con su puerto el cual estará en espera de la interacción con el usuario, ver Figura 18.

Activación de puertos locales

```
var server = restify.createServer();
        server.listen(process.env.port || process.env.PORT || 3978, function() {
            console.log('listening');
        });
```

Figura 18. Código para la activación de puertos locales
Elaborado por: Christian Abata y Pedro Ramírez

Además, se dará inicio al diálogo, como se aprecia en la Figura 19.

Inicio de un diálogo

```
bot.dialog('/', dialog);
```

Figura 19. Código para el inicio de un diálogo
Elaborado por: Christian Abata y Pedro Ramírez

dialogos.js, contendrá el código con el qué se hará la conexión con LUIS, como se aprecia en la Figura 20.

Conexión con LUIS

```
const model = process.env.LUIS_MODEL;
const recognizer = new builder.LuisRecognizer(model)
const dialog = new builder.IntentDialog({ recognizers: [recognizer] });
```

Figura 20. Código para el inicio de reconocedor de LUIS
Elaborado por: Christian Abata y Pedro Ramírez

La identificación de las intenciones y entidades junto a las funciones que se encargarán de extraer datos y mostrarlos al usuario como se tiene previsto, como se aprecia en la Figura 21.

Identificación de intenciones

```
module.exports = dialog
    .matches('Buscar', [
        confirmarQuery, buscarPerfiles
    ])
```

Figura 21. Código la identificación de intenciones y ejecución de funciones
Elaborado por: Christian Abata y Pedro Ramírez

Como se puede ver, “Buscar” es nuestra intención y la primera función confirmarQuery identifica la entidad sobre la pregunta escrita por el usuario la cual en LUIS fue nombrada como “persona”, como se aprecia en la Figura 22.

Identificación de entidades

```
var query = builder.EntityRecognizer.findEntity(args.entities, 'persona');
```

Figura 22. Código la identificación de entidades
Elaborado por: Christian Abata y Pedro Ramírez

buscarPerfiles es la función que mostrará la información extraída por medio de la función ejecutarBusqueda la misma que ejecuta otra función llamada cargarDatos encargada de realizar un método GET sobre la URL pública de GitHub como se puede ver en la Figura 23.

Función para la lectura de un API

```
function ejecutarBusqueda(query, callback) {
    cargarDatos('/search/users?q=' + querystring.escape(query), callback);
}
function cargarDatos(path, callback) {
    var options = {
        host: 'api.github.com',
        port: 443,
        path: path,
        method: 'GET',
        headers: {
            'User-Agent': 'sample-bot'
        }
```

Figura 23. Código la lectura de una URL que contiene una API tipo GET
Elaborado por: Christian Abata y Pedro Ramírez

Adicionalmente la información que se mostrará al usuario será por medio de una tarjeta de presentación misma, que se codifica en la función traerDatosPerfil, y se llama en la función buscar perfiles. Ver Figura 24.

Variable de creación de tarjetas

```
var thumbnails = profiles.items.map(function(item) { return traerDatosPerfil(session, item)});
                var message = new builder.Message(session).attachments(thumbnails).attachmentLayout('carousel');
                session.send(message);
```

Figura 24. Código para presentación de la información en tipo de tarjetas
Elaborado por: Christian Abata y Pedro Ramírez

inicio.js, contiene los llamados a las librerías cargadas en packgage.json y se realiza la llamada al archivo conexión que levantará el servidor como ya se mencionó anteriormente, como se aprecia en la Figura 25.

Variable que instancian librerías

```
var builder = require('botbuilder');
var http = require('http');
var querystring = require('querystring');
var restify = require('restify');
var dotenv = require('dotenv');
dotenv.load();
var ConnectorBot = require('./conexion');
var connectorBot = ConnectorBot.start();
```

Figura 25. Código para el llamado a las librerías de Node Js
Elaborado por: Christian Abata y Pedro Ramírez

.env, contiene el URL que conecta la aplicación con los servicios de LUIS, como se aprecia en la Figura 26.

Token de LUIS

```
LUIS_MODEL=https://westus.api.cognitive.microsoft.com/luis/v2.0/apps/e9a90b0c-775c-4168-81ec-7cec10778560?subscription-key=79448f1c5
```

Figura 26. Código identificar el token de acceso a LUIS
Elaborado por: Christian Abata y Pedro Ramírez

En cuanto a Kohonen, a pesar de ser un algoritmo con una amplia trayectoria, las librerías disponibles libremente no se encuentran preparadas para realizar una clasificación directa de términos. Por ejemplo, la solución propuesta por John McCullock recibe vectores de números enteros (McCullock, 2013).

En esta iteración, con el fin de seleccionar el lenguaje más eficiente para resolver la clasificación se utiliza una misma solución de clasificación dispuesta en varios lenguajes multipropósito como lo son .Net y Python.

El algoritmo propuesto por John McCullock (ver el anexo 1), realiza una clasificación de caracteres de diferentes fuentes siguiendo la siguiente secuencia:

1. Inicializa los vectores que representan a cada carácter.
2. Entrena a la RNA (Establece los pesos aleatorios inicialmente).
3. Recorre cada ejemplo, actualizando el vector de pesos e imprimiendo el respectivo resultado.

El cálculo de cada uno de los ejemplos responde a la regla de aprendizaje de los pesos, ejecutada de acuerdo con la función presentada en el Figura 27.

Aprendizaje de Pesos

```
def compute_input(self, vectorArray):
    self.d = [0.0] * self.mMaxClusters

    for i in range(self.mMaxClusters):
        for j in range(self.mVectorLen):
            self.d[i] += math.pow((self.w[i][j] - vectorArray[j]), 2)

    return
```

Figura 27. Aprendizaje de Pesos
Elaborado por: Christian Abata y Pedro Ramírez

2.5. Pruebas

Para ejecutar las pruebas dentro de la consola se debe tipear la palabra node y el nombre del archivo ejecutable llamado inicio junto a su extensión js haciendo referencia a Javascript.

Es necesario ejecutar el emulador Bot Framework Emulator y conectarlo al servidor local como se aprecia en la Figura 28.

Interfaz de conexión local

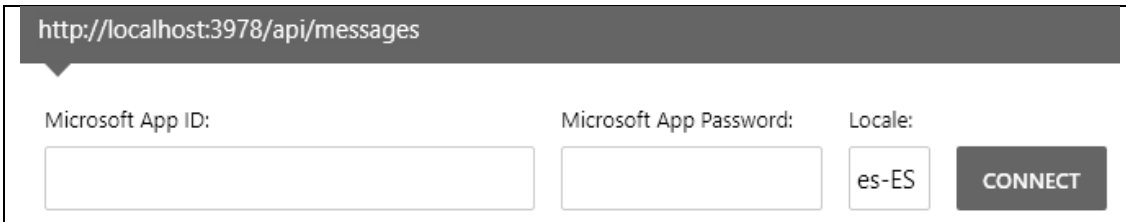

Figura 28. Interfaz de conexión para pruebas locales.
Elaborado por: Christian Abata y Pedro Ramírez

Una vez realizado todo el procedimiento anterior el servidor de pruebas se encontrará ejecutándose y el usuario ya podrá interactuar con el agente. Ver Figura 29.

Primera interacción del usuario

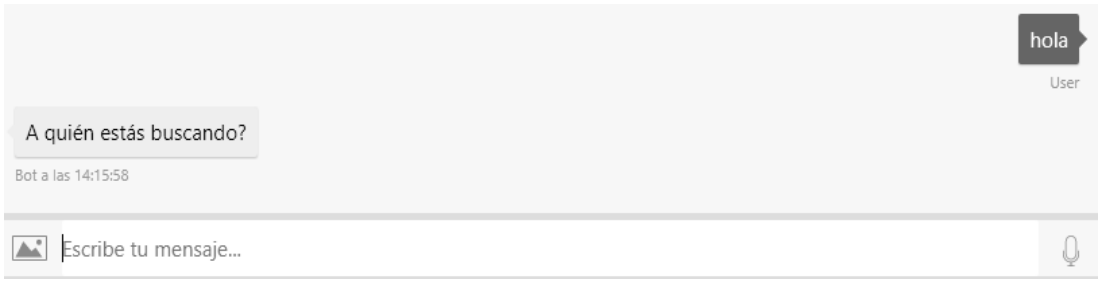

Figura 29. Inicio de la interacción usuario con agente
Elaborado por: Christian Abata y Pedro Ramírez

Y si el usuario escribe en la interfaz una oración con intención de búsqueda como se configuró en LUIS, la respuesta obtenida será el perfil del usuario de GitHub identificado como entidad como se muestra a en la Figura 30.

Respuesta de servicio GitHub

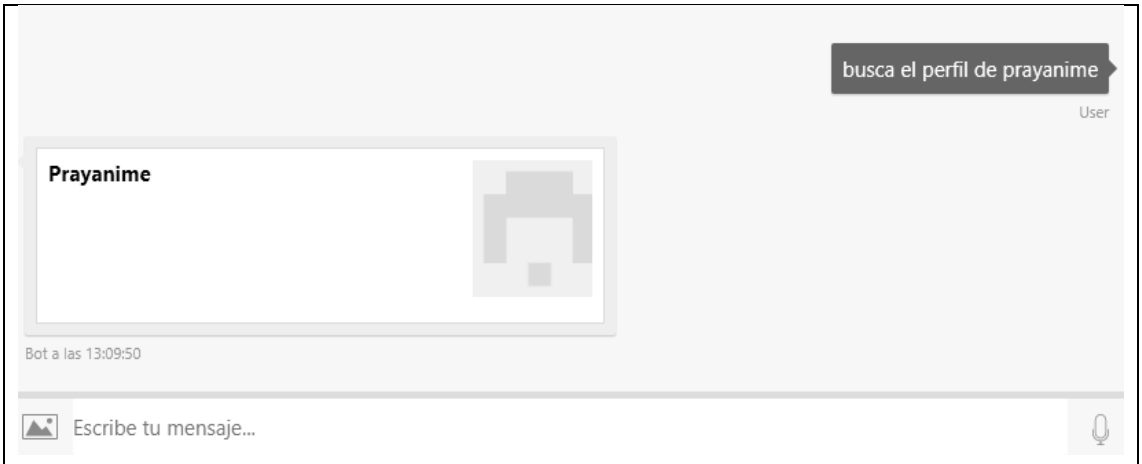

Figura 30. Respuesta tipo tarjeta del servicio GitHub
Elaborado por: Christian Abata y Pedro Ramírez

Por otro lado, John McCullock propone un algoritmo disponible en .Net, Java y Python. Por lo cual se probó su clasificación de términos, modificando el algoritmo para imprimir el tiempo de inicio, tiempo de fin y el tiempo de duración en terminar de ejecutar el algoritmo (todos en milisegundos).

Las pruebas realizadas se ejecutaron en .Net y en Python en un equipo cuyas características son:

- CPU: Intel® Core(TM) i5-3337U CPU @ 1.80GHz
- Memoria (RAM): 4.00 GB (3,88 GB utilizable)
- Disco: 117GB (Estado sólido)

Debido a la inicialización aleatoria de los pesos se realizaron 100 pruebas en cada lenguaje obteniendo los resultados visualizados en la Tabla 6.

Tabla 6. Resultados de las pruebas de Kohonen en .Net y Python.

	.Net	**Python**
Tiempo inicio medio	1525020705360,37	1.52502724371e+12
Tiempo fin medio	1525020705870,41	1.52502724646e+12
Duración media	510,04	2746,01

Nota: La tabla presenta los resultados de las pruebas de Kohonen en .Net y Python.

2.6. Resultados

- Hasta este punto la implementación de un agente que entienda y responda a una pregunta realizada por el usuario está completa, pero al estar tratando con la información de una fuente externa al proyecto como la de GitHub previamente creada con un método GET el cual está parametrizado para que realice búsquedas de perfiles dentro de su base de datos, el trabajo de pruebas se complica y se limita a usar lo disponible o a buscar en otras fuentes libres que sirvan para alimentar la base de conocimientos del agente. Además, la incertidumbre de poder encontrar información que se apegue al escenario de trabajo es muy alta, y en caso de tener éxito nuevamente se requiere designar un tiempo amplio para entender la documentación de dichos repositorios, lo que expone claramente una necesidad de creación de un repositorio de datos propios al proyecto, el cual se podrá manipular y publicar como API para poder ser consumido y probado las veces que sean necesarias.
- Las pruebas realizadas con el algoritmo de Kohonen, propuesto por McCullock, revelan que es más eficiente ejecutarlo en .Net. Sin embargo, es muy probable que exista una librería o código más eficiente y/o robusto con algún otro método de resolución de clasificación. Finalmente, en la siguiente iteración se considera necesario un ajuste en el algoritmo para la clasificación de términos y alinearlo a la clasificación de respuestas oportunas e inoportunas.
- Los diagramas BPMN describen procesos de negocio secuenciales, en el desarrollo de este proyecto demostraron ser de gran utilidad para el análisis de la interacción de componentes, facilitando la delimitación sobre ¿qué tarea deben realizar? y ¿cuándo lo deben hacer?

CAPÍTULO 3

Escenario

En cuanto a la información que se convertirá en el origen de los datos se plantea un escenario apegado a la educación, para ello es necesaria la búsqueda de repositorios API públicos que provean de información básica como el tratamiento de una nómina, información sobre la gestión de procesos, un inventario del cual se pueda disponer de información de adquisición de recursos físicos. Dependiendo de la disponibilidad de la información se pondrá en consideración el cambio de escenario apegado a la educación por uno más apegado a la información que se encuentre en los repositorios de API públicos.

Segunda Iteración

3.1. Requisitos y tecnologías

Para empezar a solventar la necesidad presentada al término de la primera iteración es necesario iniciar una selección de tecnologías las cuales vayan a aportar con objetivo de crear un repositorio API que contenga los datos necesarios para realizar las pruebas con un escenario empresarial y para esto existen varios requerimientos descritos en la Tabla 7.

Tabla 7. Requisitos, tecnologías y criterios de la segunda iteración

Requisitos	Tecnologías evaluadas	Tecnologías Escogida	Criterios
Orígenes de información y conocimiento.	SQL Server My SQL PostgreSQL Oracle SQL	SQL Server	Motor de Base de datos que se apague a la tecnología seleccionada en la nube. Implementación de API's con una misma codificación. Publicación de servicios desde una mismo IDE.
Orígenes de información y conocimiento.	Bases de datos API's Gestores de Contenido	API's	Información pública y disponible de forma inmediata para pruebas.

Requisitos	Tecnologías evaluadas	Tecnologías Escogida	Criterios
	Archivos planos		
Disponibilidad de la información	Servidor local de base de datos Servidor en la nube Azure Servidor en la nube Amazon Servidor en la nube Google	Azure	Acceso a recursos gratuitos. Disponibilidad 100% de la información en cualquier momento. Visibilizar la comparativa.
Inferencia de los resultados	Kohonen SVM	SVM	Eficiencia en clasificación de términos

Nota: Esta tabla contiene los requisitos, tecnologías y criterios considerados en la segunda iteración.

La necesidad de una base de datos en un escenario controlado es evidente y la primera interrogante se presenta al momento de elegir el motor de la misma, es aquí en donde empieza la evaluación de motores de bases de datos como se describe en el cuadro de requerimientos y se escoge a SQL Server, éste se puede integrar fácilmente con la creación de API's sin tener que escribir más código, a diferencia de haber seleccionado otro motor y además se integra rápidamente con los servicios en la nube provistos por Microsoft.

Tabla 8. Comparación entre tecnologías de la nube

	Azure	IBM	Google
Costo	5	0	0
Disponibilidad del motor SQL Server	5	5	5
Configuración del servidor	5	5	5
Facilidad para el levantamiento de servidor	5	5	5
Integración con Visual Studio	5	0	0
Total	25	15	15

Nota: Esta tabla contiene criterios de evaluación para seleccionar una tecnología de la nube.

Dado que el proyecto requiere información disponible siempre para la experimentación, la necesidad de usar un servicio en la nube es primordial, y como se lo menciona en el cuadro de requerimientos y se demuestra en la matriz multicriterio se evaluaron varias tecnologías de las cuales Azure es la propuesta que se usará por contar con una suscripción gratuita para estudiantes y se integra con Visual Studio. La selección de

Azure, se realizó en bajo los criterios presentados en la Tabla 8, en dónde 5 es disponible gratuitamente (para el desarrollo de este proyecto) y 0 que requiere una inversión.

Si bien, en la anterior iteración se tenía una preferencia porque el algoritmo sea de autoaprendizaje, porque no es necesario una etapa de entrenamiento extenuante, la eficiencia es el factor primordial. Durante la investigación realizada para identificar la aproximación matemática más eficiente se encontró un estudio lanzado recientemente publicado en octubre del 2017, llamado "Re-evaluating the need for Modelling Term-Dependence in Text Classification Problems" en éste se realiza una comparación entre diferentes soluciones para la clasificación de términos en dónde se muestra cómo claramente SVC (Support Vector Classification) es más eficiente (Banerjee, Majumder, & Mitra, 2017). Ver Figura 31. Por lo cual se decidió cambiar de algoritmo de clasificación a pesar de ser éste supervisado y por ende requerir el flujo de la información para poder mejorar la precisión del asistente en el transcurso del tiempo, crecimiento y madurez de la organización.

Plano de puntajes para todos los clasificadores

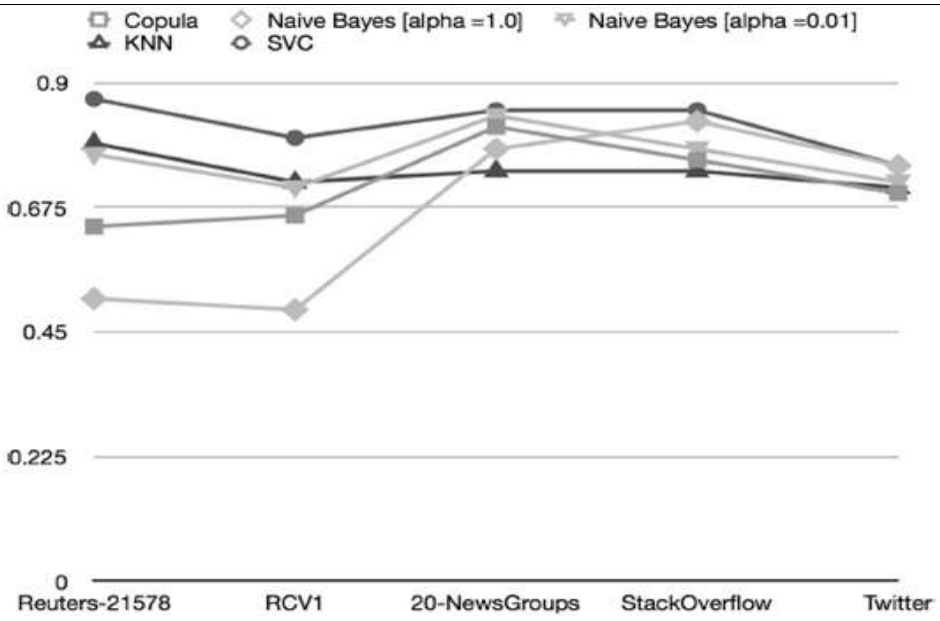

Figura 31. Plano de puntajes para todos los clasificadores
Elaborado por: Sounak Banerjee, Prasenjit Majumder y Mandar Mitra

3.2. Análisis

Para alcanzar el objetivo y la necesidad planteada para esta iteración es necesario empezar con la extracción de la información de repositorios públicos. Se inicia la búsqueda de API's que provean información en el escenario educativo, sin éxito.

Dada la necesidad de encontrar un camino para continuar con la extracción de información y futura creación de nuestra base de datos, se evalúan las API's que brinden más información y se encuentren en el escenario médico es el que más provee información apegada al propósito del agente, que es brindar respuestas en un entorno empresarial de tipo pequeña o mediana empresa.

Con la información obtenida se analiza los actores, quienes deben interactuar con el agente en este escenario médico, los mismos que serán: doctores, enfermeras y pacientes. Terminado dicho análisis es evidente que durante el proceso será necesario conocer información sobre pacientes, doctores, enfermedades y medicinas o tratamientos para esas enfermedades. En esta iteración resulta imprescindible identificar

la interacción entre los subcomponentes de los orígenes de datos. Ver Tabla 9 y Figura 32.

Tabla 9. Interacciones Origen de datos – cliente REST y usuario – componente de tratamiento de la información

Escenario	Descripción	Descripción sub-escenario
Interacción Origen de datos - CLIENTE REST	Un cliente HTTP hace el consumo de la información de la base de datos por medio de API's REST	**Interacción: base de datos – API's REST** Los datos se encuentran alojados en un origen de datos y son publicados como API's REST.
		Interacción: API's REST – cliente HTTP Un ente externo puede consumir el origen que contiene la información mediante una petición HTTP.
Interacción usuario/interfaz - componente de tratamiento de la información	Por medio de la interfaz realiza una pregunta, que pasa por el componente de tratamiento de información, en donde: - Se realiza una comprensión de la petición (pregunta) del usuario. - Se realiza la consulta de la información en los orígenes de datos competentes. - Se analiza las respuestas obtenidas y se discierne las más oportunas para el usuario. Obtiene retroalimentación sobre las respuestas presentadas al usuario y aprende futuras consultas.	**Interacción: Componente de tratamiento de información - Origen de datos** El componente de tratamiento de información traduce la pregunta del usuario en una petición o consulta de acuerdo con el tipo de origen de datos. El origen de datos provee toda la información que posee.
		Interacción: servicio cognitivo de búsqueda y compresión de lenguaje natural – sub-componente de inferencia LUIS comprende la petición del usuario, traduce la pregunta a una consulta sobre el origen de datos, al obtener las respuestas envía las mismas al componente de inferencia que discierne las respuestas, devuelve las mismas y envía al usuario final. En la retroalimentación el sub-componente de inferencia aprende de la respuesta que considera más oportuna el usuario.

Nota: La tabla muestra la interacción de subcomponentes entre el Origen de datos – cliente REST y usuario – componente de tratamiento de la información.

Diagrama de procesos: Origen de datos - cliente REST

Interacción: Origen de datos cliente REST

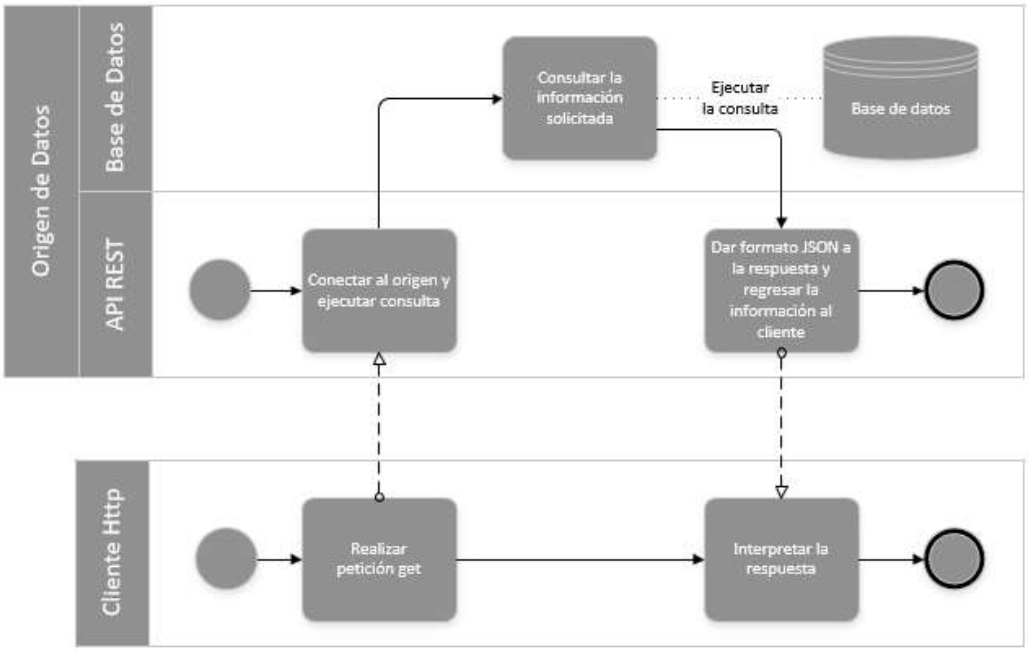

Figura 32. Interacción: Origen de datos – cliente REST
Elaborado por: Christian Abata y Pedro Ramírez

En cuanto al componente de tratamiento de información, sería necesario una modificación en el proceso puesto que ahora se debería realizar un entrenamiento progresivo de acuerdo con una retroalimentación por parte del usuario, como se aprecia en la Figura 33.

Diagrama de procesos: Interacción usuario – componentes del prototipo de asistente virtual

Interacción: Usuario – componentes del prototipo de asistente virtual

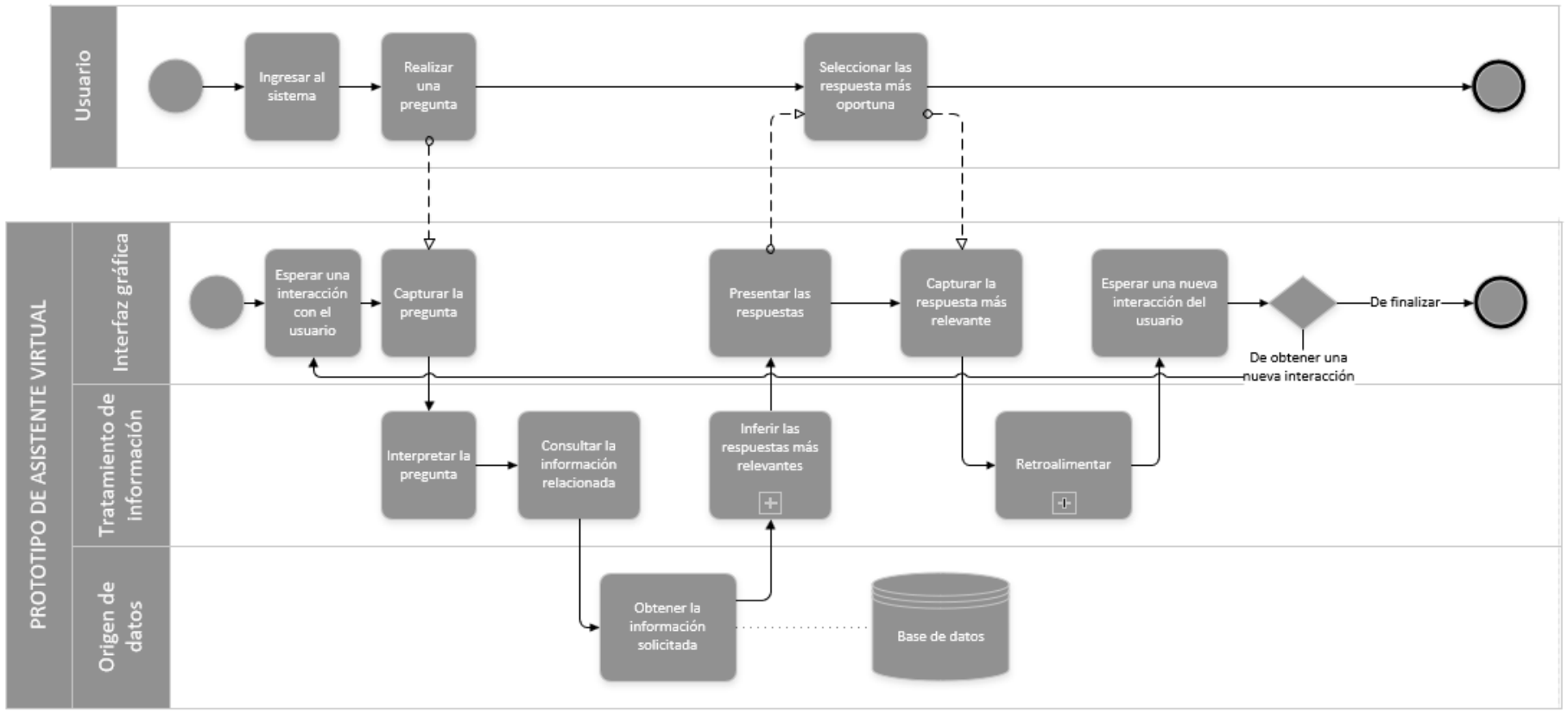

Figura 33. Interacción: Usuario – componentes del prototipo de asistente virtual
Elaborado por: Christian Abata y Pedro Ramírez

3.3. Diseño

En la primera iteración se presentaron dos diferentes perspectivas para visualizar la solución informática, la primera en un nivel arquitectónico de software y la segunda en un nivel modular sobre el producto.

En esta iteración se destaca en un nivel arquitectónico más específico sobre el componente de Origen de información y conocimiento, como se aprecia en la Figura 34.

Segunda propuesta arquitectónica

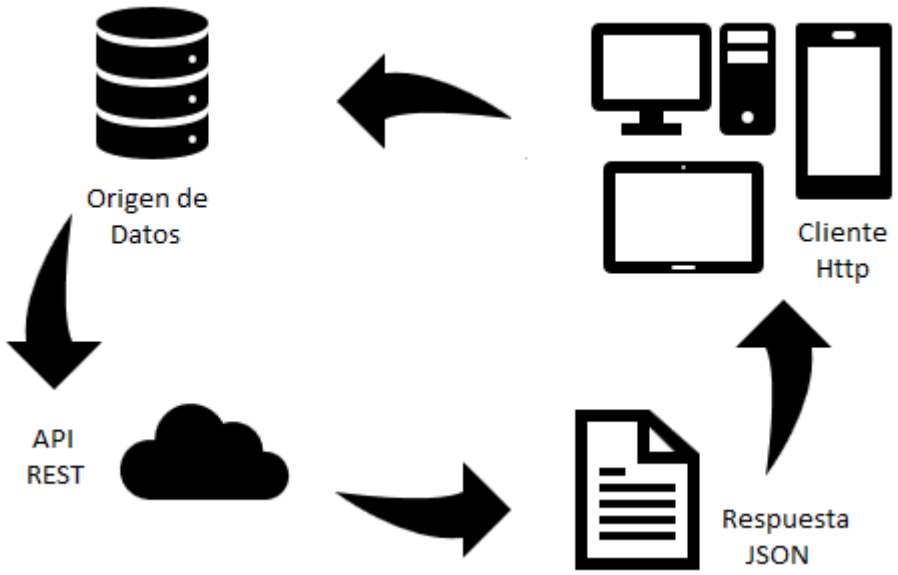

Figura 34. Propuesta arquitectónica considerando los componentes de la nube
Elaborado por: Christian Abata y Pedro Ramírez

3.4. Codificación

Para empezar con la extracción es necesario el uso del aplicativo Postman, este permite realizar las consultas tipo GET de los recursos API públicos, para ello es necesario el URL y configurar el aplicativo para que realice la llamada como se aprecia en la Figura 35.

Captura de interfaz Postman

Figura 35. Apartado para colocar la URL que direcciona a un API
Elaborado por: Christian Abata y Pedro Ramírez

La información entregada por el API deberá ser cargada por medio de una cuenta cuyo registro se debe realizar en la web https://www.cliniko.com/ y dependiendo de lo que se guarde en esta cuenta obtendremos datos como lo muestra su repositorio de GitHub https://GitHub.com/redguava/cliniko-api/blob/master/sections/patients.md en el apartado *example response*, como se aprecia en la Figura 36.

Respuesta de JSON


```
{
  "patients": [
    {
      "accepted_privacy_policy": true,
      "address_1": "1 Smith Street",
      "address_2": "",
      "address_3": "",
      "archived_at": "",
      "city": "Melbourne",
      "country": "Australia",
      "created_at": "2013-03-26T14:00:00Z",
      "date_of_birth": "2001-05-26",
      "deleted_at": "",
      "email": "peter@example.com",
      "emergency_contact": "",
      "first_name": "Peter",
      "invoice_default_to": "Super Insurance",
      "invoice_email": "super.insurance@example.com",
      "invoice_extra_information": "Insurance #123456\r\nClaim #123456",
      "gender": "Male",
      "id": 1,
      "last_name": "Patientman",
      "medicare": "",
      "notes": "",
      "occupation": "",
      "old_reference_id": "",
      "post_code": "3000",
      "referral_source": "",
      "reminder_type": "SMS & Email",
```


Figura 36. Respuesta JSON que entrega el servicio GET de Cliniko
Elaborado por: Christian Abata y Pedro Ramírez

La información, obtenida se filtra de manera que sólo se usan los campos necesarios para obtener la estructura de doctores, pacientes, enfermedades y medicinas. La base de datos es simulada a partir de la información obtenida del API pública responde al modelo relacional de la Figura 37.

Modelo entidad relación planteado para la base de datos

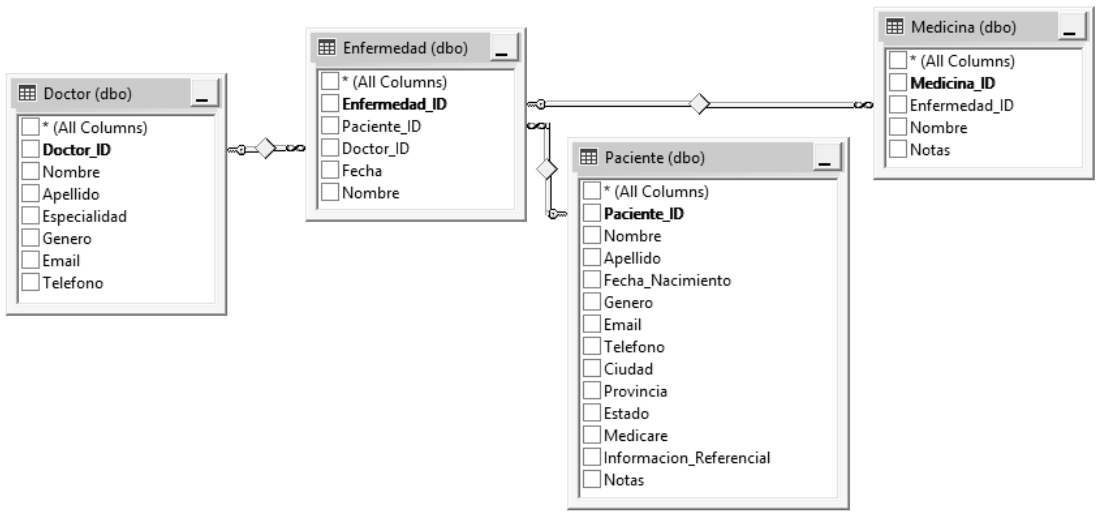

Figura 37. Modelo relacional de la base de datos de prueba
Elaborado por: Christian Abata y Pedro Ramírez

En la creación de la base de datos para publicar la información en servicios REST (y por la disponibilidad de aplicaciones en Azure) se creó un proyecto con una plantilla web API y MVC en C# .Net.

En este proyecto se siguen los siguientes pasos:

1. Se crean las clases objeto como modelos y se compila la solución.
2. Se generan los controladores. Ver Figura 38.

Estructura del proyecto WebApiHistorialMedico

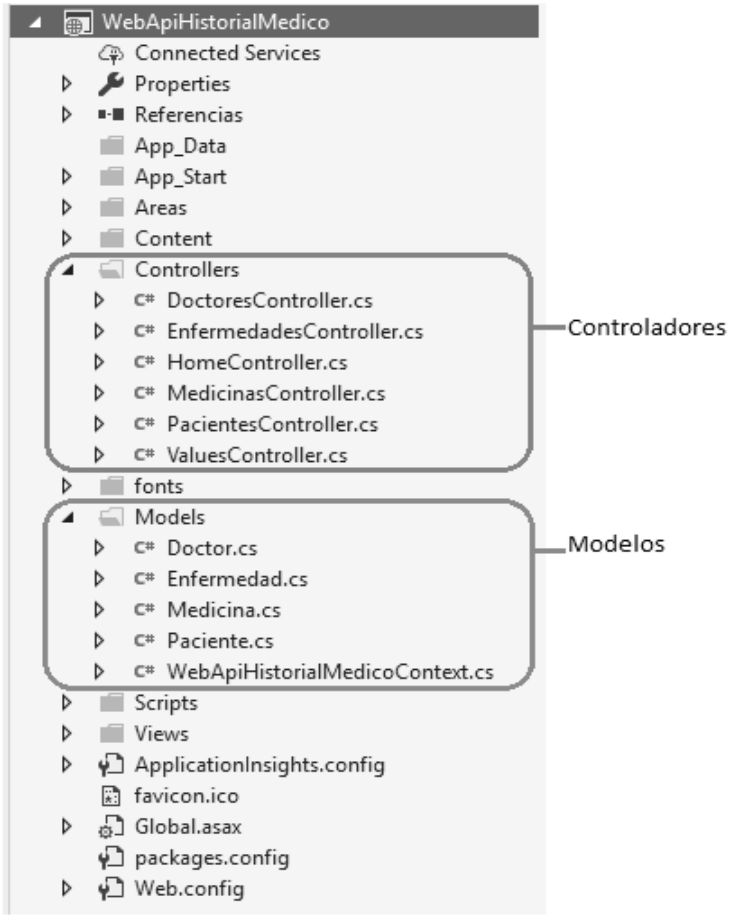

Figura 38. Estructura de archivos del proyecto API simulado
Elaborado por: Christian Abata y Pedro Ramírez

3. Se ingresa la cadena de conexión a la base de datos en el fichero web.config (alojada en Microsoft Azure). Ver Figura 39.

Cadena de conexión hacia la base de datos

```
<connectionStrings>
  <add name="WebApiHistorialMedicoContext" providerName="System.Data.SqlClient"
       connectionString="Server=tcp:proyectog.database.windows.net,1433;Initial
       Catalog=HistorialMedico;Persist Security Info=False;User ID=pramirez;
       Password=Hmedico01!;MultipleActiveResultSets=False;Encrypt=True;
       TrustServerCertificate=False;Connection Timeout=30;" />
</connectionStrings>
```

Figura 39. Cadena de conexión hacia la base de datos creada
Elaborado por: Christian Abata y Pedro Ramírez

4. Se publica el proyecto (seleccionando el recurso correspondiente). Ver Figura 40.

Publicación del Servicio

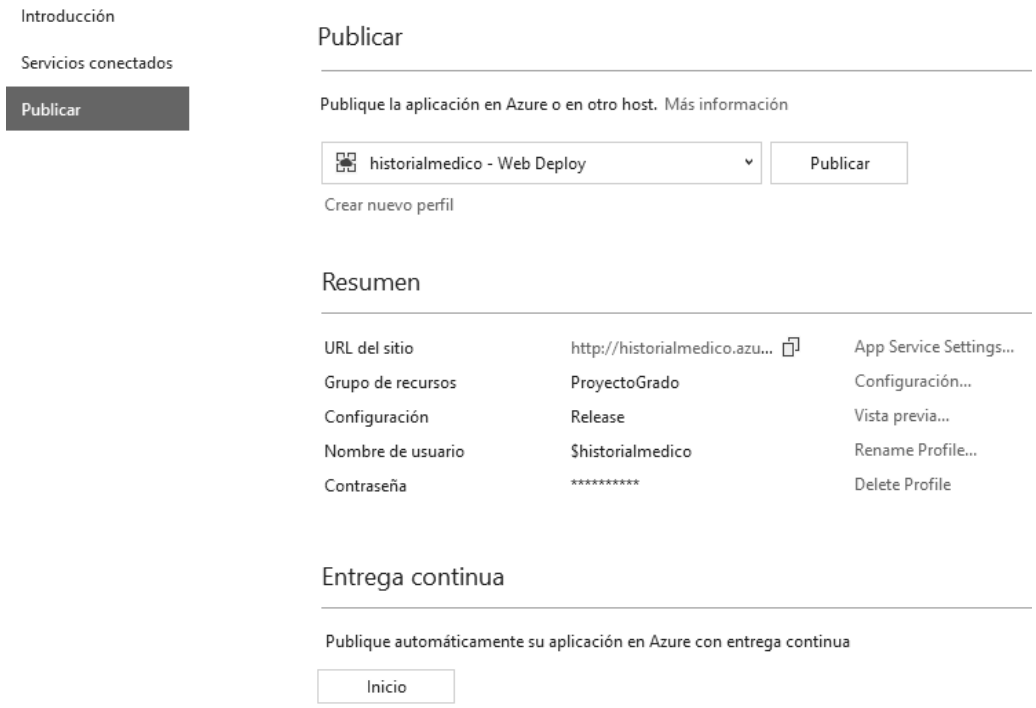

Figura 40. Interfaz para la publicación del servicio
Elaborado por: Christian Abata y Pedro Ramírez

A continuación, se creó un nuevo proyecto para la simulación de datos, éste se conecta directo a la base de datos y hace inserciones bajo distribuciones uniformes explicadas en cada paso.

En este nuevo proyecto se utiliza como base las mismas clases objeto del proyecto web API. Agregando métodos para la generación simulada de datos. En el caso de doctores:

1. Se crean arreglos de cadenas que contienen nombres y especialidades.
2. Se recorre un arreglo en donde se genera aleatoriamente un número que simula el 40% de registros correspodientes a un género masculino.

3. Se genera un número aleatorio para la simulación de un 70% de doctores con una especialidad en Medicina General.
4. Se crean los correos mediante la primera letra del nombre, seguido del apellido y el dominio.
5. Se genera aleatoriamente un número de teléfono.
6. Finalmente, debido a que existe la probabilidad de repitencia de más de un médico con el mismo nombre y apellido, se agrupa por email reduciendo el número de registros.

Con la implementación de API's concluida, se procede a realizar pruebas en el código terminado en la primera iteración cambiando los parámetros de la API pública de GitHub por la generada con el servicio de Azure historial médico.

Los primeros parámetros en cambiar son los de URL en la función cargarDatos como se aprecia en la Figura 41.

Función cargarDatos modificada

```
function cargarDatos(path, callback) {
    var options = {
        host: 'historialmedico.azurewebsites.net',
        port: 443,
        path: path,
        method: 'GET',
        headers: {
            'User-Agent': 'sample-bot'
        }
    };
```

Figura 41. Función modificada para cargar los datos del API generada y publicada para pruebas
Elaborado por: Christian Abata y Pedro Ramírez

Y los de la función ejecutarBusqueda, como se aprecia en la Figura 42.

Función ejecutarBusqueda modificada

```
function ejecutarBusqueda(username, callback) {
    cargarDatos('/api/Pacientes/1' + querystring.escape(username), callback);
}
```

Figura 42. Función modificada para ejecutar una búsqueda
Elaborado por: Christian Abata y Pedro Ramírez

Además, es necesario modificar la forma en la que se mapean los datos en la función traerDatosPerfil con respecto al nuevo URL, los cuales serán tomados de acuerdo con la necesidad de la respuesta y lo que quiera ver el usuario como, por ejemplo, para este caso en particular los campos de Nombre, Apellido, Cuidad, y Notas como se aprecia en la Figura 43.

Función traerDatosPerfil modificada

```
function traerDatosPerfil(session, profile) {
    var thumbnail = new builder.ThumbnailCard(session);
    thumbnail.title(profile.Nombre);

    if(profile.Nombre) thumbnail.subtitle(profile.Nombre);

    var text = '';
    if (profile.Apellido) text += profile.Apellido + ' \n';
    if (profile.Ciudad) text += profile.Ciudad + ' \n';
    if (profile.Notas) text += profile.Notas;
```

Figura 43. Función modificada para extraer ciertos datos del API
Elaborado por: Christian Abata y Pedro Ramírez

Otro cambio necesario se da en la función buscar perfiles que a diferencia de nuestra primera versión no existe la necesidad de usar el método map para mostrar los datos, como se aprecia en la Figura 44.

Función buscarPerfiles modificada

```
function buscarPerfiles(session, results, next) {
    var username = results.response;
    if (username.entity) username = session.dialogData.username = username.entity;
    else session.dialogData.user = username;
    if (!username) {
        session.endDialog('Petición cancelada..');
    } else {
        ejecutarBusqueda(username, (profile) => {
            if (profile && profile.message !== 'No encontrado') {
                var message = new builder.Message(session).attachments([traerDatosPerfil(session, profile)]);
                session.send(message);
```

Figura 44. Función modificada para presentar los datos al usuario en forma de tarjetas
Elaborado por: Christian Abata y Pedro Ramírez

Ya que se dispone de un servicio web que permite la inserción de nuevos registros en la tabla de Doctor. Haciendo el llamado al método generador de registros de doctores, como se aprecia en la Figura 45.

Método generador de registros

```
public static async Task RunAsync()
{
    client.BaseAddress = new Uri("https://historialmedico.azurewebsites.net/");
    client.DefaultRequestHeaders.Accept.Clear();
    client.DefaultRequestHeaders.Accept.Add(
        new MediaTypeWithQualityHeaderValue("application/json"));
    List<Doctor> doctores = generarDoctores(250);
    foreach (var doctor in doctores)
    {
        var url = await crearDoctor(doctor);
        Console.WriteLine(url);
    }
}
```

Figura 45. Método que generar los registros de doctores
Elaborado por: Christian Abata y Pedro Ramírez

Para la simulación las enfermedades, es necesario identificar al doctor y al paciente que se atendió en dicha enfermedad. Para esto se recurrió a un fichero CSV con dos columnas, la primera sobre una especialidad médica y la segunda especificaba las enfermedades que suelen atender. Y para ello se siguió los siguientes pasos:

1. Se consulta los doctores que ya se encuentran registrados en la base de datos.
2. Se genera un número aleatorio de enfermedades que se asignará a cada paciente entre 2 y 6.

3. Se generaron fechas aleatorias de atención de la enfermedad en base a días entre 1 y 28, meses entre 1 y 12 y años entre 1989 y 2018.
4. Se genera un número aleatorio para obtener un doctor aleatorio.
5. Se leyó el fichero de Especialidades – Enfermedades y se extrajo las enfermedades que puede atender el doctor de acuerdo con su especialidad.
6. Se extrae una enfermedad al azar de la lista de enfermedades.
7. Finalmente, se llama al método generador de enfermedades y se registra cada una mediante una conexión directa a la base de datos (con el fin de disminuir el tiempo de carga de los datos). Ver Figura 46.

Métodos de generación de enfermedades

```
public void crearEnfermedad(Enfermedad appointment)
{
    EnfermedadTableAdapter adapter = new EnfermedadTableAdapter();
    adapter.InsertQuery(
        appointment.Paciente_ID,
        appointment.Doctor_ID,
        appointment.Fecha,
        appointment.Nombre
        );
}

public void run()
{
    List<Enfermedad> enfermedades = generarEnfermedades();
    foreach (var enfermedad in enfermedades)
    {
        try
        {
            crearEnfermedad(enfermedad);
        }
        catch (Exception ex)
        {
            Console.WriteLine(enfermedad.Paciente_ID);
            Console.WriteLine(ex.Message);
        }
    }
}
```

Figura 46. Método que genera los registros de enfermedades
Elaborado por: Christian Abata y Pedro Ramírez

Para la simulación de medicinas, igualmente se posee un fichero CSV que posee las especialidades, medicamentos y observaciones. Y se siguen los siguientes pasos:

1. Se consultan las enfermedades registradas.
2. Se consultan los doctores registrados.
3. Se lee el fichero CSV.
4. Se genera un número aleatorio.
5. Se consulta la especialidad del doctor que atienda cada enfermedad registrada.
6. Se consultan las medicinas y observaciones que suelen sugerir dichas especialidades.
7. Se agrega cada una de las medicinas a esa enfermedad en específico.
8. Una vez generados todos los registros de medicinas se graban en la base de datos con una conexión directa. Ver Figura 47.

Métodos de generación de medicinas

```
public void crearMedicina(Medicine medicine)
{
    MedicinaTableAdapter adapter = new MedicinaTableAdapter();
    adapter.Insert(
        medicine.Enfermedad_ID,
        medicine.Nombre,
        medicine.Notas
        );
}

public void run()
{
    List<Medicine> medicines = generarMedicina();
    foreach (var medicine in medicines)
    {
        try
        {
            crearMedicina(medicine);
        }
        catch (Exception ex)
        {
            Console.WriteLine(medicine.Enfermedad_ID);
            Console.WriteLine(ex.Message);
        }
    }
}
```

Figura 47. Método que genera los registros de medicinas
Elaborado por: Christian Abata y Pedro Ramírez

En el componente de inferencia el código implementado para el algoritmo SVM, se recurrió al repositorio del profesor Chih-Jen Lin quien recopila varias opciones de interfaces y extensiones de LIBSVM (librería de SVM), de la cual se seleccionó la propuesta por Nicolas Panel desarrollada en C#. Para lo cual se utilizaron los siguientes criterios:

- El tiempo de respuesta obtenido en la anterior iteración por Kohonen.
- Alexandre Kowalczyk, desarrollador de software con amplia experiencia en la implementación de SVM, propone una solución que permite hacer la clasificación de términos mediante la utilización de la librería de Nicolas Panel.

El código propuesto por Kowalczyk (ver anexo 1) inicialmente crea un vocabulario a partir de todas las diferentes palabras encontradas en cada uno de los ejemplos, que a su vez se alojan en el fichero CSV y se construye el problema. Ver Figura 48.

Lectura de los ejemplos y etiquetas, y construcción del problema

```
const string dataFilePath = @"..\..\..\PruebasSVM\Entrenamiento.csv";
var dataTable = DataTable.New.ReadCsv(dataFilePath);
List<string> x = dataTable
    .Rows
    .Select(row => row["Descripcion"])
    .ToList();
double[] y = dataTable
    .Rows
    .Select(row => double.Parse(row["Soleado"]))
    .ToArray();
var vocabulary = x.SelectMany(GetWords)
    .Distinct()
    .OrderBy(word => word)
    .ToList();
var problemBuilder = new TextClassificationProblemBuilder();
var problem = problemBuilder.CreateProblem(
    x,
    y,
    vocabulary.ToList()
    );
```

Figura 48. Lectura de las características de los ejemplos y sus etiquetas, y construcción del problema a partir del vocabulario
Elaborado por: Christian Abata y Pedro Ramírez

En la Figura 49, se crea un modelo a partir del problema, se define el *kernel* y se obtiene la precisión del modelo.

Generación del modelo

```
const double C = 1;
var model = new C_SVC(problem, KernelHelper.LinearKernel(), C);
var accuracy = model.GetCrossValidationAccuracy(10);
Console.Clear();
Console.WriteLine("Exactitud del modelo: {0:P}", accuracy);
```

Figura 49. Generación del modelo y cálculo de su exactitud
Elaborado por: Christian Abata y Pedro Ramírez

En la Figura 50, se establece el valor correspondiente a las diferentes clases: soleado (1) y lluvioso (-1).

Etiquetas de predicción

```
_predictionDictionary = new Dictionary<int, string> {
    { -1, "Lluvioso" },
    { 1, "Soleado" }
};
```

Figura 50.Etiquetas para la predicción
Elaborado por: Christian Abata y Pedro Ramírez

Ejecución de pruebas

```
const string dataFilePathTest = @"..\..\..\PruebasSVM\Test.csv";
var dataTableTest = DataTable.New.ReadCsv(dataFilePathTest);
List<string> x_Test = dataTableTest
    .Rows.Select(row => row["Descripcion"]).ToList();
double[] y_Test = dataTableTest
    .Rows.Select(row => double.Parse(row["Soleado"])).ToArray();
double[] y_Hat = new double[y_Test.Count()];
for (int i = 0; i < x_Test.Count(); i++)
{
    var newX = TextClassificationProblemBuilder.CreateNode(
        x_Test[i],
        vocabulary
        );
    var predictedY = model.Predict(newX);
    y_Hat[i] = predictedY;
    Console.WriteLine(
        x_Test[i]+" \n Es {0}",
        _predictionDictionary[(int)predictedY]);
}
```

Figura 51. Ejecución de pruebas
Elaborado por: Christian Abata y Pedro Ramírez

Se modifica la predicción, de tal forma que se lee un fichero CSV de pruebas. Y se procede a la predicción típica por cada uno de los ejemplos, en dónde se crea un nodo y se realiza la predicción. Ver Figura 51.

En la Figura 52, se añadió un método para calcular el error de clasificación.

Método para calcular el Error de Clasificación

```
public static double ErrorClasificacion(double[] y, double[] yhat)
{
    double ec = 0.0;
    int equivocacion = 0;
    for (int i = 0; i < y.Count(); i++)
    {
        if (y[i] != yhat[i])
        {
            equivocacion += 1;
        }
    }
    Console.WriteLine(new string('=', 50));
    Console.WriteLine("Error de Clasificación");
    Console.WriteLine("Equivocaciones: "+equivocacion);
    Console.WriteLine("Ejemplos: "+ y.Count());
    ec = equivocacion/(double)y.Count();
    return ec;
}
```

Figura 52.Método para el error de clasificación.

Kowalczyk en su solución propone dos métodos fundamentales para ajustar la clasificación de términos en SVM.

El primero para la creación de un nodo. Éste básicamente trata al ejemplo para convertirlo en uno que contenga valores numéricos interpretables por SVM. En cada uno de los ejemplos, tanto durante el entrenamiento y las pruebas se calcula el número de veces que cada palabra del vocabulario se encuentra en el ejemplo.

El segundo para construir el problema, ejecutado sólo en el entrenamiento, llama a la creación de un nodo para cada uno de los ejemplos y lo asigna al vector de características, en tanto asigna a la etiqueta el valor real de dicho ejemplo 1 o -1 (soleado o lluvioso).

3.5. Pruebas

Una vez cargada la información en la base de datos Microsoft Azure, se probó el funcionamiento de cada uno de los API's REST.

Para lo cual se utilizó como cliente HTTP a Postman, teniendo como resultado lo que se aprecia en la Figura 53, 54, 55 y 56.

Resultado del API Pacientes en Postman

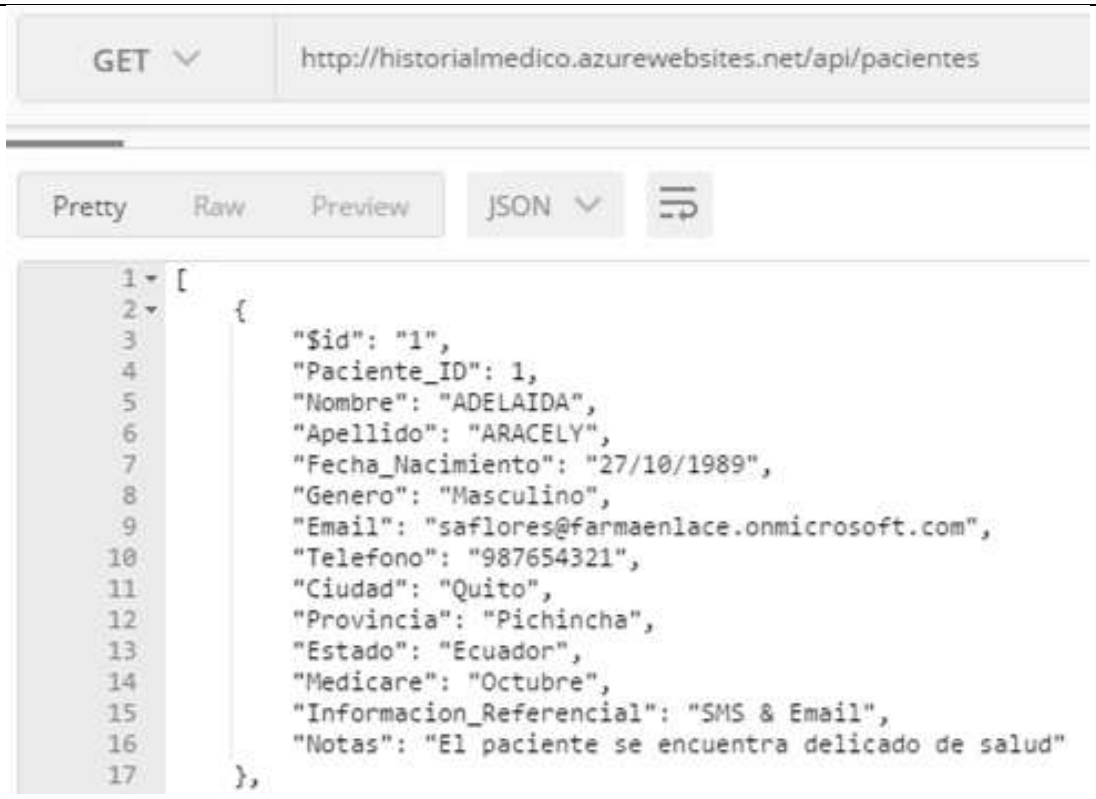

Figura 53. Respuesta JSON del API pacientes
Elaborado por: Christian Abata y Pedro Ramírez

Resultado del API doctores en Postman

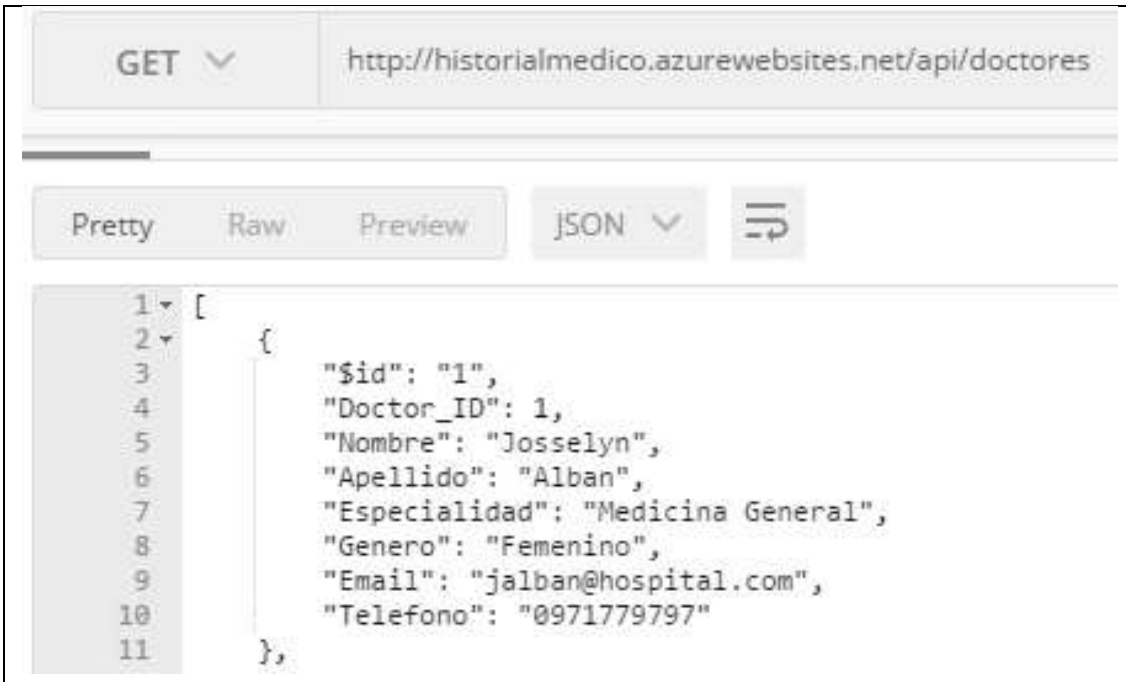

Figura 54. Respuesta JSON del API doctores
Elaborado por: Christian Abata y Pedro Ramírez

Resultado del API enfermedades en Postman

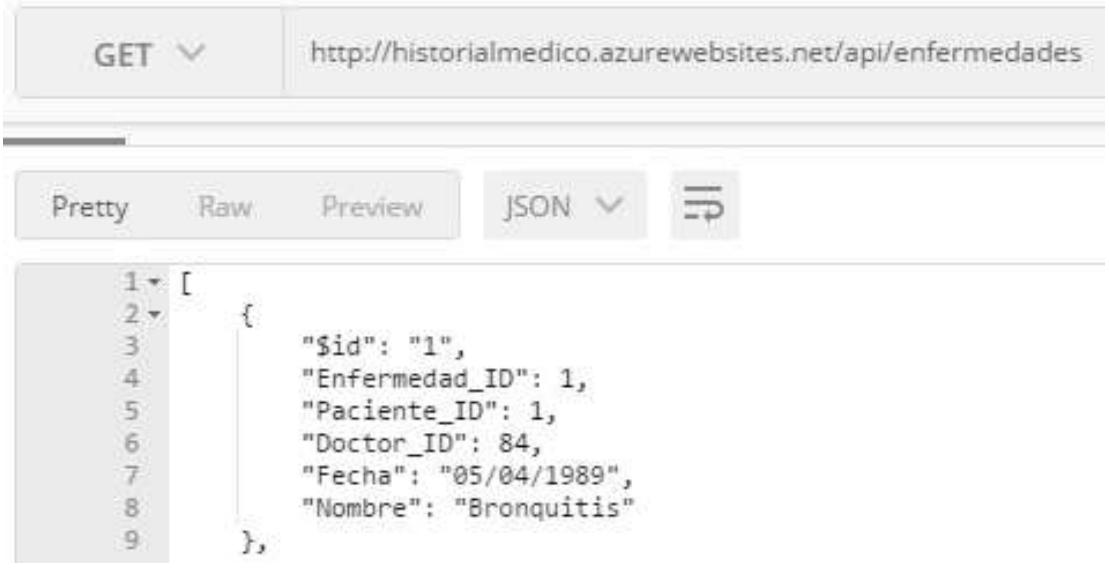

Figura 55. Respuesta JSON del API enfermedades
Elaborado por: Christian Abata y Pedro Ramírez

Resultado del API medicinas en Postman

Figura 56. Respuesta JSON del API medicinas
Elaborado por: Christian Abata y Pedro Ramírez

Es importante mencionar que durante el desarrollo se realizó una prueba exitosa de un método POST para subir la información de los doctores. Por lo cual se encuentran disponibles en el servidor las peticiones de carga (POSTs) y obtención de información (GETs) para cada una de las tablas de la base de datos.

Tras realizar todos los cambios necesarios en el código, la pantalla muestra la información que fue mapeada de acuerdo con la pregunta del usuario, como se aprecia en la Figura 58.

Resultado en Bot Framework Emulator con APIs Sumuladas

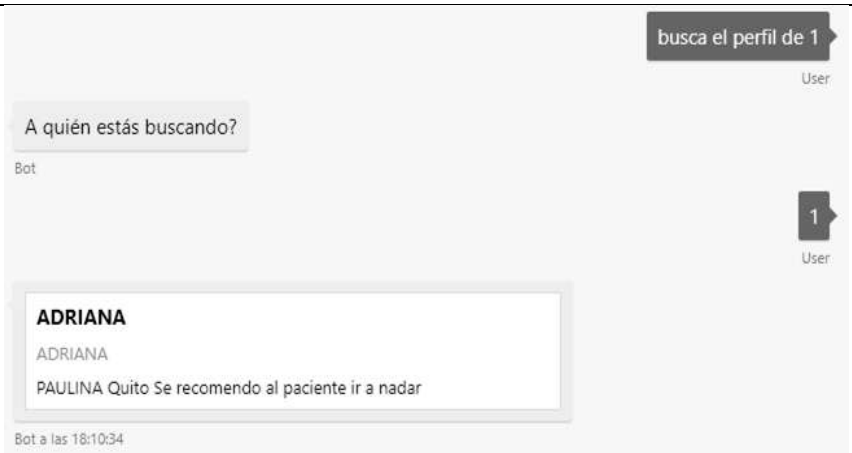

Figura 57. Resultado después de la implementación de API simulada
Elaborado por: Christian Abata y Pedro Ramírez

Pero como se puede ver, dicha información hace referencia a un ID más no a un nombre y esto se debe a que el API de pruebas construido anteriormente no consideró una búsqueda por perfiles o por un campo en específico sino más bien la entrega de la información general de la base de datos.

Esto hace que para efectos de estas pruebas el trabajo de crear un API que sí realice una función de búsqueda en específico sea más grande sin considerar que sólo se está trabajando para una URL sin hacer ningún tipo de relación.

Para las pruebas de SVM, se utilizaron los ficheros de entrenamiento y pruebas como se puede ver en la Figura 58 y Figura 59 respectivamente. En el último se escriben los resultados reales con el objetivo de validar el error de clasificación, cuyos resultados se encuentran en la Figura 60.

Ejemplos de entrenamiento

```
Descripcion, Soleado
soleado,1
lluvioso,-1
soleado soleado,1
lluvioso soleado,-1
soleado lluvioso,-1
lluvioso lluvioso,-1
soleado soleado soleado,1
soleado soleado lluvioso,1
soleado lluvioso lluvioso,-1
```

Figura 58. Ejemplos de entrenamiento
Elaborado por: Christian Abata y Pedro Ramírez

Resultados de pruebas SVM

```
Descripcion, Soleado
soleado lluvioso soleado,1
lluvioso soleado soleado,1
lluvioso lluvioso soleado,-1
lluvioso soleado lluvioso,-1
soleado lluvioso lluvioso,-1
lluvioso lluvioso lluvioso,-1
```

Figura 59. Resultado pruebas SVM
Elaborado por: Christian Abata y Pedro Ramírez

Exactitud del modelo y error de clasificación

```
Exactitud del modelo: 77,78 %
soleado lluvioso soleado
 Es Soleado
lluvioso soleado soleado
 Es Soleado
lluvioso lluvioso soleado
 Es Lluvioso
lluvioso soleado lluvioso
 Es Lluvioso
soleado lluvioso lluvioso
 Es Lluvioso
lluvioso lluvioso lluvioso
 Es Lluvioso
=====================================================
Error de Clasificación
Equivocaciones: 0
Ejemplos: 6

Error de Clasificación: 0
```

Figura 60. Resultado de error de clasificación y exactitud del modelo
Elaborado por: Christian Abata y Pedro Ramírez

A pesar de tener una precisión del 77,78%, los resultados obtenidos en la clasificación son muy satisfactorios al obtener un error de clasificación del 0%. En adición, si bien es cierto que los ejemplos son pequeños, la intención de estas pruebas eran validar la utilidad de SVM para la clasificación de términos.

3.6. Resultados

- Al trabajar con un API se muestra muy clara la necesidad que tendría una empresa con respecto a su nivel de madurez tecnológica el cual debe estar en Integración para asegurar que el departamento de TI este muy familiarizado con la creación de métodos tipo GET los cuales entreguen información que se apegue a las preguntas de un usuario, por eso para esta iteración el trabajar con API sin parametrizar un campo de búsqueda no resulta eficiente.
- Las respuestas que se muestran al usuario se encuentran limitadas a un solo URL ya que un API no considera las relaciones existentes cuando un usuario necesite una respuesta, por ello es necesario experimentar con un método más nativo y versátil para el consumo de datos como el lenguaje SQL.
- SVM es un algoritmo muy potente de clasificación, se puede ajustar fácilmente a la clasificación de respuestas, sin embargo, en una siguiente iteración será necesario cambiar el ambiente en donde resuelve la clasificación y pasarlo de consola a un servicio web (para que se pueda comunicar con el resto de los componentes). Así también resulta necesario definir el momento en el que se debe reentrenar dado el crecimiento de los ejemplos en un ambiente de producción.

CAPÍTULO 4

Tercera Iteración

4.1. Requisitos y tecnologías

Con la necesidad de manejar los datos flexiblemente de tal manera que las preguntas realizadas por el usuario se apeguen a las relaciones existentes entre tablas, se evalúa un motor de base de datos que se integre a las soluciones construidas.

Además, el objetivo primordial es integrar los componentes y verificar su validez como una solución informática. Para ello resulta indispensable que cada componente se encuentre preparado para la comunicación e interoperabilidad con el resto de los componentes. Los requisitos de esta iteración se resumen en la Tabla 10.

Tabla 10. Tabla de requisitos, tecnologías y criterios para la tercera iteración.

Requisitos	**Tecnologías evaluadas**	**Tecnologías Escogida**	**Criterios**
Orígenes de información y conocimiento.	SQL Server My SQL PostgreSQL Oracle SQL	SQL Server	Motor de Base de datos que se apegue a la tecnología seleccionada en la nube. Implementación de API's con una misma codificación. Publicación de servicios desde un mismo IDE.
Usabilidad	Creación de una interfaz desktop Creación de una aplicación web	Creación de una aplicación web	Integrable con una intranet. Mayor cobertura de métricas de usabilidad.
IA	Cognitive Services Amazon transcribe API SPEECH	Cognitive Services	Integración con Bot Framework. Trial gratuito para pruebas.
Inferencia de los resultados (interoperabilidad)	Servicios web Embebidos	Servicios web	Interoperabilidad. Agilidad para realizar cambios.

Nota: Requisitos, tecnologías y criterios para la tercera iteración.

Como ya se conoce el motor de base de datos usado en la segunda iteración (SQL Server), se procede a utilizar este mismo recurso para evitar esfuerzos innecesarios construyendo un nuevo motor de base de datos.

De acuerdo con el escenario empresarial propuesto, el medio más viable para desplegar el agente sería la intranet de la empresa, visto como un canal de comunicación directo con el usuario y por ende una aplicación web brinda la flexibilidad del caso ya que su característica principal es ser multiplataforma.

Si bien es cierto, existen varias opciones que permiten el reconocimiento de voz que aportarían con el objetivo del asistente, para poderlos usar es necesario contar con un capital monetario que permita acceder a ellos como es el caso de Amazon transcribe y API SPEECH, por esto Cognitive services es la opción más adecuada ya que maneja un tiempo de prueba suficientemente amplio con respecto al uso de sus API's permitiendo proceder con las pruebas necesarias.

El componente de inferencia pudiera ser resuelto en el lado del cliente, es decir, ejecutar la clasificación en el equipo del usuario y no en el servidor. Sin embargo, esto conllevaría un costo de procesamiento computacional muy extenuante para el cliente, así como un desarrollo que se ajuste a la plataforma específica del usuario. Un servicio web es más conveniente, al realizar dicho procesamiento del lado del servidor la interacción entre el usuario y el aplicativo es más transparente, así como más flexible para interoperar con más componentes y/o plataformas.

4.2. Análisis

Para iniciar esta iteración se efectúa pruebas locales con un espectro de preguntas con las que el usuario interactuará con el asistente, considerando al menos 8 de la cuales 3 reflejarán el aprendizaje brindando respuestas que permitan la toma de decisiones, 2

reflejarán respuestas precisas que permitan al usuario obtener información rápida y las últimas 3 enfocadas a entregar información de procesos previamente documentados.

PREGUNTA 1

¿Qué medicamentos serían los más oportunos para recetar al paciente Diomedes Eduardo?

PREGUNTA 2

¿Qué doctores son los más recomendados para tratar el asma alérgica?

PREGUNTA 3:

¿Qué medicamentos son los más eficaces para tratar el asma alérgica?

PREGUNTA 4:

¿Qué enfermedades se registran de forma más frecuente en la clínica?

PREGUNTA 5:

¿Cuántos pacientes se han enfermado de asma alérgica?

PREGUNTA 6:

¿Qué debo hacer para registrar un paciente por consulta externa?

PREGUNTA 7:

¿Qué debo hacer para registrar un servicio médico de un paciente interno?

PREGUNTA 8:

¿Qué debo hacer para agendar una operación?

En esta iteración ya se poseen algunos componentes aptos para la integración. Como la base de datos que, a pesar de ser consumida por las API's, ya se encuentra lista y no requiere un cambio en su estructura o en su información.

El componente de tratamiento de la información y la interfaz de usuario son los que más se tratarán y por ello se especifica en la Tabla 11 la forma en que se relacionan.

Tabla 11. Componentes de tratamiento de la información.

Escenario	Descripción
Componente de tratamiento de información – Origen de datos. Ver Figura 61.	El componente de tratamiento de información una vez que ha entendido la pregunta del usuario consulta los datos relacionados que posee en su origen de datos.
Subcomponente de Inferencia – Componente de tratamiento de información. Ver Figura 62.	El subcomponente de inferencia clasifica las respuestas consultadas en el origen de datos en respuestas oportunas e inoportunas. El componente de tratamiento de la información brinda una retroalimentación al subcomponente de inferencia permitiendo mejorar la precisión de clasificación.

Nota: La tabla describe los componentes de tratamiento de la información

Diagrama de procesos: componente de tratamiento de información – origen de datos

Proceso que identifica intención

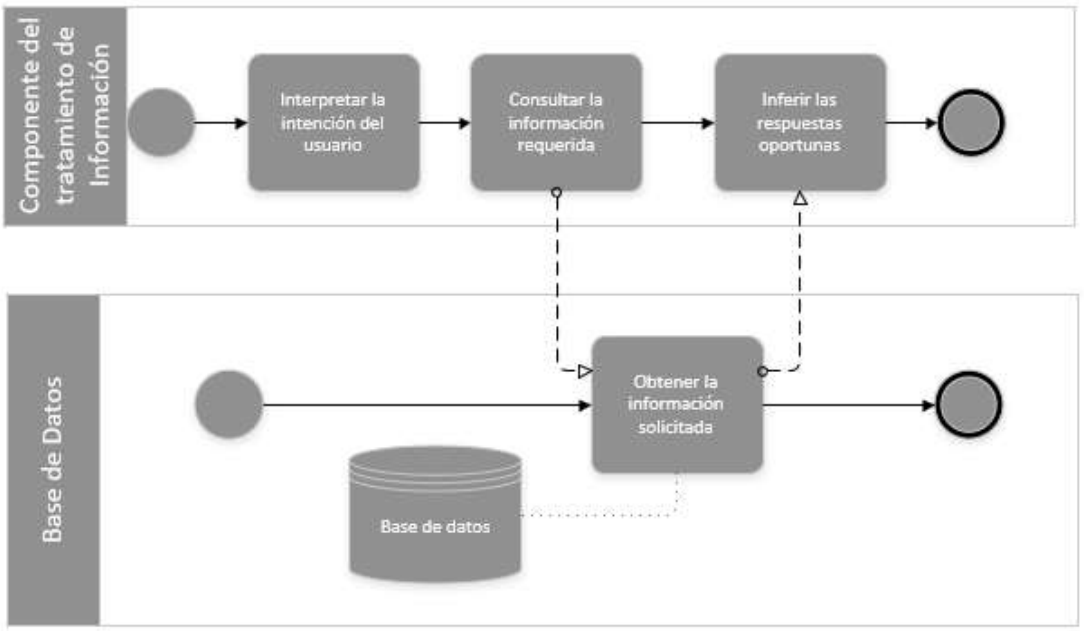

Figura 61. Proceso que identifica la intención y la consulta que se hace a la base
Elaborado por: Christian Abata y Pedro Ramírez

Diagrama de procesos: Subcomponente de inferencia – componente de tratamiento de información

Subcomponente de inferencia

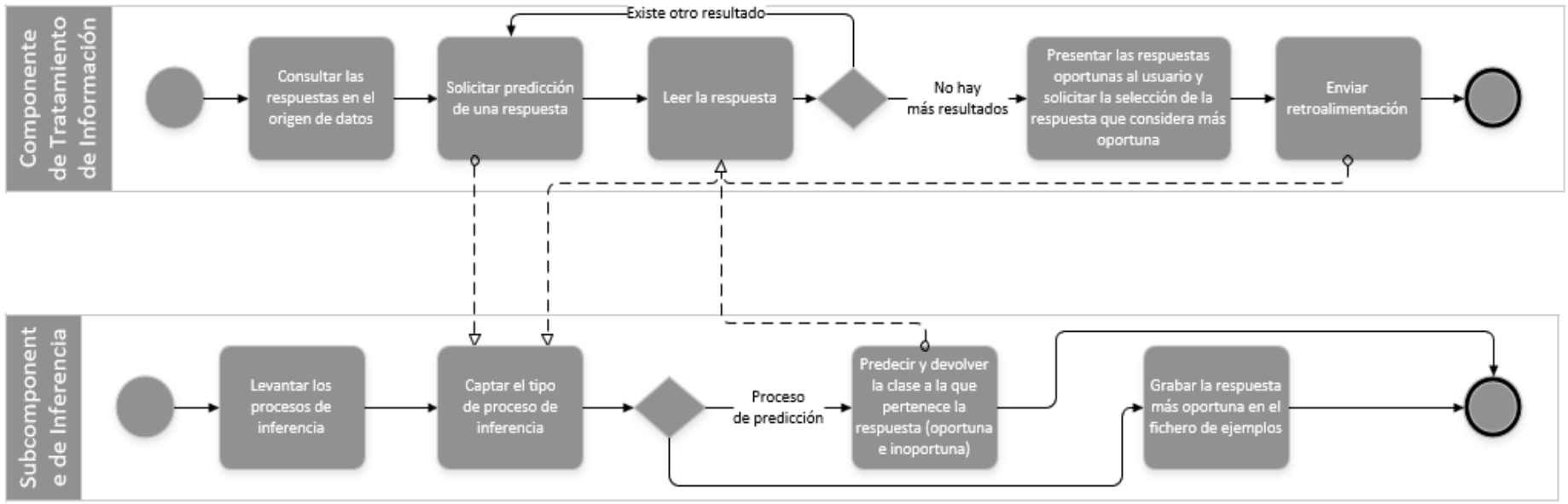

Figura 62. Proceso que identifica la inferencia desde que se hace una pregunta hasta la retroalimentación de SVM
Elaborado por: Christian Abata y Pedro Ramírez

4.3. Diseño

En esta iteración ya se encuentra muy bien definida la arquitectura, en donde se identifican principalmente tres componentes, descritos a continuación y graficados en la Figura 63.

- Componente de orígenes de información y conocimiento: componente que posee una base de datos de información operativa de la clínica referente al departamento de atención hospitalaria y ficheros planos sobre los procedimientos.
- Componente de tratamiento de información: posee dos subcomponentes:
    - Procesamiento de lenguaje natural y comunicación: comprende la intención y funciona como una interfaz de comunicación entre el origen de datos, el subcomponente de inferencia y la interfaz de usuario.
    - Inferencia: se encarga de filtrar la información oportuna y/o relevante para el usuario.
- Componente de interfaz de usuario: interfaz de chat que permite una comunicación fluida entre el usuario y el asistente.

Arquitectura final

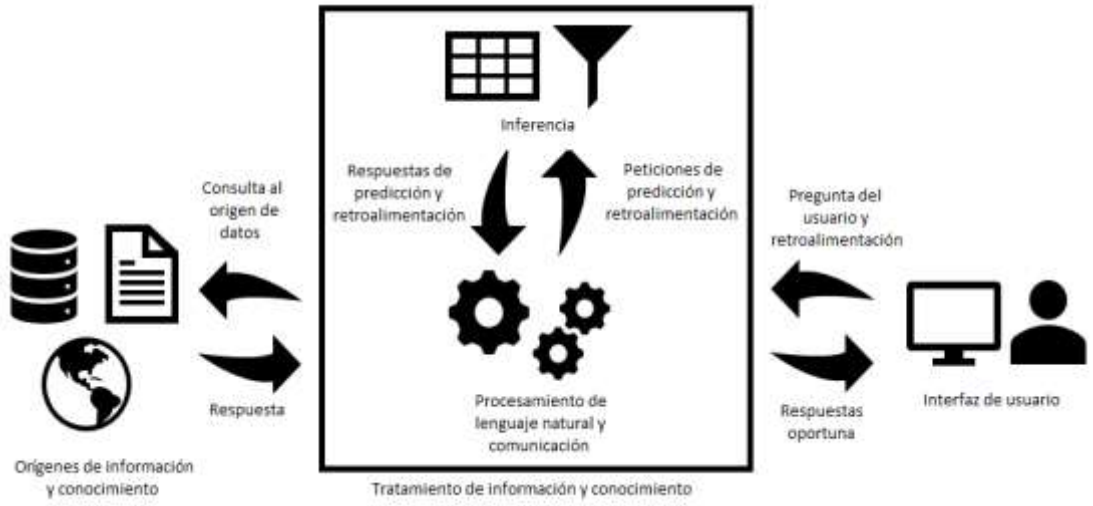

Figura 63. Arquitectura final con la incorporación de todos los componentes
Elaborado por: Christian Abata y Pedro Ramírez

4.4. Codificación

Siendo SQL server el motor de base de datos a usar es necesario implementar el código que realice la conexión directa con el servidor esto implica la instalación de dos librerías nuevas, Tedious brindará los métodos necesarios para leer la información y Bluebird será la que permita usar el concepto de Promises o promesas en Node Js.

Para realizar la instalación de las librerías es necesario correr las siguientes líneas de código sobre nuestra terminal integrada, como se aprecia en la Figura 64.

Instalación de librerías

```
npm install --save tedious-promises
npm install tedious
npm install --save bluebird
```

Figura 64. Instalación de librerías adicionales para lectura de la base de datos
Elaborado por: Christian Abata y Pedro Ramírez

Dentro del archivo dialogos.js es necesario crear la conexión a la base de datos y esto se lo realiza de la siguiente manera, como se aprecia en la Figura 65.

Conexión SQL

```
var config =
   {
     userName: 'pramirez',
     password: 'Hmedico01!',
     server: 'proyectog.database.windows.net',
     options:
        {
           database: 'HistorialMedico'
           , encrypt: true
        }
   };
```

Figura 65. Código de conexión con la base de datos para SQL Server
Elaborado por: Christian Abata y Pedro Ramírez

Adicionalmente se debe crear un archivo nuevo que será llamado msSqlConnecter.js el cual contendrá las funciones de tipo *Promises* o Promesas que lean la información de la base de datos como se muestra a continuación, como se aprecia en la Figura 66.

Extracción de datos de la base

```
currentRequest.Run = function () {
                var request = new Tedious.Request(currentRequest.sql, function (err, rowCount, rows) {
                    if (err) {
                        currentRequest.errorHandler(err);
                    }
                    else {
                        currentRequest.onComplateHandler(rowCount, currentRequest.result);
                    }
                });
                 request.on("row", function (columns) {
                    var item = {};
                    columns.forEach(function (column) {
                        item[column.metadata.colName] = column.value;
                    });
                    currentRequest.result.push(item);

                });
                 for (var i in currentRequest.params) {
                    var item = currentRequest.params[i];
                    request.addParameter(item.key, item.type, item.value);
                }
                 currentConnect.connection.execSql(request);
                return currentRequest;
            };
```

Figura 66. Función que recorre la tabla de la base de datos y la extrae
Elaborado por: Christian Abata y Pedro Ramírez

Ya que existen preguntas para realizar pruebas, es necesario crear sus intenciones y entidades en la plataforma de LUIS, al igual que en la primera iteración, por ejemplo, para la pregunta: ¿Qué medicamentos son los más eficaces para tratar el asma alérgica? Se crea la intención recomendación con la entidad paciente como se aprecia en la siguiente imagen, como se aprecia en la Figura 67.

Creación de intención para una pregunta

Figura 67. Se crea una intención en base a una pregunta de nuestro espectro de datos
Elaborado por: Christian Abata y Pedro Ramírez

Con las entidades e intenciones configuradas se deben modificar las líneas de código que detecten qué función se debe iniciar, como se aprecia en la Figura 68.

Módulo de identificación de funciones

```
.matches('Recomendacion', [
idenficaPaciente, traerPaciente, confirmaPaciente
])
```

Figura 68. Se identifica la entidad y se enlazan las funciones que serán ejecutadas
Elaborado por: Christian Abata y Pedro Ramírez

Función identificaPaciente

```
function idenficaPaciente(session, args, next) {
    session.dialogData.entities = args.entities;
    var pacienteChat = builder.EntityRecognizer.findAllEntities(args.entities, 'paciente');
    if (pacienteChat.length > 0) {
        let mensaje = 'Se ha identificado el paciente ';
        mensaje += `: **${pacienteChat[0].entity}**`;
        session.send(mensaje);
        builder.Prompts.choice(session, '¿Quieres saber algo referente a ese Paciente?',
 'OK', { listStyle: builder.ListStyle.button });
    }
    else {
        builder.Prompts.text(session, '¿Qué quieres saber exactamente?');
    }
}
```

Figura 69. La función identificaPaciente encuentra una entidad sobre la pregunta hecha por el usuario.
Elaborado por: Christian Abata y Pedro Ramírez

La función identificaPaciente será la que busque la entidad dentro de la intención, valide con el usuario si la entidad encontrada es correcta y de paso a la siguiente función, como se aprecia en la Figura 69.

La función traerPaciente es la función que carga los datos sin filtrar dependiendo de la consulta que se apegue a la pregunta del usuario por medio de otra función llamada queryPaciente, para a continuación mapear dichos datos y tratarlos para que se envíen por medio de un método POST al servicio web de forma que éste entregue una respuesta con datos filtrados y oportunos a la pregunta del usuario.

Lo primero que es necesario conocer es la extracción de los datos, sin filtrar de la base de datos para ello se procede a realizar el código de la función queryPaciente como se aprecia en la Figura 70.

Función queryPaciente

```
var nombrePaciente = pacienteChat.substr(0,pacienteChat.indexOf(' '));
      var apellidoPaciente = pacienteChat.substr(pacienteChat.indexOf(' ')+1);
      var con = new msSqlConnecter.msSqlConnecter(config);
      var formatear = "El paciente "+nombrePaciente+" "+apellidoPaciente+ " debe tomar";
      con.connect().then(function () {
          new con.Request("SELECT '"+formatear+"', Medicina.Nombre, Medicina.Notas FROM Medicina WHERE
              Medicina.Enfermedad_ID=(SELECT Enfermedad.Enfermedad_ID FROM Enfermedad WHERE
              Enfermedad.Paciente_ID=(SELECT Paciente.Paciente_ID FROM Paciente
              WHERE LOWER(Nombre)=LOWER('"+ nombrePaciente +"') AND LOWER(Apellido)=LOWER('"+ apellidoPaciente +"'))
              AND Enfermedad.Fecha=(SELECT MAX(Enfermedad.Fecha) FROM Enfermedad WHERE
              Enfermedad.Paciente_ID=(SELECT Paciente.Paciente_ID FROM Paciente WHERE LOWER(Nombre)=LOWER('"+ nombrePaciente +"')
              AND LOWER(Apellido)=LOWER('"+ apellidoPaciente +"')) GROUP BY Paciente_ID)) GROUP BY Medicina.Nombre, Medicina.Notas")
          .onComplate(function (count, datas) {
              if (callback)
              callback(datas);
          })
          .onError(function (err) {
              console.log(err);
          }).Run();
          }).catch(function (ex) {
          console.log(ex);
      });
```

Figura 70. La función queryPaciente realiza la consulta a la base de datos.
Elaborado por: Christian Abata y Pedro Ramírez

Con los datos disponibles es necesario mapearlos y tratarlos para que se apeguen al formato del servicio web creado, este tratamiento no es más que la eliminación de ciertos términos que no serán interpretados y la conversión a formato JSON de nuestros datos. Esto se logra como se aprecia en la Figura 71.

Formateo de datos

```
var respuestaSyc = JSON.stringify(data);
var cleanFotmats = respuestaSyc;
cleanFotmats = cleanFotmats.replace(/""/g, '"Respuesta"');
cleanFotmats = cleanFotmats.replace(/Nombre/g, '');
cleanFotmats = cleanFotmats.replace(/Notas/g, '');
cleanFotmats = cleanFotmats.replace(/","":"/g, ' ');
var newJsonRes = cleanFotmats.replace(/([a-zA-Z0-9]+?):/g, '"$1":');
newJsonRes = newJsonRes.replace(/[']/g, '"');
```

Figura 71. Se formatean los datos rescatados de la base de datos para que puedan ser leído por el servicio.
Elaborado por: Christian Abata y Pedro Ramírez

Teniendo los datos en formato JSON ya mapeados y tratados se envían al servicio web de manera que éste nos entregue sólo las respuestas oportunas para la pregunta de usuario, como se aprecia en la Figura 72.

Envío de datos formateados al servicio

```
var request = require('request');
      var options = {
          method: 'POST',
          url: 'http://svmapi.azurewebsites.net/api/SVM/Predice',
          headers: {
              'Content-Type': 'application/json'
          },
          json: miVaResp

      };
```

Figura 72. Envío de datos hacia el servicio por medio de POST
Elaborado por: Christian Abata y Pedro Ramírez

Tal como se recibe la repuesta no es como el usuario debe verla, por lo que para identificar lo que el servicio SVM está entregando, se marca la respuesta más oportuna con un visto, de esa forma el usuario será quien decida sobre la respuesta más efectiva, como se aprecia en la Figura 73.

Formateo de datos para presentar

```
var mijsonAns = JSON.stringify(info);
var respuestaBotShow = mijsonAns;
var formatoLimpio = respuestaBotShow.substring(1, respuestaBotShow.length-1);
formatoLimpio = formatoLimpio.replace(/[&\/\\#,+()$~%'":*?<>{}]/g, '');
formatoLimpio = formatoLimpio.replace(/Respuesta/g, '|');
formatoLimpio = formatoLimpio.replace(/OportunoSVM/g, '');
formatoLimpio = formatoLimpio.replace(/OportunoUsuario/g, '');
formatoLimpio = formatoLimpio.replace(/inoportuno/g, '');
formatoLimpio = formatoLimpio.replace(/null/g, '');
formatoLimpio = formatoLimpio.replace(/oportuno/g, ' ✓');
let showAns = formatoLimpio;
```

Figura 73. Cambio de formato String para mostrar al usuario.
Elaborado por: Christian Abata y Pedro Ramírez

Hasta este punto el usuario recibe una respuesta ya filtrada a la pregunta que ha realizado, pero la inferencia para el aprendizaje del agente aún no está considerada, para ello se debe usar la segunda vía de comunicación con el servicio web que recibirá una confirmación por parte del usuario sobre el total de preguntas que se mostraron anteriormente, la cual en realidad sí fue satisfactoria para sus fines.

Función confirmaPaciente

```
findAndReplace(dataU, entidad, "Oportuno");
var request = require('request');
var options = {
method: 'POST',
url: 'http://svmapi.azurewebsites.net/api/SVM/Retroalimenta',
headers: {
'Content-Type': 'application/json'
},
json: dataU
};

function callback(error, response, body) {
    if (!error) {
        var infoFinal = JSON.parse(JSON.stringify(body));
        console.log(infoFinal);
        }
        else {
             console.log('Error happened: '+ error);
            }
        }
        request(options, callback);
}
```

Figura 74. Envío de datos haca el segundo servicio.
Elaborado por: Christian Abata y Pedro Ramírez

Para lograr lo antes mencionado se procede a la ejecución de la última función en el orden de identificación de intenciones llamada confirmaPaciente, la que repite la extracción de la base de datos, transforma dicha información a un formato que pueda recibir el servicio web de predicción para después enviar la respuesta al servicio de retroalimentación, como se aprecia en la Figura 74.

En el subcomponente de inferencia se construyó un API REST que ofrece esencialmente tres servicios:

- EntrenarSVM: permite entrenar el algoritmo de SVM, bajo ejemplos definidos para las respuestas a las preguntas previamente explicadas, para ello estas respuestas se encuentran documentadas en un fichero de formato CSV. Éste distingue dos columnas correspondientes a la respuesta y al resultado, 1 si es oportuno o -1 caso contrario (ver anexo 2): Ejemplos.csv. Se lee el fichero CSV separándolo en una lista las características y en un arreglo las etiquetas, apreciado en la Figura 75.

EntrenarSVM: obtención de características y etiquetas

```
string fichero = HttpContext.Current.Server.MapPath(
    @"..\..\App_Data\Ejemplos.csv");
var tablaDatos = DataTable.New.ReadCsv(fichero);
List<string> x = obtenerCaracteristicas(tablaDatos);
double[] y = obtenerEtiquetas(tablaDatos);
```

Figura 75. Lectura de características y etiquetas.
Elaborado por: Christian Abata y Pedro Ramírez

Se obtiene el vocabulario de los ejemplos de entrenamiento, se guarda dicho vocabulario y se construye el problema, ver Figura 76.

EntrenarSVM: generación del vocabulario y construcción del problema

```
var vocabulario = x
    .SelectMany(ObtenerPalabras)
    .Distinct()
    .OrderBy(word => word)
    .ToList();
guardarVocabulario(vocabulario);

var constructorProblema = new TextClassificationProblemBuilder();
var problema = constructorProblema.CreateProblem(
    x,
    y,
    vocabulario.ToList());
```

Figura 76. Generación del vocabulario y construcción del problema.
Elaborado por: Christian Abata y Pedro Ramírez

La generación del modelo requiere una selección de *kernel* y para su selección óptima se requiere un análisis extenuante de la distribución de los datos, sin embargo, se optó por realizar pruebas que permitan obtener un *kernel* que brinde la mejor precisión. Aunque no se encontraron resultados definitivos para los parámetros, se encontró valores óptimos dentro de algunos rangos establecidos, por lo que para obtener un modelo óptimo se recorre el *kernel* polinomial de segundo grado con un gama (g) entre 0.1 y 1 y un r entre 0.1 y 1. En cuanto a la definición de la constante C que permite un "balance entre la maximización del margen y la violación a la clasificación" (Betancourt, 2005), se estableció con un valor de 10000 para favorecer la exactitud. Se calcula la exactitud del modelo. Ver Figura 77.

EntrenarSVM: definición del *kernel* y construcción del modelo

```
double C = 10000;
var modelo = new C_SVC(problema, KernelHelper.LinearKernel(), C);

var exactitud = modelo.GetCrossValidationAccuracy(10);
for (double g = 0.1; g <= 1; g += 0.1)
{
    for (double r = 0.1; r <= 1; r += 0.1)
    {
        var modeloAuxiliar = new C_SVC(problema, KernelHelper.PolynomialKernel(2, g, r), C);
        var exactitudAuxiliar = modeloAuxiliar.GetCrossValidationAccuracy(10);
        if (exactitudAuxiliar > exactitud)
        {
            modelo = modeloAuxiliar;
            exactitud = exactitudAuxiliar;
        }
    }

}
respuesta = "La exactitud del modelo es " + exactitud + "\n";

modelo.Export(HttpContext.Current.Server.MapPath(
    @"..\..\App_Data\modelSVM.model"));
```

Figura 77. Definición del *kernel* y construcción del modelo
Elaborado por: Christian Abata y Pedro Ramírez

Se define el diccionario de predicción. Ver Figura 78.

EntrenarSVM: diccionario de predicción

```
diccionarioPrediccion = new Dictionary<int, string> {
    {
        -1,
        "inoportuna"
    },
    {
        1,
        "oportuna"
    }
};
```

Figura 78. Diccionario de predicción
Elaborado por: Christian Abata y Pedro Ramírez

- Predice: en una iteración intermedia se utilizaba el modelo obtenido en el entrenamiento, y se predecía su clasificación de oportuna o inoportuna para el usuario. Sin embargo, al probar el funcionamiento y la interacción de los componentes se halló más adecuado calcular el modelo cada vez que se iba a

realizar una predicción, puesto que tras la retroalimentación las características y etiquetas ya no concordaban con el modelo generado previamente. Este método, presentado en la Figura 79, al igual que el anterior construye el listado de respuestas y el arreglo de etiquetas, genera el vocabulario, construye el problema, obtiene el método óptimo y define el diccionario de predicción; con la diferencia de que toma un parámetro de tipo lista correspondiente a las respuestas que debe predecir y devuelve los resultados tras la tarea de clasificación.

Predice: predicción de un conjunto de respuestas

```
foreach (var respuesta in respuestas)
{
    var newX = TextClassificationProblemBuilder.CreateNode(
        respuesta.Respuesta,
        vocabulario
        );
    var predictedY = modelo.Predict(newX);
    respuesta.OportunoSVM = diccionarioPrediccion[(int)predictedY];
}
return respuestas;
```

Figura 79. Predicción de un conjunto de respuestas
Elaborado por: Christian Abata y Pedro Ramírez

- Retroalimenta: por cada respuesta presentada al usuario, este último ofrece una retroalimentación seleccionando la respuesta que le ha sido más oportuna, llamando a este método. Se toma el listado de respuestas con retroalimentación como un parámetro, se retira los vistos y los espacios en blanco al principio o fin que pudiesen existir. Se lee el fichero con los datos utilizados para el entrenamiento obteniendo las respuestas en una lista y las etiquetas en un arreglo. Ver Figura 80.

Retroalimenta: Lectura de respuestas de entrenamiento

```
foreach (var item in lista)
{
    item.Respuesta = item.Respuesta.Replace("✓", "");
    item.Respuesta = item.Respuesta.Trim();
}
string fichero = HttpContext.Current.Server.MapPath(
    @"..\..\App_Data\Ejemplos.csv");
var tablaDatos = DataTable.New.ReadCsv(fichero);
List<string> X = obtenerCaracteristicas(tablaDatos);
double[] y = obtenerEtiquetas(tablaDatos);
List<string> respuestas = new List<string>();
```

Figura 80. Lectura de respuestas de entrenamiento
Elaborado por: Christian Abata y Pedro Ramírez

Se recorre el listado de respuestas de entrenamiento y se comparan con las respuestas provenientes de la retroalimentación de tal manera que de existir una respuesta en el entrenamiento se reemplaza el valor de la etiqueta por el correspondiente al que el usuario decidió si era oportuno o no. De no existir una respuesta de retroalimentación en el entrenamiento esta se añade al listado de entrenamiento. Finalmente se escribe los nuevos valores en el fichero de entrenamiento Ejemplos.csv. Ver Figura 81.

Una vez comprobado el funcionamiento del asistente en un ambiente de pruebas local, este será publicado como una aplicación web que permite la incorporación de conceptos de usabilidad, disponibilidad y de multiplataforma, sobre el ambiente cloud de Azure.

Retroalimenta: Sobreescritura de los ejemplos ya existentes y escritura de los nuevos.

```
for (int i = 0; i < X.Count; i++)
{
    string respuesta = null;
    Ejemplo itemExistente = null;
    foreach (var item in lista)
    {
        if (X.ElementAt(i).Equals(item.Respuesta))
        {
            if (item.OportunoUsuario == null)
            {
                respuesta = item.Respuesta + ", -1";
            }
            else
            {
                respuesta = item.Respuesta + ", 1";
            }
            itemExistente = item;
        }
    }
    if (itemExistente != null)
    {
        lista.Remove(itemExistente);
    }
    if (respuesta == null)
    {
        respuestas.Add(X.ElementAt(i) + ", " + y[i]);
    }
    else
    {
        respuestas.Add(respuesta);
    }
}
foreach (var item in lista)
{
    if (item.OportunoUsuario == null)
    {
        respuestas.Add(item.Respuesta+ ", -1");
    }
    else
    {
        respuestas.Add(item.Respuesta + ", 1");
    }
}
StreamWriter ficheroSW = new StreamWriter(fichero, true);
foreach (var x in respuestas)
{
    ficheroSW.WriteLine(x);
}
ficheroSW.Close();
return true;
```

Figura 81. Sobreescritura de los ejemplos existentes y escritura de los nuevos
Elaborado por: Christian Abata y Pedro Ramírez

4.5. Pruebas

Tal como se realizó la codificación cuando el usuario realiza una pregunta buscando encontrar qué medicamento es el más oportuno para un paciente, el agente identificará la intención y entidad e inmediatamente confirmará si esto es correcto con el usuario, como se aprecia en la Figura 82.

Pruebas locales finales parte 1

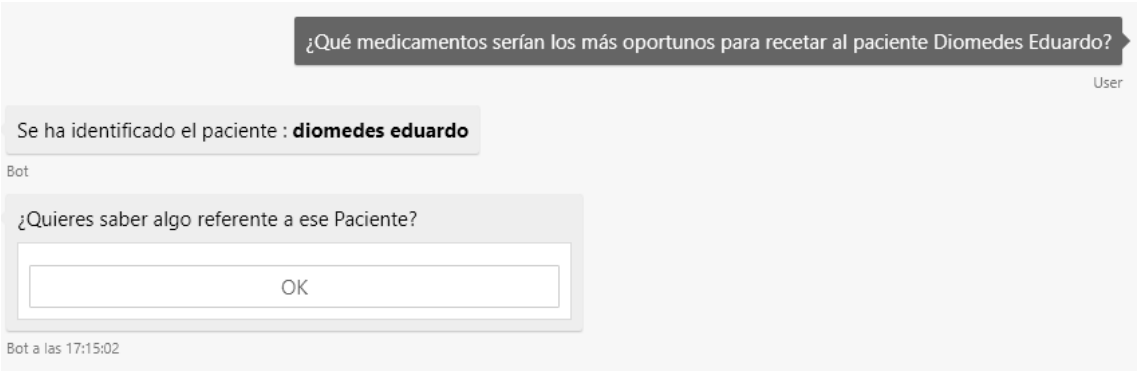

Figura 82. Resultados de pruebas finales locales 1
Elaborado por: Christian Abata y Pedro Ramírez

Para así mostrar las respuestas que serán las más oportunas en ese momento con un visto, como se aprecia en la Figura 83. Dejando al usuario la opción de seleccionar la más apegada a su necesidad, de forma que el agente pueda aprender sobre dicha pregunta.

Pruebas locales finales parte 2

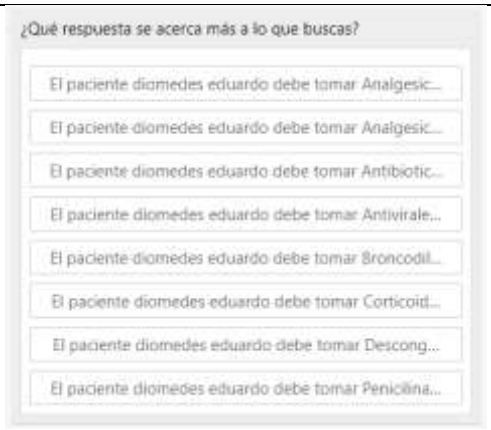

Figura 83. Resultados de pruebas finales locales 2
Elaborado por: Christian Abata y Pedro Ramírez

Las pruebas realizadas para encontrar el *kernel* y los parámetros apropiados en la implementación de SVM arrojaron los resultados presentados en la Tabla 12.

Tabla 12 Exactitud por *kernel*.

Primera Prueba				
Kernel	**Exactitud**	**Grado**	**Gamma**	**r**
Lineal	0,8372093023	-	-	-
Polinomial	0,8604651162	2	0,2	0,3
Polinomial	0,8837209302	2	1,7	0,5
Segunda Prueba				
Kernel	**Exactitud**	**Grado**	**Gamma**	**r**
Lineal	0,7906976744	-	-	-
Polinomial	0,8372093023	2	0,1	0,1
Polinomial	0,8604651162	2	0,2	0,2
Tercera Prueba				
Kernel	**Exactitud**	**Grado**	**Gamma**	**r**
Lineal	0,7674418604	-	-	-
Polinomial	0,8139534883	2	0,1	0,1
Polinomial	0,8372093023	2	0,1	0,3
Polinomial	0,8604651162	2	0,4	2,2
Polinomial	0,8837209302	2	0,1	2
Cuarta Prueba				
Kernel	**Exactitud**	**Grado**	**Gamma**	**r**
Lineal	0,8139534883	-	-	-
Polinomial	0,8372093023	2	0,1	0,5
Polinomial	0,8604651162	2	0,4	2,1
Quinta Prueba				
Kernel	**Exactitud**	**Grado**	**Gamma**	**r**
Lineal	0,7674418604	-	-	-
Polinomial	0,8139534883	2	0,1	0,2
Polinomial	0,8372093023	2	0,1	2.4

Polinomial	0,8604651162	2	0,2	0,3
Polinomial	0,8837209302	2	0,3	2.4
Sexta Prueba				
Kernel	**Exactitud**	**Grado**	**Gamma**	**r**
Lineal	0,8372093023	-	-	-
Polinomial	0,8604651162	2	0,7	1,2
Polinomial	0,8837209302	2	1,6	2,9
Séptima Prueba				
Kernel	**Exactitud**	**Grado**	**Gamma**	**r**
Lineal	0,8372093023	-	-	-
Polinomial	0,8604651162	2	0,1	1,9
Polinomial	0,8837209302	2	0,5	2,9
Octava Prueba				
Kernel	**Exactitud**	**Grado**	**Gamma**	**r**
Lineal	0,7441860465	-	-	-
Polinomial	0,7906976744	2	0,1	0,1
Polinomial	0,8372093023	2	0,1	0,2
Polinomial	0,8604651162	2	0,3	2.4
Polinomial	0,8837209302	2	1	0,8

Nota: Exactitud por *kernel*

Una vez subida la aplicación a un ambiente *cloud* el asistente presentado a los usuarios será la que se puede apreciar en la Figura 84.

Asistente virtual en la web

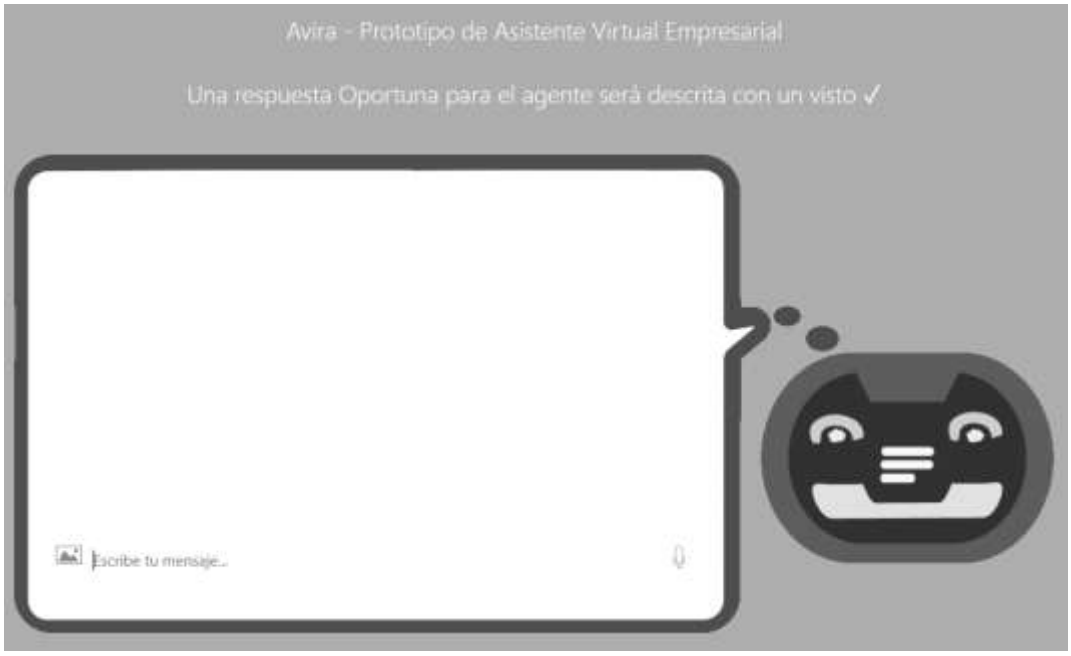

Figura 84. Prototipo final web
Elaborado por: Christian Abata y Pedro Ramírez

Al igual que en el ambiente local el asistente es capaz de aprender sobre las intenciones e identifica la o las entidades para entregar una respuesta, como se aprecia en la Figura 85 y 86.

Pruebas del prototipo virtual en entorno web parte 1

Figura 85. Pruebas del prototipo virtual en entorno web parte 1
Elaborado por: Christian Abata y Pedro Ramírez

Pruebas del prototipo virtual en entorno web parte 2

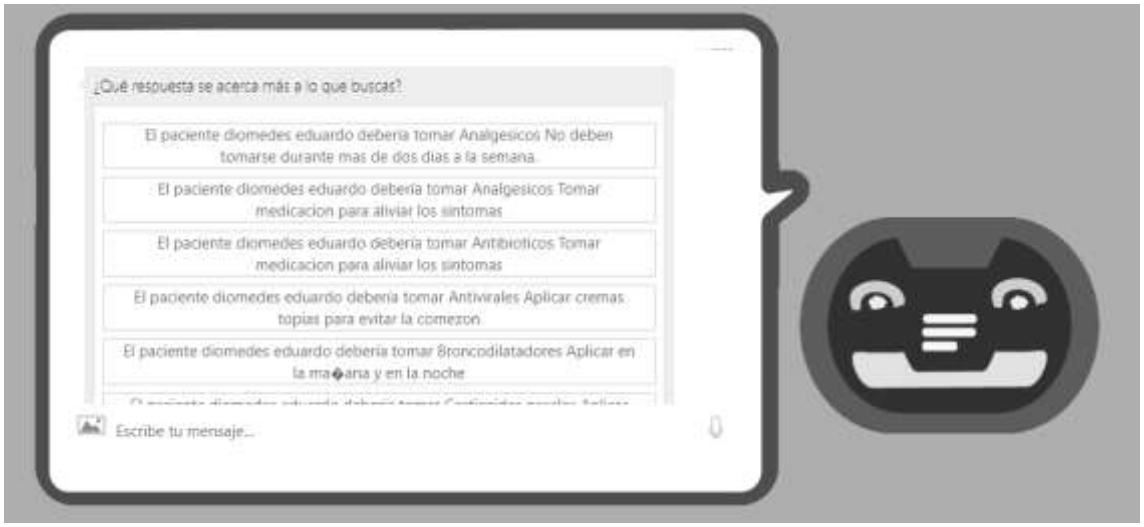

Figura 86. Pruebas del prototipo virtual en entorno web parte 2
Elaborado por: Christian Abata y Pedro Ramírez

4.6. Resultados

- Las pruebas de SVM demostraron ofrecer la mayor precisión con *kernels* lineal y polinomial, puesto que la sigmoidal y radial no mostraron mejorar la exactitud de las antes mencionadas.

- Las pruebas realizadas en SVM logran alcanzar una precisión del 88,37%, con más de un parámetro distinto, esto implica que necesariamente para obtener una mayor precisión se debe establecer un rango o establecer un valor específico. Así para la función polinomial se determinaron los valores de grado 2, gamma entre 0,1 - 2 y r entre 0,1 y 3. Los parámetros de gamma y r permiten establecer la tolerancia de la función, así como el número de vectores de soporte que se computan.

CAPÍTULO 5

Cuarta Iteración

Concluido el prototipo de asistente virtual es necesario probarlo con usuarios, definiendo los escenarios de uso para el agente, en los que la retroalimentación recibida será en primer lugar por el apoyo de un médico, en segundo lugar, el de un enfermero y en tercer lugar el de un conocedor de procesos.

El alcance definido para las pruebas se enfoca en identificar si el agente entrega información de calidad al usuario mediante varias preguntas, apegándose a los siguientes criterios:

- Que el agente identifique la intención de la pregunta.
- Que el agente identifique la entidad en la pregunta.
- Que el agente entregue información oportuna al usuario.
- Que el agente aprenda sobre las respuestas seleccionadas por el usuario.
- Que el agente prediga ciertas respuestas en base a la selección de respuestas.
- Repetir el mismo procedimiento para 2 preguntas más con diferentes entidades.

Siendo cada uno positivamente respondido, ya que el asistente puede identificar con éxito la intención de la pregunta, la entidad en la pregunta así esta tenga una palabra o dos, la información que ve el usuario es oportuna, el usuario se da cuenta que el agente está aprendiendo con cada pregunta, que prediga ciertos resultados y todo esto independientemente de la pregunta. Todos los resultados mencionados pueden ser ahondados en los videos de pruebas en los anexos.

Conclusiones

- Las pruebas en la tercera iteración demostraron que el prototipo tiene, tras el entrenamiento, una precisión variante entre 86,04% y 88,37%, considerada aceptable para la cantidad de ejemplos e información en el origen de datos. Dicha predicción refiere a la probabilidad de que una respuesta sea oportuna o no para el usuario, ya que todas las respuestas siempre son correctas porque su origen de datos pertenece a la simulación.
- Bot Framework a pesar de ser una solución muy efectiva en cuanto a manejar el hilo de intercambio de información usuario/aplicación, para realizar un cambio profundo sobre la interfaz no se debe usar Bot Framework porque es muy poco manipulable.
- Al término del prototipo de asistente virtual se puede concluir que Node Js es el lenguaje de programación oportuno para codificar las funciones que interaccionan con el usuario, que comunican los servicios y les dan formato a las respuestas, porque al ser código apenado al cliente este disminuye considerablemente el tiempo de respuesta.
- La forma más pertinente de identificar el orden para poder presentar la información es por medio de un hilo de conversación al cual el usuario se encuentre vinculado.
- Del nivel de madurez empresarial dependerá el desempeño el agente, el asistente virtual no podrá ser puesto en producción si la empresa no cuenta al menos con una base de datos o archivos digitales de los cuales se pueda obtener información.

Recomendaciones

- De utilizar Node Js como lenguaje de programación se recomienda utilizar para las funciones los conceptos de CallBack *y* Promises ya que de esa forma se puede trabajar con asincronismo.
- Si se usa Bot Framework con miras a la creación de un asistente virtual, es recomendable usar los servicios de Microsoft Bot Website desde el inicio ya que las funciones cognitivas como el reconocimiento de voz no están habilitadas por defecto en el ambiente de pruebas.
- Se recomienda hacer un análisis del nivel de madurez tecnológico de la empresa para decidir si es o no factible la implementación del asistente virtual, si la empresa se encuentra en un nivel de madurez de integración se recomienda conectar el aplicativo por medio de un API.
- SVM al soportar *kernels* personalizados y por su construcción calculada de vectores de soporte es recomendable para resolver cualquier problema de clasificación, sin embargo, esto no implica que sea el óptimo en todos los escenarios.
- A partir del prototipo obtenido, se pueden desarrollar varios proyectos, pero sin lugar a duda el más relevante es su implementación en un escenario distinto con el fin de validarlo como una solución informática aplicable en diversos contextos empresariales y no solo el médico.

Glosario de términos

Bot: Se considera como un usuario autónomo que podrá ser automatizado por medio botones interactivos, secuencias de texto o IA (García Quesada, 2017).

Eficacia: Según la RAE es la capacidad para lograr el efecto deseado o esperado. (RAE)

Key: Que en concepto es una pieza de metal que se usa para abrir puertas o candados, en esencia una llave que permite el acceso (Cambridge, 2018).

API: *Application Programming Interface,* se entiende "como las funcionalidades que aporta un determinado cierto servicio software facilitando que pueda ser utilizado por otro software" (Plaza Estévez, Ramírez Lamela, & Acosta Morales, 2016).

Cliente: "Persona que compra en una tienda, o que utiliza con asiduidad los servicios de un profesional o empresa" (RAE, 2018).

Usuario: Persona que usa algo y tiene derecho, con limitaciones, de usar una cosa ajena (RAE, 2018).

MVC: Es una guía para el diseño de aplicaciones, dividida en tres modelos separados, el modelo que presenta los datos y reglas, un conjunto de vistas y un conjunto de controladores. (Gutiérrez., 2017)

Fichero: Se define como una colección de información relacionada. (Uji, 2018)

ID: Se encuentra como un membrete de un identificador único, usado para bases de datos.

URL: En inglés significa Uniform Resource Locutor es decir un localizador uniforme de recurso, que contiene una secuencia de caracteres apegados a un estándar usado para nombrar recursos en internet (bibing.us.es, 2018).

SQL: Es el estándar ANSI/ISO para la definición, manipulación y control de bases de datos relacionadas (Martín Escofet, 2017).

Solución Informática: Dado el concepto de informática se puede definir a una solución informática que solventa un problema con el uso de la tecnología computacional (Prieto Espinosa & Lloris Ruiz y Jua, 2001).

Referencias

Afsari, K., Eastman, C., & Castro-Lacouture, D. (2017). JavaScript Object Notation (JSON) data serialization for IFC schema in web-based BIM data exchange. *ScienceDirect*, 26.

Almuiñas Rivero, J. L., Passailaigue Baquerizo, R., & Galarza López, J. (2015). La gestión de la información y el conocimiento: Una oportunidad para las instituciones de educación superior. *Revista Científica de la Universidad de Cienfuegos*, 16-22.

Amaro, L. R. (2012). Usuarios de nuevos medios, Comunicación e interacción social. *Comunite*, 100.

asd. (2000). asdfqweqwr. *qewr*, 12-12.

Azure Microsoft. (2018, 25 03). *microsoft.com*. Retrieved from microsoft.com: https://azure.microsoft.com/en-us/services/cognitive-services/computer-vision/?v=18.05

Banerjee, S., Majumder, P., & Mitra, M. (2017). Re-evaluating the need for Modelling Term-Dependence in Text Classification Problems. *Cornell University Library*, 1-23.

Becerra Fernandez, I., & Sabherwal, R. (2015). *Knowledge Management Systems and Processes.* New York: Routledge.

Bengio, Y. (2016). Aprendizaje Profundo. *Investigación y Ciencia*, 47-53.

Benítez, R., Escudero, G., & Kanaan, S. (2013). *Inteligencia artificial avanzada.* Catalunya: UOC.

Besteiro, M., & Rodríguez, M. (2016). *Web Services.*

Betancourt, G. A. (2005). Las máquinas de soporte vectorial (SVMs). *Scientia et Technica*, 67-72.

bibing.us.es. (2018). *bibing.us.es.* Retrieved from bibing.us.es: http://bibing.us.es/proyectos/abreproy/11247/fichero/Memoria%252F6-Conceptos+previos.pdf

Bizagi Suite. (2014). *BPMN 2.0.* Bizagi.

Bonaparte, U. J. (2012). *Proyectos UML Diagramas de clases y aplicaciones JAVA en NetBeans 6.9.1.* Buenos Aires: Editorial de la Universidad Tecnológica Nacional.

Cabero J. (2007). Nuevas tecnologías aplicadas a la educación. 3.

Cambridge. (2018). *cambridge.org*. Retrieved from cambridge.org: https://dictionary.cambridge.org/dictionary/englis

Cárcel Carrasco, F. J., & Roldán Porta, C. (2013). Principios básicos de la Gestión del Conocimiento y su aplicación a la empresa industrial en sus actividades tácticas de mantenimiento y explotación operativa: Un estudio cualitativo. *Intangible Capital*, 91-125.

Carmon Suárez, E. J. (2016). Tutorial sobre Máquinas de Vectores de Soporte SVM. *Dpto. de Inteligencia Artificial, ETS de Ingeniería Informática, Universidad Nacional de Educación a Distancia*.

Casali, A. (2015, Marzo 31). *ResearchGate.* Retrieved from www.researchgate.net: https://www.researchgate.net/publication/268275299_Que_es_la_Inteligencia_Artificial

Chicaiza Paredes, A. (2016, 08 09). SISTEMA DE CONTROL ELECTORAL. *SISTEMA DE CONTROL ELECTORAL*. Quito, Pichincha, Ecuador.

CIDEC. (2004). *CIDEC.* Retrieved from http://www.cidec.net: http://www.cidec.net/cidec/pub/archivos/31.pdf

Corral Mendivil, C. R., Gil Palomares, M. G., Velasco Cepeda, R. I., & Serrano Cornejo, M. L. (2011). La importancia del programa de inducción, en las empresas del sector de servicio. *El Buzón de Pacioli*.

Cruz Martínez, A., & Alarcón Armenteros, A. (2016, 10 18). Base de datos interna de pérdidas operacionales. *Un desafío en la banca cubana para gestionar el riesgo operacional.*

de Moya Anegón, F., Herrero Solana, V., & Guerrero Bote, V. (n.d.). *La aplicación de Redes Neuronales Artificiales (RNA): a la recuperación de la información.* Revistes Catalanes amb Accés Obert.

Diarlu. (2017, Diciembre 07). *diarlu.com.* Retrieved from Diarlu: https://www.diarlu.com/mejores-aplicaciones-mensajeria-cifrada/

DREYFUS, H., & DREYFUS, S. (2017). DE SOCRATES A LOS SISTEMAS EXPERTOS: Los límites y peligros de la racionalidad calculante. *Revista de la Facultad de Artes Y Humanidades*, 10.

Felman, M. (2017, 06 03). *mva.microsoft.com.* Retrieved from mva.microsoft.com: https://mva.microsoft.com/es-es/training-courses/desarrollo-de-bots-con-microsoft-bot-framework-y-nodejs-17712

Fernández González, J. (2013). Introducción a las metodologías ágiles. *Otras formas de analizar y desarrollar*. Universidad Oberta de Catalunya.

Fernández Martínez, A., & Llorens Largo, F. (2011). *Gobierno de las TI para universidades.* Madrid: CRUE.

Fontalvo Herrera, T., Quejada, R., & Puello Payares, J. G. (2011). La gestión del conocimiento y los procesos de mejoramiento. *Dimens. empres*, 80-87.

Gámez Albán, H. M., Orejuela Cabrera, J. P., Salas Achipiz, Ó. A., & Bravo Bastidas, J. J. (2016). Aplicación de mapas de Kohonen para la priorización de zonas de mercado: Una aproximación práctica. *Revista EIA*, 157-169.

García P., V., & Gutiérrez C., L. (2011). Manual de géneros periodísticos. Segunda edición. In V. M. García P., & L. M. Gutiérrez C., *Manual de géneros periodísticos. Segunda edición* (p. 238). Bogotá: ECOE.

García Quesada, M. J. (2017, 09). BOT PARA TELEGRAM QUE EJECUTE AVENTURAS CONVERSACIONALES. Jaén, España: Escuela Politécnica Superior de Jaén.

Godoy Viera, Á. F. (2017). Técnicas de aprendizaje de máquina utilizadas para la minería de texto. *Bibliotecológica*, 103-126.

González Herrero, D. (2017). *Diseño y construcción de una herramienta tecnológica basada en técnicas de procesamiento de lenguaje natural y aprendizaje automático para clasificación de correros de una entidad bancaria.* Madrid: Universidad Politécnica de Madrid.

Guarda, L., & Carrasco, J. (2017). Comparación entre SVM y RNA para clasificación de sujetos con ojos abiertos o cerrados mediante EEG. *Departamento de Ingeniería Mecánica*, 1-4.

Guerrero Bote, V. P., López Pujalte, C., Faba Pérez, C., Reyes Barragán, M. J., Zapico Alonso, F., & de Moya Anegón, F. (2002). Clasificación de términos mediante el algoritmo de Kohonen. *Facultad de Biblioteconomía y Documentación (Alcazaba de Badajoz), Universidad de Extremadura*.

Gurugé, A. (n.d.). *Web Services.* Digital Press.

Gutiérrez., J. J. (2017). ¿Qué es un framework web? . 4.

IBM. (2018, Mayo 19). *IBM Knowledge Center*. Retrieved from www.ibm.com/support/knowledgecenter: https://www.ibm.com/support/knowledgecenter/es/SS3RA7_17.1.0/modeler_mainhelp_client_ddita/clementine/kohonennode_general.html

Isasi Viñuela, P. (n.d.). Modelos Neuronales Competitivos: Kohonen & ART. *Universidad Carlos III*, 141-177.

Joskowicz, J. (2008). Reglas y Prácticas en eXtreme Programming. *Reglas y Prácticas en eXtreme Programming*. España: Universidad de Vigo.

Latorre Ariño, M. (2017). Contenidos declarativos (factuales, conceptuales), procedimentales y actitudinales. *Universidad Marcelino Champagnat*, 1-9.

Leal, C., Perpinyà Morera, P., & Remei. (2013). *Cómo y dónde buscar fuentes de información.* Bellatera.

López Yolanda, B. (2016, 07 02). Metodología Ágil de Desarrollo de Software – XP. *Metodología Ágil de Desarrollo de Software – XP*. ESPE, MEVAST.

Martín Escofet, C. (2017). El lenguaje SQL.

Mazza, N. (2012). *Agentes Virtuales.*

McCullock, J. (2013). *Mnemosyne_Studio*. Retrieved from mnemstudio.org: http://mnemstudio.org/neural-networks-som1.htm

Microsoft. (2018, 03 12). *docs.microsoft.com*. Retrieved from docs.microsoft.com: https://docs.microsoft.com/en-us/azure/bot-service/?view=azure-bot-service-3.0

Mukherjee, R., & Mao, J. (2004). Enterprise Search: Tough Stuff. *QUEUE*, 36-46.

Murazzo, M., Millán, F., Rodríguez, N., Segura, D., & Daniela, V. (2012). DESARROLLO DE APLICACIONES PARA CLOUD COMPUTING. *Universidad Nacional de San Juan, Facultad de Ciencias Exactas, Físicas y Naturales*, 109.

Nagles G., N. (2007). La Gestión del Conocimiento como fuente de Innovación. *Revista Escuela de Administración de Negocios*, 77-87.

Nievas Lio, E. (2016). *Aplicando máquinas de soporte vectorial al análisis de pérdidas no técnicas de energía eléctrica.* Córdoba: Facultad de Matemática, Astronomía, Física y Computación Universidad Nacional de Córdoba.

Nieves Lahaba, Y., & León Santos, M. (2001). La gestión del conocimiento: una nueva perspectiva en la gerencia de las organizaciones. *ACIMED*, 121-126.

Pablos Heredero, C., López Hermoso Agius, J., Romo Romero, S., & Salgado, S. (2012). *Organización y transformación de los sistemas de información en la empresa.*

Palacios, J. P. (2008). La Gestión del Conocimiento como Disciplina Empresarial. *Revista de Investigación de la Facultad de Ciencias Administrativas*, 29-34.

Pérez Rodríguez, Y., & Coutín Domínguez, A. (2005). La gestión del conocimiento: un nuevo enfoque en la gestión empresarial. *ACIMED*.

Pignani, J. (1998). *Orientación I: Informática aplicada a la Ingeniería de Procesos 1*. Universidad Tecnológica Nacional, Facultad Regional Rosario.

Plaza Estévez, S., Ramírez Lamela, N., & Acosta Morales, C. (2016). *API de servicios web orientados a accesibilidad.* Madrid: Facultad de Informática Universidad Complutense de Madrid.

Prieto Espinosa, A., & Lloris Ruiz y Jua, A. (2001). Introducción a la Informática.

Quintanilla Juárez, N. A. (2014). Herramientas TICs y la Gestión del Conocimiento. *Universidad Don Bosco*, 1-12.

RAE. (2018, Junio 1). *Real Academia de la Lengua*. Retrieved from rae.es: http://dle.rae.es/?id=9SnxU0N

RAE. (2018, Junio 1). *Real Academia Española*. Retrieved from rae.es: http://dle.rae.es/?id=bBsqKPc

RAE. (n.d.). *dle.rae.es.* Retrieved from dle.rae.es: dle.rae.es/srv/search?m=30&w=eficacia

Real Academia Española. (2018). *Diccionario de la lengua española*. Retrieved from dle.rae.es: http://dle.rae.es/srv/fetch?id=3IWZ4nr

Real Academia Española. (2018, Mayo 24). *Diccionario de la lengua española*. Retrieved from dle.rae.es: http://dle.rae.es/?id=AMrJ4zs

Redbooks, I. (2014). *Using Web Services for Business Integration.* IBM.

Reyes Barragán, M. J., Guerrero Bote, V., & Zapico Alonso, F. (n.d.). Uso del algoritmo de Kohonen, aplicado al estudio de la localización y accesibilidad de

revistas científicas en bibliotecas universitarias. *Facultad de Biblioteconomía y Documentación Universidad de Extremadura.*

Rodríguez Cruz, Y., & Pinto, M. (2017). Requerimientos informacionales para la toma de decisiones estratégicas en organizaciones de información. *TransInformacao*, 175-189.

Rodríguez Rodríguez, A., Hernández Cabrera, J., & Plácido Castro, A. M. (2006). *Ingeniería del conocimiento Adquisición del conocimiento.* Las Palmas de Gran Canaria: Facultad de Informática Universidad de Las Palmas de G.C.

Róspide, C., & Puente, C. (2012). *Agente Virtual Inteligente Aplicado a un Entorno Educativo.*

Russell, S., & Norvig, P. (2008). *Inteligencia Artificial Un Enfoque Moderno.* Madrid: PEARSON Prentice Hall.

Sánchez Anzola, N. (2015). Máquinas de soporte vectorial y redes neuronales artificiales en la predicción de movimiento USD/COP spot intradiario. *ODEON*, 113-172.

Schneider, C. (2016, Mayo 25). *IBM.* Retrieved from www.ibm.com: https://www.ibm.com/blogs/watson/2016/05/biggest-data-challenges-might-not-even-know/

Segarra Ciprés, M., & Bou Llusar, J. C. (2005). Concepto, tipos y dimensiones del conocimiento: configuración del conocimiento estratégico. *Revista de Economía y Empresa*, 175-195.

SEO Cómo triunfar en buscadores. (2010). In M. Orense Fuentes, & O. Isaac , *SEO Cómo triunfar en buscadores* (p. 22).

SyTrue. (2015, Marzo 31). *Hit Consultant*. Retrieved from http://hitconsultant.net: http://hitconsultant.net/2015/03/31/tapping-unstructured-data-healthcares-biggest-hurdle-realized/

Uji. (2018). *FICHEROS Y BASES DE DATOS (E44).* Retrieved from http://www3.uji.es/~aliaga/e44/Tema_01.pdf

Valladares, L., & Olivé, L. (2015). *¿Qué son los conocimientos tradicionales? Apuntes epistemológicos para la interculturalidad.* México: SciELO.

Venkataramanan, M. (2015, Junio 19). *BBC.* Retrieved from BBC: http://www.bbc.com/mundo/noticias/2015/06/150618_vert_fut_super_reconocedores_rostros_yv

Vidal González, M. (2015). *El uso del Perceptrón Multicapa para la clasificación de patrones en conductas adictivas.* Palma: Universidad de las Islas Baleares.

Villena Román, J., Crespo García, R., & García Rueda, J. (2012). *Sistemas basados en conocimientos.* Universidad Carlos III de Madrid.

Zapata Restrepo, M. P., Villada Arango, J. F., & Arias Chaverra, J. A. (2017). *Propuesta de un modelo de gestión del conocimiento para la gerencia de base de datos del grupo Bancolombia.* Medellín: Universidad de San Buenaventura Medellín.

Anexos

Anexo 1.

Ver documento Tabla de enlaces.docx

Anexo 2.

Ver documento Ejemplos.csv

Anexo 3.

Ver documento Plan de Pruebas.docx