DISEÑO Y DESARROLLO DE UN AGENTE INTELIGENTE CON LA CAPACIDAD DE INTERACTUAR DE FORMA ESCRITA CON SERES HUMANOS MEDIANTE EL USO Y ANÁLISIS DEL LENGUAJE NATURAL CASTELLANO CON EL FIN DE UTILIZARLO COMO AYUDANTE VIRTUAL EN SITIOS WEB

MARCO ANTONIO ALMANZA IBARRA

EFRAÍN HERRERA JIMÉNEZ

JUAN SALVADOR NORIEGA MADRID

UNIVERSIDAD TECNOLÓGICA DE BOLÍVAR

FACULTAD DE INGENIERÍA DE SISTEMAS

CARTAGENA DE INDIAS D.T. Y C.

2004

DISEÑO Y DESARROLLO DE UN AGENTE INTELIGENTE CON LA CAPACIDAD DE INTERACTUAR DE FORMA ESCRITA CON SERES HUMANOS MEDIANTE EL USO Y ANÁLISIS DEL LENGUAJE NATURAL CASTELLANO CON EL FIN DE UTILIZARLO COMO AYUDANTE VIRTUAL EN SITIOS WEB

MARCO ANTONIO ALMANZA IBARRA

EFRAÍN HERRERA JIMÉNEZ

JUAN SALVADOR NORIEGA MADRID

UNIVERSIDAD TECNOLÓGICA DE BOLÍVAR

FACULTAD DE INGENIERÍA DE SISTEMAS

CARTAGENA DE INDIAS D.T. Y C.

2004

DISEÑO Y DESARROLLO DE UN AGENTE INTELIGENTE CON LA CAPACIDAD DE INTERACTUAR DE FORMA ESCRITA CON SERES HUMANOS MEDIANTE EL USO Y ANÁLISIS DEL LENGUAJE NATURAL CASTELLANO CON EL FIN DE UTILIZARLO COMO AYUDANTE VIRTUAL EN SITIOS WEB

MARCO ANTONIO ALMANZA IBARRA

EFRAÍN HERRERA JIMÉNEZ

JUAN SALVADOR NORIEGA MADRID

Tesis para optar al título de

Ingenieros de Sistemas

Director

MOISÉS QUINTANA ÁLVAREZ

MSC en Informática

UNIVERSIDAD TECNOLÓGICA DE BOLÍVAR

FACULTAD DE INGENIERÍA DE SISTEMAS

CARTAGENA DE INDIAS D.T. Y C.

2004

Cartagena de Indias, 15 de noviembre de 2004

Señores
UNIVERSIDAD TECNOLÓGICA DE BOLÍVAR
COMITÉ DE EVALUACIÓN
L.C.

Respetados señores:

Cordialmente nos dirigimos a ustedes para informarles que el trabajo de grado titulado "**Diseño y desarrollo de un agente inteligente con la capacidad de interactuar de forma escrita con seres humanos mediante el uso y análisis del lenguaje natural castellano con el fin de utilizarlo como ayudante virtual en sitios web**" ha sido desarrollado de acuerdo a los objetivos establecidos. Como autores del proyecto consideramos que el trabajo es satisfactorio y amerita ser presentado ante ustedes.

Agradeciendo su atención a la presente.

_____	_____
MARCO ANTONIO ALMANZA IBARRA	EFRAÍN HERRERA JIMÉNEZ
Código 9905008	Código 9905035

_____
JUAN SALVADOR NORIEGA MADRID
Código 9905045

Cartagena de Indias, 15 de noviembre de 2004

Señores
UNIVERSIDAD TECNOLÓGICA DE BOLÍVAR
COMITÉ DE EVALUACIÓN
L.C.

Respetados señores:

Cordialmente me dirijo a ustedes para informarles que el trabajo de grado titulado "**Diseño y desarrollo de un agente inteligente con la capacidad de interactuar de forma escrita con seres humanos mediante el uso y análisis del lenguaje natural castellano con el fin de utilizarlo como ayudante virtual en sitios web**" ha sido desarrollado de acuerdo a los objetivos establecidos.

Como director del proyecto considero que el trabajo es satisfactorio y amerita ser presentado ante ustedes.

Agradeciendo su atención a la presente.

_____
MOISÉS QUINTANA ÁLVAREZ

Nota de aceptación

_____

_____

_____

_____

_____
Presidente del jurado

_____
Jurado

_____
Jurado

Cartagena de Indias (día,mes,año)

A Dios por ser tan hermoso con nosotros quien nos inspira y nos da las fuerzas para seguir adelante.

A mis padres por estar siempre a mi lado, formándome en cariño, amor y valores.

A mis lindas hermanas Erika y Naty, quienes con su amor traen cada día felicidad a mi corazón.

A Adriana, por apoyarme y motivarme en todo momento con su cariño.

A todos mis amigos y hermanos quienes con su amistad me enriquecen cada día.

MARCO ANTONIO ALMANZA IBARRA

A Dios todopoderoso quien me inspira y me guía en todo momento.

A Auristela Jiménez Olivo, por incentivar en mí, con su alegría en cada uno de mis logros, las ganas de luchar para ser siempre mejor.

A Ana, Mary, Angélica, mis sobrinos, mis primos, tías y tíos.

A Nelsi, por encargarse de todo lo demás durante mis estudios.

A mi papá, por financiarme los estudios.

A todos mis amigos y profesores de la universidad por contribuir en mi formación académica y personal.

EFRAÍN HERRERA JIMÉNEZ

A Dios todopoderoso quien siempre está con nosotros y nos guía en todo momento.

A mi Mamá por apoyarme con todo su amor, por estar siempre a mi lado y por todos los esfuerzos que hizo para que yo pudiera estudiar.

A mi Abuela que aunque ya no esta conmigo me inspiró a siempre seguir adelante.

A todos mis amigos por entenderme y apoyarme.

A los profesores que me sirvieron de ejemplo y me dieron su apoyo.

JUAN SALVADOR NORIEGA MADRID

AGRADECIMIENTOS

Los autores expresan sus agradecimientos a:

Moisés Quintana Álvarez, Msc. y Director de la investigación, por todo el apoyo brindado, por los conocimientos transmitidos y por el buen ejemplo que siempre significó para nosotros.

Gonzalo Garzón, Ingeniero de Sistemas y Director del programa de Ingeniería de Sistemas por escucharnos y colaborarnos durante toda la carrera.

Giovanny Vásquez Mendoza, Ingeniero de Sistemas, quien fue la persona que nos guió a dar el primer paso en el mundo de la lógica y la programación.

Nuestros amigos de carrera, con quienes estudiamos, trasnochamos y pasamos buenos momentos, a quienes nunca olvidaremos y consideraremos siempre nuestros amigos.

Todas aquellas personas que de una u otra manera colaboraron con el desarrollo de nuestro proyecto y nos impulsaron a soñar.

CONTENIDO

LISTAS DE TABLAS

Pág.

LISTAS DE FIGURAS

Pág.

LISTAS DE ANEXOS

Pág.

GLOSARIO

Agente Inteligente: El concepto de agente proviene de la capacidad de delegar en una persona el poder para gestionar tareas en nombre de otra. Este concepto se incorpora al software bajo el nombre de agente inteligente. Entre las capacidades de estos agentes figura la movilidad, la capacidad de toma de decisiones, intercomunicación y su autonomía reconociendo los intereses del usuario al que representa. Pueden realizar tareas guiados por reglas lógicas, utilizar gran cantidad de ejemplos para dotarlos de capacidades inductivas o simplemente observar a un usuario y agilizarle su trabajo. Se encargan por ejemplo de realizar tareas laboriosas, repetitivas, de control, etc.

Applet: Programa en Java, ejecutable por un navegador. Dícese también de cualquier pequeño programa que se acopla al sistema.

Chatbot: Es un programa que intenta simular una conversación escrita con el propósito de hacerle creer a un humano que están hablando con otra persona. Los chatbots analizan los patrones de diálogo empleados por su interlocutor humano, para luego buscar una respuesta adecuada, por ello, aunque no entiendan lo que se les está diciendo, pueden analizar la frase y presentar una respuesta totalmente lógica e incluso sensible.

Cliente: Programa que se usa para contactar y obtener datos de un programa de servidor localizado en otro ordenador, a menudo a gran distancia. Cada programa cliente está diseñado para trabajar con uno o más tipos de programas servidores específicos, y cada servidor requiere un tipo especial de cliente.

Dirección IP: Es un número de 32 bits que identifica a cada ordenador que se encuentre conectado a la red. Esta dirección debe ser única para cada host, y normalmente suele representarse como cuatro cifras de 8 bit separadas por puntos.

Gramática: Una fórmula o conjunto de fórmulas usada para describir relaciones lógicas entre unidades léxicas u otros símbolos terminales. La gramática se usa en una lengua, en matemática y en lógica para describir o prescribir la estructura de un lenguaje ya sea humano, ya sea artificial.

Gramática Libre de Contexto: Un conjunto finito de símbolos o variables que representan categorías aplicables a elementos de léxico. La gramática libre de contexto es muy útil para definir relaciones entre objetos sintácticos tales como la sintaxis de un lenguaje de programación. Una gramática libre de contexto tiene un símbolo de arranque o de objetivo, y utiliza símbolos terminales que representan ejemplos de variables y reglas de producción que combinan entre sí para el logro del objetivo.

IA (Inteligencia Artificial): Término aplicado tanto a sistemas y programas informáticos capaces de realizar tareas complejas simulando el funcionamiento del pensamiento humano, como al estudio metodológico de estos sistemas y sus capacidades.

JAVA: Lenguaje de programación orientado a objeto parecido al C++. Usado en WWW para la telecarga y telejecución de programas en el computador cliente. Desarrollado por Sun Microsystems.

JRE(Java Runtime Environment): Es el entorno mínimo para ejecutar programas de Java. Incluye el Java Plug-in que necesitan lo navegadores (Explorer o Netscape) para poder ejecutar los applet.

JDBC(Java Database Connectivity): es un API (Application Programming Interface) usado para enviar comandos SQL hacia una base de datos relacional, como Oracle, Infomix, MySQL entre otras.

OpenSource: Es un tipo de software que la persona que quiera puede usar y modificar. Cualquiera puede descargarlo de Internet y usarlo sin pagar por ello. Inclusive, cualquiera que lo necesite puede estudiar el código fuente y cambiarlo de acuerdo a sus necesidades.

PLN (Procesamiento del Lenguaje Natural). Rama de la Inteligencia Artificial encargada de la comprensión y generación del lenguaje natural por parte de sistemas computacionales inteligentes.

Sintaxis: Rama de la gramática que se encarga del estudio de la funcionalidad de las palabras dentro de las oraciones.

Semántica: Rama de la gramática que se encarga del estudio del significado de las palabras.

Silogismo: Una fórmula lógica con premisas seguidas de una conclusión: SI a Y b, ENTONCES c.

Usenet: Derivado de Users Network (red de usuarios) es un medio de comunicación en el cual los usuarios leen y envian mensajes (denominados "artículos") a distintos foros distribuidos (denominados "grupos de noticias"). El medio se sostiene gracias a un gran número de servidores, que guardan y se pasan los mensajes de unos a otros. Usenet tiene una importancia cultural significativa en el mundo reticulado, habiendo dado lugar al nacimiento, o popularizado, conceptos ampliamente reconocidos, como "FAQ" y "spam".

RESUMEN

En este trabajo de investigación se describe el diseño de una aplicación capaz de mantener una conversación coherente con seres humanos utilizando el lenguaje castellano. La aplicación es el resultado de una investigación en el área de Procesamiento del Lenguaje Natural, gramática castellana e inteligencia humana, con lo cual se ha logrado diseñar un esquema, a nivel de base de datos, de la forma como se organizan las ideas en el cerebro y de la estructura sintáctica de oraciones simples dentro de una gramática castellana abreviada.

Por medio del análisis del lenguaje castellano en cuanto a morfología, sintaxis y semántica, se obtuvo un conjunto de reglas básicas para conformar oraciones simples y oraciones interrogativas, teniendo en cuenta los diferentes tipos de palabras y un conjunto de sus propiedades. Tanto los tipos de palabras y sus respectivas propiedades fueron registradas en la base de datos, al igual que las reglas gramaticales que utiliza el sistema para reconocer el texto que recibe. El sistema luego de recibir una oración, busca en la base de datos el tipo y las propiedades de todas las palabras que conforman la oración, para realizar con esto, el proceso de revisión de las reglas gramaticales, el cual se efectúa descomponiendo la oración en pares de palabras y verificando que existe una regla en la cual encajen; si existe dicha regla, se toma el producto (sintagma) de ella y este se analiza con los sintagmas producidos por los demás pares de

palabras, con lo cual se obtiene un árbol binario cuya raíz es el producto final de la unión de todos los sintagmas y que además indica si la frase es una oración simple o una pregunta, mientras que los nodos hojas del árbol, son cada una de las palabras escritas. Si la frase es una oración, se procede a tomar cada uno de los sintagmas para almacenarlos como complementos de las ideas que almacena el sistema, teniendo como núcleo de la idea el verbo principal o núcleo del predicado de la oración. Si la frase es una pregunta se busca dentro de las ideas del sistema si existe una respuesta para el tipo de pregunta, teniendo en cuenta los sintagmas encontrados en dicha frase interrogativa. Las ideas son almacenadas de tal forma que los sintagmas que hacen parte de ellas no contengan exclusivamente las palabras recibidas sino el contexto en el que ellas se encuentran, obteniendo así un sistema más inteligente que los ya conocidos. Los contextos donde se encuentran las palabras son principalmente conjuntos de sinónimos, y son revisados constantemente por el sistema con el fin de maximizar su capacidad expresiva, ya que es normal para una persona decir la misma información utilizando diferentes palabras una y otra vez.

Abstract: This work describes the design of an application that is able to keep conversations with human beings using spanish language. The application is the result of an investigation in the area of Natural Language Processing, spanish grammar and human intelligence, which has made possible to design a database level scheme of the way the brain organizes the ideas and the syntactic structure of simple sentences within an abbreviated spanish grammar.

Keywords: Spanish grammar. Natural Language Processing. Intelligent software.

INTRODUCCIÓN

La Inteligencia Artificial, debido a las muchas conjeturas planteadas durante sus inicios y a lo difícil que ha significado lograr que máquinas se comporten y piensen como seres humanos, se ha convertido en un reto para los ingenieros de sistemas actuales, pero además empieza a vislumbrarse como una herramienta útil en muchas áreas de interés, como son el entretenimiento, la salud, los viajes espaciales, los trabajos riesgosos, la toma de decisiones, etc.

En esta época, donde la información juega un papel esencial en la vida cotidiana, el uso y desarrollo de inteligencia artificial con fines informativos, puede generar cambios muy positivos dentro de la sociedad, por ejemplo aumentaría la disponibilidad y la fiabilidad de los datos, posibilitaría la traducción instantánea en múltiples idiomas, aumentaría la efectividad de los sistemas expertos, entre muchas otras nuevas aplicaciones y avances que se pueden generar.

Este proyecto de investigación muestra una metodología para diseñar un agente inteligente cuyo objetivo principal es ofrecer cualquier tipo de información y que puede ser utilizado en cualquier área, ya sea con fines netamente informativos, publicitarios, de entretenimiento o de servicio técnico.

En el primer capitulo de este documento se presenta una recopilación de todo el material de investigación que fue útil para el desarrollo del proyecto. El material descrito pertenece a diversas disciplinas como son, Inteligencia Artificial, gramática castellana, pensamiento humano y programación de computadores.

En el segundo capítulo se detalla paso a paso el diseño del agente inteligente, describiendo la gramática que utiliza el agente, la base de conocimiento, la estructura de las bases de datos necesarias, los procesos de análisis y almacenamiento de oraciones y la creación de nuevas ideas.

En el tercer capítulo se describe la implementación de un prototipo creado utilizando herramientas de software gratuitas y ampliamente reconocidas en el actual ámbito de la programación.

Finalmente se especifican las áreas en donde se puede aplicar un agente inteligente con las características descritas en los capítulos anteriores, además se dejan ciertas pautas para futuros adelantos en este proyecto.

1. MARCO TEÓRICO

1.1 INTELIGENCIA ARTIFICIAL

De acuerdo con la Enciclopedia Encarta [EMENCARTA, 2003], El término IA se aplica a sistemas y programas informáticos capaces de realizar tareas complejas, simulando el funcionamiento del pensamiento humano, aunque todavía muy lejos de éste. En esta esfera los campos de investigación más importantes son el procesamiento de la información, el reconocimiento de modelos, los juegos y las áreas aplicadas, como el diagnóstico médico. Algunas áreas de la investigación actual del procesamiento de la información están centradas en programas que permiten a un computador comprender la información escrita o hablada, generar resúmenes, responder a preguntas específicas o redistribuir datos a los usuarios interesados en determinados sectores de esta información. En esos programas es esencial la capacidad del sistema de generar frases gramaticalmente correctas y de establecer vínculos entre palabras e ideas.

Según Sergio A. Moriello [MORIELLO, 1999]: La inteligencia artificial fundamentalmente intenta simular en sistemas informáticos la mayoría de actividades cognitivas complejas que caracterizan a los seres humanos. Casi desde su inicio, en la década del 50, la IA se dividió en dos paradigmas: el enfoque simbólico y el enfoque conexionista.

1.1.1 Enfoque simbólico: Para esta corriente, que dominó durante las décadas del 60 al 80, la mente es una máquina de procesamiento de información, en donde no interesan los mecanismos biológicos subyacentes. Su estrategia recibe el nombre de "enfoque descendente o Top-Down", ya que a través de un exhaustivo y detallado conjunto de reglas pre-programadas, se espera que el sistema sea capaz de desenvolverse en el mundo real haciendo énfasis en los aspectos más cognitivos del ser humano, como la lógica, las habilidades deductivas, la capacidad de procesamiento, el razonamiento, la manipulación de símbolos mentales y la combinación de conceptos. Desde aquí se hicieron grandes avances en algunas áreas específicas, generalmente ligadas a actividades concretas y susceptibles de formalización. Los ejemplos más significativos son los Sistemas Expertos, los programas de resolución de problemas matemáticos o los programas de juegos, como el ajedrez.

1.1.2 Enfoque conexionista: La corriente conexionista, que resurgió a partir de la década del 80 gracias al progreso de las neurociencias, intenta imitar la esencia procedente de la inteligencia, reproduciendo las propiedades observables de los mecanismos biológicos. Su estrategia recibe el nombre de "enfoque ascendente o Bottom-Up" ya que dejan que el sistema se desarrolle a partir de su propia experiencia haciendo énfasis en los aspectos más perceptivos del ser humano, como el aprendizaje, las habilidades asociativas, las capacidades sensoriales, la generalización, la autonomía y el reconocimiento de patrones. Los ejemplos más significativos son las Redes Neuronales Artificiales y los Algoritmos Genéticos.

1.1.3 Comparación de ambos enfoques: Para los simbólicos la esencia de la mente se encuentra en el software; por eso simulan directamente las características inteligentes que se pretenden conseguir (es decir, la actividad racional), manipulando símbolos por medio de reglas lógicas y utilizando poderosos computadores de un solo procesador, que ejecutan miles de instrucciones una después de la otra. Para los conexionistas, en cambio, el fundamento del intelecto está en el hardware; por eso imitan los elementos de más bajo nivel que componen o intervienen en los procesos inteligentes, con la esperanza en que de su combinación emerja de forma espontánea el comportamiento inteligente, a través de una red densamente conectada de unidades simples que procesan simultáneamente partes interactuantes del mismo problema. A las máquinas deductivas hay que decirles todo, pues su conducta sigue un camino orientado por reglas fijas; a las máquinas inductivas no hay que decirles nada, pues pueden aprender y generalizar a partir de ejemplos.

La lucha por crear máquinas inteligentes lleva ya muchos años, un ejemplo claro de esto ocurre entre el matemático Douglas Lenat, de la empresa Cyc y el profesor Rodney Brooks, del Instituto Tecnológico de Massachusetts (MIT). Conceptualmente sus propuestas representan dos caminos en extremo diferentes: Cyc es un acercamiento descendente (lo último en máquinas de sentido común preprogramadas) y Cog es un acercamiento ascendente (lo más avanzado en robots humanoides).

Cyc, cuyo nombre proviene de “enCYClopedic”, es un programa que cuenta con un enorme poder inferencial y que trabaja con una gran base de conocimientos, la cual contiene más de un millón de reglas de sentido común acerca del mundo, así como deducciones hechas por el mismo Cyc. La esperanza de Lenat es que Cyc se convierta en una gigantesca base de conocimientos que pueda servir de plataforma para dotar a las futuras computadoras con una parte significativa del conocimiento general y de sentido común de un adulto. Lenat cree que Cyc comenzará a aprender por sí mismo y con mayor rapidez (leyendo libros, diarios y revistas electrónicas) cuando cuente, finalmente, con la increíble cifra de 100 millones de enunciados de conocimientos generales. Mientras tanto algunas decenas de personas examinan y analizan, oración por oración y sentencia a sentencia, artículos de libros y revistas, noticias y novelas, definiciones en enciclopedias y anuncios, y luego introducen en su base de conocimientos no sólo lo que se expresa explícitamente, sino también los datos que el escritor asume que cualquier lector conoce. Es este conocimiento previo, común y generalizado (y no el contenido del texto), lo que hay que codificar.

Del otro lado está Cog, cuyo nombre proviene de “COGnitivo”. Se trata de un ambicioso experimento inspirado en estructuras biológicas, en vez de lógicas, que consiste en dejar que un robot humanoide descubra por sí mismo el complejo entorno en el que se encuentra, al igual que lo hacen los niños, en lugar de llenar su memoria con una detallada descripción del mundo desde la particular perspectiva humana. Cog es como un recién nacido que debe aprender a hablar, caminar y tomar decisiones. A través de sus sentidos artificiales de visión,

audición y tacto (cámaras fotográficas, micrófonos y una membrana sintética sensible al contacto) absorbe información que alimenta su sentido común. La idea es darle a la máquina unos conceptos básicos, situarla en un entorno que pueda percibir y en el que pueda actuar y, de esa manera, averiguar qué logra aprender por sí misma. Más adelante, una vez que conozca los objetos del mundo que le rodea, podrá aprender mediante interacciones con los seres humanos. Se espera que en algunos años más tenga la capacidad mental de un bebé de seis meses y su capacidad intelectual futura se potenciará enormemente cuando cuente con casi 300 microprocesadores. La meta de Brooks es alcanzar una inteligencia humana más que una simple inteligencia. Para eso, la máquina debe tener experiencias humanas y la capacidad de relacionarse con su entorno como lo hace un ser humano, es decir, percibiendo el mundo a través de los sentidos y desenvolviendo su conducta en un contexto social, en interacción con otras personas. Cog no funciona bajo un programa, sino que se reprograma según su experiencia.

1.1.4 Procesamiento del Lenguaje Natural: El procesamiento del lenguaje natural es el área de la IA encargada de estudiar la comprensión y generación de lenguaje natural en sistemas computacionales.

El procesamiento del lenguaje natural (PLN) es una de las áreas de la IA que se puede utilizar en un gran número de aplicaciones, entre las cuales se encuentran [Paredes, 2002]:

- Traducción automática: Básicamente es la traducción correcta de un lenguaje a otro tanto sintáctica como semánticamente.
- Recuperación de la Información: En la cual un sistema puede buscar información utilizando como referencia datos introducidos por el usuario en lenguaje natural.
- Análisis de un Texto y Extracción de Resúmenes: Proporcionándole cualquier texto, el sistema será capaz de analizarlo y extraer un resumen coherente basado en las ideas del contenido.
- Tutores Inteligentes: Sistemas capaces de transmitir conocimiento, con la capacidad de evaluar y de adaptarse a los diferentes tipos de estudiantes.
- Sistemas expertos avanzados: Sistemas expertos capaces de recibir las consultas en lenguaje natural.

La investigación, dentro de las aplicaciones de PLN, se enfocó hacía la recuperación de información (sistemas de búsqueda y respuesta) debido a que esta sería la principal funcionalidad del agente.

Esta investigación reveló [CIMIA, 2004] que los sistemas actuales de búsqueda y respuesta se orientan en responder preguntas simples sobre hechos concretos a partir de una colección de documentos donde la respuesta se encuentra en forma explícita en un sólo documento, sin embargo, lo que se quiere en un sistema inteligente es que pueda resolver preguntas más complejas a partir de la fusión de la información contenida en varios documentos. Los sistemas inteligentes de búsquedas y respuestas típicamente consideran los siguientes procesos: El

análisis de la pregunta, la recuperación de documentos relacionados, la selección de pasajes relevantes, y la extracción de fragmentos respuesta. Dependiendo de las técnicas y herramientas de PLN empleadas en estos procesos, los sistemas de búsqueda y respuesta se pueden clasificar en las siguientes cuatro categorías:

1.1.4.1 Sistemas que no utilizan técnicas de PLN: Estos sistemas se basan en la recuperación de extractos de texto relativamente pequeños con la suposición de que dichos extractos contendrán la respuesta esperada. Generalmente utilizan varias formas de seleccionar aquellos términos de la pregunta que deben aparecer cerca de la respuesta. Normalmente, se eliminan las palabras vacías y se seleccionan aquellos términos con mayor "valor discriminatorio". Estos términos se utilizan para recuperar directamente fragmentos relevantes de texto que se presentan directamente como respuestas.

1.1.4.2 Sistemas que usan información léxico-sintáctica: En esta categoría se incluyen la mayoría de los sistemas existentes. Estos, al igual que los anteriores, emplean técnicas de recuperación de documentos para seleccionar aquellos documentos o pasajes de la colección documental que son más relevantes a la pregunta, sin embargo usan técnicas de PLN para analizar las preguntas y facilitar el proceso de identificación y extracción final de las respuestas. Los sistemas de esta categoría se caracterizan, en primer lugar, por la realización de un análisis detallado de la pregunta que permite conocer o aproximar el tipo de entidad que cada pregunta espera como respuesta. Estas entidades están organizadas en conjuntos de clases semánticas como por ejemplo, "persona", "organización",

"tiempo", "lugar", etc. La identificación del tipo de respuesta esperada se suele afrontar mediante el análisis de los términos interrogativos de la pregunta. Para realizar el análisis de la pregunta se suelen utilizar etiquetadores léxicos y analizadores sintácticos, así como métodos de aprendizaje automático. Por otra parte, el proceso de extracción de la respuesta combina el uso de técnicas de recuperación de documentos para la valoración de extractos reducidos de texto, con el uso de clasificadores de entidades. Estas herramientas permiten localizar aquellas entidades cuya clase semántica corresponde con aquella que la pregunta espera como respuesta. De esta forma, el sistema sólo tiene en cuenta aquellos extractos de texto que contienen alguna entidad del tipo requerido como respuesta. Algunas variantes de esta estrategia general consideran un uso más intensivo de la información sintáctica para evaluar la similitud entre la pregunta y las posibles respuestas; la aplicación de técnicas de aprendizaje basadas en modelos de máxima entropía para estimar la probabilidad de que una respuesta sea correcta; y el uso de la redundancia en la información para enfrentar el problema de la expresividad del lenguaje humano, a través de únicamente información léxica.

1.1.4.3 Sistemas que usan información semántica: El uso de técnicas de análisis semántico es escaso debido fundamentalmente a las dificultades intrínsecas de la representación del conocimiento. De hecho, sólo un grupo reducido de sistemas aplica herramientas que realizan este tipo de análisis. Estas técnicas se utilizan en los procesos de análisis de la pregunta y de extracción final de la respuesta. De forma general, estos sistemas obtienen la representación

semántica de la pregunta y de aquellas sentencias que son relevantes a dicha pregunta. Con ello, la extracción de la respuesta se realiza mediante procesos de comparación y/o unificación entre las representaciones de la pregunta y las frases relevantes. Las representaciones semánticas usadas hasta el momento son: las tripletas semánticas formadas por una entidad del discurso, el rol semántico que dicha entidad desempeña y el término con el que dicha entidad mantiene la relación, y fórmulas lógicas para representar las preguntas y las frases candidatas a contener la respuesta.

1.1.4.4 Sistemas que usan información contextual: La aplicación de técnicas de análisis contextual se restringe a la incorporación de conocimiento general del mundo asociado a mecanismos de inferencia que facilitan el proceso de extracción de respuestas y a la aplicación de procesos de resolución de correferencias. Los trabajos que hasta ahora consideran conocimiento del mundo lo obtienen de ontologías de conceptos como Wordnet, y lo aplican en el análisis de la pregunta así como en la extracción de la respuesta. La resolución de correferencias constituye el conjunto de técnicas de análisis contextual más utilizado en procesos de búsqueda y respuesta. En consecuencia, y aunque estas técnicas se enmarcan en el último nivel del análisis del lenguaje natural, se puede afrontar su utilización sin la aplicación previa de técnicas de análisis semántico. Esta circunstancia provoca que algunos de los sistemas del nivel léxico-sintáctico también apliquen estrategias de resolución de correferencias en sus procesos. Generalmente, las técnicas de resolución de la anáfora se aplican en dos etapas diferentes del proceso: en la extracción de las respuestas y en el análisis de las preguntas. En el

primer caso, la resolución de correferencias se realiza sobre aquellos documentos que son relevantes a la pregunta con la finalidad de facilitar la localización y extracción de entidades relacionadas con la pregunta y la respuesta. En el segundo caso, los sistemas utilizan estas técnicas para seguir la pista de aquellas entidades del discurso referidas de forma anafórica a través de series de preguntas individuales que interrogan al sistema acerca de diferentes aspectos relacionados todos en un mismo contexto.

1.1.4.5 Bots y Chatbot: Un bot se refiere a un programa de computador que recopila información o realiza un servicio. Un bot (algunas veces llamado un agente) típicamente explora Internet, recopila información relativa a sus intereses y la maneja según sus propios parámetros. Los bots obtuvieron notoriedad en IRC (Grupos de Chat en Internet) donde realizaban funciones tales como saludar a nuevos participantes, controlar el uso de lenguaje ofensivo, y algunas veces estar abiertamente molesto. Hoy retoman información, y responden a sucesos en Usenet, la Web y otras esquinas del ciberespacio.

Mientras que algunos bots deambulan por Internet silenciosamente a su orden, otros son interactivos, como por ejemplo los chatbots o chatterbots. Un Chatbot es un software que tiene por objetivo, establecer una conversación coherente con cualquier ser humano (que hable su mismo idioma). Al igual que un sistema experto, tiene la capacidad de almacenar conocimientos y aprender. Está basado en los sistemas expertos, pero es mucho más interactivo y su función no es otra sino conversar.

El primer chatbot, fue Eliza, creado en 1966 por el Profesor Joseph Weizenbaum de MIT (Instituto de Tecnología Massachusetts), para estudiar la comunicación con lenguaje natural entre el hombre y el computador.

1.1.5 Actualidad: A mediados del siglo pasado se soñaba con que una máquina de cómputo lograra ser inteligente o mostrara principios de inteligencia, este sueño se convirtió en el objetivo principal de muchos centros de investigación en el mundo entero, por esta razón la Inteligencia Artificial (IA) hoy día se ha convertido en uno de los principales enfoques de la investigación en universidades e instituciones.

Algunos expertos consideran que en la actualidad no se ha logrado avanzar mucho en el área de la Inteligencia Artificial, teniendo en cuenta todas las predicciones que aparecieron hace ya varias décadas, y que aún queda mucho camino por recorrer hasta lograr, por ejemplo, realizar con éxito el experimento de Turing [Turing, 1950, 433], en el cual un ser humano no debería distinguir entre las respuestas de otro ser humano y las respuestas de una máquina para probar que la máquina era inteligente.

Entre las predicciones de hace varias décadas y los comentarios recientes cabe destacar: “Dentro de una generación, el problema de crear inteligencia artificial estará sustancialmente solucionado” [M. Minsky, 1967], “Los logros de la investigación en IA son exíguos, incluso de juguete, si se comparan con el objetivo final de construir un computador que funcione al nivel de un ser humano

inteligente en el complejo y caótico mundo real" [J. Copeland, 1993, 85], "El área en la que IA se encuentra con las mayores dificultades es la programación del sentido común…. Hace tiempo que se ha reconocido que es mucho más fácil escribir un programa que lleve a cabo complicadas operaciones formales, que un programa que refleje el sentido común de un perro" [Winograd et al, 1986, 98], "Quizá la crítica más importante al trabajo reciente en IA es que todavía no ha conseguido formalizar lo que llamamos sentido común" [Waltz, 1982, 122], "Veinticinco años de investigación en IA han servido para hacer realidad muy pocas de sus promesas y han fracasado en producir evidencia alguna de que lo harán en el futuro" [Dreyfus, 1986, xi] y "Las actuales afirmaciones y esperanzas sobre progresos en la construcción de computadores inteligentes son como la creencia de que alguien subiendo a un árbol está haciendo progresos en su objetivo de alcanzar la luna" [10].

Entre los pocos éxitos que se han logrado en el área de la IA se pueden mencionar [UNAV]: Deep Thought y Hitech, los cuales juegan ajedrez al nivel de los maestros; El General Problem Solver, el cual resuelve problemas como el de las Torres de Hanoi o el de los misioneros y caníbales; El robot Shrdlu, el cual sostiene conversaciones en inglés coloquial sobre su mundo de bloques y pirámides, puede formar y ejecutar planes, y puede discutir sus motivos; El Logic Theorist, el cual demuestra teoremas en lógica simbólica; El sistema experto Mycin, el cual puede diagnosticar la enfermedad de un paciente a partir de la información sobre los síntomas de la enfermedad y el resultado de los análisis.

Recientemente se ha hecho más significativa la necesidad de que los sistemas puedan comunicarse de forma natural con el usuario, y que sean capaces de entenderlo, y uno de los campos de la inteligencia artificial, llamado Procesamiento del Lenguaje Natural (PLN), se encarga de este tipo de sistemas. El PLN es una de las ramas de la IA en las cuales se ha hecho mucho énfasis en los últimos años, buscando el ideal de que el ser humano pueda comunicarse con las máquinas hablando o escribiendo naturalmente, sin necesidad de complicados comandos, haciendo más sencilla la interacción con ellas.

En la actualidad, estudiosos del tema se han congregado con el fin de establecer grupos de investigación para desarrollar agentes inteligentes con la capacidad de interactuar con los seres humanos con la modalidad pregunta-respuesta, además se busca que estos agentes aprendan o enriquezcan su conocimiento a medida que interactúan con personas.

A pesar que aún no se han logrado las metas propuestas en los inicios de IA, en la última década PLN ha ido evolucionando, quizás con mayor celeridad que otras disciplinas. Actualmente se pueden encontrar comunidades opensource de desarrollo como A.L.I.C.E liderada por el Dr. Richard Wallace, creador de ALICE BOT con el lenguaje AIML, cuyos avances son muy significativos en esta área [ALICE, 2004]. Otro grupo con muy buenos logros es Thinking Bits, ellos a diferencia de A.L.I.C.E. no son opensource pero entregan demos de pruebas para que se evalué el comportamiento de sus agentes. También existen muchas empresas interesadas en el proceso del lenguaje natural, Microsoft por ejemplo

con su ayudante en MS Office, ofrece la posibilidad de buscar en los archivos de ayuda sólo con describir el problema, esto no es PLN pero aplica muchos de los conceptos de esta rama. Otras empresas se dedican también a crear programas que puedan entender lo que el usuario desea.

Desde 1990, se realiza anualmente un concurso, llamado “The Loebner Prize”, en el cual se implementa el Test de Turing, entregando un premio de US$2.000 a aquel programa de computador cuyas respuestas no sean diferenciadas de las de un humano (o por lo menos, que se acerque a ello). El último ganador de este concurso fue el Dr. Richard Wallace con A.L.I.C.E..

1.2 PENSAMIENTO HUMANO

El diseño de la relación de las ideas está basado en la representación preposicional de la memoria del modelo de Quillan y la teoría de silogismos.

1.2.1 Modelo de Quillan: Quillan desarrolló su estudio en el ámbito de la lingüística computacional. Creó un modelo de ordenador, el TCL (Teachable Language Comprehender), cuyos objetivos eran la “comprensión” de textos ingleses y la capacidad de responder a preguntas. Quillan fue el primero que empleó la denominación “memoria semántica” para referirse al conocimiento conceptual organizado, almacenado en la memoria de un hablante. También tiene el mérito de haber elegido el formato proposicional para representar la estructura de datos en la memoria TCL y, por extensión en la memoria humana.

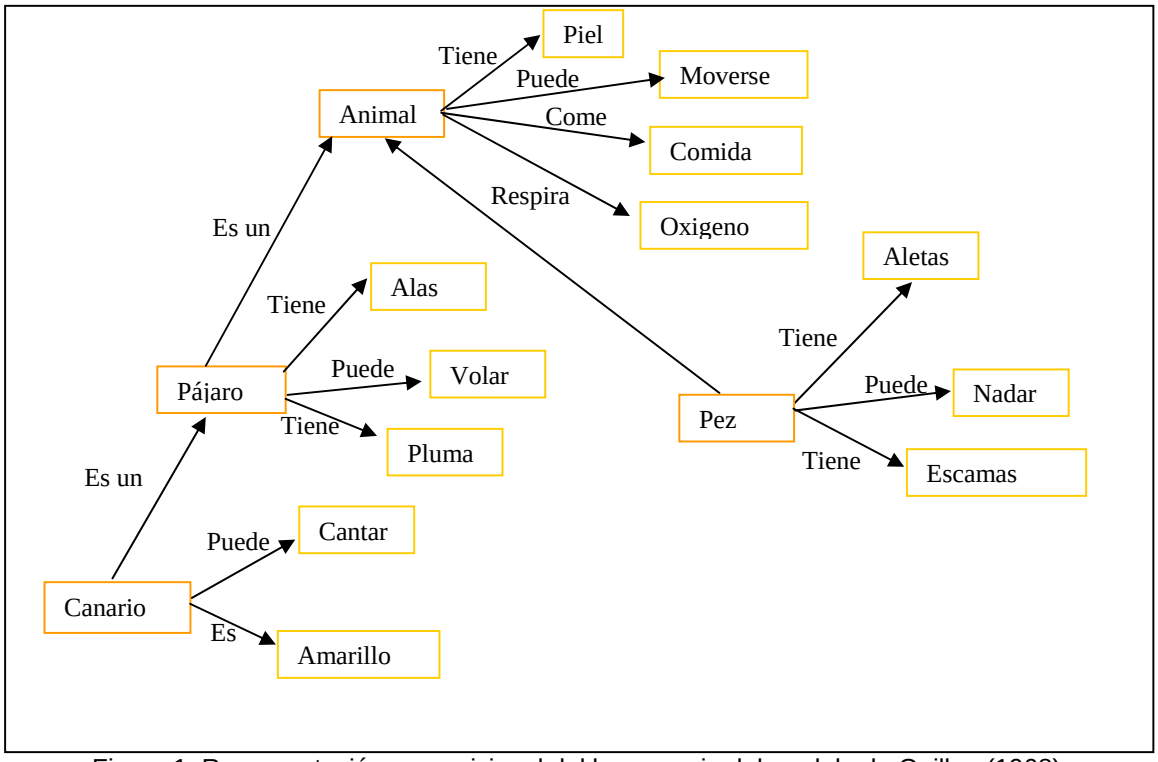


Figura 1. Representación proposicional del la memoria del modelo de Quillan (1968) [De la vega,1998]

A pesar de las críticas recibidas por Quillan y dirigidas sobre todo a los presupuestos teóricos del modelo, sus trabajos abrieron paso a nuevas investigaciones entre las que cabe destacar el proyecto ACT de Anderson (1983). El modelo de Anderson puede considerarse dentro de las teorías proposicionales de amplio espectro, es decir, pretende explicar un buen número de procesos mentales (comprensión y generación de frases, memoria, razonamiento) frente a la especialización de otras teorías proposicionales que se aplican al ámbito de una sola tarea experimental. [Luengo,1991]

1.2.2 Silogismos: Un silogismo es un conjunto de reglas que establece qué conclusiones pueden ser alcanzadas desde un conjunto de proposiciones o

"premisas". Cada regla del silogismo establece una transición desde las premisas a la conclusión que es intuitivamente evidente. El proceso de alcanzar una conclusión es también llamado "hacer una inferencia". Un ejemplo que puede ilustrar este método de razonamiento es un silogismo de la lógica tradicional denominado *silogismo categórico*. Este silogismo exige que las premisas y la conclusión estén escritas en la forma de proposiciones categóricas, es decir, en una de cuatro formas que caracterizan una posible relación entre dos clases. Estas proposiciones adoptan la siguiente estructura general:

Cuantificador Sujeto Cópula Predicado

Las cuatro formas de proposiciones son: Todo S es P (afirmación universal); Ningún S es P (negación universal); Algún S es P (afirmativo particular); Algún S es no P (negativo particular).

El *silogismo categórico* dice que si es verdadera una instancia cualquiera de las siguientes dos premisas:

(Premisa mayor) Todo M es A. (Todos los mamíferos son mortales)

(Premisa menor) Todo B es M. (Todos los hombres son mamíferos)

Entonces la siguiente *conclusión* es válida:

Todo B es A. (Todos los hombres son mortales)

Aquí, M es llamado término medio, A es llamado término mayor y B es llamado término menor.

Un paso deductivo o instancia de dos premisas y su conclusión es también llamado argumento simple. Un argumento complejo es un argumento compuesto de muchos pasos deductivos. Un argumento complejo es lógicamente válido si cada paso puede explicarse mediante una regla del silogismo. Así, si una persona considera como válidas las premisas, entonces debe considerar válida la conclusión que se sigue de ellas, ya que ésta sólo expresa de manera explícita una información que está contenida de manera implícita en aquéllas.

1.3 GRAMÁTICA

El diseño de una aplicación con la capacidad de comunicarse con seres humanos en un idioma en particular, requiere obviamente de un estudio que analice las características principales de su gramática.

El estudio de la gramática castellana, aparte de confirmar la complejidad de esta lengua, permitió obtener una clasificación de las palabras en cuanto a sus propiedades morfológicas y semánticas, y además extraer un conjunto de reglas sintáctico-semánticas que ayudan a analizar tanto oraciones simples como interrogativas.

1.3.1 Oración: Una oración es un conjunto de palabras que tiene sentido por sí misma y transmite un mensaje. Generalmente, las oraciones están formadas por dos partes o miembros llamados sujeto y predicado, por eso se dice que son bimembres.

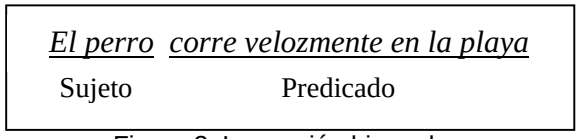


Figura 2. La oración bimembre

Las oraciones que no pueden dividirse en sujeto y predicado se llaman oraciones unimembres. En algunas oraciones se habla de un sujeto pero no aparece en la oración, aunque no aparezca, la oración es bimembre y al sujeto se le llama sujeto tácito.

Las ciencias que se encargan de estudiar las oraciones y sus componentes son:

- Morfología: Estudia los accidentes de las palabras (género, número, persona, tiempo, modo, etc.).
- Sintaxis: Estudia la función de las palabras y las estructuras dentro de la oración en relación con las demás palabras.
- Semántica: Estudia la palabra según su contenido y significado.

1.3.2 Tipos de palabras en castellano: La clasificación más importante y reconocida de las palabras de una gramática es la agrupación en tipos de palabras. En castellano los tipos de palabras son:

Sustantivos: Son palabras que identifican objetos, animales o personas. Los sustantivos comunes señalan a las cosas en general y los sustantivos propios señalan una cosa en particular identificándola por su nombre. Los sustantivos comunes, en singular, pueden señalar un único objeto o un conjunto de ellos, por eso se dividen en sustantivos individuales y colectivos.

Artículos: Son palabras que anuncian a los sustantivos y van delante de ellos. Concuerdan siempre en género y número con los sustantivos que acompañan, aunque existen algunas excepciones, como sucede cuando el sustantivo es femenino y comienza con “a” o “ha” acentuadas, ya que en estos casos se utiliza el artículo masculino “el”.

Adjetivos: Son palabras que acompañan a los sustantivos y siempre concuerdan en género y número con ellos. Describen características de los mismos y de acuerdo a ella los podemos clasificar en:

- Calificativos: Señalan una cualidad.
- Gentilicio: Señalan un lugar de origen o nacionalidad.
- Numeral: Señalan una cantidad u orden.
- Indefinido: Señalan una cantidad en forma imprecisa.
- Demostrativo: Señalan una ubicación.
- Posesivo: Señalan a quien pertenece el sustantivo

Pronombres: Los pronombres personales son palabras que señalan a una persona, animal o cosa, reemplazando al sustantivo en el sujeto, objeto directo, objeto indirecto o circunstancial. Se dividen en tres personas gramaticales. Existen otros tipos de pronombres llamados posesivos, demostrativos e indefinidos, que indican pertenencia, lugar o una cantidad imprecisa respecto de las personas gramaticales, reemplazando al sustantivo.

Verbos: Son palabras que indican una acción que realiza una persona, animal o cosa. Estas acciones pueden realizarse en diferentes tiempos pasado o pretérito, presente y futuro, y se les llama accidentes gramaticales. El verbo está formado por dos partes: la raíz y la terminación. Esta última nos indica el tiempo de la conjugación del verbo. El modo de un verbo indica la manera en que la acción es expresada por quien habla.

Adverbios: Son palabras que modifican a un verbo, un adjetivo o a otro adverbio. En la oración funcionan como circunstanciales o formando parte de modificadores. Son invariables, ya que no tienen género ni número.

Además están las conjunciones y preposiciones, las cuales se utilizan como nexos entre oraciones o sintagmas.

Al sustantivo principal del sujeto de una oración se le llama núcleo del sujeto y al verbo principal del predicado se le denomina núcleo del predicado. Dentro del sujeto hay otras palabras que acompañan al núcleo, llamados modificadores directos o indirectos. A los artículos y adjetivos que acompañan al sustantivo núcleo del sujeto y concuerdan con él en género y número, se les llama modificador directo. Al conjunto de palabras que se unen al sustantivo núcleo del sujeto por medio de preposiciones, se le llama modificador indirecto.

Dentro del predicado de una oración hay otras palabras y construcciones que acompañan a los núcleos, se llaman modificadores del predicado y son objeto

directo, objeto indirecto, complemento agente, circunstancial y predicativo subjetivo. El objeto directo es el modificador del verbo que puede reemplazarse por lo(s) o la(s), ya que es una construcción de núcleo sustantivo. El objeto indirecto está formado por las preposiciones “a” o “para” y un término de núcleo sustantivo, y puede reemplazarse por los pronombres “le” o “les”. El complemento agente es quien realiza la acción cuando la oración está en voz pasiva. El circunstancial es el adverbio o construcción formada por un subordinante y un término, que acompaña al verbo y no puede ser reemplazada por pronombres. El predicativo subjetivo es un sustantivo o construcción de núcleo sustantivo que modifica al verbo y al núcleo del sujeto.

1.4 HERRAMIENTAS, TÉCNICAS Y METODOLOGÍAS DE PROGRAMACIÓN

Entre las herramientas, técnicas y metodologías más recomendadas en la actualidad para desarrollar una aplicación se encuentran, la metodología orientada a objetos, el modelo Cliente-Servidor, sockets, programación con hilos, modelo de Entidad-Relación y el lenguaje de programación JAVA.

1.4.1 Metodología orientada a objetos: La metodología orientada a objetos consiste en construir un modelo de un dominio de aplicación añadiéndosele detalles de implementación durante el diseño de un sistema. Esta metodología es llamada Técnica de Modelado de Objetos (OMT) y consta de las siguientes fases:

- Análisis: El modelo del análisis es una abstracción resumida de lo que debe hacer el sistema deseado.
- Diseño del sistema: El sistema de destino se organiza en subsistemas basados tanto en la estructura del análisis como en la arquitectura propuesta.
- Diseño de objetos: El diseñador de objetos construye un modelo de diseño basándose en el modelo de análisis que lleven incorporados detalles de implementación.
- Implementación: Las clases objetos y las relaciones desarrolladas durante su diseño se traducen finalmente a un lenguaje de programación concreto, a una base de datos o a una implementación de hardware.

1.4.2 El lenguaje de programación java: Java fue diseñado en 1990 por James Gosling, de Sun Microsystems, como software para dispositivos electrónicos de consumo. Fue diseñado para dispositivos electrónicos como calculadoras, microondas y la televisión interactiva y se le conocía como OAK. Java como tal nace en 1994 con el navegador HotJava y seria utilizado para hacer dinámicas las páginas e Internet, para 1995 Sun Microsystems lanzaba el primer alpha del JDK 1.0.

Java es un lenguaje que posee muchas características que resultan muy útiles al momento de desarrollar cualquier software.

1.4.2.1 Características de Java: Las principales características que podemos destacar de Java son:

- Simple: Java ofrece toda la funcionalidad de un lenguaje potente, pero suprimiendo las características menos usadas y más confusas de éstos.
- Orientado a objetos: Java implementa la tecnología básica de objetos de C++ con algunas mejoras eliminando algunas cosas para mantener la simplicidad del lenguaje. Java trabaja sus datos como objetos e interfaces. Soporta las tres características propias del paradigma de la orientación a objetos: encapsulación, herencia y polimorfismo.
- Distribuido: Java se ha construido con extensas capacidades de interconexión TCP/IP.
- Robusto: Java realiza verificaciones en busca de problemas tanto en tiempo de compilación como en tiempo de ejecución. Comprobación de tipos, límites de arreglos, etc.
- Multiplataforma: Lo que le permite ser independiente al sistema operativo en el cual se utilice.
- Seguro: la seguridad en Java tiene dos facetas. En el lenguaje, características como los punteros o el casting implícito que hacen los compiladores de C y C++ se eliminan para prevenir el acceso ilegal a la memoria. El código Java pasa muchos tests antes de ejecutarse en una máquina.
- Multihilo: Java permite muchas actividades concurrentes en un programa utilizando hilos, con interfaces sencillas de implementar.

Pero Java también posee un gran limitante:

- La velocidad: Puesto que Java es un lenguaje interpretado, los programas desarrollados no tienden a ser muy rápidos, esto supone que se necesita una

máquina más rápida para alcanzar la misma velocidad que un programa compilado.

1.4.2.2 JDK: El **J**ava **D**evelopment **K**it es un conjunto de herramientas que permiten realizar programas en java. Este kit permite hacer cualquiera de las aplicaciones Java posibles, no tiene limitaciones en este sentido, sin embargo, no es lo más adecuado para desarrollar, ya que carece de un editor propio de código fuente, de un gestor de proyectos, de una herramienta de diseño visual de interfaces, etc.

1.4.2.3 NetBeans [NETBEANS, 2004]: NetBeans es un proyecto open source (software libre) de gran éxito con una gran base de usuarios, una comunidad en constante progresión, y con cerca de 100 sociedades (¡y contando!) en todo el mundo. Sun MicroSystems fundó el proyecto open source NetBeans en junio 2000 y continúa siendo el patrocinador principal de los proyectos.

El IDE NetBeans es un entorno de desarrollo, una herramienta para programadores para escribir, compilar, corregir errores y para ejecutar programas. Está escrito en Java pero puede servir de soporte a cualquier otro lenguaje de programación. Existe también un número enorme de módulos para extender el NetBeans IDE. El NetBeans IDE es un producto libre y gratuito sin restricciones de utilización.

1.4.3 Modelo Cliente-Servidor: Los sistemas Cliente/Servidor son aplicaciones que suelen ejecutarse en al menos dos sistemas distintos, uno que hace de cliente y otro que hace de servidor, aunque, es posible que ambos se encuentren en un mismo sistema. Generalmente un servidor proporciona servicios a varios sistemas clientes, pero puede haber un único cliente. [Spencer, 1997]

El modelo Cliente/Servidor implica que siempre que un usuario necesite recuperar o almacenar información, la parte cliente de la aplicación ejecuta una solicitud, que se envía, generalmente a través de una red, al servidor, y el servidor ejecuta la solicitud y devuelve la información al cliente. [Spencer, 1997]

Una base de datos no constituye un sistema Cliente/Servidor, aunque estos pueden utilizar una base de datos para realizar la actividad del servidor. [Spencer, 1997]

1.4.3.1 Descripción: Es un modelo de interacción entre dos procesos, que se ejecutan en forma simultánea. Este modelo es una comunicación basada en una serie de preguntas y respuestas en el cual un proceso puede proporcionar unos servicios a los restantes procesos del sistema.

Se destaca lo siguiente:

- El término servidor se aplica a cualquier programa que ofrece un servicio y que puede ser accedido a través de la red.
- Los servidores aceptan peticiones recibidas a través de la red, realizan el servicio y regresan el resultado al que generó la petición.

- Un cliente es un programa que envía una petición a un servidor y espera una respuesta.

1.4.3.2 Sockets: Los sockets son puntos finales de enlaces de comunicaciones entre procesos de diferentes máquinas en una red. Los sockets se manejan en los procesos de Java como descriptores de archivos, de forma que se pueden intercambiar datos con otros procesos escribiendo y leyendo a través de ellos.

1.4.3.3 Utilización de Sockets en el modelo cliente-servidor: Utilizando Sockets se pueden crear programas con el modelo cliente servidor, que puedan funcionar en una red en la cual el servidor estaría en una máquina y los clientes estarían distribuidos a través de la red.

1.4.4 Programación con hilos (THREADS): La programación con hilos es empleada en un modelo Cliente–Servidor, debido a la necesidad que se presenta en el servidor de soportar la concurrencia de múltiples clientes. El modelo de programación de hilos permite ignorar los detalles de la arquitectura, como el número de procesadores, para concentrarse en el algoritmo adecuado que exprese la concurrencia. Es pues un modelo adecuado para la programación concurrente. La utilización de la concurrencia en un lenguaje de programación requiere una cierta disciplina de programación junto a la utilización de nuevas técnicas.

Los hilos siempre pertenecen a un proceso, y no tienen existencia propia independiente. Cada hilo tiene un flujo de control, una pila y un estado propio. Ya que todos los recursos, a excepción de la CPU, son gestionados por el proceso, la comunicación entre hilos de un proceso es mucho más rápida y eficiente al compartir el mismo espacio de direcciones. Al igual que los procesos tradicionales, los hilos pueden estar en alguno de los siguientes estados:

- En ejecución (Running): Cuando el hilo posee la CPU y se encuentra activo.
- Bloqueado (Suspend): Cuando el hilo se encuentra esperando algún evento o a la espera de que otro libere el bloqueo por el que se encuentra detenido.
- Listo o preparado (Ready): Cuando el hilo está preparado para su ejecución, y se encuentra a la espera de ser elegido por el planificador.
- Terminado (Finished): Cuando el hilo ha finalizado, pero todavía no ha sido recogido por el hilo padre, aunque no puede ser planificado nunca más.

Un proceso puede tener uno o más hilos, donde estos comparten los recursos (memoria, archivos, etc.), y son la unidad de planificación. Así, un proceso será un objeto estático que posee un conjunto de recursos para una serie de hilos, que son los objetos dinámicos planificables.

1.4.4.1 Ventajas de los hilos: La principal ventaja de los hilos es el rendimiento. Normalmente el rendimiento es una cantidad percibida y no dada. Unas aplicaciones se beneficiarán de los hilos mientras que otras no.

- Recursos compartidos: Los hilos comparten la mayoría de los recursos en un proceso. Comparten el acceso a los ficheros, memoria compartida, y el espacio de direcciones virtual. El empleo de los hilos permite a los programadores conservar los recursos del sistema. El rendimiento puede beneficiarse también porque los recursos compartidos entre hilos necesitan menos gestión que los recursos compartidos entre procesos.
- Velocidad de las operaciones con hilos: Los hilos tienen un menor contexto que los procesos, así pues, las operaciones con hilos son generalmente más rápidas que las operaciones similares con procesos. En especial, la creación, finalización y cambio de contexto de un hilo, son operaciones más rápidas con los hilos que con los procesos.
- Tiempo de respuesta: Si es posible separar operaciones en un programa, los hilos se pueden emplear para mejorar los tiempos de respuesta de la aplicación, ya que cada operación que puede ejecutarse independientemente de las otras.

1.4.4.2 Aplicaciones de los hilos: Algunos programas presentan una estructura que puede hacerles especialmente adecuados para entornos multihilo. Normalmente estos casos involucran operaciones que pueden ser solapadas. La utilización de múltiples hilos, puede conseguir un grado de paralelismo que incremente el rendimiento de un programa, e incluso hacer más fácil la escritura de su código. Algunos ejemplos son:

- Utilización de los hilos para expresar algoritmos inherentemente paralelos. Se pueden obtener aumentos de rendimiento debido al hecho de que los hilos funcionan muy bien en sistemas multiprocesadores. Los hilos permiten expresar el paralelismo de alto nivel a través de un lenguaje de programación.
- Utilización de los hilos para solapar operaciones de E/S lentas con otras operaciones en un programa. Esto permite obtener un aumento del rendimiento, permitiendo bloquear a un simple hilo que espera a que se complete una operación de E/S, mientras otros hilos del proceso continúan su ejecución, evitando el bloqueo entero del proceso.
- Utilización de los hilos para solapar llamadas RPC salientes. Se obtiene una ganancia en el rendimiento permitiendo que un cliente pueda acceder a varios servidores al mismo tiempo en lugar de acceder a un servidor cada vez. Esto puede ser interesante para servidores especializados en los que los clientes pueden dividir una llamada RPC compleja en varias llamadas RPC concurrentes y más simples.
- Utilización de los hilos en interfaces de usuario. Se pueden obtener aumentos de rendimiento empleando un hilo para interactuar con un usuario, mientras se pasan las peticiones a otros hilos para su ejecución.
- Utilización de los hilos en servidores. Los servidores pueden utilizar las ventajas del multihilo, creando un hilo gestor diferente para cada petición entrante de un cliente.
- Utilización de los hilos en procesos pipeline: Se puede implementar cada etapa de una tubería o pipeline mediante un hilo separado dentro del mismo proceso.

- Utilización de los hilos en el diseño de un kernel multihilo de sistema operativo distribuido que distribuya diferentes tareas entre los hilos.
- Utilización de los hilos como soporte de aplicaciones de tiempo real acelerando los tiempos de respuesta para los eventos asíncronos a través de la gestión de señales.

1.4.4.3 Problemas potenciales de los hilos: Entre las principales causas de problemas con hilos se encuentran:

- Complejidad: La programación con hilos es más difícil que la clásica programación secuencial, por ejemplo, cuando se mantiene, depura u optimiza una aplicación con hilos, se deben tener en cuenta la existencia de múltiples hilos de código en ejecución, por ello generalmente es más fácil depurar y optimizar un programa de hilo simple.
- Sincronización: Se debe coordinar el acceso a los datos compartidos mediante bloqueos. Olvidar un bloqueo puede producir la corrupción de los datos.
- Depuración difícil: Debido a las dependencias entre los datos y las dependencias de tiempo. Depurar en un sistema multiprocesador es más complicado, más costoso, y los errores no siempre son reproducibles. Una biblioteca de hilos puede ser útil para detectar y analizar errores en un entorno uniprocesador antes de ser probada en un sistema multiprocesador. Existen dos tipos de errores, los *errores serie* que pueden ocurrir en un entorno uniprocesador, y los *errores paralelos* que son inherentes a una ejecución paralela, y son difíciles de detectar.
- Rompen la abstracción: lo que impide diseñar módulos independientes.

- Obtener un buen rendimiento es difícil: Los bloqueos demasiado simples (grano grueso) disminuyen el nivel de concurrencia, mientras que los bloqueos de grano fino son demasiado complicados. Además la velocidad de cambio de contexto del sistema operativo influye notablemente en el rendimiento.
- Soporte escaso: El código es difícil de transportar a otras arquitecturas que no soportan hilos. Las librerías estándar no suelen ser hilo-seguras. Las llamadas al kernel o al sistema de gestión de ventanas pueden no estar preparadas para soportar múltiples hilos.
- Disponibilidad de herramientas: Para favorecer el desarrollo de aplicaciones multihilo, la industria necesitará crear herramientas de depuración y optimización más refinadas. Sin embargo, la tecnología de los depuradores y optimizadores es relativamente joven lo que a corto plazo supone un problema para los programadores. Un entorno de programación multihilo debe proporcionar depuradores con soporte multihilo. La información debe ser extraída del TCB (bloque de control del hilo) y hacerla disponible al usuario. Los cambios de contexto deberían ser visibles para el usuario. Por ejemplo, cuando se va a producir un cambio de contexto, el usuario podría elegir si continúa depurando después del punto de suspensión o si cambia al contexto de otro hilo. Además se podrían presentar ventanas de depuración separadas para cada hilo dentro de un proceso.
- Utilización de código no seguro: Se debe tener mucho cuidado a la hora de mezclar código multihilo con código no hilo-seguro, es decir, código no preparado para ejecución en un entorno multihilo.

1.4.5 El motor de base de datos MySQL: MySQL es un sistema de gestión de bases de datos relacional. Posee un diseño multihilo que le permite soportar una gran carga de forma muy eficiente. MySQL fue creada por la empresa sueca MySQL AB.

Este gestor de bases de datos es, probablemente, el gestor más usado en el mundo del software libre, debido a su gran rapidez y facilidad de uso. Esta gran aceptación es debida, en parte, a que existen infinidad de librerías y otras herramientas que permiten su uso a través de gran cantidad de lenguajes de programación, además de su fácil instalación y configuración.

MySQL es muy rápido, utiliza pocos recursos del sistema en comparación con los demás motores de bases de datos, Tiene un buen manejo de usuarios y de la seguridad. Una característica muy importante es que dispone de APIs para varios de los lenguajes de programación más utilizados, como C++, PHP o Java, esto permite una mayor flexibilidad al escoger el que sea de mayor utilidad al momento de realizar un programa.

1.4.6 Modelo de Entidad-Relación: [HAWRYSZKIEWYCZ,1994] El modelo entidad-relación (E-R) lo introdujo P.P. Chen en 1976 y se ha aceptado mucho en el diseño de base de datos. Ayuda al analista con tres conceptos semánticos principales: entidades, relación y atributos.

Los diseñadores que usan el modelo E-R en el análisis de datos deben describir la empresa en términos de:

- Entidades, que son objetos distintos en un entorno.
- Relaciones, que son interacciones significativas entre objetos.
- Atributos que describen y las relacionan. Cada atributo se asocia con un conjunto de valor (dominio) y puede adquirir un valor de este conjunto.

2. DISEÑO DEL AGENTE INTELIGENTE (CHATBOT)

2.1 DISEÑO DE LA GRAMÁTICA DEL AGENTE

Una vez realizado el estudio de la lengua castellana se procedió a restringir o limitar la gramática para su fácil adaptación a una estructura de base de datos, evitando llegar al extremo en donde se perdiera la naturalidad del lenguaje; por esta razón, el lenguaje que utiliza el agente inteligente para comunicarse con los seres humanos es sólo un subconjunto del idioma castellano.

La gramática abreviada obtenida contiene un conjunto de tipos de palabras con sus respectivas propiedades morfológicas y semánticas, y un conjunto de reglas gramaticales que admiten oraciones simples afirmativas, negativas e interrogativas.

2.1.1 Tipos de palabras: Teniendo en cuenta la clasificación de las palabras de la lengua castellana y los requerimientos del lenguaje del agente, se consideran tipos de palabras los siguientes:

Sustantivos: Palabras que se refieren de forma general a objetos, animales, personas, lugares, etc., es decir, son las palabras que representan a todo aquello de lo que se puede expresar algo. Ej: casa, país, arañas, universidad, etc.

Nombres: Nombres propios de personas, animales, lugares, etc. También se incluyen en este tipo de palabras, los pronombres personales. Ej: Colombia, Amazonas, etc.

Adjetivos: Palabras que se refieren a cualidades de los sustantivos. Ej: bellas, grandes, colombianos, etc.

Adverbios: Palabras que afectan o detallan alguna característica específica de los verbos, adjetivos u otros adverbios. Ej: mucho, bien, sencillamente, etc.

Determinantes: En esta categoría se incluyen los artículos, los pronombres (excepto los personales) y los adjetivos determinantes. Ej: La, unos, los, etc.

Preposiciones: Son palabras que carecen de significado por sí mismas pero que junto con otras se utilizan para complementar al sujeto o al predicado de la oración. Ej: a, para, con, de, etc.

Verbos: Son las palabras que se refieren a acciones. Ej: comer, estudias, vivimos, salieron, etc.

Verbos Copulativos: Son los verbos que permiten relaciones directas entre el sujeto y el predicado de una oración. Ej: ser, estar.

Verbos de Acción: Son los verbos que se utilizan para describir otro conjunto de verbos o acciones. Ej: hacer.

Interrogativos: Son los términos que se utilizan para hacer preguntas. Ej: Qué, cómo, dónde, entre otros. Según su utilización se han clasificado en 2 tipos:

- Tipo A: Los que pueden ser seguidos por verbos o por otros tipos de palabras antes del verbo principal. Ej: Cuántos, Qué, etc.
- Tipo B: Los que sólo pueden ser seguidos por verbos. Ej: Cómo, Cuándo, etc.

Negativos: En esta categoría se incluyen los adverbios negativos. Ej: No, nunca, etc.

Saludos: Son los términos utilizados para iniciar o dar fin a una conversación. Ej: Hola, chao, etc.

2.1.2 Oraciones: En la gramática del agente son aceptadas los siguientes tipos de oraciones:

- Las oraciones bimembres que sean explicitas, es decir, oraciones con sujeto no tácito (sintagma nominal) y con predicado (sintagma verbal).
- Las oraciones unimembres conformadas por una palabra de tipo saludo.
- Las oraciones interrogativas que inician con algún término interrogativo (palabra interrogativa o preposición seguida de palabra interrogativa) y finalizan con un sintagma verbal.

Los sintagmas de las oraciones bimembres e interrogativas pueden estar subdivididos en otros sintagmas y/o complementos.

2.1.2.1 Sintagmas: En términos generales los sintagmas se pueden definir como una o más palabras dispuestas de manera que en conjunto expresan un significado global y forman una unidad de función y de significado, que se integran en una oración o en un texto, por este motivo, las oraciones han sido divididas en sintagmas para facilitar el análisis y organizar mejor las ideas en la base de conocimientos del agente.

Dentro de la categoría de sintagmas en la gramática del agente se consideran, el núcleo del sujeto, los modificadores del sujeto, los modificadores del predicado, sustantivos, adverbios, adjetivos y los nombres.

El modificador directo del sujeto, unido con el núcleo del sujeto, son llamados “sintagma nominal”, al igual que los objetos directos y predicativos subjetivos del predicado que inician con determinante. Los objetos directos del predicado que inician con adjetivo son llamados “sintagma adjetivo”. El modificador indirecto del sujeto es considerado “sintagma preposicional nominal”, al igual que los objetos indirectos del predicado, y los circunstanciales (de núcleo nominal) y predicativos subjetivos que inician con preposición. Un modificador circunstancial compuesto por preposición seguida de verbo es considerado “sintagma preposicional verbal”.

2.1.2.2 Complementos: Son agrupaciones de dos o más sintagmas y son empleados durante el proceso de análisis de oraciones.

2.1.3 Reglas gramaticales: En la teoría de los compiladores se señala que las Gramáticas Libres de Contextos están compuestas por producciones que a su vez están conformadas por símbolos terminales y no terminales combinados entre sí, con los cuales se puede evaluar si una cadena pertenece a la gramática o no. De esta misma forma, la gramática del agente utiliza reglas gramaticales (producciones) basadas en combinaciones entre pares de tipos de palabras, de sintagmas o de complementos, con las cuales es posible conocer si una oración recibida es una afirmación o una pregunta.

Cada uno de los diferentes tipos de palabras forma parte de los símbolos terminales de las reglas gramaticales.

Tabla 1. Símbolos terminales en la gramática del agente

Tipo de palabra	**Símbolo Terminal**
Sustantivos	SUST
Nombres	NOM
Adjetivos	ADJ
Adverbios	ADV
Determinantes	DET
Verbos Copulativos	VERBOC
Verbos de Acción	VERBOH
Preposiciones	PREP
Interrogativos	INTA
Interrogativos (seguidos sólo de verbos)	INTB
Negativos	NEGACION
Saludos	SALUDO

Las múltiples combinaciones de los terminales producen símbolos no terminales y en algunos casos otros terminales. Los símbolos no terminales combinados entre sí (e incluso con terminales), pueden también producir otros símbolos no terminales. Los símbolos no terminales producidos por todas las combinaciones mencionadas son los siguientes:

Tabla 2. Símbolos no terminales en la gramática del agente

Sintagma o complemento	**Símbolo no Terminal**
Sintagma nominal	SN
Sintagma adjetivo	SADJ
Sintagma verbal	SV
Sintagma preposicional nominal	SPN
Sintagma preposicional verbal	SPV
Complemento sintagma preposicional	CSP
Complemento nominal-nominal	CNN
Complemento ad-nominal	CAN
Complemento nominal-SPV	CNPV
Complemento ad-SPV	CAPV
Interrogativo acción	INTC

Los sintagmas nominales inician con un determinante y terminan con un sustantivo, nombre o adjetivo. En una oración bimembre el sujeto (sintagma nominal) puede ser un SN sencillo o estar acompañado de otros sintagmas y/o complementos.

Los sintagmas adjetivos son los que inician con un adjetivo y terminan con un sustantivo.

Los sintagmas verbales inician con el verbo principal de la oración y pueden estar seguidos por cualquier otro sintagma. Estos sintagmas son el predicado de la oración.

El sintagma preposicional nominal inicia con una preposición y tiene como núcleo un sustantivo o nombre. El sintagma preposicional verbal inicia con una preposición y termina con un verbo (diferente del principal). El complemento sintagma preposicional inicia y termina con un sintagma preposicional (nominal o verbal).

El complemento nominal-nominal agrupa dos sintagmas cuyos núcleos son sustantivos o nombres. El complemento ad-nominal agrupa un adjetivo o un adverbio con un sintagma nominal o complemento nominal-nominal o complemento ad-nominal.

El complemento nominal-SPV agrupa un sintagma o complemento que inicie con núcleo sustantivo con un sintagma o complemento que termine con sintagma preposicional verbal. El complemento ad-SPV agrupa un sintagma o complemento que inicie con adverbio o adjetivo y termine con un sintagma preposicional verbal.

El Interrogativo acción es utilizado para indicar que la oración corresponde a una oración interrogativa de acción.

Los posibles estados de aceptación que se pueden alcanzar por medio de todas las combinaciones de los símbolos terminales y no terminales dentro de la gramática del agente, son (Véase Tabla 3):

Tabla 3. Estados de aceptación en la gramática del agente

Estado de aceptación	**Símbolo**
Oración simple afirmativa o negativa	AFIRMACIÓN
Oración interrogativa	PREGUNTA
Oración interrogativa de acción	PREGUNTAACCION
Saludo	SALUDO

2.1.4 Estructura de la gramática en la base de datos: La gramática en la base de datos está compuesta por siete tablas en las cuales se almacenan las palabras según su tipo, propiedades, relaciones con otras palabras y las reglas gramaticales. Una de estas tablas es la tabla de los tipos de palabras, donde se guarda cada tipo con su nombre y el conjunto de propiedades que pueden tener las palabras clasificadas en él. La tabla de las propiedades guarda el nombre de cada propiedad, el grupo al que pertenece y el conjunto de propiedades que son incompatibles con ella. La tabla de las palabras almacena cada palabra y un vínculo a la tabla de las propiedades de la palabra en la cual se guarda el tipo de la palabra y el conjunto de sus propiedades. La tabla de los verbos y sus infinitivos, guarda como su nombre lo indica, la relación entre los verbos conjugados y sus respectivos verbos en infinitivo. La tabla de los contextos, también hace parte de la gramática, ya que en ella se guarda la posible relación entre dos o más palabra, ya sea de sinonimia, antonimia, plural-singular, primera persona–segunda persona, etc. Por último se encuentra la tabla de las reglas

gramaticales, en la cual se guardan todas las posibles producciones sintáctico-semánticas del lenguaje del agente. (Véase Anexo D).

2.2 DISEÑO DE LA BASE DE CONOCIMIENTO DEL AGENTE

Una conversación es un proceso que tiene como finalidad el intercambio o flujo de información entre emisor y receptor, por lo cual, el contenido de los mensajes transmitidos durante este proceso se convierten en parte esencial de su desarrollo. Debido a esto, se hizo necesario el diseño de una estructura adecuada para albergar toda la información que pudiera ser objeto de conversación entre el agente y los usuarios, es decir se hizo necesario el diseño de una base de conocimientos que pudiera ser utilizada por el agente al igual que una persona se vale de todo lo que sabe o conoce para hablar sobre un tema determinado. (Véase Anexo E).

2.2.1 Ideas: La base de conocimientos diseñada para el agente está compuesta principalmente por las ideas, las cuales se encuentran representadas en un conjunto de tablas de la base de datos, con el lógico propósito de conservarlas permanentemente. Las ideas son básicamente oraciones simples (Véase Figura 3) que son recibidas, luego procesadas a través de un análisis sintáctico-semántico en el cual son divididas entre el verbo principal y un conjunto de sintagmas, con el fin de organizar mejor su contenido, y por último almacenadas en las tablas mencionadas anteriormente.

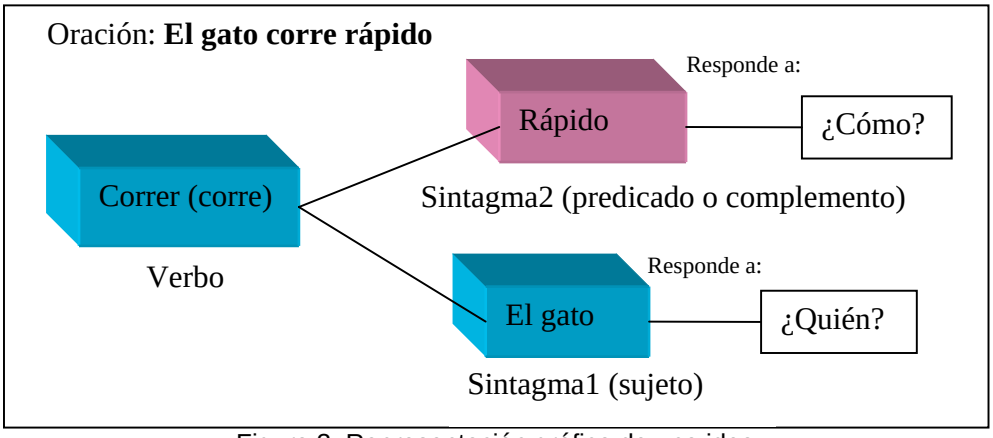


Figura 3. Representación gráfica de una idea

2.2.2 Estructura de las ideas en la base de datos: La estructura de las ideas en la base de datos consta básicamente de dos tablas, una en la cual se almacena el verbo principal de la oración “ideaverbo”, y otra donde se almacenan cada uno de los sintagmas que forman parte de la oración “ideapalabra”.

El verbo es núcleo del predicado de una oración y por ende se convierte en el núcleo de la idea, ya que él indica la acción que se realiza y en torno a él giran los demás componentes de la oración, razón por la que la tabla donde se almacena, llamada “ideaverbo”, es la tabla central de las ideas. Esta tabla, además, tiene un campo que indica si la idea corresponde a una acción afirmativa o negativa.

Cada sintagma de las oraciones es almacenado en un registro de la tabla “ideapalabra” como un conjunto de contextos de las palabras que hacen parte de él, es decir, que no se guardan los índices de las palabras sino los índices de los contextos a los que estos pertenecen, con el fin de poder tener el adecuado contexto de cada palabra según el sentido que quiera expresar la idea. Junto con

cada sintagma, se guarda también, el conjunto de palabras interrogativas a las cuales es posible que responda cuando se haga una pregunta relacionada con el verbo y los demás sintagmas de la idea, es decir que las ideas tienen prácticamente definidas las respuestas a las preguntas que se hacen relacionadas con ellas. Además, se almacena también, un indicador de si el sintagma hacía parte del sujeto o del predicado de la oración que la generó, esto con el fin de tener una mejor organización del contenido de las respuestas.

2.3 ANÁLISIS DE ORACIONES

El agente al momento de recibir una oración, busca en ella siglas, contracciones o abreviaturas y las separa. La oración que se obtiene es separada en las palabras que la conforman, luego cada una de estas se busca en la tabla “palabra” mediante varios ciclos en los cuales los términos formados por varias palabras se unen para formar una sola palabra, en esta búsqueda también se obtiene el tipo y las propiedades de las palabras guardadas en la tabla “propiedadpalabra”. Ej:

Se introduce la oración “La UTB es una universidad”.

- Se separa UTB, Universidad Tecnológica de Bolívar.
- Se descompone la oración en palabras y se determina cuales son los términos formados por varias palabras “Universidad Tecnológica de Bolívar”.
- Se busca en las tablas “palabra” y “propiedadpalabra” el tipo y las propiedades de cada palabra.

La	Universidad Tecnológica de Bolívar	es	una	Universidad
DET	SUST	VERBOC	DET	SUST

Figura 4. Oración descompuesta en tipos de palabras

Para el análisis de la oración se procede a tomar grupos de dos palabras y verificar si la agrupación de sus tipos produce algún resultado en las reglas almacenadas, hasta completar toda la oración. Este procedimiento se repite recursivamente hasta que se obtiene un sólo resultado, el cual puede ser:

- **SALUDO**: Son los términos utilizados para iniciar o dar fin a una conversación
- **AFIRMACIÓN**: Son las oraciones simples, las cuales pueden ser afirmativas o negativas. Son utilizadas para formar las ideas de la base de conocimiento cuando el usuario es administrador o para hacer preguntas cuyas respuestas son “Sí”, “No” o “no sé”, cuando el usuario es un cliente normal.
- **PREGUNTA**: Son las oraciones interrogativas que tendrán como respuesta alguno de los sintagmas almacenados en la idea
- **PREGUNTAACCION**: Son las oraciones interrogativas que tendrán como respuesta una acción, es decir, un verbo.

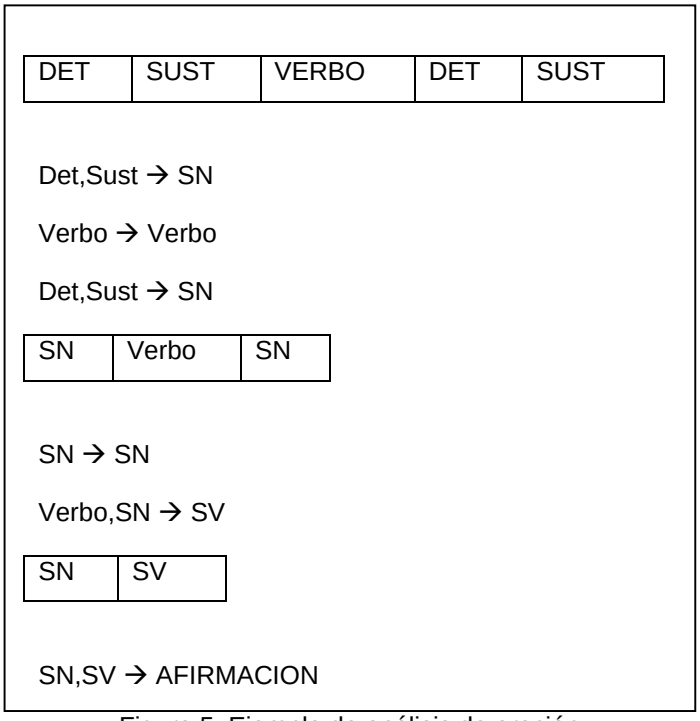


Figura 5. Ejemplo de análisis de oración

Durante el análisis de la oración se crea un árbol binario (Véase Figura 6) que contiene las combinaciones y los resultados que se dieron en cada uno de los pasos del análisis. En este árbol el resultado es el nodo padre y los elementos que conformaron la combinación son los hijos. Si la combinación es de un sólo elemento el nodo hijo pasa a ser el nodo padre.

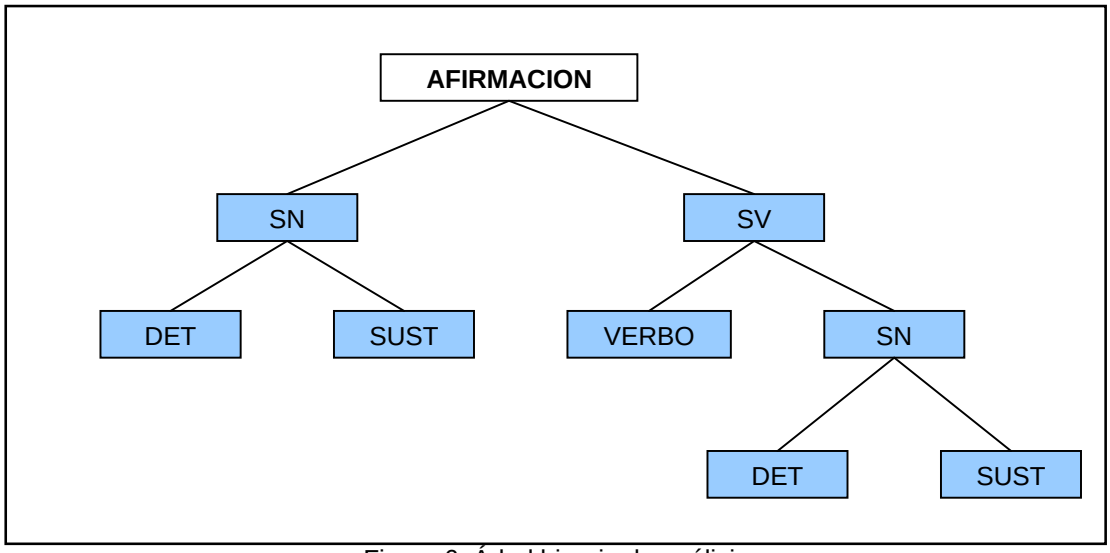


Figura 6. Árbol binario de análisis

El árbol se recorre en preorden para obtener las partes con mayor relevancia de la oración como el verbo principal, sintagmas o complementos, a estos últimos se les agrega además la pregunta a la cual responden según lo almacenado en la tabla “pregunta”. Luego estas partes serán utilizadas para almacenar la idea, generar una respuesta o saludar, dependiendo si el resultado obtenido en el análisis fue afirmación, pregunta o saludo respectivamente.

2.3.1 Almacenamiento de ideas: Las ideas son almacenadas luego de ser recibidas en forma de frases con sentido coherente, es decir que sólo las oraciones cuya estructura gramatical sea aprobada luego del análisis gramatical serán consideradas aptas para formar parte de la base de conocimiento del agente. Cabe anotar que se ha tomado como medida preventiva que este proceso sólo pueda ser llevado a cabo por usuarios denominados administradores, los cuales se encargarían del mantenimiento general del agente. Esto indica que

además de revisar la estructura gramatical de la oración recibida, también se revisa si el usuario con quien se conversa es un administrador.

Antes de iniciar el procedimiento de almacenamiento se lleva a cabo otro proceso de revisión con el fin de conocer si la oración recibida está contenida o es igual a alguna de las ideas ya almacenadas.

El proceso que almacena las ideas recibe una oración dividida en los sintagmas del sujeto, sintagmas del predicado y el verbo principal. Debido a que el verbo es el núcleo de las ideas, es lo primero que se almacena, pero no como el verbo en sí, sino como la relación infinitivo - conjugado, por lo que se hace necesario ubicar esa relación dentro de la tabla “verboinfinitivo”, y una vez ubicada, se verifica el indicador de si la oración es afirmativa o negativa y se procede a insertar un nuevo registro en la tabla “ideaverbo”, guardando así, el núcleo de la nueva idea.

Una vez se ha almacenado el verbo de la idea, se procede con cada uno de los sintagmas del sujeto y luego con los del predicado. El primer paso con un sintagma es descomponerlo, ya que es recibido como un conjunto de palabras y sus respectivos interrogantes. El conjunto de palabras es lógicamente la secuencia ordenada de cada una de las palabras que forman el sintagma y los interrogantes son las palabras interrogativas para las cuales el sintagma es respuesta. Por cada palabra del sintagma se hace una búsqueda en la tabla de contextos para obtener los contextos (sinónimos) en los que se encuentra la palabra y en caso de no encontrar uno, se agrega. Finalmente, luego de tener los

contextos de cada palabra, se ejecuta un proceso en el cual se efectúan diferentes combinaciones de los contextos sin perder el orden de cada palabra en el sintagma, y se inserta un nuevo registro en la tabla de los sintagmas "ideapalabra", con la mejor combinación y los respectivos interrogantes de cada sintagma, relacionándolos a su vez con el núcleo de la idea en "ideaverbo".

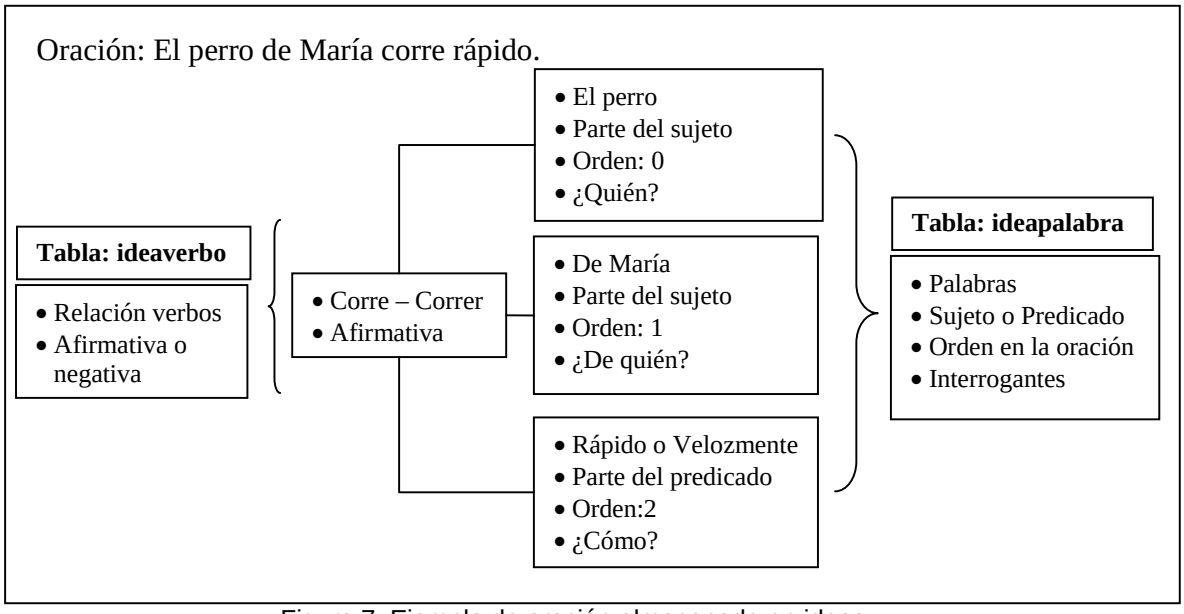


Figura 7. Ejemplo de oración almacenada en ideas.

2.3.2 Generación de respuestas: Luego de reconocer que la oración recibida es una pregunta, se procede a buscar una respuesta dentro del conjunto de ideas almacenadas en la base de conocimientos.

El primer paso en la búsqueda de una respuesta, es precisamente buscar dentro de las ideas todas aquellas que tengan como verbo principal al verbo utilizado en la pregunta o a una conjugación relacionada con él, y que dentro de sus sintagmas se encuentre alguno que responda a la palabra interrogativa (si está en plural se

busca también en como si fuera singular), es decir, inmediatamente se busca si hay respuesta dentro de todas las ideas relacionadas al verbo sin analizar aún los sintagmas complementarios de la pregunta quienes al final se encargarán de restringir el conjunto de ideas correctas.

El siguiente paso, luego de tener un conjunto de ideas como posibles respuestas, es analizar si ellas se refieren a lo mismo que se refieren los demás sintagmas de la pregunta, es decir si dentro de sus sintagmas están todos los sintagmas de la pregunta. Este proceso se inicia obteniendo los contextos de cada palabra de los sintagmas que hacen parte de la pregunta, con el fin de poder comparar con los sintagmas de las ideas, debido a que en estas se almacenan contextos y no palabras. Con base a la comparación de los sintagmas, se concluye si la idea seguirá siendo parte del conjunto de respuestas o no. Si la idea responde a la pregunta se inicia el proceso de obtención de los otros sintagmas que hacen parte de la idea, luego de esto, se analiza si la idea es apta para ser mostrada como respuesta, teniendo en cuenta la conversación que se ha mantenido y el contenido de las otras respuestas. Por último, se revisan todas las respuestas que han quedado y se elige en todas ellas, cada palabra que va a ser mostrada dependiendo de la variedad que haya en su respectivo contexto.

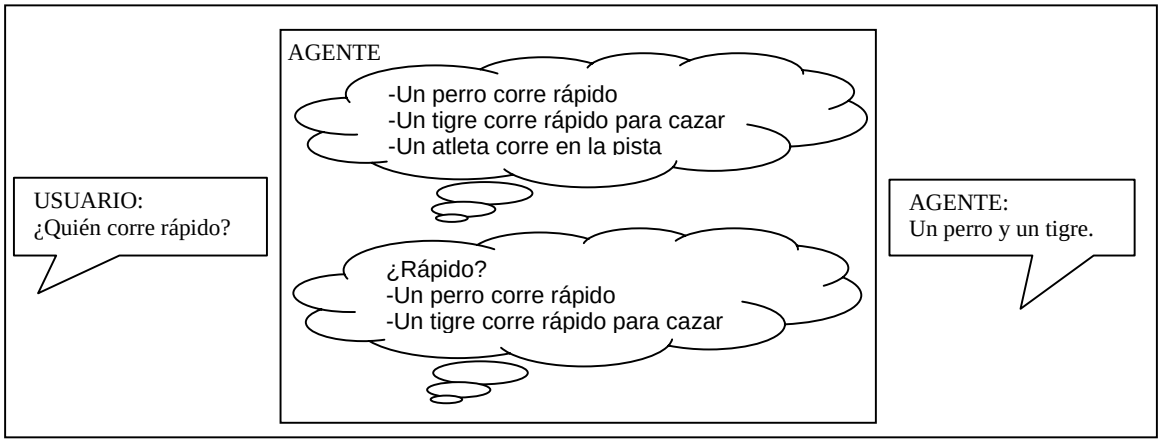


Figura 8. Esquema de respuesta a pregunta.

Si la pregunta que se obtuvo debe tener como respuesta una acción, debido a que el análisis de la oración produjo “PreguntaAcción”, se buscan las ideas que tengan los sintagmas que correspondan a los ingresados, se obtienen los verbos de dichas ideas y se envían como respuesta.

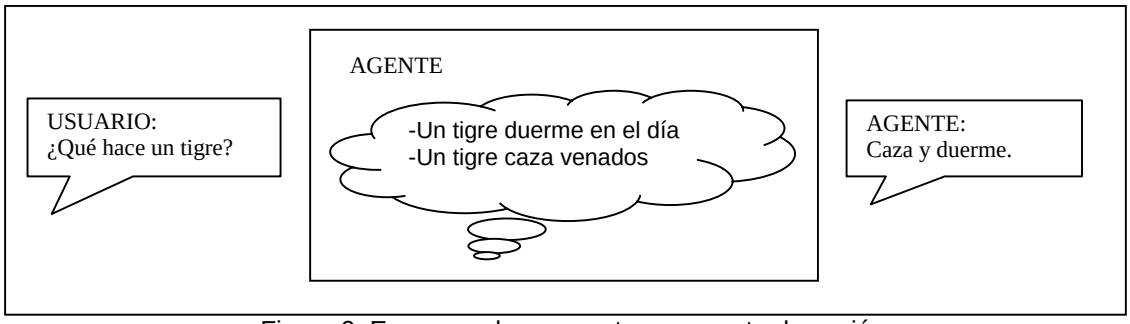


Figura 9. Esquema de respuesta a pregunta de acción.

Existe otro tipo de preguntas que pueden formular los usuarios del agente, las cuales son de acuerdo a su estructura gramatical, oraciones simples afirmativas o negativas, pero que en el lenguaje oral son oraciones interrogativas. Esta clase de preguntas generalmente conllevan a una respuesta de tipo “Sí”, “No” o “No sé”. El agente en estos casos examina dentro de sus ideas buscando la idea

correspondiente a la oración y si la encuentra responde “Sí”, si la encuentra con signo opuesto responde “No” y si no la encuentra responde “No sé”.

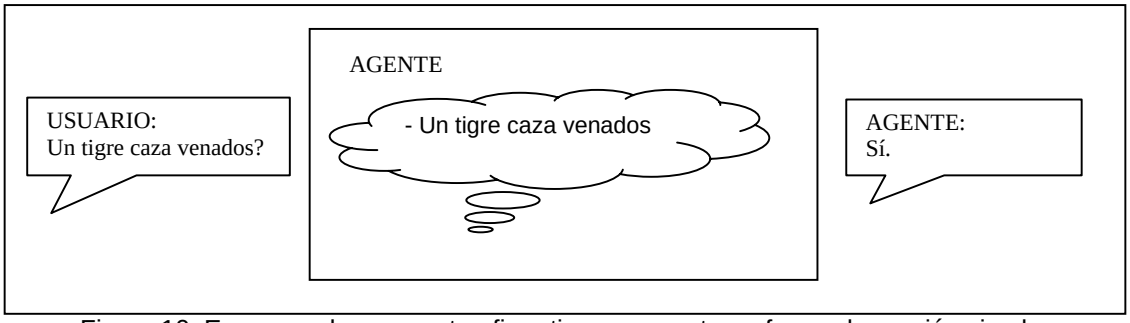


Figura 10. Esquema de respuesta afirmativa a pregunta en forma de oración simple.

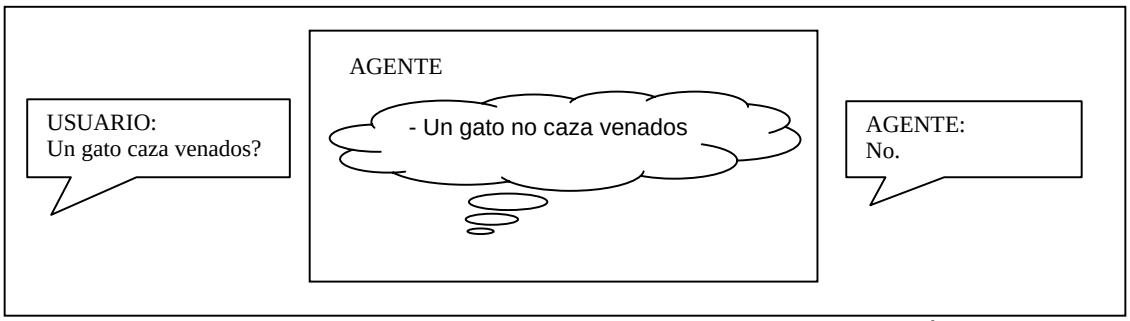


Figura 11. Esquema de respuesta negativa a pregunta en forma de oración simple.

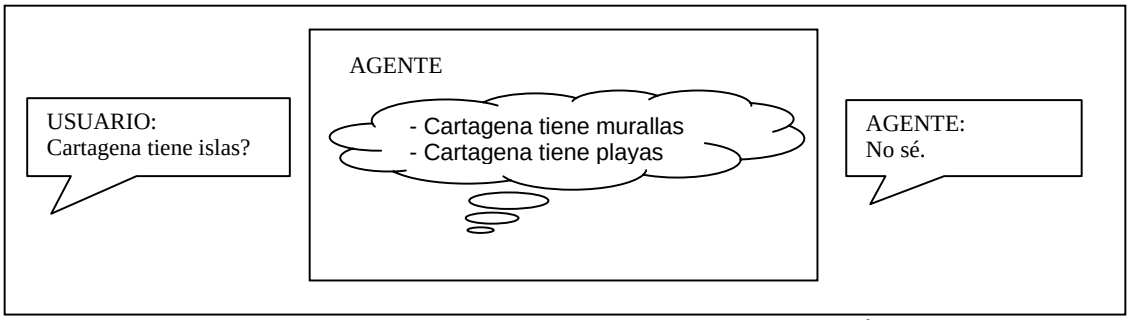


Figura 12. Esquema de respuesta a preguntas en forma de oración afirmativa.

2.4 CONTROL DE IDEAS

2.4.1 Creación y relación de ideas por medio de silogismos deductivos: El programa puede crear nuevas ideas partiendo de las relaciones entre proposiciones es decir el agente escapas de realizar silogismos entre ideas. Para esto el agente hace los siguientes pasos:

1. El agente busca en su base de conocimiento las ideas que estén relacionadas con el verbo ser en infinitivo. Ejemplo:

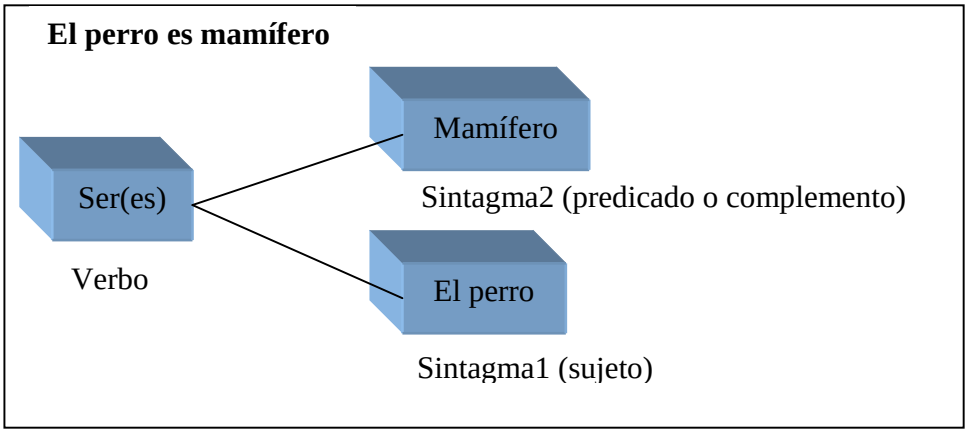


Figura 13. Idea relacionada con el verbo ser.

2. Después de tener el resultado, el agente consulta que ideas tienen relación con el Sintagma2 de esa idea.

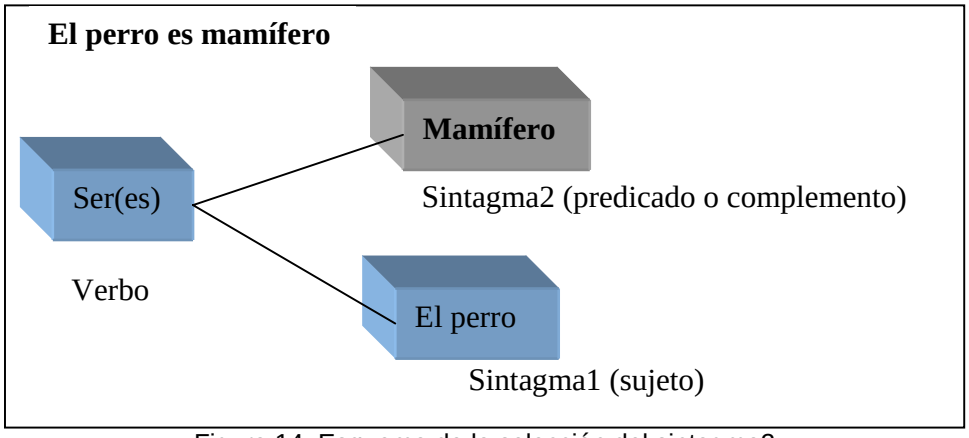


Figura 14. Esquema de la selección del sintagma2

3. De encontrar una o más ideas relacionadas al sintagma2, el agente toma esas ideas y las clona con el fin de cederlas en herencia al Sintagma1. Ejemplo:

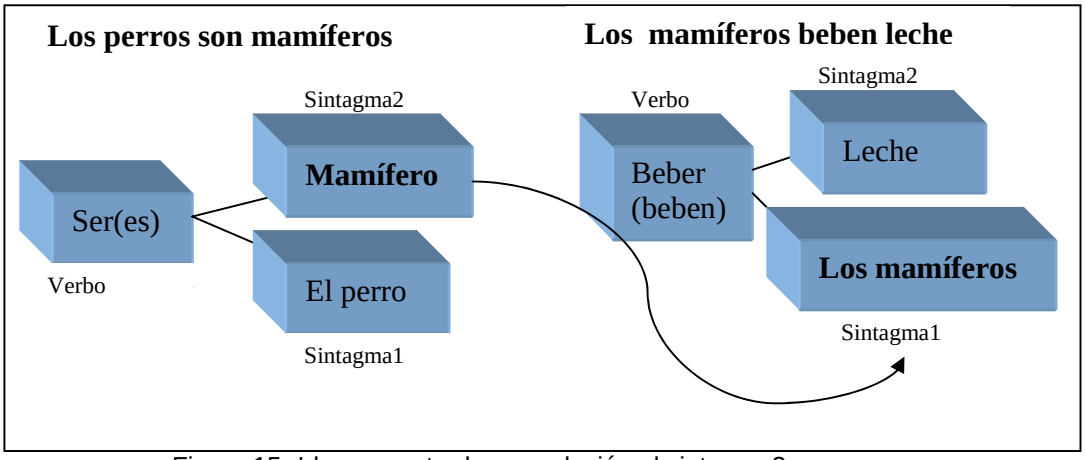


Figura 15. Idea encontrada con relación al sintagma2.

El proceso de clonación consiste en copiar las ideas encontradas y entrarlas como nuevas ideas, donde el sujeto o el sintagma1 de esas ideas son reemplazados por el sujeto o sintagma1 de la idea anterior es decir la idea que originó la búsqueda.

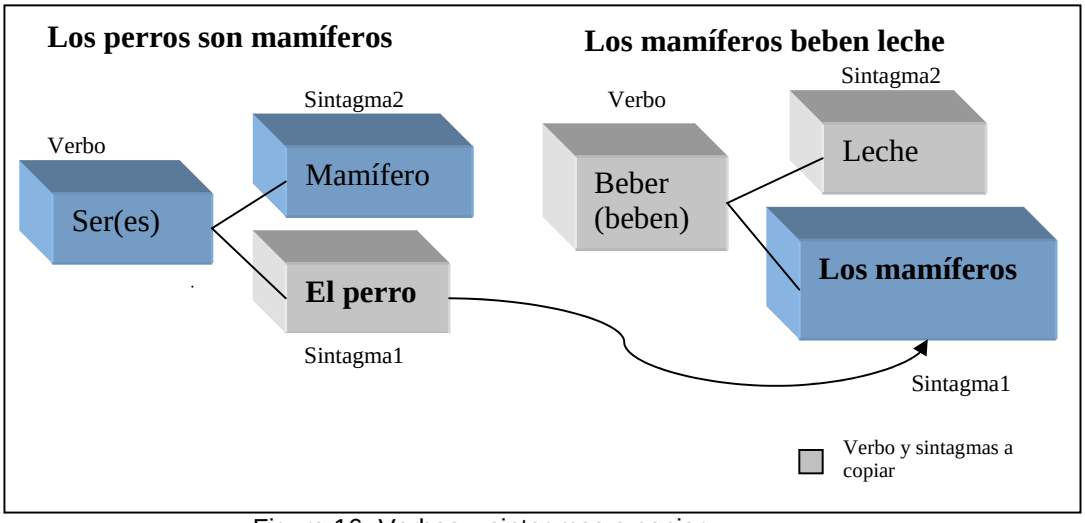


Figura 16. Verbos y sintagmas a copiar.

4. En este paso el agente crea las nuevas ideas y las agrega en su base de conocimiento.

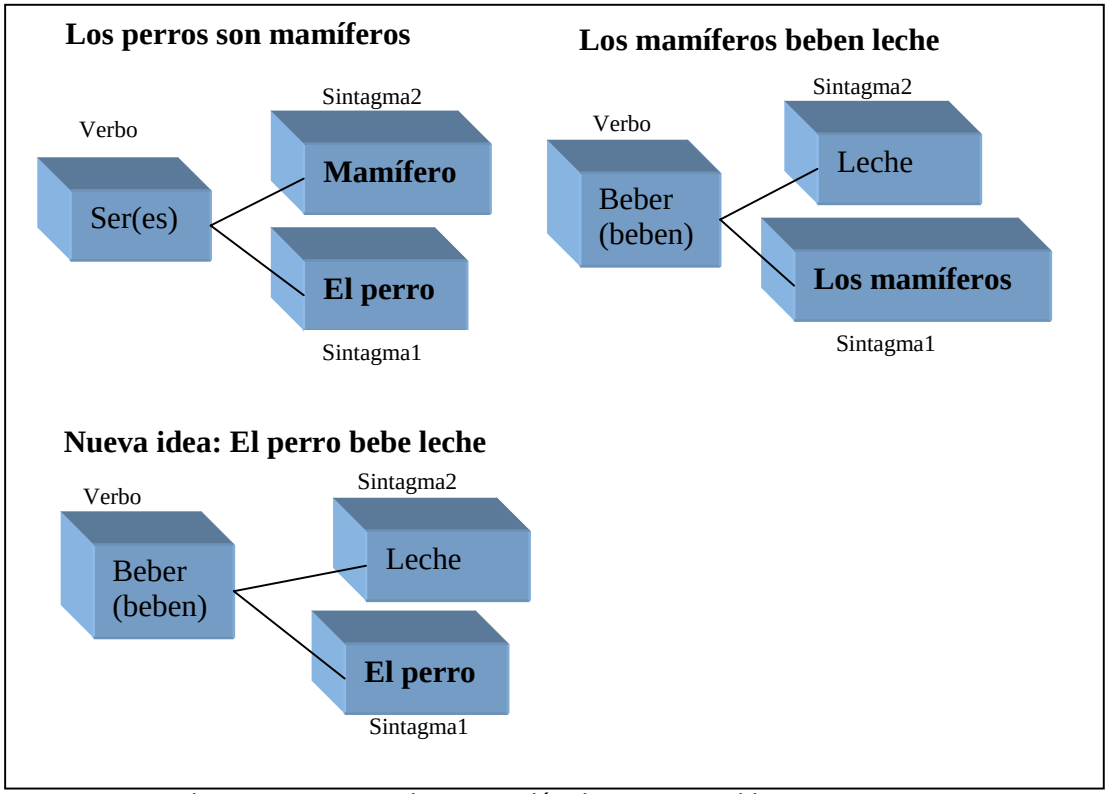


Figura 17. Proceso de generación de una nueva idea.

2.4.2 Proceso a nivel de base de datos

Paso 1. Se escogen las ideas formadas con el verbo ser en infinitivo. El agente realiza una consulta a la base de datos seleccionando los índices de la tabla "ideaverbo" donde "idve_idverboinfinitivo" sea igual al índice de la tabla "verboinfinitivo" donde "vein_idpalabrainfinitivo" sea igual al índice del verbo ser de la tabla palabras. Para una mayor rapidez en la consulta el índice del verbo ser está almacenado en la tabla "variables", por lo cual el agente compara "vein_idpalabrainfinitivo" con el registro "verboser" de dicha tabla.

Paso 2. El agente busca en la tabla "ideapalabra" los registros relacionados con el índice de la tabla "ideaverbo. Una vez en los registros el programa almacena los datos de los campos en variables en memoria. De estos datos el programa toma el de "idpa_contextos" cuando "idpa_sujetopredicado" indica que es predicado, y realiza una consulta a la base de datos donde selecciona los campos de las ideas que tengan este valor como sujeto en la tabla "ideacontexto".

Paso 3. De encontrase una idea atribuida a este sujeto se copian los datos de los campos de las tablas que conforman esta ideas.

Paso 4. Los datos seleccionados ingresan nuevamente en las tablas como idea nueva, pero en el campo "idpa_contexto" de la tabla "idea_palabra" no se copiará el sujeto (predicado) sino el sujeto de la idea del verbo "ser" en infinitivo.

3. PROTOTIPO

3.1 COMPONENTES DEL PROTOTIPO

El objetivo general de implementar un agente inteligente con la capacidad de conversar puede verse materializado a través de un prototipo creado con ciertas capacidades para aprender y responder preguntas.

El prototipo consta de dos aplicaciones, una basada en el modelo cliente-servidor con la cual se establecerá la interacción entre los usuarios y el agente, y la otra es una aplicación administrativa o de configuración del agente.

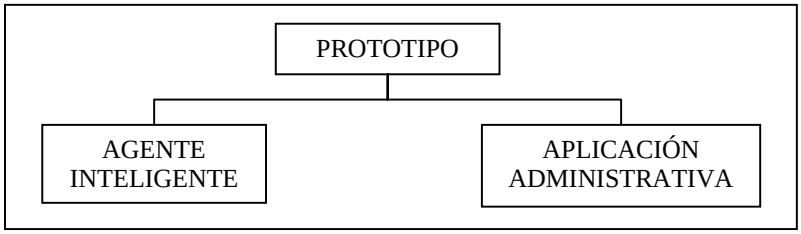


Figura 18. Esquema del prototipo

El agente inteligente es la aplicación que se encarga de establecer y controlar las conversaciones con los usuarios, basándose en el diseño del agente explicado en el capítulo anterior. Este agente tiene como finalidad responder preguntas.

La aplicación administrativa se encarga de ofrecer al usuario administrador del agente, la opción de visualizar y controlar de manera más fácil la información almacenada en la base de datos.

3.2 AGENTE INTELIGENTE (CHATBOT)

3.2.1 Descripción: La aplicación del agente se basa en un modelo de tres capas (Clientes – Servidor Agente – Bases de datos).

Los clientes interactuarán con el servidor mediante un applet desarrollado en JDK 1.4.2.05, al cual podrán acceder mediante una página web desde cualquier computador que tenga instalado el JRE 1.4.2 o superior, con un navegador compatible con MS Internet Explorer o Netscape.

El applet posee una interfaz sencilla parecida a un Chat, su función es conectarse al servidor, recibir las preguntas de los clientes y enviarlas. Este se descarga al computador cliente y se ejecuta en él, para luego establecer una conexión directa con el servidor por medio de sockets.

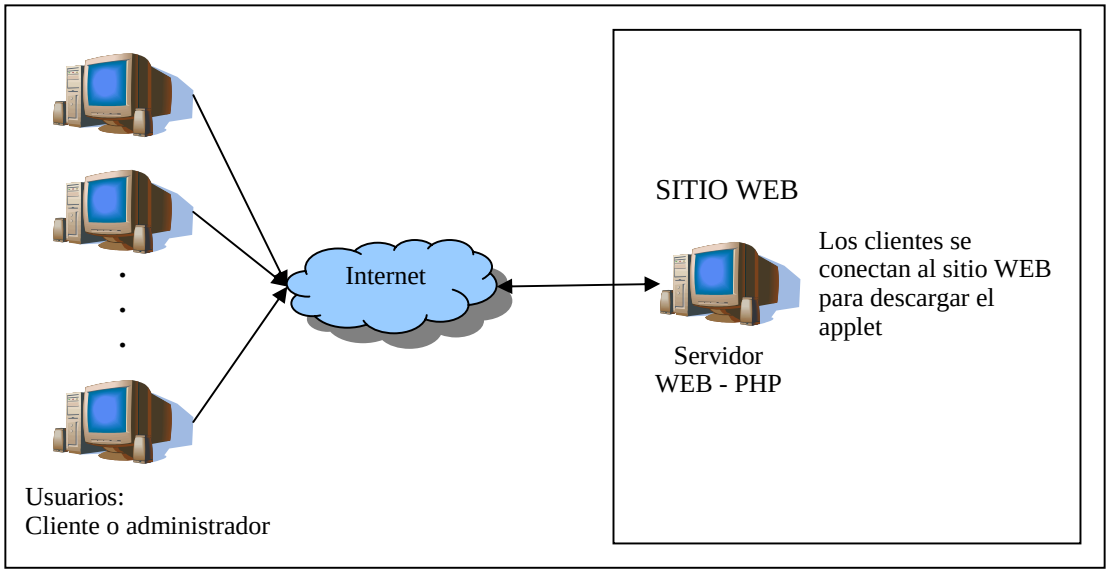


Figura 19. Paso 1 - Clientes acceden al sitio WEB

En el servidor existen dos puertos en espera de conexión de clientes uno para los clientes administradores y otro para los clientes normales, estos últimos no tienen permitido enseñarle nada al agente. Los puertos deben ser elegidos por el administrador del servidor y el que será utilizado para clientes administradores deberá ser restringido por motivos de seguridad a sólo unos pocos equipos destinados para ese propósito.

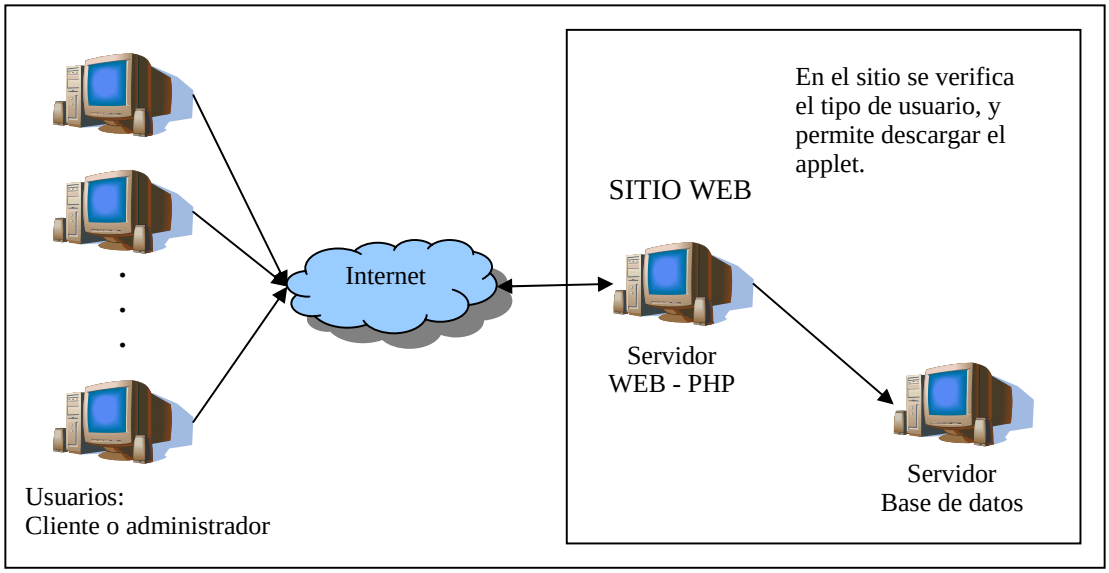


Figura 20. Paso 2 - Seguridad en el Sitio WEB

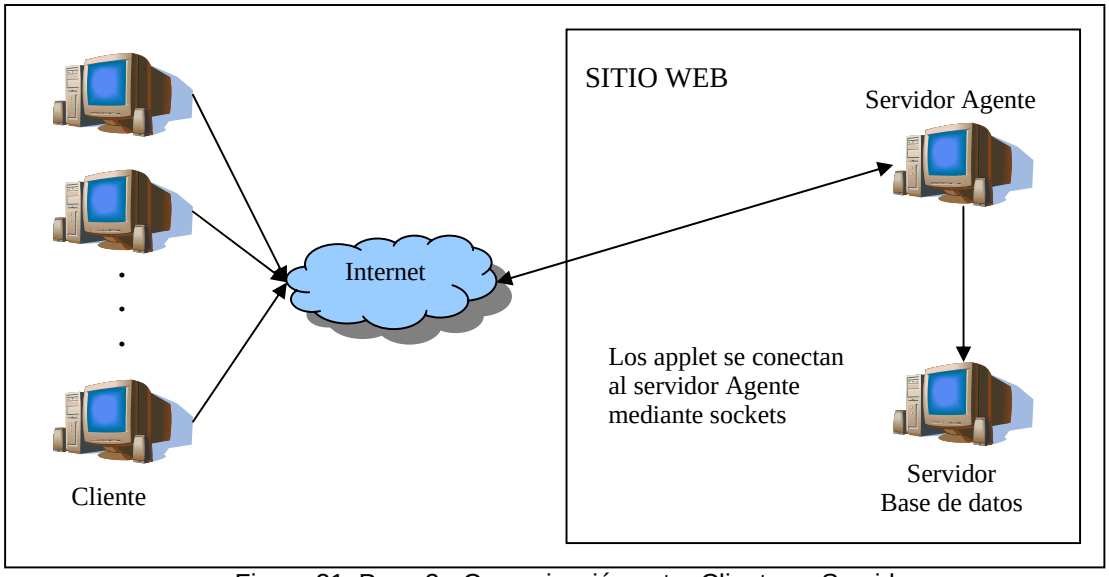


Figura 21. Paso 3 - Comunicación entre Clientes y Servidor

Cada cliente que se conecta al servidor se convierte en un hilo, dejando así, cabida a la conexión de más clientes, sin importar si son administradores o clientes normales.

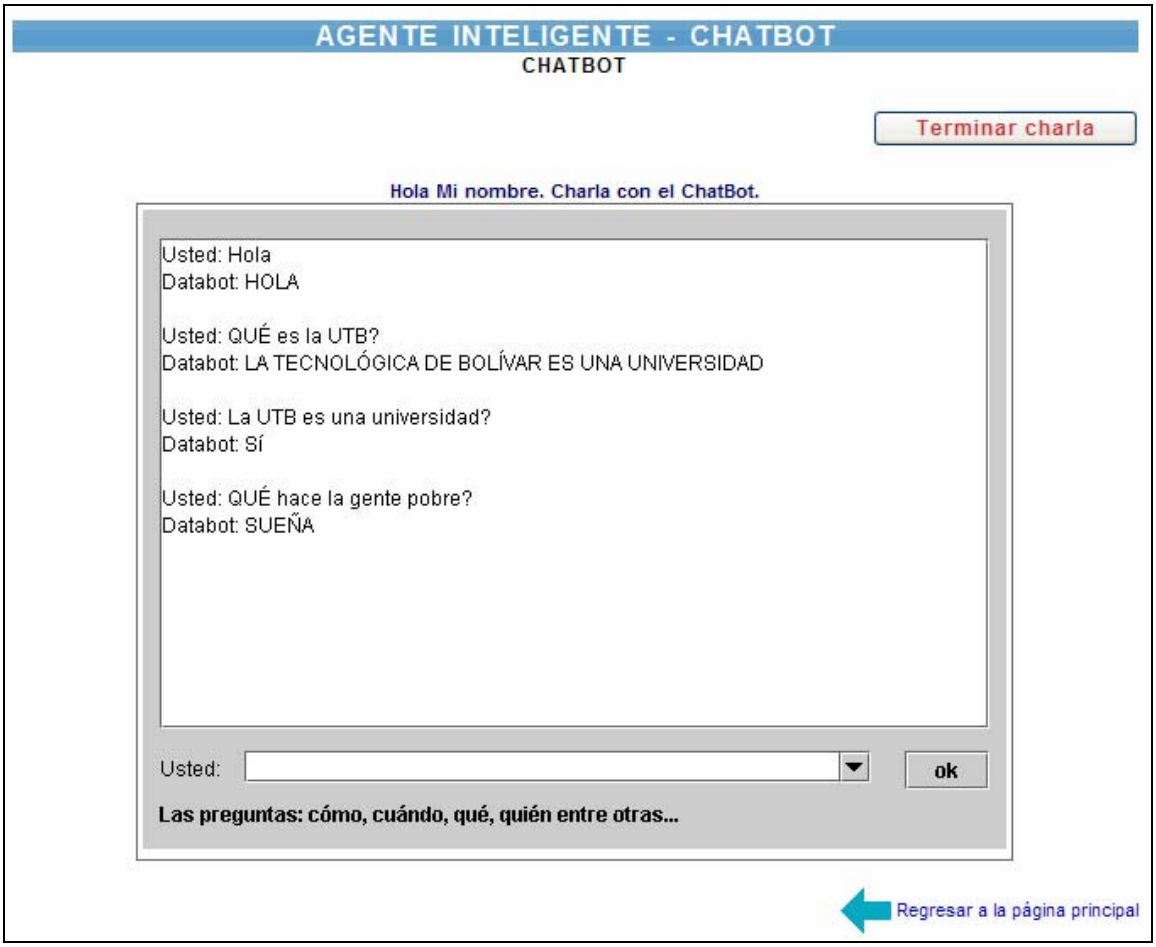


Figura 22. Interfaz cliente normal (applet)

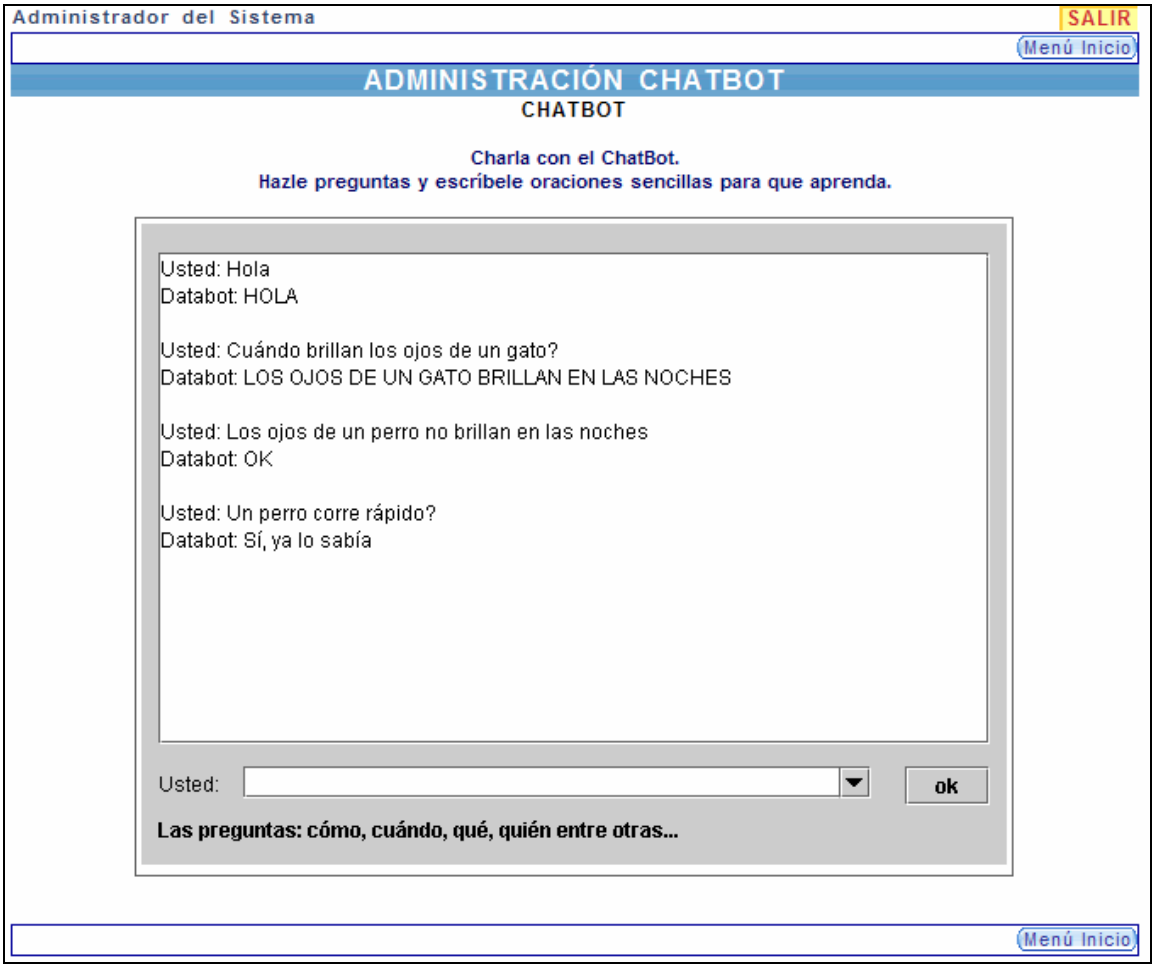


Figura 23. Interfaz cliente administrador (applet)

3.2.2 Archivos: Las aplicaciones constan de los siguientes archivos:

- servidor.class: Debe ser ejecutado en el computador servidor. Se encarga de recibir las oraciones que envían los clientes y darles respuesta. Dentro del servidor cada cliente es un hilo independiente. Este programa será transparente a los usuarios.
- chatbot.class: Es la clase que contiene los métodos de análisis del agente relacionados con análisis léxico-sintáctico, almacenamiento de ideas y a la generación de respuestas.

- dbase.class: Es la clase que se encarga de ejecutar las diferentes operaciones sobre la base de datos del agente, como consultas o modificaciones a las tablas.
- str.class: Contiene métodos para el manejo de cadenas utilizados en el análisis sintáctico.
- Sintaxis.class: Contiene la estructura del árbol sintáctico.
- webcbot.class: Es el cliente en forma de applet diseñado para conectarse al servidor por medio de sockets. Su función es enviar al servidor las consultas hechas por el usuario y mostrar las respuestas que este le envía. Este es la parte del programa que utilizaran los usuarios finales.

3.3 APLICACIÓN ADMINISTRATIVA

3.3.1 Descripción: La aplicación de administración o configuración del agente, desarrollada con el lenguaje de programación PHP, fue diseñada con el propósito de que un usuario administrador controlara toda la información del agente, tanto de la gramática como de su base de conocimientos. En esta aplicación, el usuario también tiene la posibilidad de conversar con el agente, al igual que los demás clientes, pero además, puede agregarle ideas utilizando oraciones simples. En el anexo H se encuentra el manual de usuario de esta aplicación.

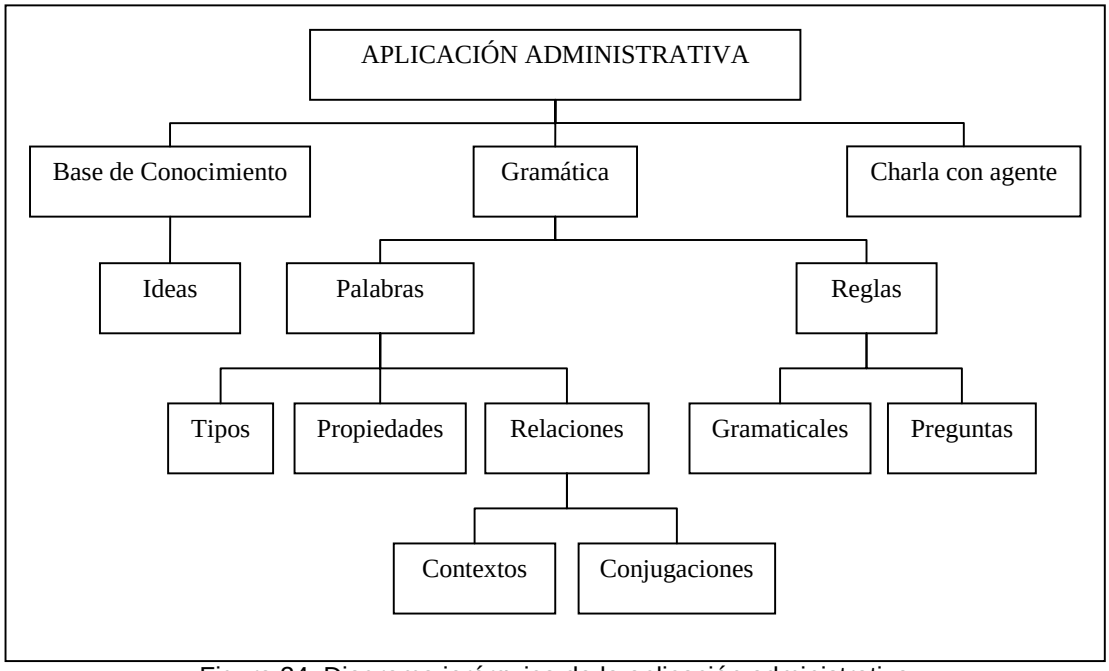


Figura 24. Diagrama jerárquico de la aplicación administrativa

3.3.2 Módulos: La aplicación está compuesta por tres módulos que son:

3.3.2.1 Base de conocimiento: Este módulo permite al usuario visualizar todas las ideas que hacen parte de la base de conocimiento del agente y además eliminar las que se consideren innecesarias o incorrectas.

3.3.2.2 Gramática: Este módulo, como su nombre lo indica, comprende todo lo referente a la gramática del agente y está dividido en dos submódulos, llamados, palabras y reglas, respectivamente.

- Palabras: Este submódulo permite la visualización, inserción, edición y eliminación de los diferentes tipos de palabras, propiedades, palabras del vocabulario del agente y relaciones entre palabras. Las relaciones entre

palabras pueden ser, de conjugación, en la que cada verbo en infinitivo es relacionado con sus diferentes conjugaciones en tiempo, modo, persona, número y voz; o de contexto, en la que se relacionan las palabras sinónimas, antónimas, de géneros opuestos (masculino-femenino), de número opuesto (singular-plural) y de persona opuesta (primera-segunda).

- Reglas: Este submódulo permite la visualización, inserción, edición y eliminación de las diferentes reglas gramaticales y las reglas de preguntas. Las reglas gramaticales, como se ha mencionado anteriormente, son utilizadas durante el análisis de las oraciones y las reglas de preguntas son las que indican cual es el respectivo interrogante de cada sintagma de una oración, de acuerdo a sus propiedades.

3.3.2.3 Charla con agente: Este módulo permite ejecutar la aplicación del agente, con el propósito principal de enseñarle nuevas ideas por medio de oraciones simples, afirmativas o negativas, y de igual forma que con los clientes, se puede utilizar para que responda preguntas acerca de los conocimientos que ya tiene.

3.4 IMPLEMENTACIÓN DEL PROTOTIPO

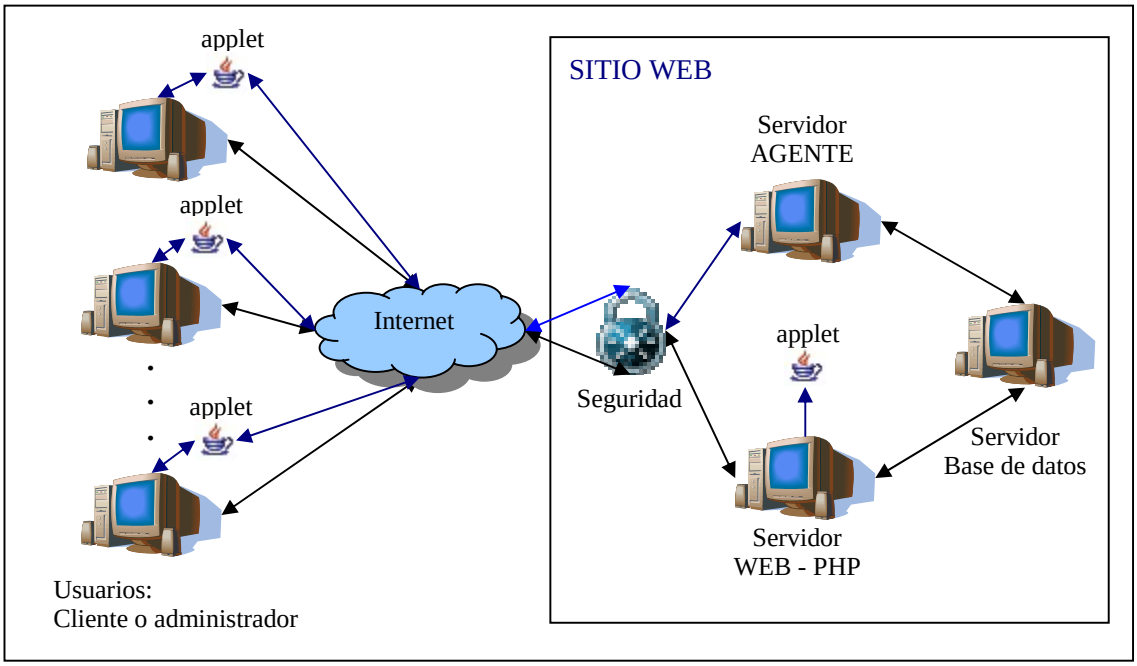


Figura 25. Esquema del funcionamiento del prototipo

Para implementar este prototipo se necesita:

Servidor AGENTE: Debe ser un computador servidor que tenga lo siguiente:

- JRE 1.4.2 o superior.
- Driver JDBC que provee MySQL LABS, para la conexión a la base de datos de MySQL (MySQLConnector).
- IP válida en Internet o en la Intranet, dependiendo del uso que se vaya a dar.
- Disponer de memoria y capacidad de procesamiento suficientes para recibir la concurrencia de muchos usuarios.

Servidor WEB-PHP: Debe ser un computador servidor que tenga lo siguiente:

- Un servidor web, tal como Apache HTTP Server o Microsoft Internet Information Service (IIS).
- PHP 4.3.0 o mayor.
- IP válida en Internet o en la Intranet, dependiendo del uso que se vaya a dar.

Servidor Base de datos: Debe ser un computador servidor que tenga lo siguiente:

- Motor de base de datos MySQL 4.0 o superior.
- Debe estar conectado con los equipos donde se instalarán las otras aplicaciones, ya sea a través de Internet o Intranet, y debe permitir el acceso a esos equipos a través del puerto de MySql (predeterminado: 3306).

PCs clientes: Deben ser computadores que tengan lo siguiente:

- JRE 1.4.2 o superior.

Es posible utilizar un mismo servidor para albergar todo el sistema, pero no es necesario ni muy recomendable, ya que pueden trabajar perfectamente por separado, evitando así la sobrecarga de un solo equipo.

La aplicación administrativa debe ser instalada en el servidor WEB-PHP. La instalación consiste en ubicar los archivos de la aplicación dentro de la carpeta que ha sido configurada en el servidor web como el directorio donde estarán las páginas y demás archivos accesibles en la red, en el caso de Apache, dentro de la carpeta asignada como “DocumentRoot”, la cual es por defecto “htdocs”.

Las herramientas utilizadas están disponibles sin costo alguno en Internet, por lo que pueden ser descargadas y utilizadas por cualquier persona. Los manuales de instalación y funcionamiento de estas herramientas están disponibles en Internet y no se detallan en este documento debido a que se asume que la persona encargada de implementar este prototipo en un sitio web debe tener un grado de conocimiento considerable en cuanto a estos temas se refieren.

El prototipo fue probado satisfactoriamente utilizando como servidor AGENTE, un PC AMD ATHLON XP2400+ de 512MB RAM con JDK1.5.0, como servidor de base de datos, un PC AMD ATHLON XP2000+ 512MB RAM con MySql 4.0.21, y como servidor WEB-PHP, un PC portátil AMD ATHLON XP-M de 248MB RAM con Apache http server 2 y PHP 4.3.4.4, conectados entre sí en una Intranet. Estos tres equipos funcionaban a la vez como clientes.

3.5 RECOMENDACIONES PARA EL MEJORAMIENTO DEL PROTOTIPO

Entre los posibles ajustes que se pueden hacer para mejorar el prototipo se encuentran:

- Permitir al usuario la utilización del lenguaje oral para la realización de consultas. Esto sería posible partiendo de las herramientas ya existentes e intentando integrarlas en el interfaz del usuario.
- Aumentar las capacidades de autoaprendizaje.
- Hacer más flexible y robusta la gramática.

- Agregar más tipos de oraciones, y que sea capaz de hacer otras operaciones según la oración ingresada.
- Aumentar su capacidad de conversar amenamente (hacerlo más amigable).

4. CAMPOS DE APLICACIÓN

4.1 ÁREAS DE INTERÉS

El servicio de atención en línea se ha convertido en una herramienta fundamental para muchas de las empresas y entidades modernas, las cuales en su mayoría tienen la necesidad de ofrecer asistencia técnica de manera personalizada a sus clientes con el fin de brindarles un respaldo o garantía sobre el uso o mantenimiento de sus productos y servicios. En la actualidad un alto porcentaje de estas empresas han aprovechado las ventajas de las nuevas tecnologías para implementar una herramienta de soporte técnico en Internet, convirtiéndola en un valor agregado para los productos y servicios que comercializan habitualmente.

Las empresas que aún no cuentan con soporte en línea son desplazadas, en el ámbito comercial, por aquellas que ya lo han implementado, debido a que para muchos clientes el soporte técnico como garantía a corto, mediano o largo plazo se convierte en un requisito fundamental al momento de adquirir un producto o servicio, además Internet es considerado como un medio fácil y rápido para consultar sobre cualquier aspecto relacionado con los productos o servicios recibidos.

El soporte en línea que se presta actualmente requiere de un personal especializado con alto conocimiento de los bienes y servicios ofrecidos, y además en algunos casos, cuando la demanda de atención es muy alta o los clientes están ubicados en países con horarios diferentes, se requiere de una disponibilidad de tiempo completo. Todo lo anterior implica una generación de costos tanto por cantidad y calidad del personal como por el número de horas de trabajo, ocasionando que muchas empresas limiten el soporte en línea a un horario fijo, con lo que se disminuye la satisfacción de los clientes.

Un problema similar al del soporte técnico también afecta a las universidades modernas que han implementado un sistema de aprendizaje virtual, donde los estudiantes reciben el material didáctico de sus clases a través de Internet y consultan a sus profesores con el fin de aclarar dudas al respecto, pero la atención de los docentes no siempre es inmediata y las respuestas a las inquietudes tardan más de lo esperado, causando que los estudiantes se desanimen con esta modalidad de educación y sientan la necesidad de ir personalmente donde el profesor, desobedeciendo así uno los principios de la educación virtual.

La atención en línea no se limita sólo al soporte técnico que se puede prestar luego de la adquisición de productos y servicios, sino que puede ser utilizada como un medio de publicidad para atraer clientes, debido a que cuando un usuario de Internet ingresa al sitio Web de una empresa, es común que por diferentes razones, entre las que se puede mencionar el mal diseño de la página, la desorganización de la información publicada y la inexperiencia del navegante, no

encuentre la información precisa de lo que está buscando, incluso después de revisar casi todos los enlaces que se hallan en la página, sintiéndose desubicado y desalentado por su búsqueda infructuosa, por lo que decide salir de allí y visitar otro sitio, lo cual significa la pérdida constante de potenciales clientes por una deficiencia en la atención ofrecida. Un ejemplo de esto podría ser el caso de un bachiller que desea buscar información detallada de la carrera que quiere estudiar, en las páginas Web de varias universidades del país, pero al ingresar a la primera de ellas se encuentra con que la información publicada no es suficiente y que para llegar a ella debió antes hacer click sobre un sinnúmero de enlaces; lógicamente el joven se va a sentir desatendido y es probable que excluya a esa universidad de su futuro educativo.

La Universidad Tecnológica de Bolívar cuenta con un sitio Web en el cual se debe mantener toda la información concerniente a las facultades, programas, estudiantes y procesos educativos que se llevan a cabo en la universidad, además cuenta con el Sistema de Aprendizaje Virtual InteractivO (SAVIO), en el cual se imparten algunas materias del pensum académico de las diferentes carreras que tiene la universidad. SAVIO posee un centro de atención a los estudiantes, al cual estos pueden enviar sus inquietudes, pero como es obvio estas inquietudes no siempre son resueltas de manera inmediata y a veces una respuesta puede tardar horas o días por lo que los estudiantes pueden dejar de realizar alguna tarea urgente o simplemente sentirse insatisfechos con dicho servicio.

4.2 JUSTIFICACIÓN

Un alto porcentaje de empresas tienen páginas de Internet donde ofrecen sus productos o servicios, dichas páginas generalmente tienen una sección donde las personas pueden hacer preguntas acerca de cualquier inquietud que tengan, pero la respuesta puede llegar muy tarde o no llegar. El agente inteligente será capaz de responder preguntas inmediatamente según la información contenida en su base de conocimientos, ofreciendo a la empresa un beneficio con relación a la atención a los usuarios, lo que seria atractivo para nuevos clientes.

En un sistema de aprendizaje virtual como SAVIO se hace necesaria una herramienta que se pueda utilizar para responder las inquietudes más frecuentes de los estudiantes, ya que generalmente las personas que se encargan de esto no se encuentran disponibles en todo momento del día.

Estos pasos en el campo de la inteligencia artificial darán la oportunidad de formar grupos de investigación en distintas áreas como lo son el procesamiento del lenguaje natural y las redes neuronales.

4.3 VENTAJAS

La implementación de este agente inteligente en la ayuda en línea o en un sistema de aprendizaje virtual como SAVIO, permitirá a las personas resolver sus inquietudes acerca de algún tema o actividad de manera inmediata, reduciendo el

número de solicitudes o de preguntas de los alumnos de manera presencial, manteniendo la virtualidad.

El uso de este agente inteligente en el sitio Web de una empresa hará que los visitantes puedan obtener información de una manera rápida e interactiva acerca de un tema específico de la entidad, logrando así un mayor impacto en el público, lo cual se verá reflejado en un mayor número de clientes o usuarios, es decir, el agente se convertirá en un medio para atraer consumidores de los bienes o servicios que ofrece la empresa.

El agente, además de dar información al instante, permitirá que las empresas que no puedan mantener un soporte o una ayuda en línea las 24 horas lo puedan hacer, mejorando así, la atención al cliente y logrando confianza por parte de los usuarios de sus productos o servicios.

CONCLUSIONES

El procesamiento del lenguaje natural ha evolucionado considerablemente durante los últimos años, llegando a ser una parte importante en las investigaciones de muchas entidades y universidades del mundo. Los programas mas representativos de esta rama de la IA son los Chatbot.

El objeto de este proyecto fue desarrollar un agente con capacidad de aprendizaje junto con una metodología que describiera la forma de realizar proyectos de esta naturaleza. La metodología desarrollada describe como hacer un agente inteligente que utilice lenguaje natural para comunicarse con seres humanos, basado en un modelo de ideas que utilizan un verbo como núcleo y varios sintagmas asociados a este.

Para el desarrollo de la gramática utilizada en la comprensión y generación del lenguaje natural, se utilizó una derivación de la gramática castellana, enfocada a oraciones simples y trasformada en reglas de producción. Los tipos de palabras y las propiedades de estas también fueron estudiadas y modificadas para lograr un óptimo resultado en el análisis de las oraciones.

Las ideas pueden ser consultadas mediante preguntas realizadas al agente y este, además, tiene la capacidad de aprenderlas es decir de almacenar nuevas ideas mediante la conversación cuando recibe afirmaciones. El agente también puede generar sus propias ideas utilizando silogismos.

El desarrollo de software enfocado a PLN está delimitado en gran manera por el idioma en el que se trabaje, tanto por la complejidad de este como por las herramientas disponibles que trabajen en él.

RECOMENDACIONES

- Uno de los principales objetivos de este proyecto fue servir de fundamento en el estudio del Procesamiento del Lenguaje Natural y los chatbot, por lo que la principal recomendación que se puede hacer es que se continúe trabajando en este tema y en las demás ramas de la Inteligencia Artificial, ya sea ampliando las fronteras de este proyecto o realizando otros.

- Crear un grupo de investigación para el desarrollo del agente con la comunidad universitaria, donde estén vinculados profesores y alumnos.

- Estudiar el comportamiento del agente, recopilar errores y buscar soluciones.

- Desarrollar en grupos de trabajos mejoras y actualizaciones a la base de conocimientos para adaptarlas a distintos ambientes o temas.

- Que la comunidad de investigadores desarrolle prototipos con base en algoritmos de redes neuronales y algoritmos genéticos.

- Invitar a otras disciplinas a que se vinculen al proyecto. La comunidad de la facultad de psicología puede ofrecer aportes claves para la representación de la lógica y el conocimiento.

- El agente asocia las ideas por medio de silogismos deductivos, se puede investigar para crear un modelo donde la relación de las ideas sea más profunda.

BIBLIOGRAFÍA

AMETAY DE SÁNCHEZ, Margarita. Desarrollo de habilidades del pensamiento: razonamiento verbal y solución de problemas: guía del instructor. Trillas, México, 1992.

Biblioteca de Consulta Microsoft Encarta 2004.

DE LA VEGA, Manuel. Introducción a la psicología cognitiva. Alianza Editorial S.A. Madrid, 1998.

DEPARTAMENTO. DE CIENCIAS DE LA COMPUTACIÓN E INTELIGENCIA ARTIFICIAL. Universidad de Sevilla. España. 2004. http://www.cs.us.es/

ENCICLOPEDIA AUTODIDACTA OCÉANO COLOR, Volumen 1, LINGÜÍSTICA, ED. OCÉANO.

GRUPO DE INVESTIGACIÓN EN TECNOLOGÍAS DEL HABLA. Universidad de Las Américas. México. 2003. http://www.udlap.mx/~sistemas/tlatoa/

PIATTINI Velthius, Mario Gerardo. Concepción y diseño de bases de datos. ADDISON-WESLEY IBEROAMERICANA.

RICH, Elaine y KNIGHT, Kevin. Inteligencia Artificial, 2 ed. McGraw Hill. 703 p.

RUMBAUH, James et al. Modelado y diseño orientados a objetos Metodología OMT. Madrid, Prentice Hall, 1996

VAN DALEN, Deobold B. y MEYER, William J. Manual de técnica de la investigación educacional, Editorial Paidos, Barcelona, 1981.

PÁGINAS CONSULTADAS

http://www.escolar.com/menule.htm
Escolar

http://es.wikipedia.org/wiki/Categoría_gramatical
Wikipedia – Categoría gramatical

http://pisuerga.inf.ubu.es/lsi/Docencia/TFC/ITIG/icruzadn/Memoria/index.htm
Robot Recuperador Web - SpiderBot 1.0
Memoria del Proyecto Fin de Carrera

http://www.aditel.org/~dpecos/docs/mysql_postgres/x57.html
MySQL

PÁGINAS WEB RECOMENDADAS

http://www.aaai.org/AITopics/html/natlang.html
Natural Language Processing

http://www.scism.sbu.ac.uk/inmandw/tutorials/nlp/syntax/syntax.html
NLP Tutorials

http://www.geocities.com/CapeCanaveral/Hangar/4434/iadex.html
Inteligencia Artificial - Carlos H. von der Becke

http://www.practique-espanol.com/gramatica/gramatica01.htm
Practique Español - Gramática

http://www.labarcadelacultura.com/gramatica01.php
La barca de la cultura – Gramática castellana

http://redcientifica.com/gaia/
GAIA – Inteligencia Artificial

http://www.cs.dartmouth.edu/~brd/Teaching/AI/Lectures/Summaries/natlang.html
Natural Language
Patrick Doyle
AI Qual Summary

http://www.abenteuermedien.de/jabberwock/faq_en.html
CHATTERBOT FAQ

http://www.loebner.net/Prizef/loebner-prize.html
LOEBNER PRIZE

http://www.rediff.com/netguide/2003/jun/10bot.htm
What a chatterbot

http://chatbotfriends.com/reference.html

REFERENCIAS BIBLIOGRÁFICAS

[ALICE, 2004] A.L.I.C.E Artifical Intelligence Foundation. http://www.alicebot.com

[CIMIA, 2004] Mexican International Conference On Artificial Intelligence, Introducción al Tratamiento Automático del Lenguaje Español**.** México, 26-30 abril, 2004. http://ccc.inaoep.mx/~mapco/MICAI04/content/c3.html

[De la vega,1998] DE LA VEGA, Manuel. Introducción a la psicología cognitiva. Alianza Editorial S.A. Madrid, 1998. p.276.

[Dreyfus, 1986, 122] Dreyfus, H.L. 1986. Dreyfus, S.E., Mind over Machine. New York:Macmillan/The Free Press.

[EMENCARTA] Enciclopedia Microsoft Encarta 2004

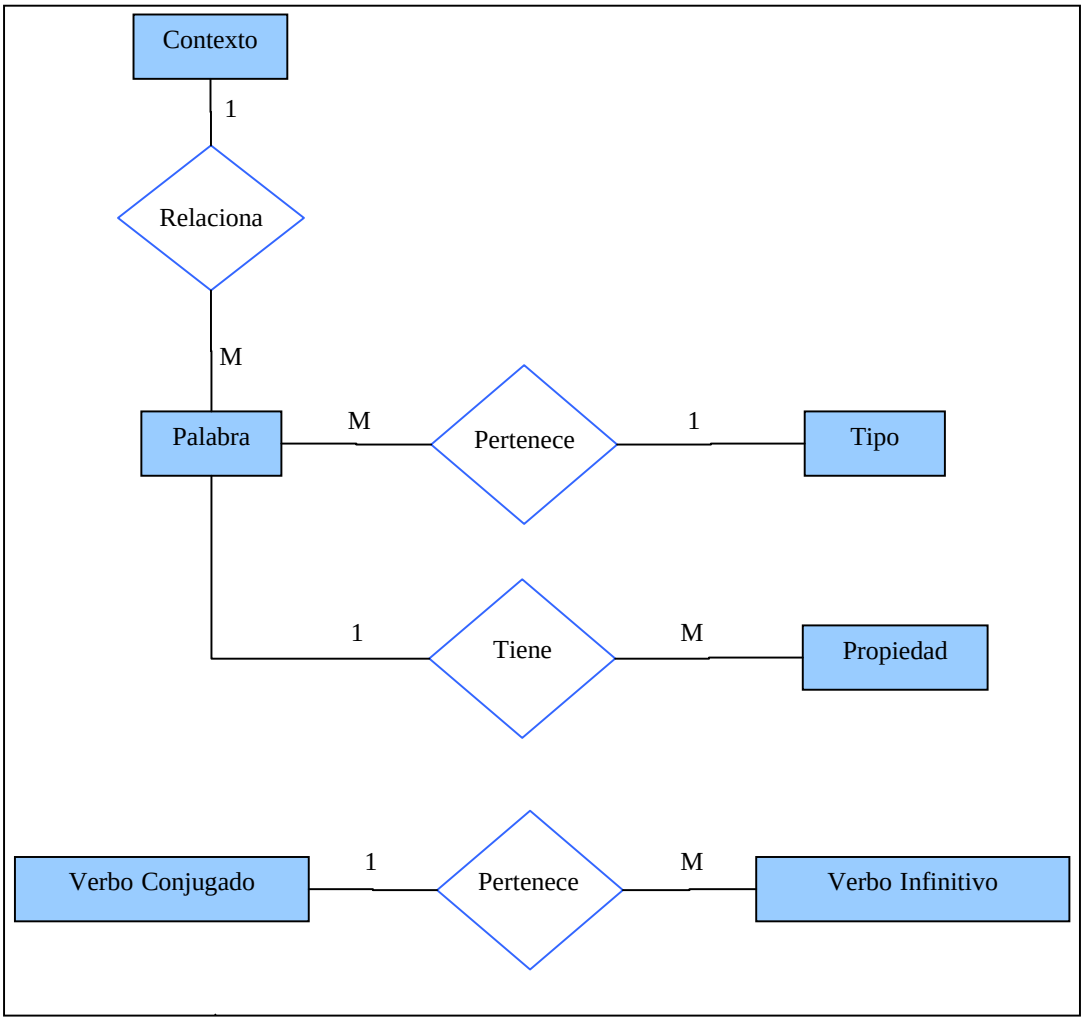
[GPLSI] Grupo de Investigación en Procesamiento del Lenguaje Natural y Sistemas de Información**.** Universidad de Alicante, España. http://gplsi.dlsi.ua.es/

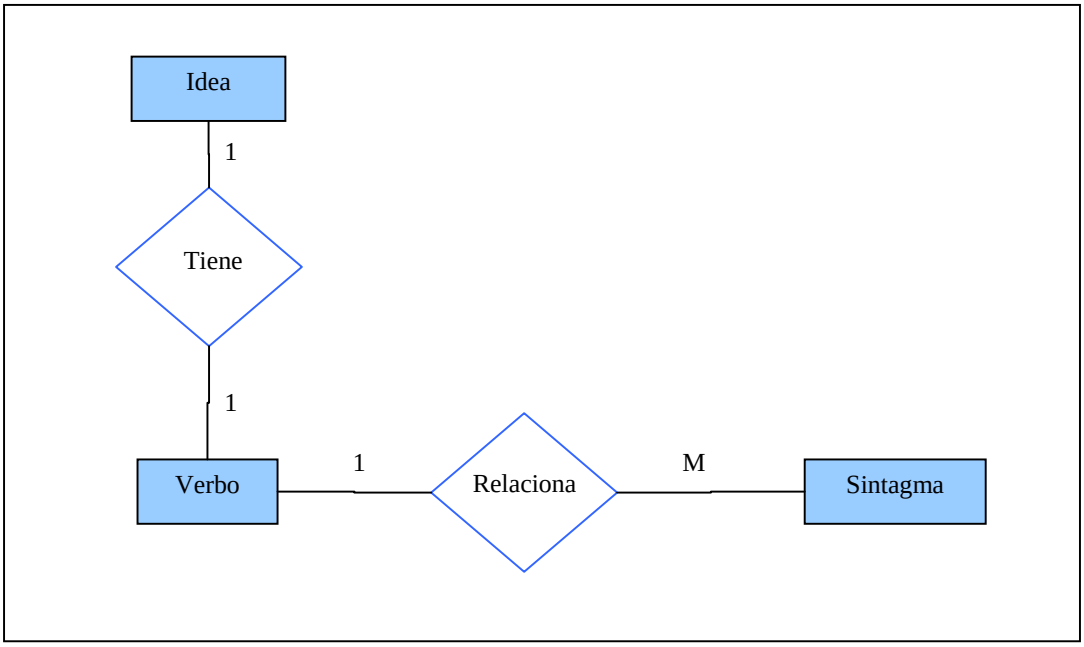
[HAWRYSZKIEWYCZ, HAWRYSZKIEWYCZ, I.T. Análisis y diseño de bases de

1994].	datos. Editorial LIMUSA, México D.F., 1994. p 140-180
[J. Copeland, 1993, 85]	J. COPELAND, B.J. 1993. Artificial Intelligence: A Philosophical Introduction. Oxford: Blackwell. p. 85-86.
[Luengo,1991]	Capítulo III Nuevas Tecnologías en Educación http://med.unex.es/Docs/TesisChavero/Cap3.pdf
[M. Minsky, 1967]	M. Minsky, 1967.
[Meyer,1999]	MEYER, Bertrand. Construcción de software orientado a objetos. 2 ed. Prentice Hal, Madrid,1999
[MORIELLO,1999]	MORIELLO, Sergio A. 1999. Intelectos Sintéticos Avanzados. http://www.swan.ac.uk/hispanic/IntelectosSinteticosAvanzados.htm
[NETBEANS, 2004]	NETBEANS. http://www.netbeans.org/index_es.html
[Spencer, 1997]	SPENCER, Kenneth L., MILLER Ken. Programación Cliente/Servidor con MicroSoft Visual Basic. McGraw Hill, Madrid, 1997. p 2-6
[Turing, 1950, 433]	TURING, A.M. Computing Machinery and Intelligence, En: Mind. Vol. 59, No.236 (1950); p. 433–460.
[UNAV]	http://web1.cti.unav.es/asignaturas/ia/programa97.html
[Waltz, 1982, 122]	Waltz, D.L. Artificial Intelligence. En: Scientific American, 247 (4), p. 122.
[Winograd, 1982, 98]	WINOGRAD, T.A., FLORES, F. Understanding Computing and Cognition. Norwood, N.J.: Ablex, 1986. p. 98.

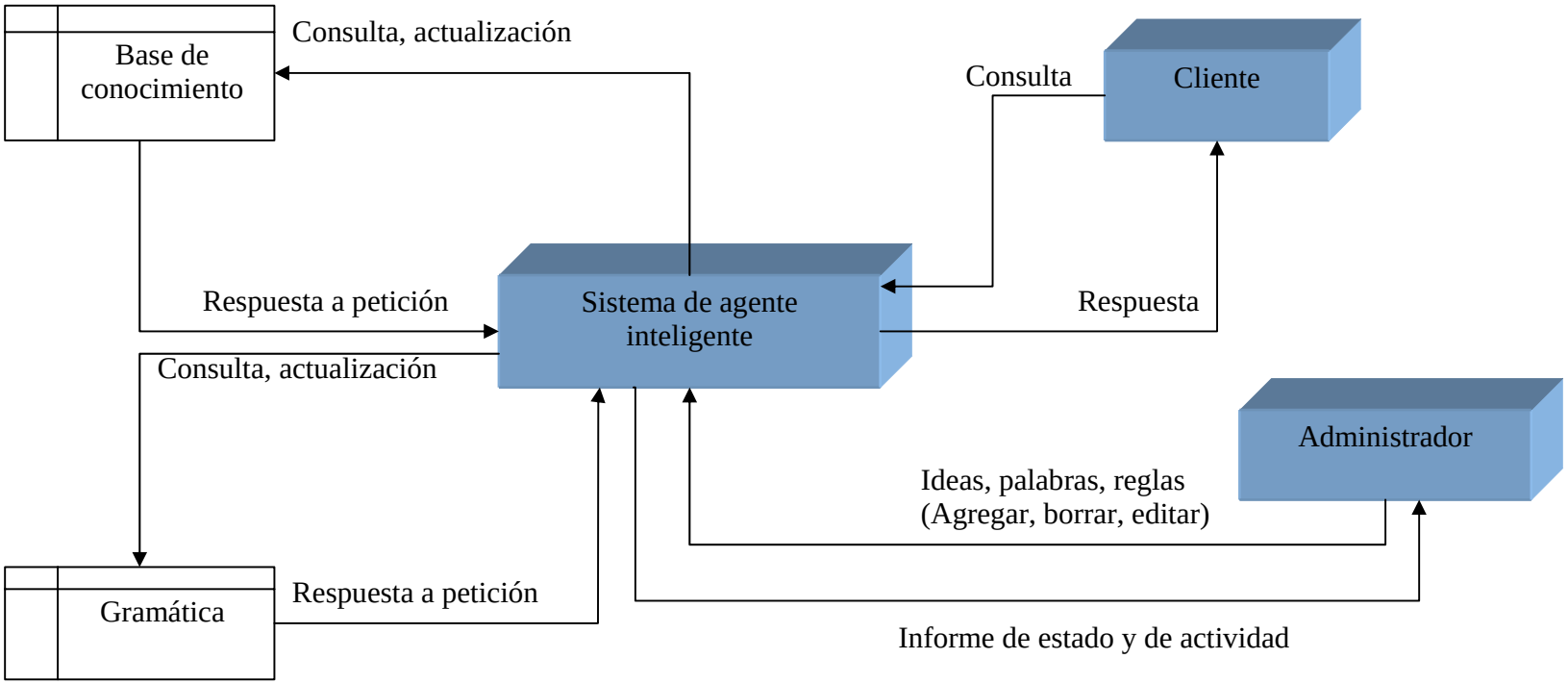
ANEXOS

ANEXO A. MODELO ENTIDAD-RELACIÓN BASE DE DATOS DEL AGENTE

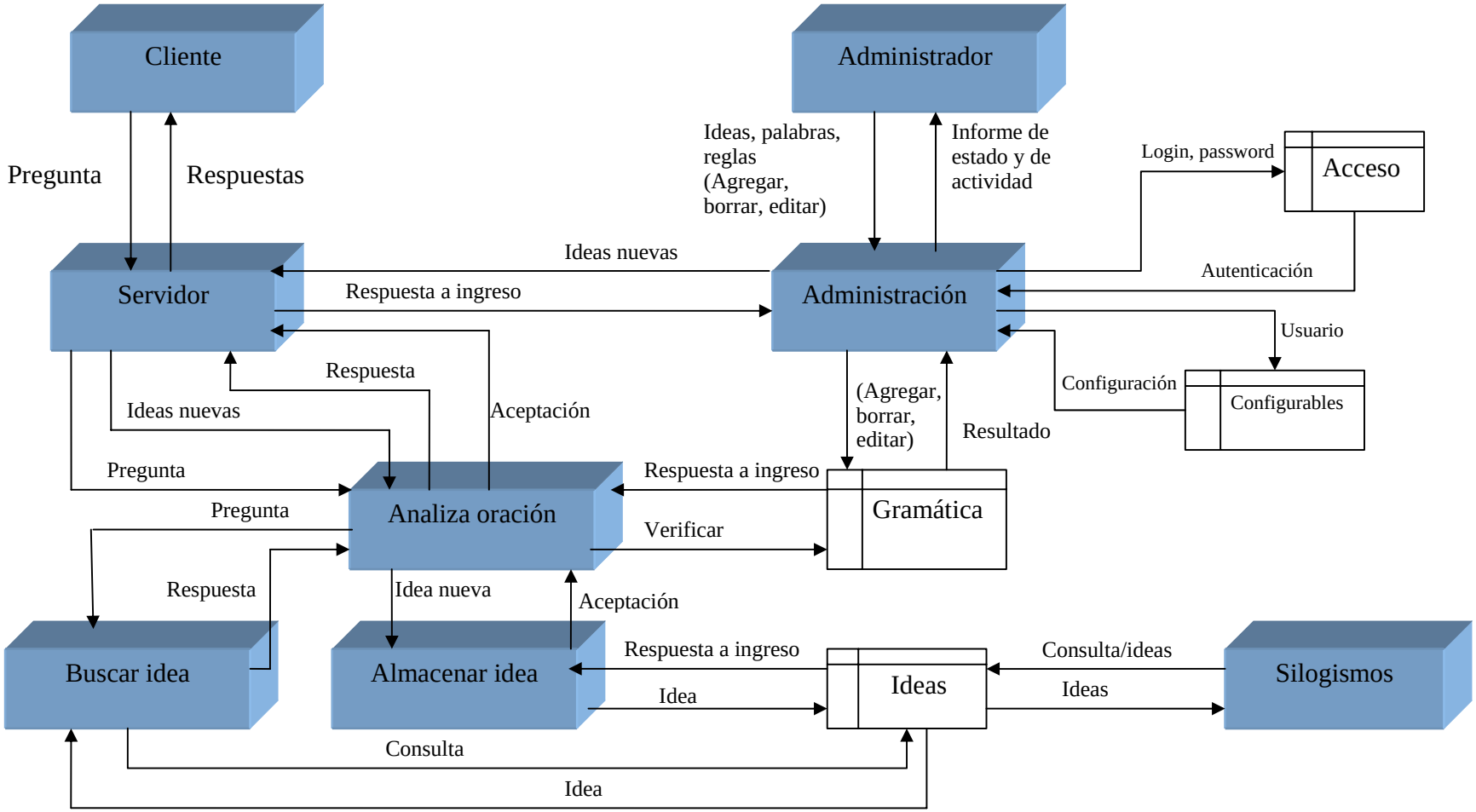



Idea
1
Tiene
1
Verbo
1
Relaciona
M
Sintagma

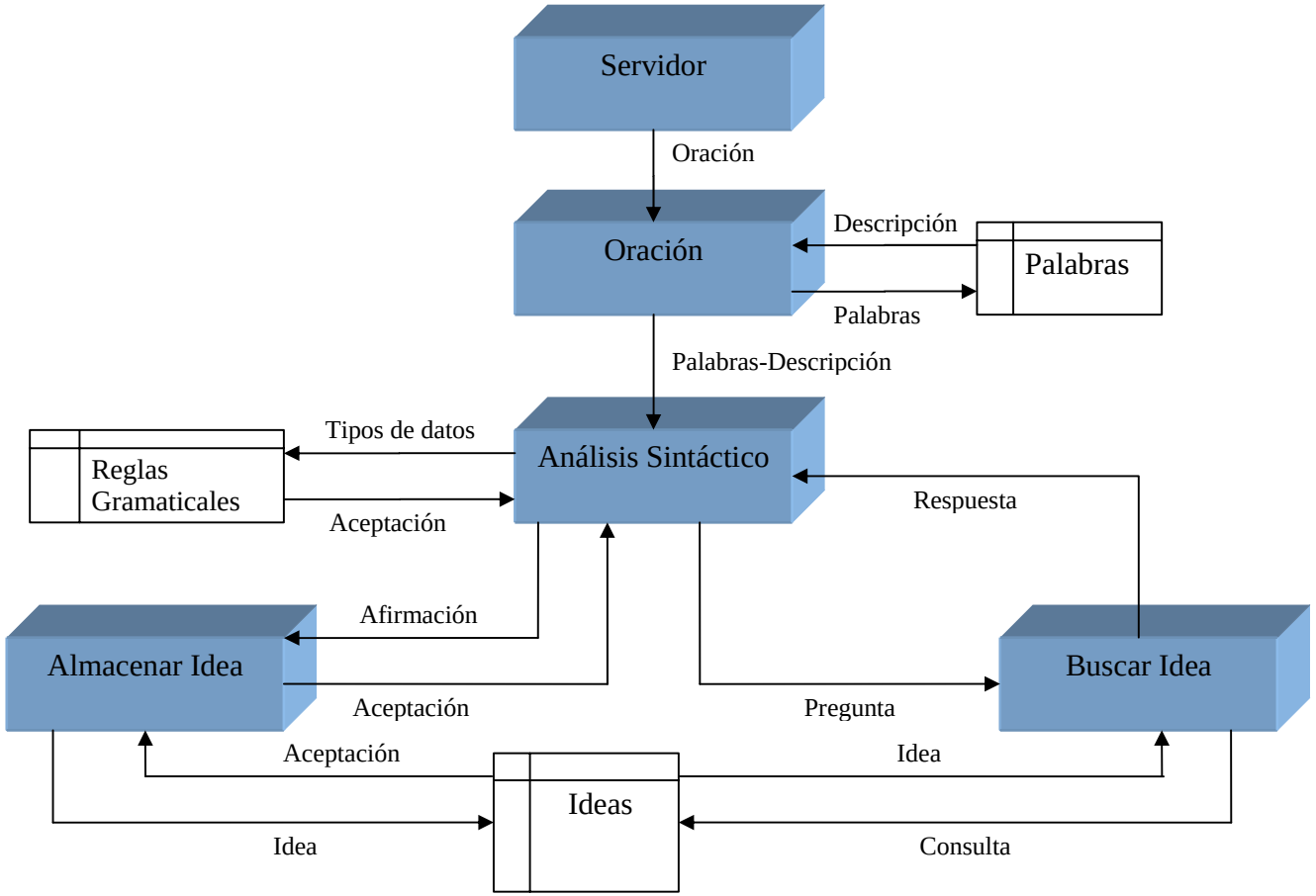
ANEXO B. DIAGRAMA DE FLUJO DE CONTEXTO



ANEXO C. DIAGRAMA DE FLUJO PRIMIGENIO



ANÁLISIS ORACIÓN



ANEXO D. DISEÑO TABLAS DE LA GRAMÁTICA DEL AGENTE

Tabla: tipo		
Nombre del campo	**Tipo**	**Descripción**
id_tipo	bigint(20) autoincremental	Campo Índice
tipo_tipo	varchar(255)	Guarda cada tipo de palabra
tipo_propiedades	Text	Guarda el conjunto de propiedades, separadas por comas (,), que puede tener el tipo de palabra

Tabla: palabra		
Nombre del campo	**Tipo**	**Descripción**
id_palabra	bigint(20) autoincremental	Campo Índice
pala_palabra	varchar(255)	Guarda las palabras que conforman el vocabulario del agente
pala_idpropiedadpalabra	bigint(20)	Establece el vínculo entre la tabla palabra y la tabla propiedad palabra en la cual se almacenan el tipo y las propiedades de las palabras

Tabla: propiedad		
Nombre del campo	**Tipo**	**Descripción**
id_propiedad	bigint(20) autoincremental	Campo Índice
prop_grupo	varchar(255)	Guarda el grupo al que pertenece la propiedad Ej.: Género
prop_valor	varchar(255)	Guarda el valor de la propiedad Ej.: Masculino, Femenino.
prop_incompatible	Text	Guarda un grupo de propiedades, separadas por comas (,), con las cuales la propiedad actual no puede coincidir en una oración

Tabla: propiedadpalabra		
Nombre del campo	**Tipo**	**Descripción**
id_propiedadpalabra	bigint(20) autoincremental	Campo Índice
prpa_idtipo	bigint(20)	Relaciona las propiedades con un tipo de palabra determinado
prpa_propiedades	Text	Guarda las propiedades, separadas por comas (,), que tiene el tipo de palabra según un grupo de palabras determinado Ej. Unos Sustantivos pueden tener las propiedades singular y masculino pero otros puede tener plural y femenino

Tabla: reglaanalisis		
Nombre del campo	**Tipo**	**Descripción**
id_reglaanalisis	bigint(20) autoincremental	Campo Índice
rean_regla	varchar(255)	Guarda las combinaciones separadas por comas (,) de los tipos de palabra, sintagmas o complementos
rean_produce	varchar(255)	Guarda el resultado producido por la combinación guardada en el campo anterior
rean_comparar	tinyint(1)	Indica cómo se deben comparar las propiedades de los elementos que conforman la combinación: ▪ 0: no realiza ninguna comparación ▪ 1: compara las propiedades del primer elemento de la combinación con las del segundo y el resultado obtenido es el conjunto unión de las propiedades. ▪ 2: igual que en 1 pero el resultado son las propiedades del primer elemento. ▪ 3: igual que en 1 pero el resultado son las propiedades del segundo elemento. ▪ 4: igual que en 1 pero sin resultado. ▪ 5: invierte el orden de los elementos y hace una comparación tipo 1 ▪ 6: igual que 5 pero hace una comparación tipo 2 ▪ 7: igual que 5 pero hace una

		comparación tipo 3 ▪ 8: no compara y el resultado son las propiedades del primer elemento. ▪ 9: no compara y el resultado son las propiedades del segundo elemento.
rean_tipo	tinyint(1)	Identifica si la combinación es de tipo especial, es decir que puede generar un resultado diferente si esta al final de una oración. Los valores que pude tener son: ▪ 1: si está al final de la oración ▪ 2: si le sigue otra palabra

Tabla: verboinfinitivo		
Nombre del campo	**Tipo**	**Descripción**
id_ verboinfinitivo	bigint(20) autoincremental	Campo Índice
vein_idpalabrainfinitivo	bigint(20) autoincremental	Guarda el índice en la tabla palabra del verbo en infinitivo
vein_idpalabra	bigint(20) autoincremental	Guarda el índice en la tabla palabra del verbo conjugado

Tabla: ideacontexto		
Nombre del campo	**Tipo**	**Descripción**
id_ ideacontexto	bigint(20) autoincremental	Campo Índice
idco_idpalabraprincipal	bigint(20) autoincremental	Guarda el índice en la tabla palabra de la palabra que se relaciona con las otras
idco_palabras	Text	Guarda el conjunto de palabras relacionadas separadas por comas (,).
idco_tipo	tinyint(3)	Indica el tipo de relación de las palabras: ▪ 1: sinónimos conceptuales ▪ 2: sinónimos connotativos ▪ 3: sinónimos eufemismo ▪ 4: Antónimos complementarios ▪ 5: Antónimos graduales ▪ 6: Antónimos recíprocos ▪ 7: Primera persona – Segunda persona ▪ 8: Plural - Singular ▪ 9: Masculino – Femenino

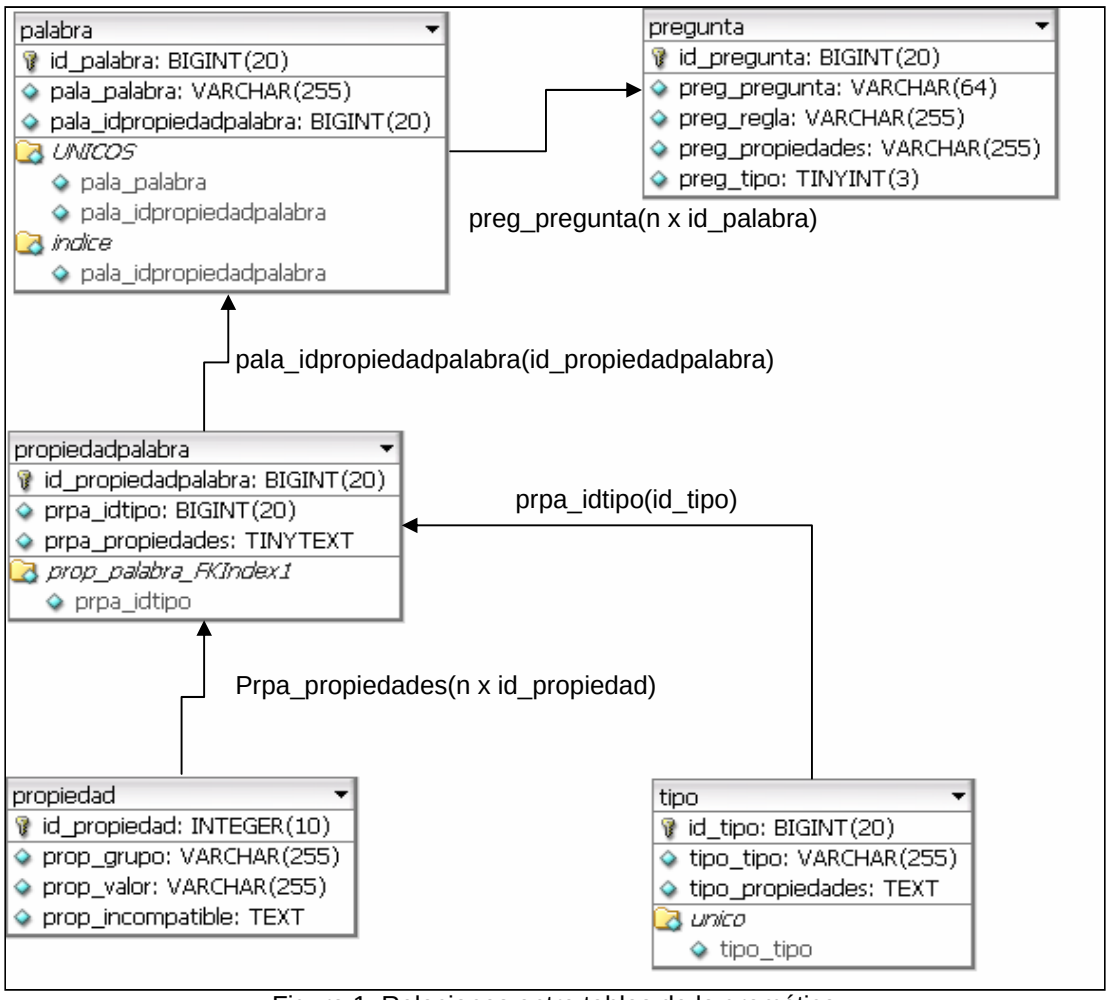


Figura 1. Relaciones entre tablas de la gramática

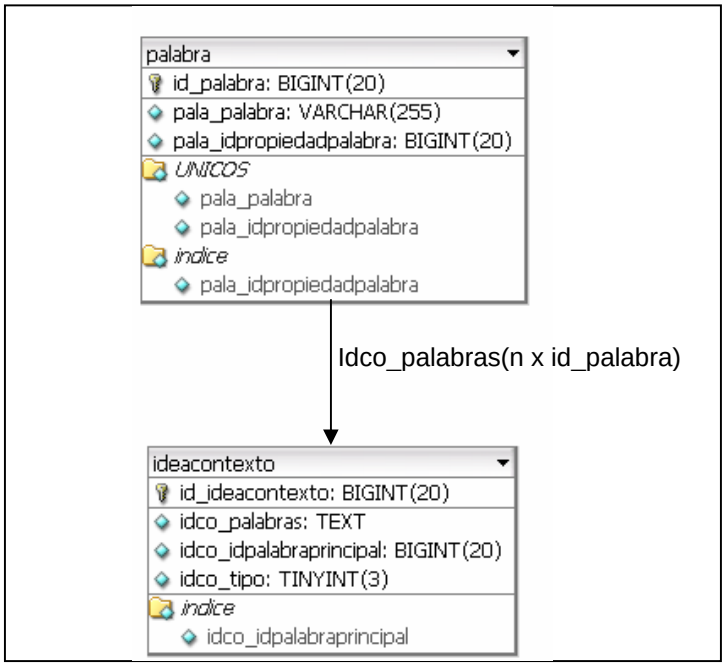


Figura 2. Relaciones entre las tablas palabra e ideacontexto

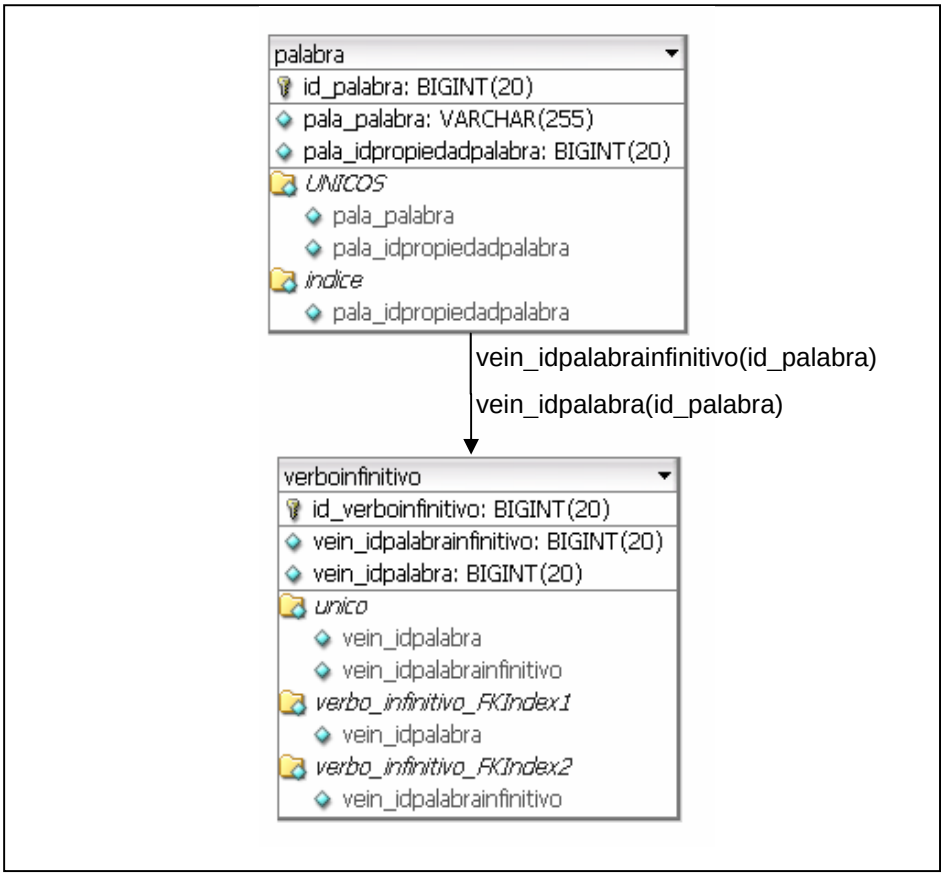


Figura 3. Relación entre las tablas palabra y verboinfinitivo

ANEXO E. DISEÑO TABLAS BASE DE CONOCIMIENTO DEL AGENTE

Tabla: ideaverbo		
Nombre del campo	**Tipo**	**Descripción**
id_ideaverbo	bigint(20) autoincremental	Campo Índice
idve_idverboinfinitivo	bigint(20)	Guarda el verbo principal de la idea, en forma de relación infinitivo y conjugado
idve_signo	tinyint(1)	Indica si la idea es afirmativa o negativa. Los valores que pude tener son: ▪ 1: si es afirmativa ▪ 2: si es negativa

Tabla: ideapalabra		
Nombre del campo	**Tipo**	**Descripción**
id_ideapalabra	bigint(20) autoincremental	Campo Índice
idpa_idideaverbo	bigint(20)	Índice del verbo principal de la idea
idpa_contextos	Text	Guarda los índices de los contextos de las palabras del sintagma separados por comas (,).
idpa_preguntas	Text	Guarda los índices de las palabras o expresiones interrogativas que corresponden al sintagma, separadas por comas (,). Las expresiones interrogativas están conformadas por preposiciones e interrogativos, cuyos índices están separados por un guión (-).
idpa_sujetopredicado	char(1)	Indica si el sintagma hace parte del sujeto o del predicado de la oración generadora de la idea.
Idpa_orden	tinyint(3)	Indica el lugar del sintagma dentro de la oración. Comienza desde cero (0).

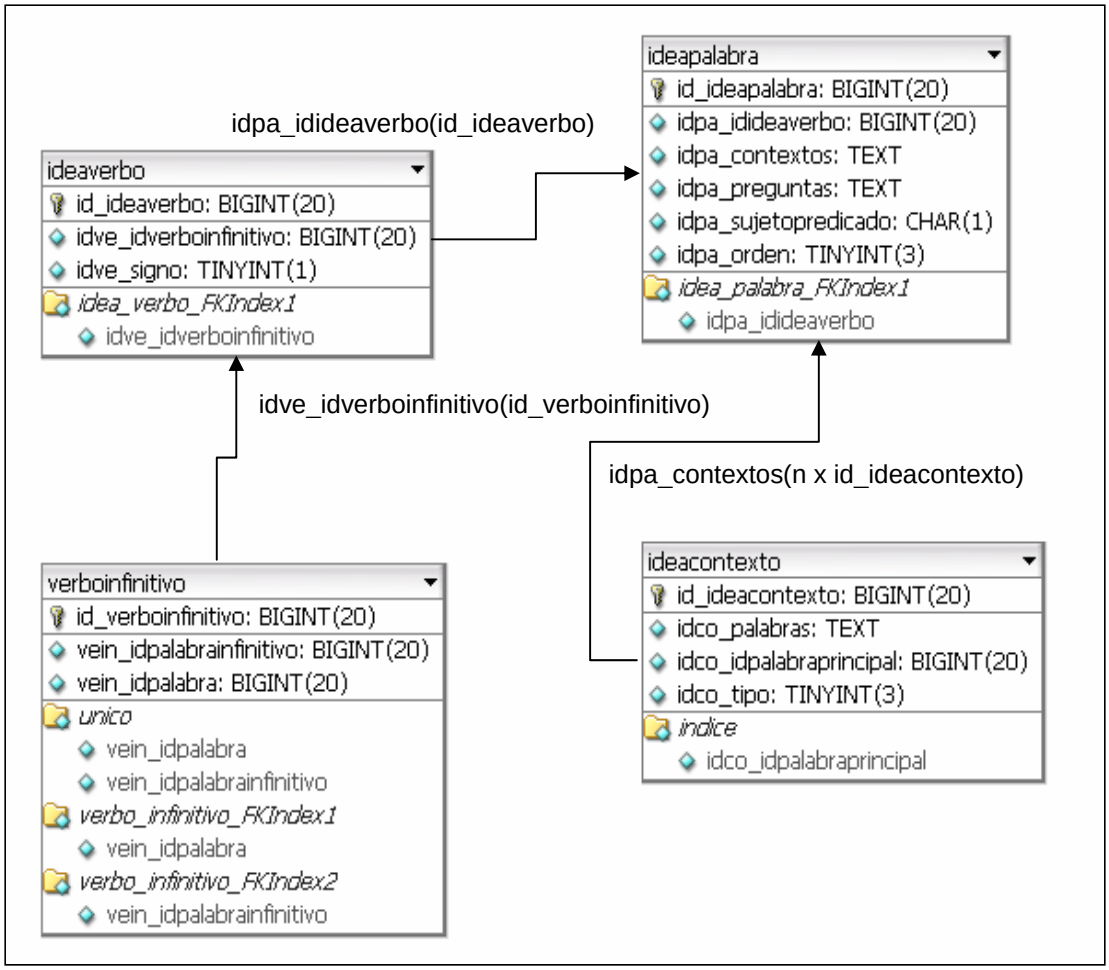


Figura 1. Relaciones entre tablas de la base de conocimiento

ANEXO F. DISEÑO TABLAS AUXILIARES DEL AGENTE

Tabla: variables		
Nombre del campo	**Tipo**	**Descripción**
vari_variable	varchar(100)	Guarda el nombre de una variable que será utilizada por las aplicaciones para almacenar cualquier tipo de valores
vari_valor	varchar (100)	Guarda el valor de las variables

Tabla: conversación		
Nombre del campo	**Tipo**	**Descripción**
*conv_*hilo	varchar(30)	Guarda el valor que identifica al usuario que está conversando con el agente
conv_tipo	varchar(50)	Indica el tipo de frase (AFIRMACIÓN, PREGUNTA, etc)
conv_verbo	varchar(255)	Almacena el índice del verbo en la tabla verboinfinitivo.
conv _sintagmas	tinytext	Guarda cada uno de los sintagmas de la oración, tal y como se obtienen del análisis, separados por punto y coma (;).
conv _interrogantes	varchar(1)	Guarda el interrogante de la oración en caso de que se trate de una pregunta.
conv_fechahora	datetime	Guarda la fecha y hora en que se registró

Tabla: palabraverificar		
Nombre del campo	**Tipo**	**Descripción**
id_palabraverificar	bingint(20) autoincremental	Campo índice
pave_palabraverificar	varchar(255)	Guarda las siglas o las contracciones que se deben separar en una oración.
pave_palabras	varchar(255)	Guarda las palabras por las que se reemplazará la sigla o la contracción separadas por espacios.

ANEXO G. DISEÑO TABLAS AUXILIARES APLICACIÓN ADMINISTRATIVA

La aplicación administrativa utiliza dos tablas adicionales a las de la gramática y las de conocimiento, una que se encarga de almacenar los datos de acceso de los usuarios y otra que se encarga de almacenar valores configurables de cada página para cada usuario específico.

Tabla: acceso		
Nombre del campo	**Tipo**	**Descripción**
id_acceso	bigint(20) autoincremental	Campo Índice
acce_usuario	bigint(20)	Guarda el verbo principal de la idea, en forma de relación infinitivo y conjugado
idve_signo	tinyint(1)	Indica si la idea es afirmativa o negativa

Tabla: configurables		
Nombre del campo	**Tipo**	**Descripción**
conf_usuario	varchar(4)	Indica el usuario al que al pertenece la variable configurada
conf_opcion	varchar (200)	Indica el nombre de la función en la aplicación administrativa
conf_variable	varchar(100)	Indica el nombre de la variable
conf_valor	varchar(100)	Guarda el valor de la variable en la función respectiva.

ANEXO H. MANUAL DE USUARIO APLICACIÓN ADMINISTRATIVA

La aplicación administrativa del agente inteligente debe ser instalada de acuerdo a lo explicado en la implementación del prototipo (véase capítulo 3.4) para su correcto funcionamiento.

La aplicación consta de los siguientes archivos:

- bot.php: Contiene las funciones principales encargadas de la ejecución de la aplicación y del acceso de los usuarios.
- configuracion.php: Contiene las funciones que se encargan de mostrar, editar, agregar y eliminar en cada una de las diferentes opciones de la aplicación, entre las que se encuentran: tipos de palabras, palabras, propiedades, relaciones de verbos, relaciones de palabras (contextos), reglas gramaticales, reglas de preguntas, ideas y charla con agente.
- objeto.php: Contiene una clase que ayuda a diseñar las páginas en el formato específico de la aplicación.
- funciones.phtml: Es la librería que utiliza el programa, ya que contiene una serie de funciones que realizan toda clase de operaciones y procesos, tales como, conexión y operaciones en la base de datos, operaciones con matrices, arreglos, etc. También contiene las constantes definidas para el funcionamiento del programa y mensajes que se muestran a los usuarios.

También hacen parte de la aplicación, las siguientes carpetas:

- html: Se utiliza para almacenar todas las carpetas de los diferentes idiomas en los cuales se desea mostrar la aplicación.
- es: Se utiliza para almacenar todos los archivos que contienen los extractos en código html que en conjunto forman las diferentes páginas de la aplicación. Es una subcarpeta de la carpeta “html” y su nombre depende del idioma (en el caso del español es “es”, en inglés sería “en”, etc).
- images: Se utiliza par almacenar las imágenes que muestra la aplicación en un idioma específico. Es una subcarpeta de las carpetas de cada idioma.

Figura 1. Estructura jerárquica de directorios de la aplicación administrativa

Los archivos de la aplicación deben ser almacenados en la misma carpeta que la carpeta “html”. En el caso de la Figura 1 de este anexo, la carpeta principal es “htdocs”, en la cual se deben estar todos los archivos y carpetas mencionados. La carpeta “htdocs” debe ser configurada como un directorio accesible desde la red en el servidor web instalado.

Funcionamiento de la aplicación: La aplicación se inicia digitando en un navegador de Internet, la dirección donde esté ubicado el archivo bot.php en el servidor o a través de un enlace (o hipervínculo) ubicado en una página web, lo cual conlleva a que se abra una página de acceso, donde se pide un nombre de usuario y una contraseña (Véase Figura 2).

Figura 2. Pantalla de ingreso.

Una vez se ingrese al sistema aparece una página de identificación (véase Figura 3) en la que se debe presionar el botón “Entrar” para continuar o en el botón “Salir” para cerrar la aplicación (el cual aparecerá en todas las páginas siguientes).

Figura 3.Pantalla de bienvenida al usuario.

La siguiente página es el menú de la aplicación (véase Figura 4), donde se puede escoger cualquier opción en la que se desee trabajar.

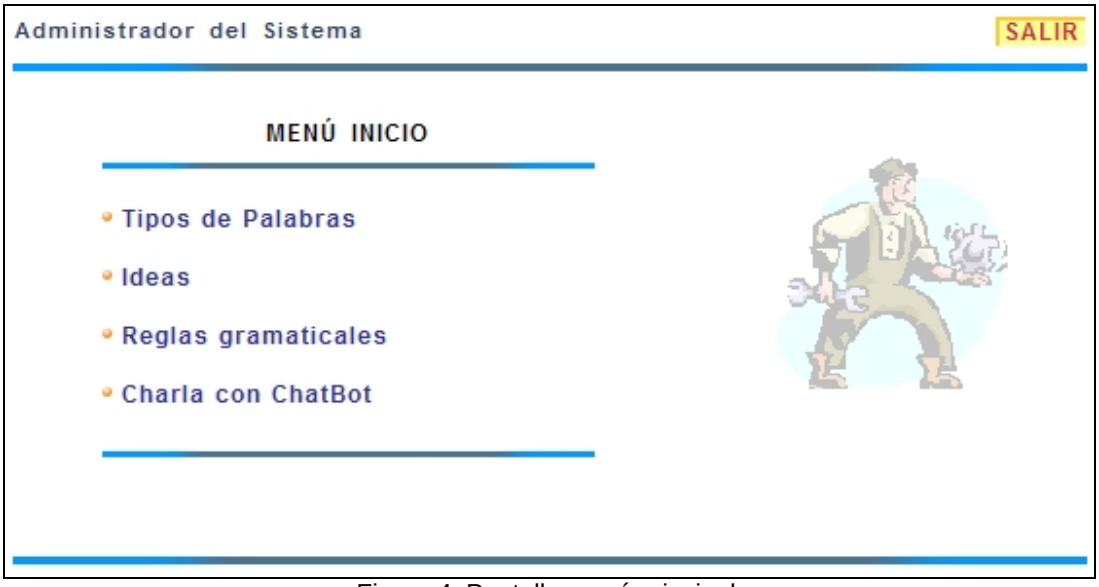


Figura 4. Pantalla menú principal.

Al seleccionar la opción “Tipos de palabras” se tiene acceso para ver, agregar, editar y eliminar, los diferentes tipos de palabras de la gramática (Véase Figura 5), las palabras del vocabulario del agente, las propiedades de los tipos de palabras y de cada una de las palabras, las relaciones entre los verbos (Véase Figura 6) y las relaciones de sinónimos, antónimos, singular-plural, etc., entre palabras (Véase Figura 7).

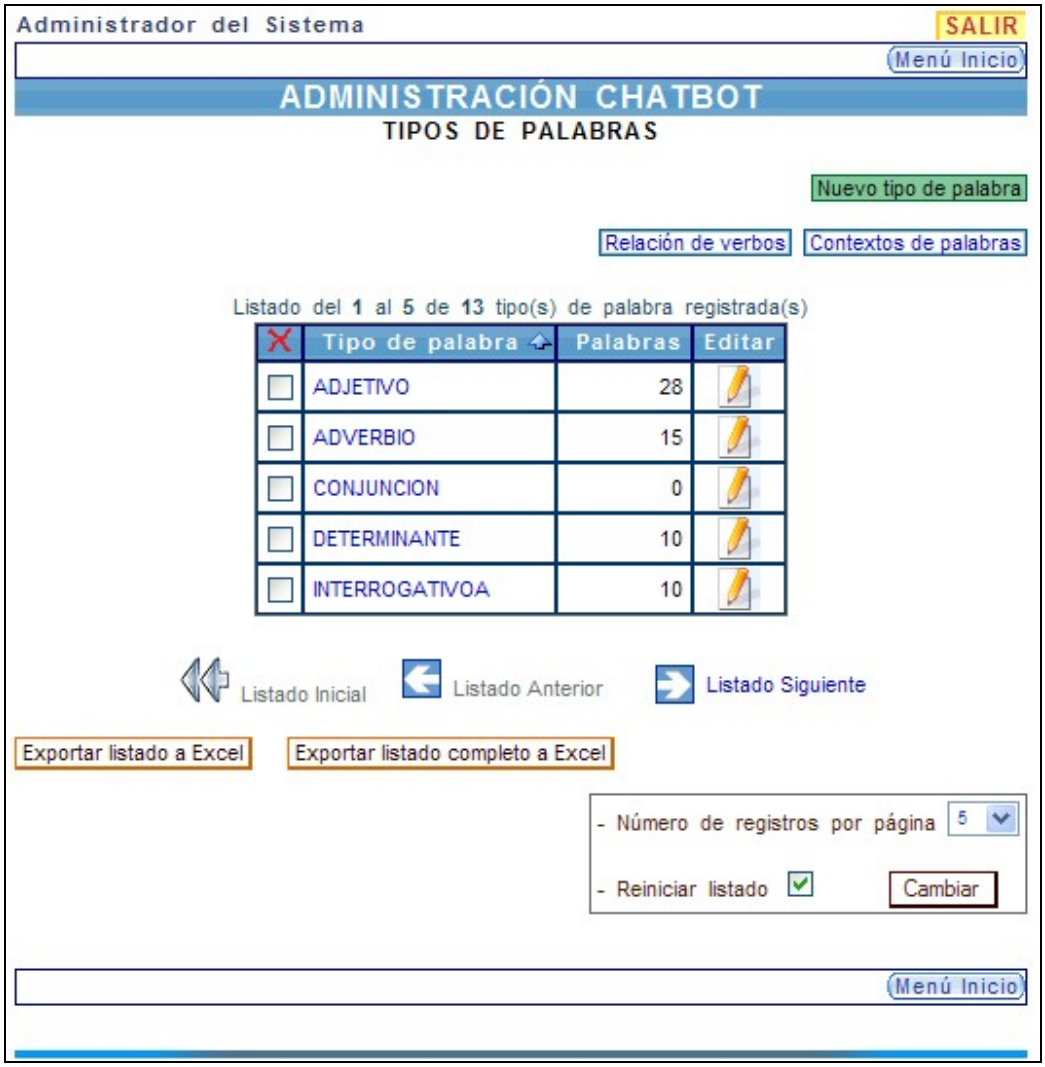


Figura 5. Pantalla listado de los tipos de palabras.

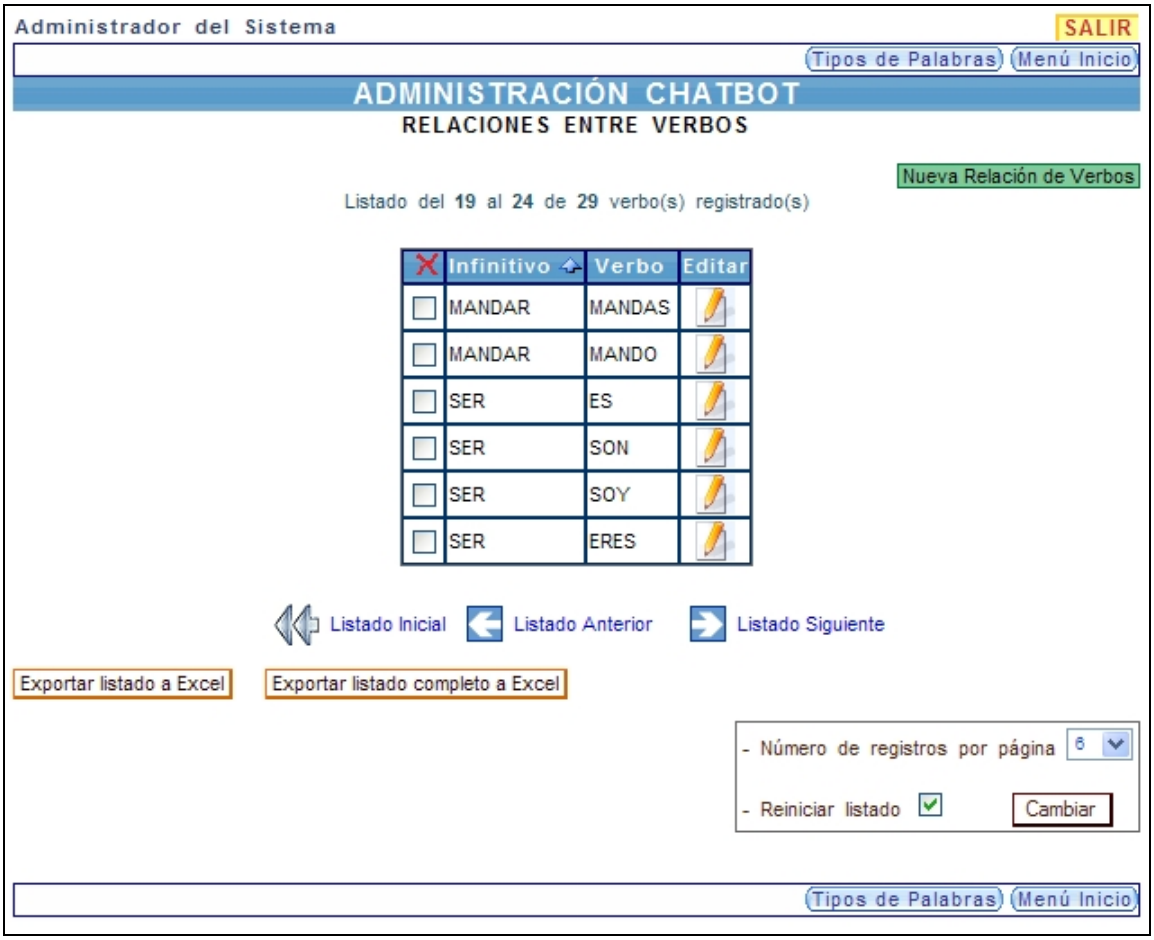


Figura 6. Pantalla listado de las relaciones entre verbos (conjugaciones).

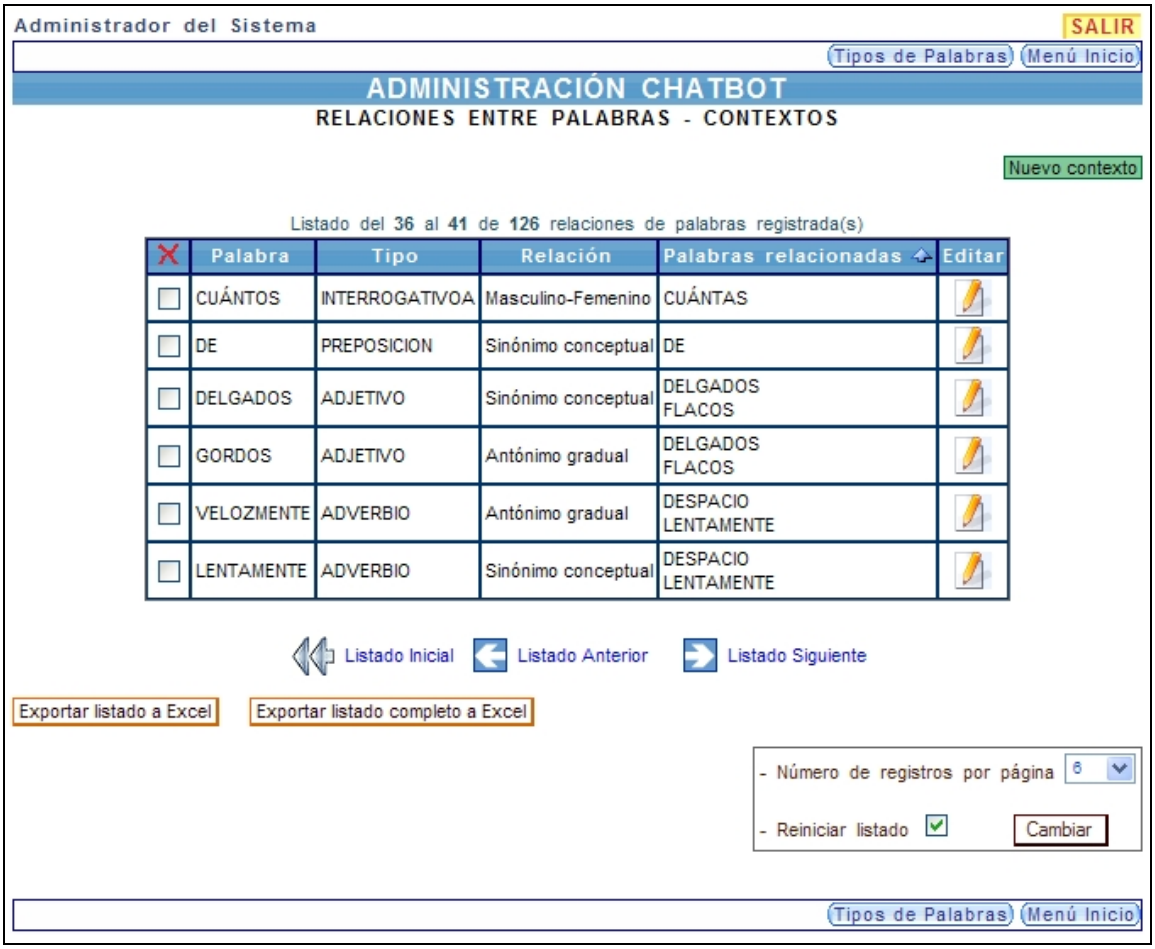


Figura 7. Pantalla listado de las relaciones entre palabras.

Al seleccionar la opción “Ideas” se tiene acceso para visualizar todas las ideas que contiene el agente en su base de conocimientos (véase Figura 8).

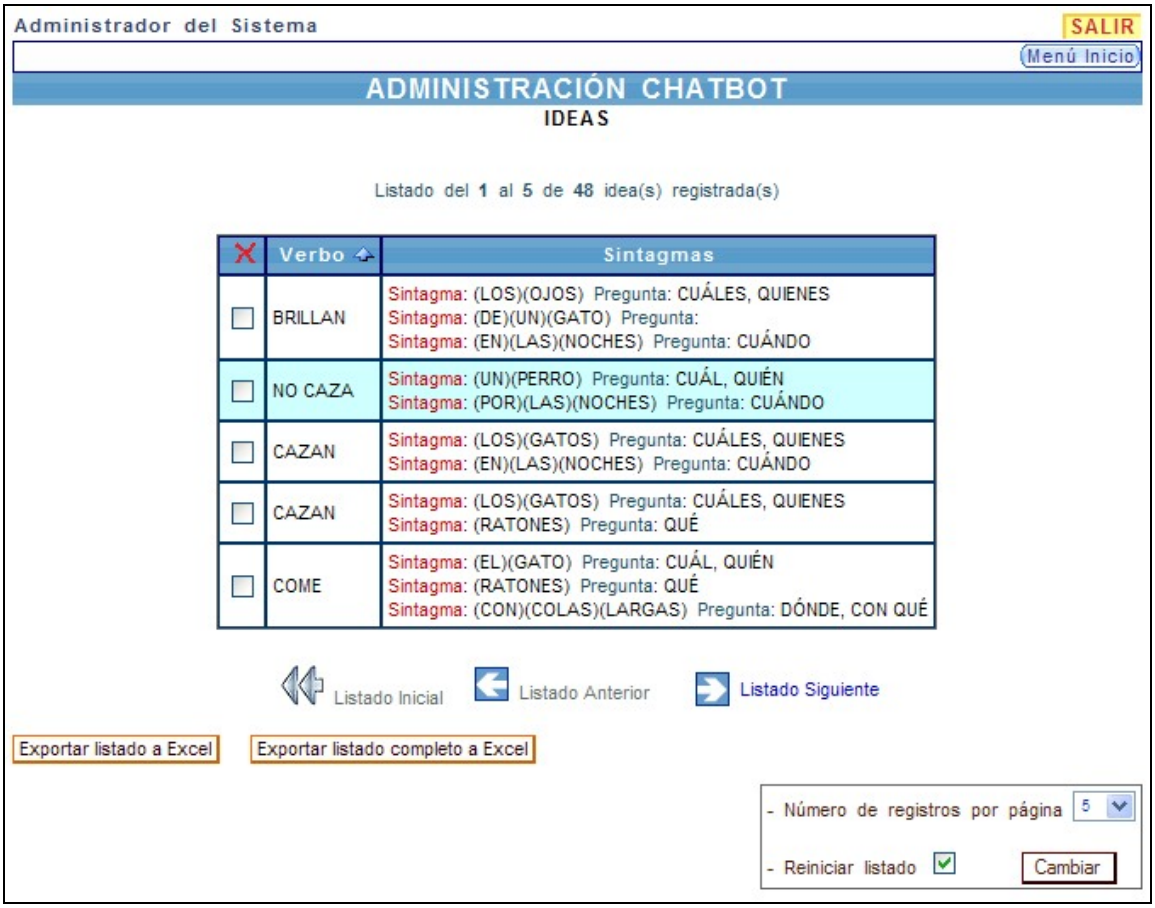


Figura 8. Pantalla listado de ideas.

Al seleccionar la opción “Reglas gramaticales” se tiene acceso para ver, agregar, editar y eliminar las reglas gramaticales (véase Figura 9) y las reglas de preguntas (véase Figura 10) que rigen al agente en el análisis y generación de oraciones.

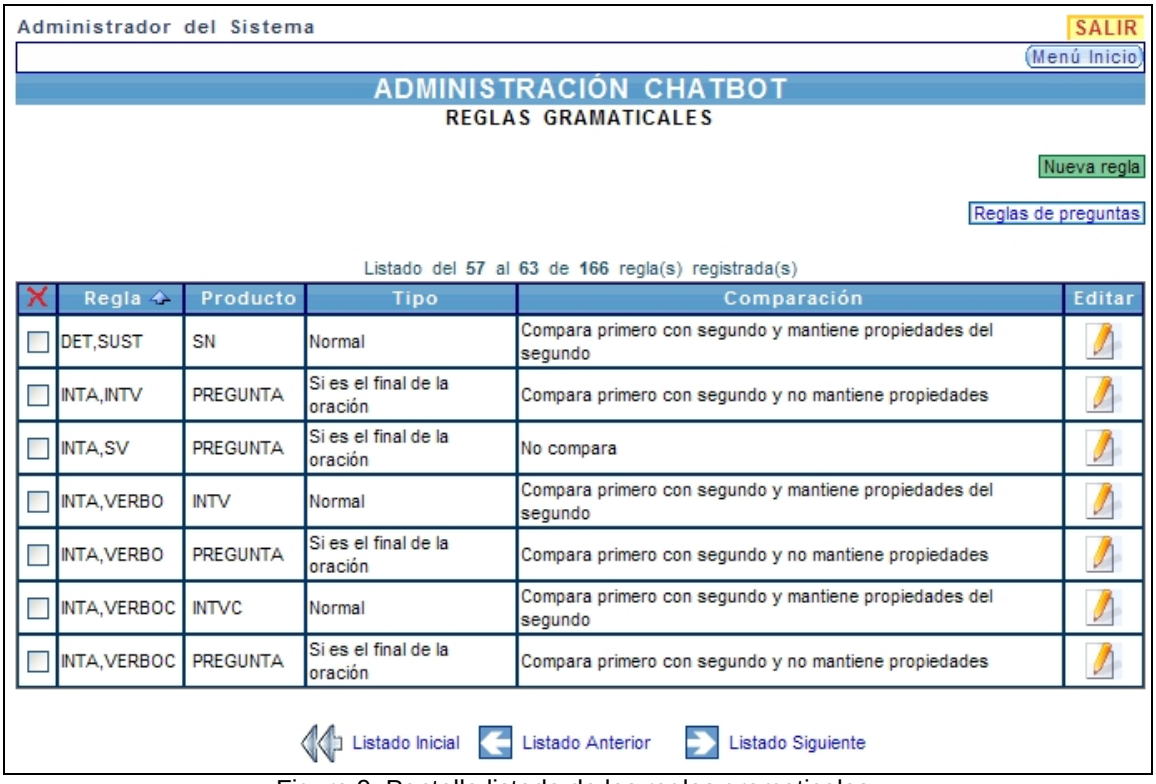


Figura 9. Pantalla listado de las reglas gramaticales.

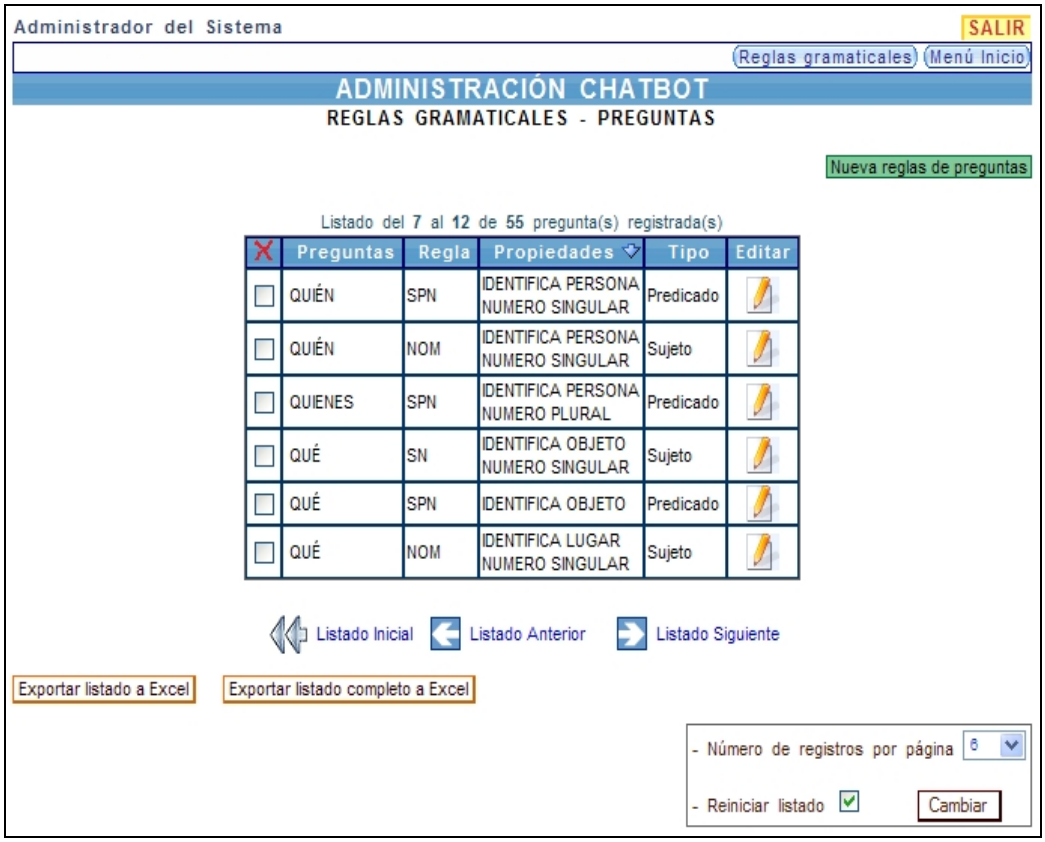


Figura 10. Pantalla listado de las reglas de preguntas.

Al seleccionar la opción “Charla con chatbot” se abre una página que contiene el applet que se conecta al agente inteligente y permite entablar una conversación con él (véase Figura 11). En esta conversación el agente tomará las oraciones simples afirmativas y negativas como ideas nuevas para almacenar en la base de conocimientos.

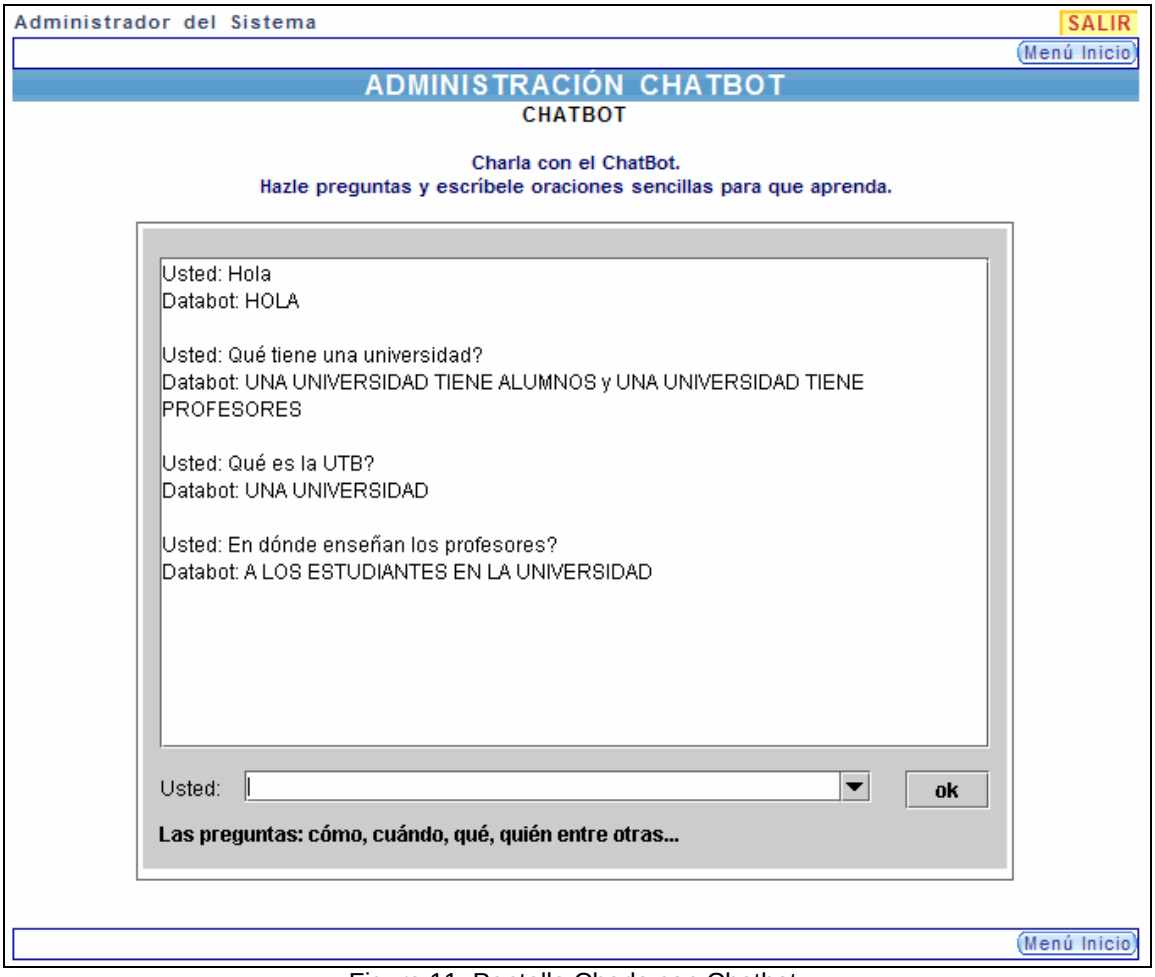


Figura 11. Pantalla Charla con Chatbot.

Las opciones de agregar o editar un registro de la base de datos se presentan al hacer clic sobre el botón respectivo, con lo cual se muestra una página con un formulario (véase Figura 12) donde se solicitan los datos del registro y se debe hacer clic sobre el botón ubicado debajo de dicho formulario para guardar dichos datos.

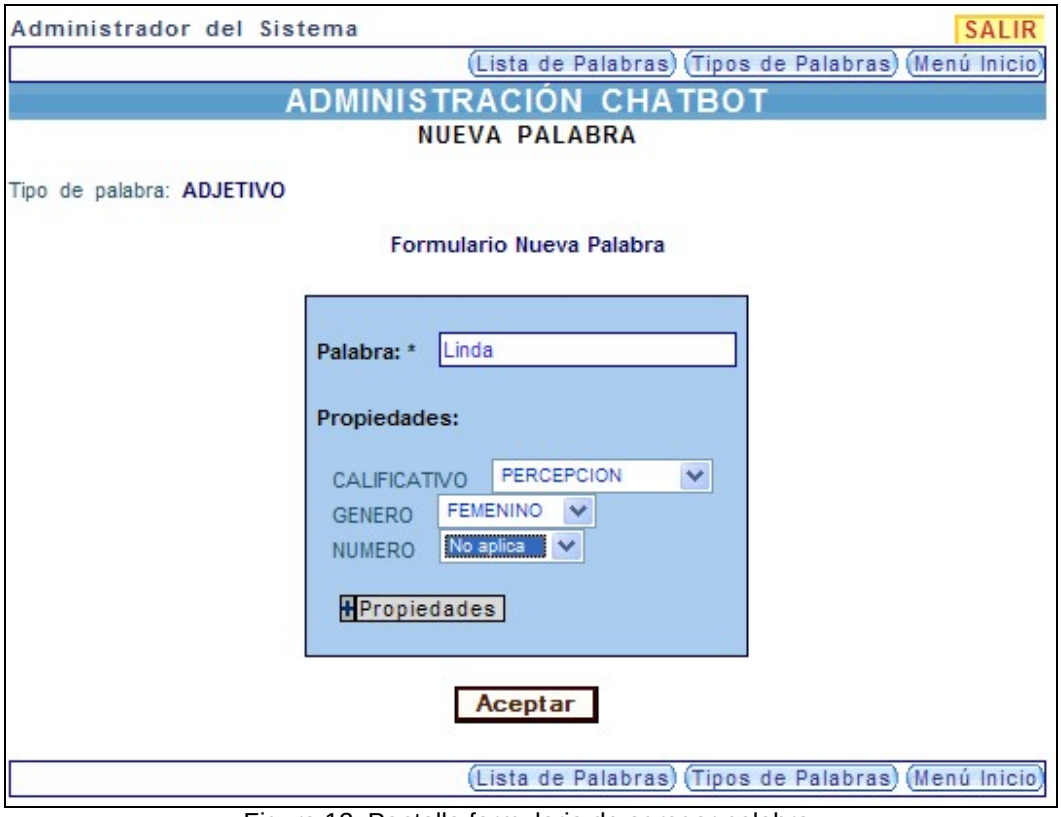


Figura 12. Pantalla formulario de agregar palabra.

La aplicación es muy intuitiva en cuanto a su manejo se refiere, es decir, en cada pantalla aparecen las diferentes opciones a las que se pueden acceder y para eso, sólo es necesario hacer clic sobre ellas. Las columnas de los listados pueden ser ordenadas al gusto del usuario, el número de registros de un listado por cada página también puede ser configurado, se pueden exportar los listados a un archivo Excel. Para eliminar un registro sólo es necesario seleccionar su casilla (a la izquierda) y luego hacer clic sobre el botón eliminar ubicado en la primera fila y columna de cada listado. Para desplazarse entre las páginas de cada listado se debe hacer clic sobre los botones de desplazamiento (si están disponibles), ubicados debajo del listado (Listado inicial, Listado anterior y Listado siguiente).

ANEXO J. REGLAS GRAMATICALES

El conjunto de reglas gramaticales utilizado actualmente es el siguiente:

1	Normal	SUST	No compara
11	Normal	VERBOH	No compara
12	Normal	SALUDO	No compara
13	Normal	VERBOC	No compara
14	Normal	NEGACION	No compara
2	Normal	DET	No compara
3	Normal	VERBO	No compara
4	Normal	ADJ	No compara
5	Normal	ADV	No compara
6	Normal	INTA	No compara
7	Normal	INTB	No compara
8	Normal	PREP	No compara
9	Normal	NOM	No compara
ADJ	Normal	ADJ	No compara
ADJ,CSP	Normal	CAN	No compara y devuelve propiedades del primero
ADJ,SPN	Normal	CAN	No compara y devuelve propiedades del primero
ADJ,SPV	Normal	CAPV	No compara y devuelve propiedades del primero
ADJ,SUST	Normal	SADJ	Compara primero con segundo y mantiene propiedades del segundo
ADV	Normal	ADV	No compara
ADV,ADJ	Normal	ADJ	Compara primero con segundo y une propiedades
ADV,ADV	Normal	ADV	Compara primero con segundo y mantiene propiedades del segundo
ADV,CNN	Normal	CAN	No compara y devuelve propiedades del primero
ADV,CNPV	Normal	CAPV	No compara y devuelve propiedades del primero
ADV,CSP	Normal	CAN	No compara y devuelve propiedades del primero
ADV,SPN	Normal	CAN	No compara y devuelve propiedades del primero
ADV,SPV	Normal	CAPV	No compara y devuelve propiedades del primero
ADV,SUST	Si es el final de la oración	CAN	No compara y devuelve propiedades del segundo

CAN	Normal	CAN	No compara
CAN,SPN	Normal	CAN	No compara y devuelve propiedades del primero
CAN,SPV	Normal	CAPV	No compara y devuelve propiedades del primero
CAPV	Normal	CAPV	No compara
CAPV,CAN	Normal	CAN	No compara y devuelve propiedades del primero
CAPV,CAPV	Normal	CAPV	No compara y devuelve propiedades del primero
CAPV,CNN	Normal	CAN	No compara
CAPV,CNPV	Normal	CAPV	No compara y devuelve propiedades del primero
CAPV,SPN	Normal	CAN	No compara
CAPV,SPV	Normal	CAPV	No compara y devuelve propiedades del primero
CAPV,SUST	Si es el final de la oración	CAN	No compara
CNN	Normal	CNN	No compara
CNN,ADJ	Si es el final de la oración	CNN	No compara y devuelve propiedades del primero
CNN,ADV	Si es el final de la oración	CNN	No compara y devuelve propiedades del primero
CNN,SPN	Normal	CNN	No compara y devuelve propiedades del primero
CNN,SPV	Normal	CNPV	No compara y devuelve propiedades del primero
CNPV	Normal	CNPV	No compara
CNPV,CAN	Normal	CNN	No compara
CNPV,CAPV	Normal	CNPV	No compara y devuelve propiedades del primero
CNPV,CNN	Normal	CNN	No compara y devuelve propiedades del primero
CNPV,CNPV	Normal	CNPV	No compara y devuelve propiedades del primero
CNPV,CSP	Normal	CNN	No compara
CNPV,SPN	Normal	CNN	No compara
CNPV,SPV	Normal	CNPV	No compara y devuelve propiedades del primero
CNPV,SUST	Si es el final de la oración	CNN	No compara
CSP,SPN	Normal	CSP	No compara y devuelve propiedades del primero
CSP,SPV	Normal	SPV	No compara y devuelve propiedades del primero
DET	Normal	DET	No compara
DET,SADJ	Normal	SN	Compara primero con segundo y mantiene propiedades del segundo
DET,SUST	Normal	SN	Compara primero con segundo y mantiene propiedades del segundo

INTA,INTV	Si es el final de la oración	PREGUNTA	Compara primero con segundo y no mantiene propiedades
INTA,SV	Si es el final de la oración	PREGUNTA	No compara
INTA,VERBO	Normal	INTV	Compara primero con segundo y mantiene propiedades del segundo
INTA,VERBO	Si es el final de la oración	PREGUNTA	Compara primero con segundo y no mantiene propiedades
INTA,VERBOC	Normal	INTVC	Compara primero con segundo y mantiene propiedades del segundo
INTA,VERBOC	Si es el final de la oración	PREGUNTA	Compara primero con segundo y no mantiene propiedades
INTA,VERBOH	Normal	INTC	No compara y devuelve propiedades del segundo
INTB,SV	Si es el final de la oración	PREGUNTA	No compara
INTB,VERBO	Normal	INTV	Compara primero con segundo y mantiene propiedades del segundo
INTB,VERBO	Si es el final de la oración	PREGUNTA	Compara primero con segundo y no mantiene propiedades
INTB,VERBOC	Normal	INTVC	Compara primero con segundo y mantiene propiedades del segundo
INTB,VERBOC	Si es el final de la oración	PREGUNTA	No compara
INTC,NOM	Si es el final de la oración	PREGUNTAACCION	Compara segundo con primero y devuelve propiedades del segundo
INTC,SN	Si es el final de la oración	PREGUNTAACCION	Compara segundo con primero y une propiedades
INTC	Normal	INTC	No compara
INTV	Normal	INTV	No compara
INTV,ADV	Si es el final de la oración	PREGUNTA	No compara
INTV,CAN	Si es el final de la oración	PREGUNTA	No compara
INTV,CAPV	Si es el final de la oración	PREGUNTA	No compara
INTV,CNN	Si es el final de la oración	PREGUNTA	No compara
INTV,CNPV	Si es el final de la oración	PREGUNTA	No compara
INTV,CSP	Si es el final de la oración	PREGUNTA	No compara
INTV,NOM	Si es el final de la oración	PREGUNTA	No compara
INTV,SADJ	Si es el final de la oración	PREGUNTA	No compara
INTV,SN	Si es el final de la oración	PREGUNTA	No compara
INTV,SPN	Si es el final de la oración	PREGUNTA	No compara
INTV,SPV	Si es el final de la oración	PREGUNTA	No compara
INTV,SUST	Si es el final de la oración	PREGUNTA	No compara
INTV,SV	Si es el final de la oración	PREGUNTA	No compara

INTVC	Normal	INTVC	No compara
INTVC,ADJ	Si es el final de la oración	PREGUNTA	Compara primero con segundo y no mantiene propiedades
INTVC,CAN	Si es el final de la oración	PREGUNTA	Compara primero con segundo y no mantiene propiedades
INTVC,CAPV	Si es el final de la oración	PREGUNTA	Compara primero con segundo y no mantiene propiedades
INTVC,CNN	Si es el final de la oración	PREGUNTA	Compara primero con segundo y no mantiene propiedades
INTVC,CNPV	Si es el final de la oración	PREGUNTA	Compara primero con segundo y no mantiene propiedades
INTVC,NOM	Si es el final de la oración	PREGUNTA	Compara primero con segundo y no mantiene propiedades
INTVC,SN	Si es el final de la oración	PREGUNTA	Compara primero con segundo y no mantiene propiedades
INTVC,SUST	Si es el final de la oración	PREGUNTA	Compara primero con segundo y no mantiene propiedades
INTVC,SV	Si es el final de la oración	PREGUNTA	No compara
NEGACION,VERBO	Normal	VERBO	No compara
NEGACION,VERBOC	Normal	VERBOC	No compara
NOM	Normal	NOM	No compara
NOM,SV	Normal	AFIRMACION	Compara primero con segundo y no mantiene propiedades
PREP	Normal	PREP	No compara
PREP,INTA	Normal	INTA	No compara y devuelve propiedades del segundo
PREP,INTB	Normal	INTB	No compara y devuelve propiedades del segundo
PREP,NOM	Normal	SPN	No compara y devuelve propiedades del segundo
PREP,SADJ	Normal	SPN	No compara y devuelve propiedades del segundo
PREP,SN	Normal	SPN	No compara y devuelve propiedades del segundo
PREP,SUST	Normal	SPN	No compara y devuelve propiedades del segundo
PREP,VERBO	Normal	SPV	No compara
SADJ,CSP	Normal	CAN	No compara y devuelve propiedades del primero
SADJ,SPN	Normal	CAN	No compara y devuelve propiedades del primero

SADJ,SPV	Normal	CAPV	No compara y devuelve propiedades del primero
SALUDO	Normal	SALUDO	No compara
SN	Normal	SN	No compara
SN,ADJ	Normal	SN	Compara primero con segundo y mantiene propiedades del primero
SN,CSP	Normal	CNN	No compara y devuelve propiedades del primero
SN,SPN	Normal	SN	No compara y devuelve propiedades del primero
SN,SPV	Normal	CNPV	No compara y devuelve propiedades del primero
SN,SV	Normal	AFIRMACION	Compara primero con segundo y no mantiene propiedades
SPN	Normal	SPN	No compara
SPN,ADJ	Normal	SPN	Compara primero con segundo y mantiene propiedades del primero
SPN,CSP	Normal	CSP	No compara y devuelve propiedades del primero
SPN,SPN	Normal	SPN	No compara y devuelve propiedades del primero
SPN,SPV	Normal	SPV	No compara y devuelve propiedades del primero
SPV	Normal	SPV	No compara
SPV,ADJ	Si es el final de la oración	SPV	No compara y devuelve propiedades del primero
SPV,ADV	Si es el final de la oración	SPV	No compara y devuelve propiedades del primero
SPV,CAN	Si es el final de la oración	SPV	No compara y devuelve propiedades del primero
SPV,CAPV	Normal	SPV	No compara y devuelve propiedades del primero
SPV,CNN	Si es el final de la oración	SPV	No compara y devuelve propiedades del primero
SPV,CNPV	Normal	SPV	No compara y devuelve propiedades del primero
SPV,CSP	Normal	CSP	No compara y devuelve propiedades del primero
SPV,PREGUNTA	Normal	PREGUNTA	No compara
SPV,SADJ	Si es el final de la oración	SPV	No compara y devuelve propiedades del primero
SPV,SPN	Normal	CSP	No compara y devuelve propiedades del primero
SPV,SPV	Normal	SPV	No compara y devuelve propiedades del primero
SPV,SUST	Si es el final de la oración	SPV	Compara primero con segundo y une propiedades
SUST	Normal	SUST	No compara

SUST,ADJ	Normal	CNN	Compara primero con segundo y mantiene propiedades del primero
SUST,ADV	Si es el final de la oración	CNN	No compara y devuelve propiedades del primero
SUST,CSP	Normal	CNN	No compara y devuelve propiedades del primero
SUST,SPN	Si es el final de la oración	CNN	No compara y devuelve propiedades del primero
SUST,SPV	Normal	CNPV	No compara y devuelve propiedades del primero
VERBO	Normal	VERBO	No compara
VERBO	Si es el final de la oración	SV	No compara
VERBO,ADV	Si es el final de la oración	SV	No compara y devuelve propiedades del primero
VERBO,CAN	Si es el final de la oración	SV	No compara y devuelve propiedades del primero
VERBO,CAPV	Si es el final de la oración	SV	No compara y devuelve propiedades del primero
VERBO,CNN	Si es el final de la oración	SV	No compara y devuelve propiedades del primero
VERBO,CNPV	Si es el final de la oración	SV	No compara y devuelve propiedades del primero
VERBO,CSP	Si es el final de la oración	SV	No compara y devuelve propiedades del primero
VERBO,SADJ	Si es el final de la oración	SV	No compara y devuelve propiedades del primero
VERBO,SN	Si es el final de la oración	SV	No compara y devuelve propiedades del primero
VERBO,SPN	Si es el final de la oración	SV	No compara y devuelve propiedades del primero
VERBO,SPV	Si es el final de la oración	SV	No compara y devuelve propiedades del primero
VERBO,SUST	Si es el final de la oración	SV	No compara y devuelve propiedades del primero
VERBOC	Normal	VERBOC	No compara
VERBOC	Si es el final de la oración	SV	No compara
VERBOC,ADJ	Si es el final de la oración	SV	Compara primero con segundo y une propiedades
VERBOC,CAN	Si es el final de la oración	SV	Compara primero con segundo y une propiedades
VERBOC,CAPV	Si es el final de la oración	SV	Compara primero con segundo y une propiedades
VERBOC,CNN	Si es el final de la oración	SV	Compara primero con segundo y une propiedades
VERBOC,CNPV	Si es el final de la oración	SV	Compara primero con segundo y une propiedades
VERBOC,CSP	Si es el final de la oración	SV	Compara primero con segundo y une propiedades
VERBOC,SN	Si es el final de la oración	SV	Compara primero con segundo y une propiedades

VERBOC,SPN	Si es el final de la oración	SV	Compara primero con segundo y une propiedades
VERBOC,SPV	Si es el final de la oración	SV	Compara primero con segundo y une propiedades
VERBOC,SUST	Si es el final de la oración	SV	Compara primero con segundo y une propiedades