FACULTAD DE INFORMATICA

UNIVERSIDAD COMPLUTENSE DE MADRID

PROYECTO DE SISTEMAS INFORMATICOS

Asistente Virtual (chatbot) para la Web de la Facultad de Informática

Luis Enrique Cubero Caba

Curso 2014-2015

Director:

Juan Pavón Mestras

FACULTAD DE INFORMATICA

UNIVERSIDAD COMPLUTENSE DE MADRID

PROYECTO DE SISTEMAS INFORMATICOS

Asistente Virtual (chatbot) para la Web de la Facultad de Informática

Luis Enrique Cubero Caba

Curso 2014-2015

Director:

Juan Pavón Mestras

El alumno abajo firmante autoriza a la Universidad Complutense de Madrid a difundir y utilizar sólo con fines académicos, no comerciales, mencionando expresamente a su autor, tanto la propia memoria, como el código, los contenidos audiovisuales incluso si incluyen imágenes del autor, la documentación y/o el prototipo desarrollado.

Madrid, 21 de Febrero de 2015.

Luis Enrique Cubero Caba

Agradecimientos

En primer lugar querría agradecer al director de proyecto Juan Pavón Mestras, la ayuda y paciencia prestada durante todo el desarrollo.

Me gustaría también dar las gracias a todas las personas con las que me he ido cruzando a lo largo de la carrera: compañeros, profesores, personal auxiliar. Sin duda alguna me han ayudado a conseguir llegar a este punto.

Por último agradecer a mis amigos y familiares, en especial a mi madre el apoyo ofrecido a los largo de todos estos años.

Resumen

Se ha implementado un asistente virtual cuyo propósito principal es facilitar la búsqueda de información dentro del dominio de la Facultad de Informática de la Universidad Complutense.

El programa es capaz de analizar las peticiones del usuario en lenguaje natural para intentar ofrecer respuestas que satisfagan sus necesidades. Para ello utiliza un analizador de oraciones, capaz de identificar los elementos más relevantes y si alguno de estos pertenece a la red semántica de la web poder dar una respuesta coherente y mostrar el origen de la información. En el caso de que los elementos no estuvieran contemplados, se ayudará de un buscador para ofrecer una serie de resultados, que puedan servir de ayuda.

Dicha funcionalidad está inmersa en un diálogo escrito con el fin de simular el comportamiento de un ser humano y así poder facilitar las búsquedas mejorando la calidad del servicio.

Palabras clave:

Asistente virtual, chatbot, crawler, seeds, indexación, applet, web semántica, gestión de diálogo, widget.

Abstract

A virtual assistant has been implemented whose main purpose is to facilitate the search of information within the website of the Facultad de Informática de la Universidad Complutense.

The program is able to analyze user requests and offer answers that meet their needs. For that, it uses a sentences analyzer, which identifies the most important elements of the sentence, and, if one them is in the semantic group of the web, it can give an appropriately answer and show the origin of the information. On the other hand, if the elements are not identified, another search engine would be used to give the results.

Previous functionality is embedded in a written chat in order to simulate the human behavior and simplify the searches improving the quality of service.

Keywords:

Virtual assistant, chatbot, crawler, seeds, index, applet, semanticgroup, dialogue management, widget.

Tabla de figuras

ÍNDICE

1. Introducción

Un agente conversacional es una herramienta capaz de procesar lenguaje natural y ofrecer información de forma coherente en tiempo real mediante un diálogo. Estas entidades también son conocidas como chatbots.

Según (1) existen varios tipos de agentes conversacionales:

- Asistentes virtuales sociales:
  Principalmente son usados para entretener a cualquier tipo de usuario, sin especializarse en ningún tema concreto.
- Asistentes virtuales educacionales:
  Su propósito principal es ayudar a adquirir conocimiento sobre un determinado tema.
- Asistentes virtuales orientados a servicios:
  Usados frecuentemente en las empresas, y su función es facilitar las búsquedas en su sitio web y resolver preguntas acerca de sus contenidos o servicios.

En particular este proyecto se plantea construir un asistente virtual orientado a servicios, que intentará facilitar las búsquedas y despejar dudas acerca de la Facultad de Informática de la Universidad Complutense.

1.1 Motivación

El proyecto viene motivado por el interés de mejorar el acceso a los contenidos de la página web de la Facultad, mediante la integración de un chatbot en la web, que facilite la interacción con los usuarios.

Actualmente se tiene implementada una función de búsqueda, donde los usuarios deben introducir las palabras que desean buscar.

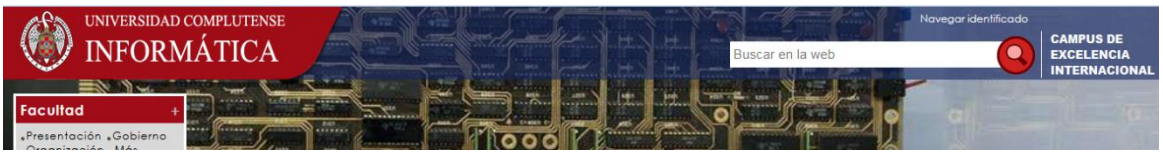

Figura 1.1: Búsqueda en la página web

Una vez el usuario ha introducido la palabra o palabras en las que está interesado se muestra un listado de los posibles resultados.

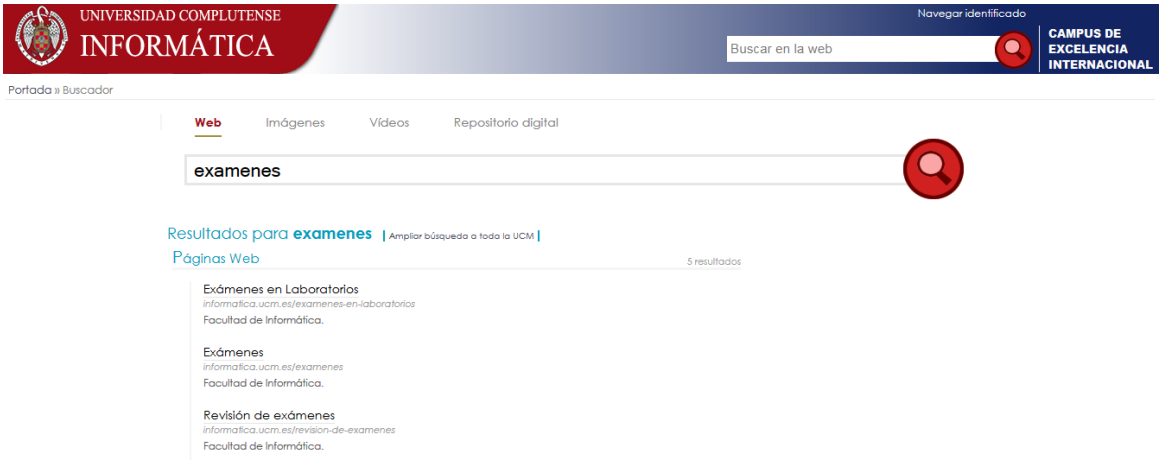

Figura 1.2: Resultados búsqueda página web

Este mecanismo de búsqueda tiene muchas limitaciones, las búsquedas tienen que ser muy precisas, en el ejemplo de la figura anterior, aparecieron una serie de resultados para la palabra exámenes, pero en el momento que se pone “exámenes junio” no ofrece ningún resultado, dejando ver alguna de sus carencias.

Otra prueba realizada fue la búsqueda de palabras que aparecen en la web:

En la dirección http://informatica.ucm.es/presentacion, aparece la palabra “especialización”, pero si se utiliza el buscador, no nos muestra ningún resultado. Se llegó a la conclusión de que la búsqueda no se realiza sobre toda la web, sino que se realiza sobre los elementos principales o sobre una base de datos.

La implantación de un asistente virtual puede suponer:

- Mejora de acceso a los contenidos de la web.
- Consultas en lenguaje natural.
- Reducción del número de llamadas a secretaría puesto que el asistente virtual podría solucionar un mayor número de consultas.
- Detección de las preguntas más repetidas.
- Distintivo respecto a otras universidades.

1.2 Objetivos

Los objetivos principales que persigue el proyecto son:

- Creación de un módulo que se pueda integrar como un complemento web en la página de la Facultad, debe funcionar como un asistente virtual capacitado para resolver las cuestiones de los usuarios en lenguaje natural. En el caso de no disponer de una respuesta en la base de conocimiento deberá realizar una búsqueda. En aquellos casos que se considere útil se deberá de redirigir al usuario para mostrarle el origen de la información.
- Adquisición de experiencia en la gestión, diseño y desarrollo de un proyecto software.
- Investigación de las tecnologías utilizadas en los asistentes virtuales, y justificación de las decisiones tecnológicas tomadas.
- Posibilidad de extender funcionalidad, o mejora de alguno de los componentes.

1.3 Metodología y plan de trabajo

Durante el desarrollo del proyecto se ha seguido una metodología ágil SCRUM.

En una primera fase de investigación, se buscó información acerca de los chatbots, buscadores actuales, y las tecnologías que usaban, viendo cuál era la opción que mejor encajaba en la web y si era viable su implementación.

Posteriormente se empezó con la fase de requisitos donde se definió el alcance del proyecto y a partir de ese momento se empezó a desarrollar prototipos que iban incrementando su funcionalidad, en sucesivas iteraciones (*sprints*).

Antes de comenzar un sprint se realiza una reunión en la que se planifica la iteración, esto consiste en:

- Definir la nueva funcionalidad que se va a incorporar.
- Elaborar la lista de tareas necesarias para la consecución del objetivo.

Dependiendo del esfuerzo de las tareas se fijará la duración del sprint que normalmente suele ser entre 2-4 semanas.
Durante el desarrollo del proyecto se han realizado 11 sprints siendo el más largo de 6 semanas, en el apartado 6 se detallarán cómo ha ido evolucionando el proyecto en cada uno de los sprints.

2. Asistentes Virtuales

A continuación se hará un breve repaso de la evolución los asistentes virtuales hasta llegar a la situación actual.

Los primeros agentes conversacionales que surgieron implementaban el denominado algoritmo de ELIZA(2). ELIZA fue desarrollado entre 1964 y 1966 por Joseph Weizenbaun(3), y su idea principal era analizar la frase y obtener los elementos más relevantes de la misma, para posteriormente reformular la oración y terminar haciendo una pregunta, como si se tratara de un "psicoterapeuta".

Ejemplo:

YOU: Hello
ELIZA: Hi. What seems to be your problem?
YOU: My problem is the darkness
ELIZA: Your problem is the darkness?
YOU: Yes
ELIZA: I understand.

Debido al interés despertado, en el año 1990 se creó un certamen anual que se conoce como Premio Loeb(4). Es una competición basada en el *Test de Turing*.

El *Test de Turing* es una prueba que consta de tres elementos: un juez, y dos evaluados, uno humano y un ordenador. Estos dos últimos se encuentran en una sala que no es visible para el juez, quien será el encargado de conversar con cada uno de ellos.

La tesis de Turing afirma que si los dos evaluados son lo suficientemente hábiles, el juez no podrá decidir cuál de los dos es el humano, y si dicha máquina es capaz de engañar al jurado durante un período de tiempo a determinar, se puede decir que dicha máquina tiene la capacidad de pensar.

Muchas organizaciones se han dado cuenta del potencial y el ahorro que este tipo de sistemas podrían suponer, por lo que han tratado de incorporarlo en sus sectores, uno de sus usos más rentables es el de soporte de primer nivel, esto significa que cuando un usuario trata de comunicarse con la compañía, la primera línea de asistencia es el chatbot, y si estos no fueran capaces de solucionar la incidencia, pasaría al segundo nivel que en muchas ocasiones es el centro de llamadas, donde unos agentes (personas) especializados tratarán de resolver el problema, este centro es conocido como call center.

Debido al rendimiento de la solución han ido surgiendo empresas dedicadas a la programación de los asistentes virtuales, una de estas, es la española Virtual Solutions que afirma en su web (5):

"Una respuesta desde la web podría llegar a **alcanzar el 80% de ahorro** en personal y medios de la empresa."

Figura 2.1: Pirámide de ahorro costes

Como se puede ver en la imagen, la compañía afirma que entre el 75-85% de las peticiones son resueltas por el asistente virtual, y como es de esperar el coste es mucho menor que el tener que ser asistido por un especialista.

En España dos de los chatbots más populares son:

- Irene que ayuda a comprar los billetes en Renfe(6).
- Anna cuya principal labor es asistir al usuario en la compra de muebles en Ikea(7).

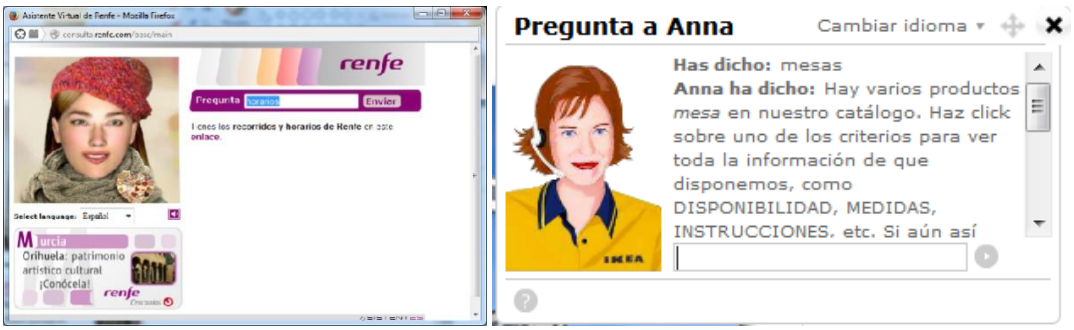

Figura 2.2: Ejemplo Asistentes Virtuales

Uno de los asistentes más populares a nivel mundial es “Siri”, es una aplicación que se utiliza en los dispositivos de Apple, y una de sus principales cualidades es el reconocimiento de voz, desarrollado por la empresa americana Nuance Comunications.

La compañía está especializada en las comunicaciones, pero también desarrolla asistentes virtuales propios. Utilizan un chatbot plantilla que se llama “Nina” y la van adaptando según las necesidades de las compañías que contratan sus servicios, entre las que destacan Coca-Cola (8) y Windstran (9).

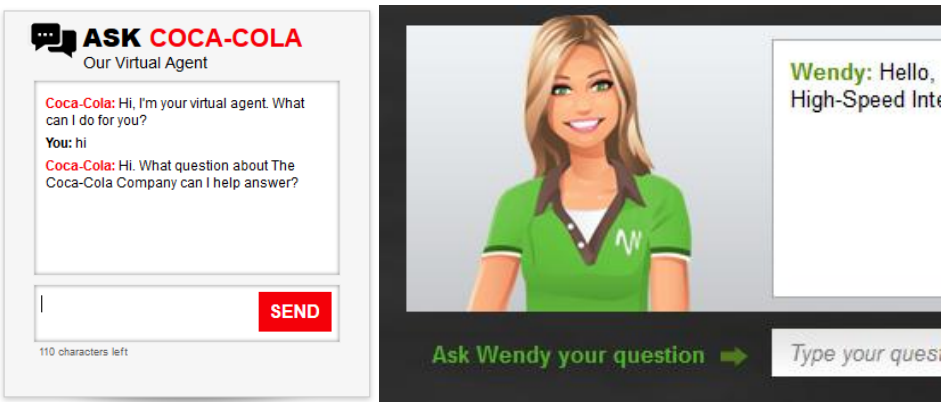

Figura 2.3: Asistentes Nuance Communications

Un asistente virtual que ha sido realizado en un contexto muy parecido al del proyecto, es Elvira, el chatbot de la Universidad de Granada (10), que ha sido desarrollado por Virtual Solutions, previamente mencionado:

Figura 2.4: Asistente Elvira

Elvira tiene una interfaz elaborada con apariencia de ser humano, y trata de dar respuesta acerca de la información de la web de la https://www.ugr.es/.

En muchas ocasiones la interfaz no es lo importante en este tipo de programas, se busca la sencillez y claridad para el usuario, pero no siempre es así, para dar mayor realismo se dota al chatbot de una interfaz humana por medio de la cual puede expresar estados de ánimo(felicidad, enfado , sorpresa…) como es el caso de Elvira.

Algunos incorporan un mecanismo conocido como “Text To Speech” que consiste en un conversor de texto a voz, para proporcionar la información por vía auditiva como complemento del texto.

Pero no sebe olvidar que el propósito de estos asistentes virtuales es resolver dudas.

Uno de los problemas más frecuentes que se encuentran, es en la actualización de productos o de sus páginas web, esta problemática es muy frecuente, y en muchas ocasiones los chatbots no han ido actualizando la información, lo que puede generar algunas incongruencias en la redirecciones a las url´s.

La mayor dificultad que reside en los chatbots es en distinguir cuando una frase es dependiente del contexto o no, cuando se trata de oraciones independientes no se debe buscar la información en el diálogo mantenido, son frases construidas a las que el asistente responde con respuestas preparadas, pero la complejidad comienza cuando se hacen referencias a partes de la conversación, en estos casos la totalidad de los asistentes virtuales pierden el hilo de la conversación, este es un campo que no se abarca en el proyecto, puesto que el objetivo es realizar un buscador de información de la web.

3. Definición del proyecto

El siguiente apartado se verá el alcance del proyecto, y se analizará la funcionalidad del sistema.

Se quiere crear un buscador para la página web de la Facultad de Informática, que a su vez sea un chatbot capaz de mantener una conversación con los usuarios. Por ejemplo el usuario escribe "quiero ver los horarios de los exámenes" el buscador deberá mostrarle un link de una página web de la facultad donde se muestren los horarios de los exámenes.

Idealmente el asistente virtual se debe poder implantar en la infraestructura web de la facultad. Al no tener acceso a los recursos de los servidores web, se optó por una solución no intrusiva, de tal forma que visualmente el módulo está integrado con la web, pero no ha sido necesario modificar el contenido original.

3.1 Descripción de la funcionalidad del sistema

En la fase de análisis se pensaron unos escenarios que ayudaron a definir los casos de uso y la funcionalidad del proyecto, a continuación enumeraremos algunos de ellos:

Escenario 1: Saludo inicial

.Buenos días.
-Hola, bienvenido al asistente virtual de la facultad de informática
.¿Cómo te llamas?
-Me llamo Nerdi, ¿En qué puedo ayudarte?

Escenario 2: Pregunta típica

.Oye, ¿Cuándo son los exámenes?
-A continuación te muestro el calendario de los exámenes. (Enlace exámenes)
.¿Pero el día 6 no es fiesta?
-Este es el calendario académico, aquí puedes ver si tienes clase (Enlace calendario académico).

Escenario 3: Pregunta típica

.Hola soy un alumno nuevo de la facultad, y tengo algunas dudas.
-Muy bien, voy a tratar de ayudarte si quieres empezamos a solucionar tus dudas.
.Me gustaría saber si la facultad tiene algún equipo de futbol.

-Si claro, si necesitas más información puedes escribir al club deportivo de la facultad, cuyo correo es: clubdeportivo.informatica@gmail.com.

Escenario 4: Insulto

.Hola
-Buenas mi nombre es Nerdi, y soy el asistente de la pagina web de la facultad.
.¿Nerdi?
-Dime.
.Pareces un poco friki.
-¿Por qué lo dices, porque estoy programando?
.No, lo digo porque pareces tonto.
-¿Por qué no vas contárselo a tus amigos?
.Eres #%&/·$/·$!!
-Modera tu lenguaje.

Escenario 5: Origen chatbot

.Buenas noches.
-Hola buenas noches, ¿tienes alguna consulta?
.¿Quién es tu creador?
-Mi creador es Luis Cubero.
.¿En qué lenguaje estas programado?
-Para programarme han utilizado JavaScript, PHP, MySQL, AIML.
.¿Qué es aiml?
-AIML es un lenguaje de etiquetas que se utiliza para programar chatbots, si quieres puedes encontrar más información en el siguiente enlace.

Escenario 6: Preguntas personales

.¿Cómo te llamas?
-Me llamo Nerdi.
.¿Cuántos años tienes?
-Pues aunque parezca mentira tengo 45 días, ¿Tienes alguna pregunta?
.¿Y qué sabes hacer?
-Soy el asistente de la Facultad de Informática, ayudo a realizar búsquedas, e intento resolver las dudas de los usuarios.
.¿Ahora mismo que estás haciendo?.
-Estoy terminando de hacer un programa, estoy un poco liado. ¿Tienes alguna pregunta?.

Escenario 7: Búsqueda Profesor

-Quería información sobre "Profesor".
.Su horarios es el de L-V de 18:00-20:00 y su despacho es el 345, y no olvides mandar un correo para concertar la tutoría (horarios profesores).

Escenario 8: Búsqueda Decano

.Me gustaría hablar con el decano.
-Te voy a facilitar que conozco del Decano de la Facultad. (Redirección información decano).

Escenario 9: Fin de conversación

-Muchas gracias.
.¿Necesitas realizar alguna consulta más?
-No, me voy a ir.
.Gracias por utilizarme, espero que haya sido útil.
-Adiós.
.Que pases un buen día.

Escenario 10: Búsqueda

-Me gustaría buscar una palabra.
.Dime.
-MIGS.
.He encontrado unos resultados quizá puedan ayudarte.

3.2 Especificación de requisitos

3.2.1 Requisitos funcionales

REQ01 Respuesta directa
En el caso de detectar una frase construida se responderá de forma directa

REQ02 Base de conocimiento
El sistema tendrá un conjunto de palabras claves, que están enlazadas con url's, y se utilizarán para redirigir al usuario a dicha página web, en caso de que la palabra sea identificada.

REQ03 El sistema tiene un buscador de la página web
En caso de que la información no esté en la web semántica definida se deberá buscar la información en los contenidos de la web, y en caso de encontrar alguna coincidencia se mostrarán todos los resultados con una breve descripción.

REQ04 El sistema tiene un sistema generador de respuestas.
Se proporcionará diferentes respuestas a la misma pregunta lo que aporta una mayor sensación de conversación real.

REQ05 Redirección url
En el caso de tener la información solicitada por el usuario se le deberá redirigir a la url donde aparece la reseña.

REQ06 Independencia del uso de acentos.

REQ07 Independencia de la morfología del término para las búsquedas
No tendrá relevancia si un término aparece en singular o plural o un verbo aparece conjugado o en infinitivo.

REQ08 Trazabilidad de las respuestas
El sistema guardará en un log las preguntas realizadas por los usuarios.

REQ09 Distinción entre búsqueda y chat
Se debe saber distinguir cuando el usuario está realizando una búsqueda, o cuando simplemente está conversando.

3.2.2 Requisitos no funcionales

REQ10Usabilidad
La interfaz debe ser sencilla e intuitiva, el usuario no debería encontrar dificultades en el uso del software por primera.

REQ11 Estándar gráfico
El diseño gráfico debe encajar con el estilo de la web de la Facultad.

REQ12 Concurrencia de usuarios
El sistema debe ser capaz de dar soporte a más de 30 usuarios a la vez, sin afectar su rendimiento.

REQ13 Actualización conocimiento
Muchos problemas surgen cuando la web y el asistente virtual no están sincronizados, por ese motivo, se tiene que poder añadir o eliminar información a la base de conocimiento de forma sencilla, semanalmente se creará una tarea que buscará nueva información en la web para actualizar el buscador.

REQ14 Multiplataforma
La aplicación podrá ser ejecutada desde cualquier entorno que tenga la posibilidad de ejecutar JavaScript, entre los cuales se encuentran los 3 navegadores más frecuentes, Internet Explorer, Firefox y Chrome, incluso se podrá llegar a ejecutar desde tablets y móviles que soporten la tecnología.

REQ15 Estabilidad
El sistema tiene que ser robusto y estar operativo 24hx7d.

REQ16 Idioma en castellano
Puesto que la web de la Facultad únicamente está en castellano no se ha contemplado que el sistema trabaje con otros idiomas.

REQ17 Extensibilidad
El sistema debe estar diseñado para que en el futuro se puedan eliminar, mejorar o incluso cambiar la funcionalidad de los componentes principales.

REQ18 Tiempo de respuesta
El sistema debe tener un tiempo de respuesta máximo de 5 segundos en todo tipo de consultas.

3.3 Casos de uso

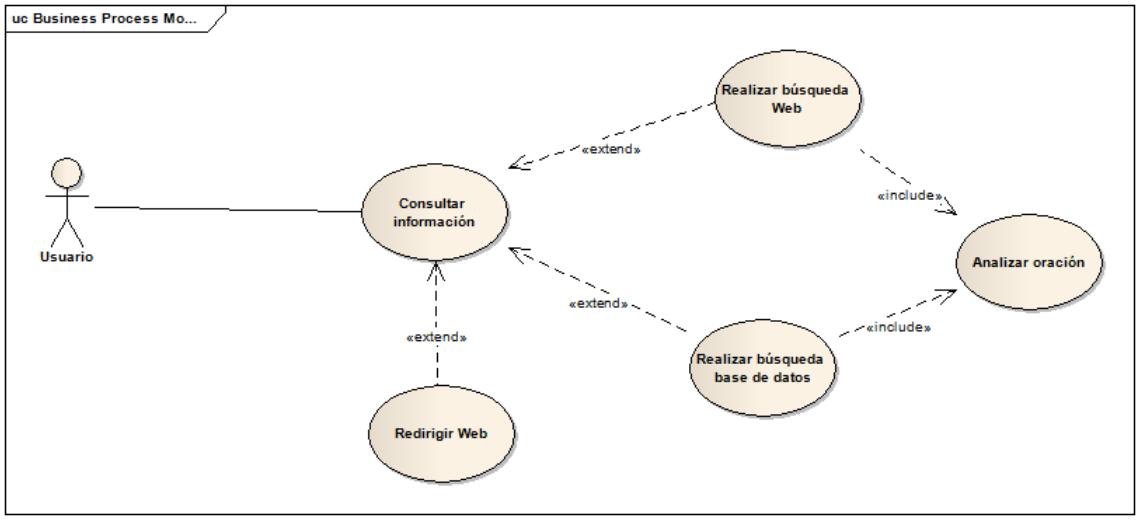

Figura 3.1: Diagrama casos de uso Usuario

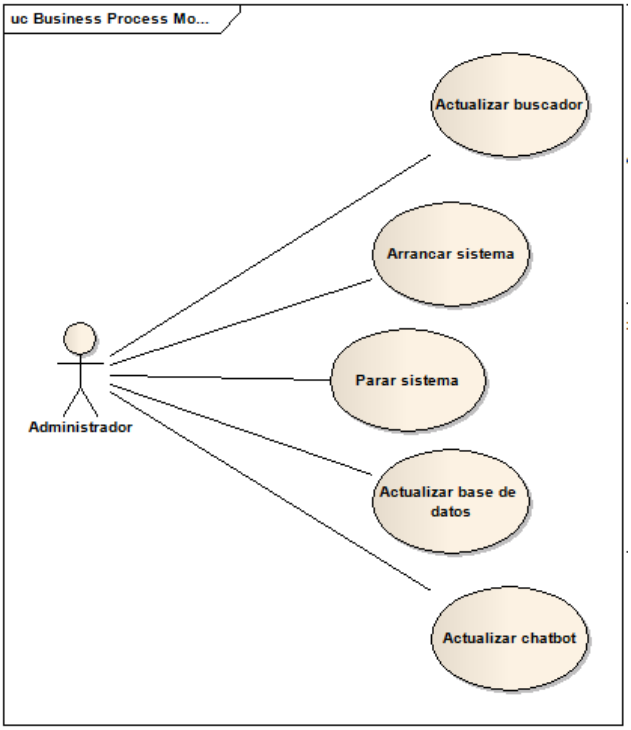

Figura 3.2: Diagrama casos de uso Administrador

CU-01	**Arranque sistema**	
Versión	2.0 (05/06/2015)	
Dependencias		
Precondición	Servidor encendido, los servicios deberán estar puestos en automáticos, para que cuando el servidor se reinicie, empiecen a levantarse sin intervención de ningún técnico, en el caso de haberlos parado manualmente seguir secuencia.	
Descripción	Se deberán seguir los siguientes pasos para arrancar el sistema	
Secuencia normal	Paso	Acción
	1	Arrancar el servicio de Apache
	2	Arrancar el servicio MySQL
	3	Arrancar el servicio Tomcat
	4	Comprobar su correcto funcionamiento
Postcondición	Se puede utilizar correctamente la aplicación	
Excepciones	Paso	Acción
	1	Si existiera algún problema se deberá revisar el log, en la ruta Xampp\apache\logs\error.log
	2	Si existiera algún problema xampp\mysql\data\mysql_error.log
	3	Si existiera algún problema tomcat\logtomcat7-stderr.log

CU-02	**Parada sistema**	
Versión	2.0 (05/06/2015)	
Dependencias	Arranque Sistema	
Precondición	Para realizar una parada del sistema, se deberá hacer en horario no crítico, recomendablemente por la noche.	
Descripción	Se deberán seguir los siguientes pasos para parar el sistema	
Secuencia normal	Paso	Acción
	1	Modificar enlace al chatbot, para que salga una alerta de mantenimiento.
	2	Si hay usuarios utilizando la aplicación les saltará el mensaje, y dejarán inmediatamente de utilizar el asistente.
	3	Parar servicios de forma controlada
Postcondición	El sistema no estará operativo, hasta que se arranque de nuevo, los usuarios no podrán interactuar con el chatbot	

CU-03	**Inicio asistente virtual**	
Versión	1.0 (05/06/2015)	
Dependencias		
Precondición	El sistema debe estar arrancado	
Descripción	Se deberán seguir los siguientes pasos para iniciar sesión en el sistema	
Secuencia normal	Paso	Acción
	1	Acceso url asistente virtual
	2	Carga de elementos web
	3	Carga interfaz del asistente
	4	Inicio sesión en Tomcat, a partir de una consulta web que devolverá una cookie JSesionID
	5	Cacheo de la cookie
Postcondición	El navegador web guardará la cookie solicitada y se mostrará la interfaz.	
Excepciones	Paso	Acción
	2	Si el sistema no puede ejecutar JavaScript, no se mostrará la aplicación y aparecerá el siguiente mensaje: "Lo sentimos, pero su navegador no soporta la tecnología necesaria para el funcionamiento del Asistente Virtual."

		E.1	Utilizar Firefox, IE, o Chrome
	4	No se devuelve JSesionID, aparece un mensaje:"En estos momentos no se pudo conectar al asistente virtual , pruebe más tarde, sentimos las molestias"	
		E.1	El usuario debe esperar, a que se restaure el funcionamiento del Tomcat

CU-04	**Pedir respuesta predefinida chatbot**	
Versión	1.0 (05/06/2015)	
Dependencias		
Precondición	Inicio asistente virtual	
Descripción	El sistema deberá comportarse como se describe en el siguiente caso de uso cuando la información solicitada está disponible por el chatbot.	
Secuencia normal	Paso	Acción
	1	El usuario escribe en el cuadro de diálogo.
	2	Detectar la pulsación de "Enter" o botón.
	3	Se envía síncronamente la petición mediante HTTP al servlet Tomcat.
	4	Se obtiene la respuesta.
Postcondición	Se muestra respuesta al usuario.	
Excepciones	Paso	Acción
	4	En el caso de que el componente chatbot no responda continuará la ejecución del programa, analizando la oración.

CU-05	**Respuesta en la base de datos**	
Versión	2.0 (05/06/2015)	
Dependencias		
Precondición	CU-04 no encuentra respuesta.	
Descripción	El sistema comprobará si aparece alguna palabra clave, y responderá al usuario.	
Secuencia normal	Paso	Acción
	1	Se analizará la oración, identificando los elementos clave.
	2	Búsqueda de elementos en la base de datos.
	3	Obtención de url de aparición en la web, y palabra clave AIML.
	4	Petición síncrona HTTP al servlet.
	5	Se obtiene respuesta.
Postcondición	Respuesta al usuario y se le redirige a la url obtenida.	
Excepciones	Paso	Acción
	1	Si el módulo de análisis no responde, se pasará al buscador.
	2	Si la base de datos no responde, se pasará al buscador, pero únicamente se buscarán los elementos principales de la oración.

CU-06	**Búsqueda web**	
Versión	2.0 (05/06/2015)	
Dependencias		
Precondición	CU-05 no encuentra palabras clave, error en base de datos o en el analizador.	
Descripción	El sistema realizará una búsqueda.	
Secuencia normal	Paso	Acción
	1	Se obtendrán los elementos clave, ya calculados previamente.
	2	Envío petición HTTP al servlet Solr.
	3	Se muestran las búsquedas encontradas.
Postcondición	Redirección a la página de búsquedas.	

CU-07	**Cierre sesión usuario**	
Versión	1.0 (05/06/2015)	
Dependencias		
Precondición	Inicio sesión.	
Descripción	El sistema cerrará las sesiones abiertas con más de 30 minutos sin interacciones.	
Secuencia normal	Paso	Acción
	1	Detención usuarios con más de 30 minutos conectado sin interactuar.
	2	Se cerrará la sesión.
	3	En el caso de volver se iniciara el CU-3.
Postcondición	Eliminación sesión en Tomcat.	

CU-08	**Búsqueda Profesor**	
Versión	1.0 (05/06/2015)	
Dependencias		
Precondición	Inicio sesión.	
Descripción	El sistema mostrará información acerca del profesor.	
Secuencia normal	Paso	Acción
	1	Identificación de profesor.
	2	Se revisará la oración en búsqueda de palabras clave, como tutoría, despacho, horario.
	3	Se redirigirá a la url de profesores y tutorías.
Postcondición	Devolución de información.	

CU-09	**Actualizar chatbot**	
Versión	2.0 (05/06/2015)	
Dependencias		
Precondición	Asistente virtual completamente operativo.	
Descripción	Se deberán seguir los siguientes pasos para actualizar la base de conocimiento del chatbot.	
Secuencia normal	Paso	Acción
	1	Modificar enlace el asistente virtual, para que salga una alerta de mantenimiento.
	2	Añadir los archivos los nuevos archivos .aiml, en la ruta correspondiente(“C:\Program Files\Apache Software Foundation\Tomcat 7.0\webapps\programd\resources\testing”)
	3	Reiniciar el servicio Tomcat.
	4	Comprobar su correcto funcionamiento.
	5	Restablecer enlace.
Postcondición	Se puede utilizar correctamente la aplicación, con los nuevos conocimientos.	
Excepciones	Paso	Acción
	4	Si existiera algún problema tomcat\logtomcat7-stderr.log.
	4	Comprobar la compilación del nuevo código aiml.

CU-10	**Actualizar base de datos**	
Versión	2.0 (05/06/2015)	
Dependencias		
Precondición	Asistente virtual completamente operativo.	
Descripción	Se deberán seguir los siguientes pasos para actualizar la base de datos del asistente virtual. No es necesario parar la instancia SQL.	
Secuencia normal	Paso	Acción
	1	Realizar una copia de seguridad.
	2	Ejecutar el script ".sql", y comprobar resultado devuelto.
	3	Comprobar su correcto funcionamiento.
Postcondición	Se puede utilizar correctamente la aplicación, con los nuevos conocimientos.	
Excepciones	Paso	Acción
	3	En caso de pérdida de la base de datos cagar la copia de seguridad.

CU-11	**Actualizar buscador manualmente**	
Versión	2.0 (05/06/2015)	
Dependencias		
Precondición	Asistente virtual completamente operativo.	
Descripción	Se deberán seguir los siguientes pasos para actualizar la base de datos del asistente virtual. No es necesario parar la instancia SQL.	
Secuencia normal	Paso	Acción
	1	Ejecutar el crawler.
	2	Modificar enlace el asistente virtual, para que salga una alerta de mantenimiento.
	2	Renombrar carpeta contenedora de la indexación antigua.
	3	Pegar los nuevos archivos del buscador, con el nombre que tenían anteriormente.
	4	Comprobar su correcto funcionamiento.
Postcondición	Se puede utilizar correctamente la aplicación, con los nuevos conocimientos.	
Excepciones	Paso	Acción
	4	En caso de fallo de funcionamiento, reinicio del servicio de Tomcat, y si persiste el fallo, volver al estado anterior utilizando los archivos antiguos.

4. Tecnología utilizada

En este apartado se detalla la tecnología utilizada para la realización del proyecto, finalizando con un resumen de las distintas versiones empleadas.

4.1 Chatbot

En la introducción se explicó qué son los chatbots y las funciones que realizan, a continuación describiremos la tecnología utilizada para desarrollar el proyecto.

Un asistente virtual se puede implementar en los lenguajes más comunes de programación como pueden ser Java, C, Python,…, pero estos lenguajes no están especializados en procesamiento de lenguaje, y menos en programación de chatbots.

En este apartado la decisión tecnológica fue fácil, el lenguaje de programación para chatbots más extendido es AIML(**Artificial Intelligence Mark-up Language**) se pueden encontrar numerosas comunidades con tutoriales y ejemplos. Es un lenguaje muy básico pero muy potente, está basado en XML, y su propósito es crear una base de conocimiento para dar respuesta mediante patrones.

4.1.1 Funcionamiento de AIML

A continuación se explica brevemente. Los tres componentes básicos son:

- **Categorías**
  Son las unidades fundamentales de conocimiento, es la etiqueta que marca el inicio de una regla de conocimiento que contiene al menos un patrón y una plantilla.
  Etiqueta: <category>

- **Patrones**
  El lenguaje AIML se basa en patrones. La entrada de texto es comparada con los patrones, estos son expresiones en lenguaje formal que puede contener:
  1. Lenguaje natural en mayúsculas.
  2. El símbolo “*” que equivale a una o más palabras.
  3. El símbolo “_”es igual que “*” pero prevalece.
  4. <name/> que es reemplazado cuando se carga por el nombre del chatbot.

  Etiqueta: <pattern>
- **Plantillas**
  En caso de coincidir el patrón, la respuesta del chatbot se sitúa en la plantilla. Dentro de la plantilla se pueden añadir otras etiquetas para personalizar las respuestas, veremos algunas de ellas más adelante.
  Etiqueta: <template>

Ejemplo 1: (Básico)

```
<category>
        <pattern>HOLA</pattern>
        <template>
        Bienvenido, espero poder ayudarte.
        </template>
</category>
```

En este caso si el usuario enviara el mensaje de "Hola", el chatbot respondería con "Bienvenido, espero poder ayudarte."

Tú: Hola
BOT: Bienvenido, espero poder ayudarte.

A partir del ejemplo se pueden ir complicando con etiquetas que aportan otras funcionalidades como son:

"<star/>", esta etiqueta es sustituida por las palabras que coinciden con "*"
"<srai>X</srai>" llama recursivamente al patrón que coincide con X
"<sr/>" = <srai><star/></srai>

Con estos nuevos elementos podemos ver que se pueden crear ejemplos más complejos.

Ejemplo 2: (Recursividad)

El Input del usuario es:
-Esto es una prueba para Nerdi

Paso	Input	pattern	template	respuesta
1	ESTO ES UNA PRUEBA PARA NERDI	_ <name/>	<sr/>	
2	ESTO ES UNA PRUEBA PARA	_ PARA	Ejemplo 2 <sr/>	Ejemplo 2
3	ESTO ES UNA PRUEBA	ESTO ES *	<sr/>	Ejemplo 2
4	UNA PRUEBA	* PRUEBA	Fin	Ejemplo 2 Fin

Partiendo de que el nombre del chatbot es Nerdi la recursión iría realizando las siguientes llamadas:
Paso 1:<sr/>=ESTO ES UNA PRUEBA PARA
Paso2: <sr/>=ESTO ES UNA PRUEBA
Paso3:<sr/>=UNA PRUEBA
La respuesta final es:"Ejemplo 2 Fin"

Otras etiquetas bastante prácticas son:

"<that></that>", se utiliza como un patrón pero de lo que dijo previamente el chatbot, frecuentemente se utiliza para preguntas.

Ejemplo 3:

```
<category>
        <pattern>TOCK TOCK</pattern>
        <template>Quién es?</template>
</category>
<category>
        <pattern>*</pattern>
        <that>¿Quién es?</that>
        <template>Sabía que eras tú, <star/></template>
</category>
```

Diálogo:

-Tock tock
.¿Quién es?
-Luis
.Sabía que eras tú, Luis

**"<random>
</random>",** permite responder aleatoriamente una de las respuestas entre las etiquetas

Ejemplo 4:

```
<category>
        <pattern>¿QUÉ TAL TE ENCUENTRAS HOY?</pattern>
        <template>
                <random>
                        <li>Bien </li>
                        <li>Regular </li>
                        <li>Mal </li>
                </random>
        </template>
</category>
```

Dialogo:

-¿Qué tal te encuentras hoy?
.Bien
-¿Qué tal te encuentras hoy?
.Mal
-¿Qué tal te encuentras hoy?
.Bien

En la siguiente dirección http://www.alicebot.org/documentation/aiml-reference.html#random se encuentran todas las etiquetas que se pueden utilizar para programar, se han explicado las que se consideran básicas para entender el funcionamiento de AIML.

4.2 Motor de Búsquedas

Un motor de búsqueda es un software diseñado para encontrar recursos digitales a partir de unas palabras clave, estos recursos pueden ser páginas web, documentos, textos, imágenes, videos…

Normalmente se aplica el concepto a Internet, pero hay muchas aplicaciones para buscar documentos en los equipos personales de los usuarios, también son usados por las empresas para clasificar su información. Por ejemplo el Microsoft Project es un gestor de proyectos que tiene implementado un motor de búsqueda.

La estructura básica de los buscadores es común para todos ellos:

-Crawler: también conocido como araña, son programas encargados de recorrer toda la web registrando toda la información y la van añadiendo a una base de datos, este proceso se conoce como indexación de información.

Uno de los crawlers más importantes es el de Google Googlebot(11) que tiene dos versiones:

- DeepBot: es una araña que intenta profundizar e ir buscando dominios nuevos a partir de referencias a otras webs
- FreshBot: se encargar de refrescar la información, ideal para periódicos o webs que añaden continuamente información.

Índice: es la base de datos, donde se almacena toda la información recogida por el crawler, puede almacenar distintos tipos de contenido un ejemplo básico sería el siguiente:

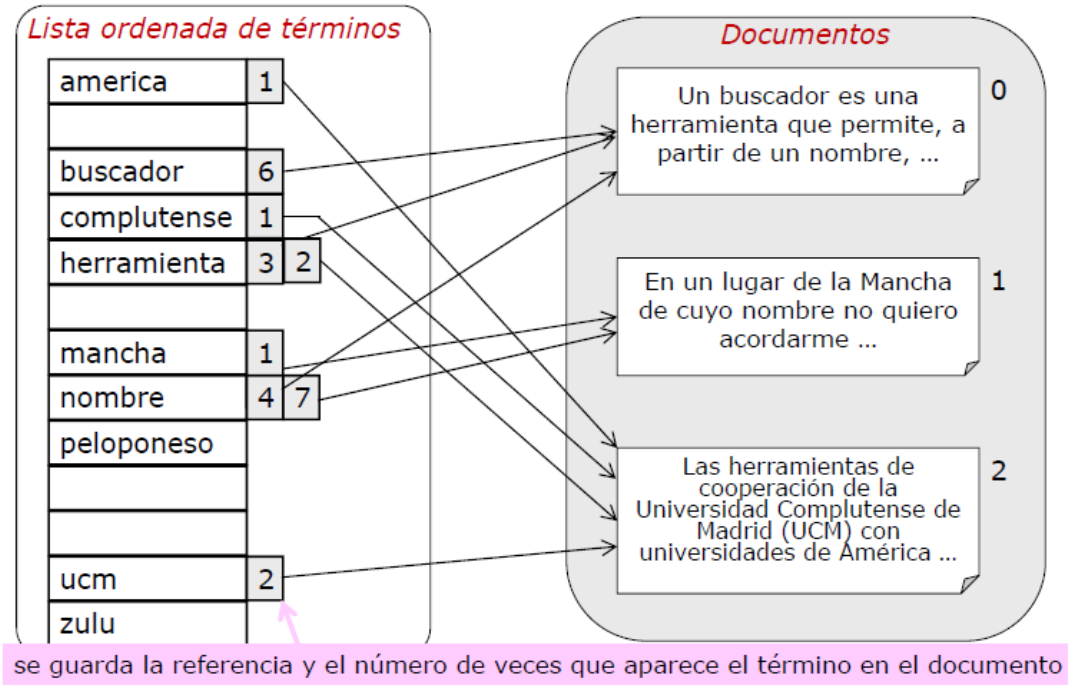

Figura 4.1: Ejemplo Indexación (12)

En este caso es un índice generado a partir de unos documentos de texto, y almacena el número de veces que aparece la referencia junto con un puntero al documento, de esta forma, si se quiere buscar una palabra se podrían recuperar los documentos en los que aparece ordenados según el número de referencias.

Para el desarrollo del proyecto se ha utilizado la tecnología Lucene, que es un producto desarrollado por Apache, lo que significa que es de código abierto.

El crawler hace uso de un producto llamado Nutch y para realizar las búsquedas Solr, se complementan a la perfección puesto que trabajan con la base Lucene. Una de las principales ventajas de Solr es su integración con todo tipo de lenguajes, para el proyecto se ha usado el formato JSON (es ligero y se utiliza para el intercambio de datos a través del protocolo HTTP) siendo una opción recomendable puesto que no necesita la instalación de ningún componente.

En el apartado numero 7 Instalación y ejecución detallaremos como configurarlos correctamente.

4.3 Hosting

Para empezar se va a explicar la diferencia entre servidor web y el contenedor web(servlet):

-**Servidor web**: Es una máquina que guarda el contenido estático de la web y se encarga de comunicar los datos al cliente a través del protocolo HTTP (**HyperText Transfer Protocol**)
-**Contenedor Web**: Es un entorno aislado dentro del servidor web, donde se ejecuta un programa Java con el fin de ofrecer contenido dinámico a los clientes. Los servlets se ejecutan en el entorno de ejecución proporcionado por el contenedor web, mediante el uso del motor JSP(Java ServletPages).

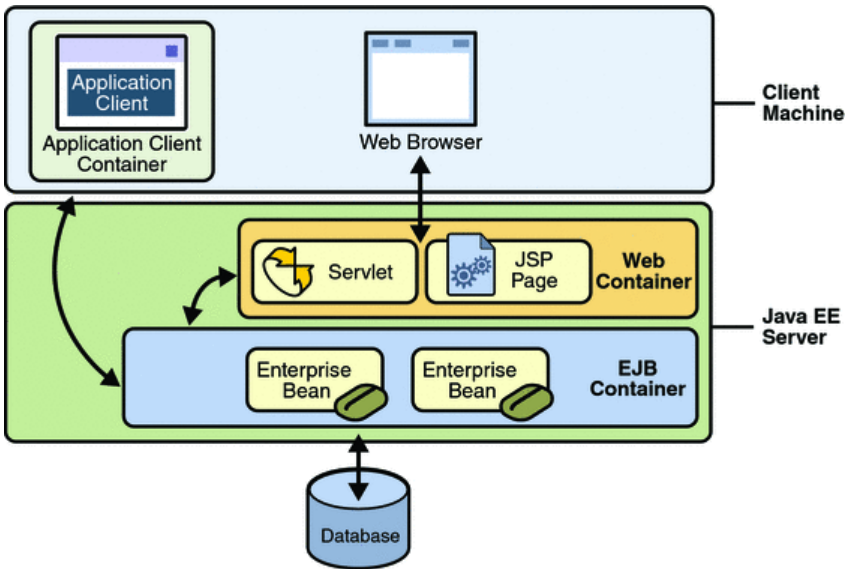

Figura 4.2: Arquitectura Servlet (13)

Para el proyecto se ha utilizado el paquete de XAMP que ofrecía servidor web, Tomcat (Contenedor Web) junto con MYSQL y PHP.

Uno de los motivos de su elección fue su integración con el resto de componentes puesto que es de código abierto, y existen muchas guías.

4.4 Applet Vs JavaScript-HTML

Al comienzo del proyecto se decidió realizar un applet puesto que se dominaba el lenguaje de programación Java, pero según iba avanzando el desarrollo se empezaron a ver las limitaciones que ofrecía, y finalmente se decidió cambiar la tecnología utilizada y se empezó a desarrollar en JavaScript.

Principales defectos de los Applet frente a JavaScript:

- Los tiempos de carga son mucho más lentos.
- No se pueden mezclar los elementos con los componentes web.
- Petición de permisos de ejecución.

JavaScript hoy en día es la solución más estándar en el desarrollo de aplicaciones cliente para el navegador, existen gran cantidad de librerías que facilitan el desarrollo de aplicaciones, jQuery es una de las que se ha utilizado, ofrece numerosas funciones que simplifican la programación web, una de sus funcionalidades es el tratamiento de las interacciones AJAX (Asynchronous JavaScript And XML) utilizadas para las peticiones HTTP.

4.5 Resumen herramientas

Las versiones de las herramientas que se han utilizado son las siguientes:

- Tomcat 7.0 (Contenedor de servlets)
- Solr 4.10.1 (Motor de búsqueda)
- Nutch-2.2.1 (Crawler)
- ProgramD 4.6 (Chatbot)
- Xamp V 3.1.0 (Servidor Web)

5. Arquitectura utilizada

A continuación se explica cuál es la estructura y funcionamiento del asistente virtual y cada una de las partes que lo componen. Para ello se va a describir la visión estática del sistema y posteriormente la visión dinámica, donde se explica cómo interaccionan esas partes para llevar a cabo su cometido.

5.1 Visión estática

5.1.1 Visión general del sistema

En la siguientes figuras se muestra las partes en las que está distribuido el sistema, la clase Asistente Virtual contiene el programa principal, que se encarga de ensamblar los módulos para que trabajen conjuntamente, cada parte es independiente por lo que se pueden realizar pruebas individualmente obteniendo un resultado, en el caso de las peticiones HTTP se han realizado a través de una aplicación de Google Chrome ”Postman –REST Client” que simula el envío de peticiones y devuelve un resultado en el caso de que existiera, se mostrarán las simulaciones cuando se vayan explicando cada uno de los componentes.

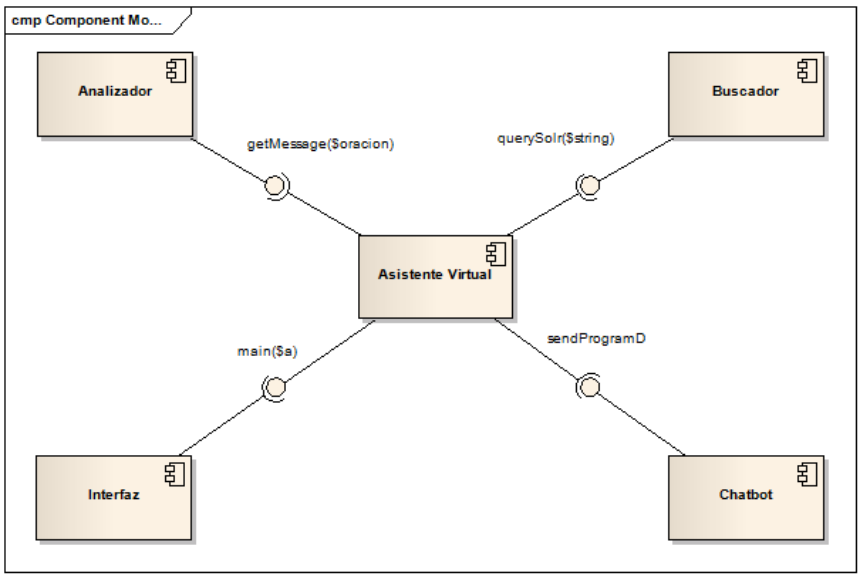

Figura 5.1: Visión general del sistema

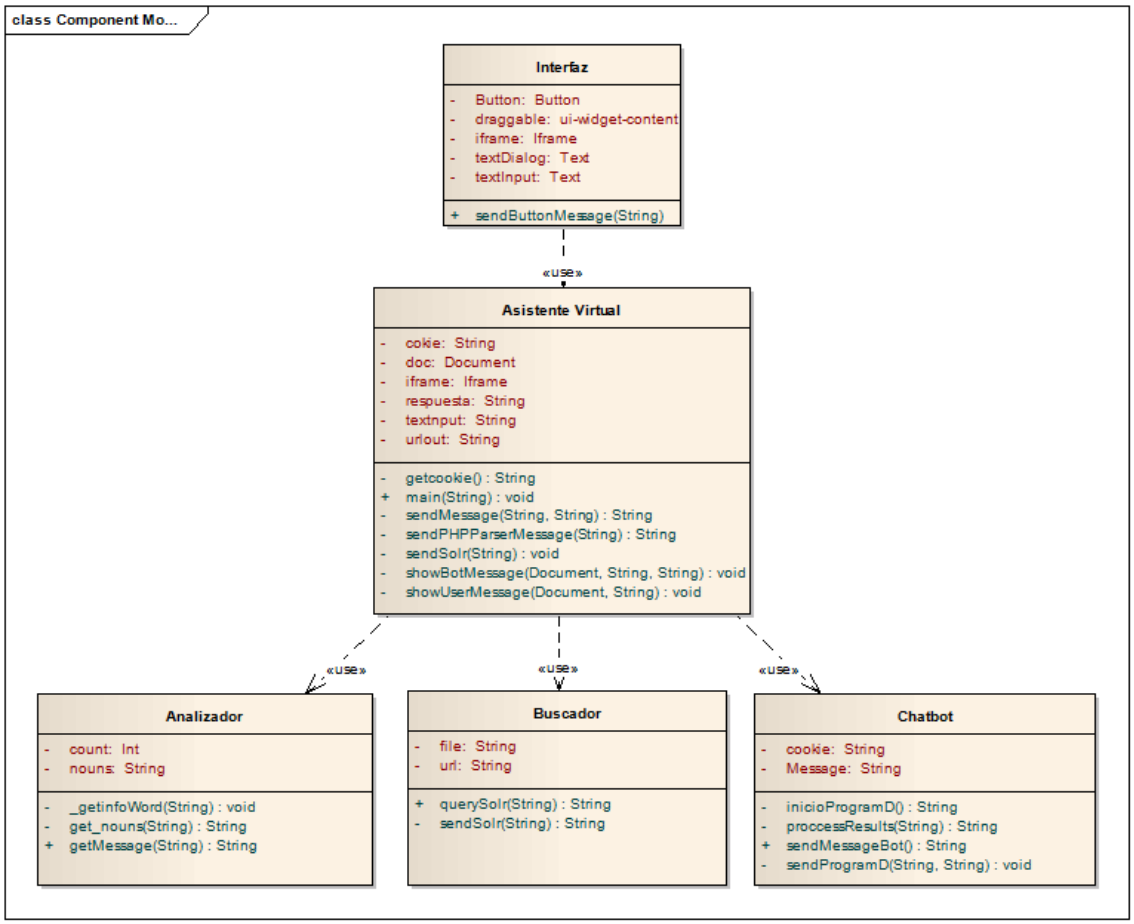

Figura 5.2: Diagrama de clases global

5.1.2 Elementos sistema

A continuación se realizará un análisis de la función de los módulos del sistema.

Analizador

El propósito del analizador es la identificación de elementos clave en las peticiones del usuario. Se considera elemento clave cualquier palabra que sea un verbo, un sustantivo, o si aparece en la base de conocimiento.

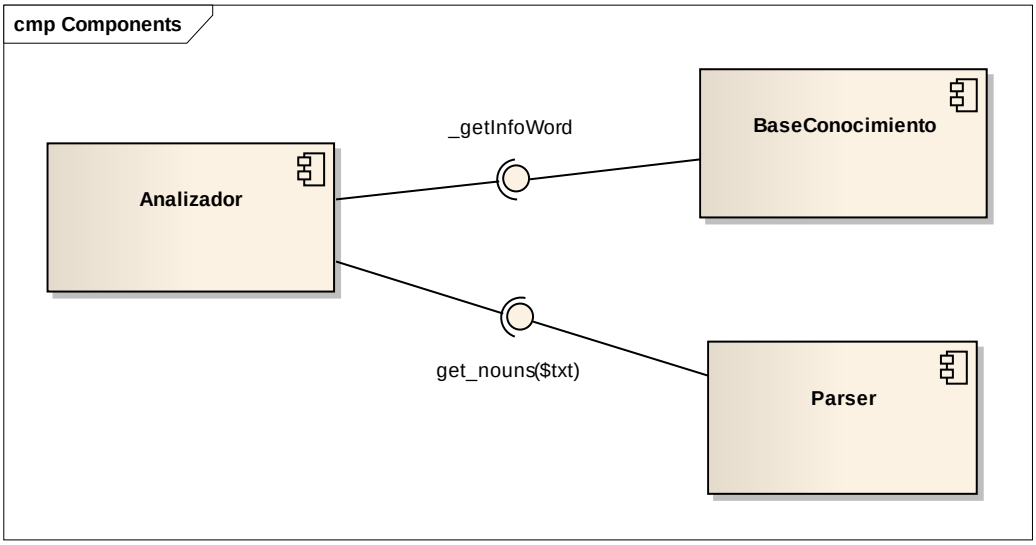

Figura 5.2: Diagrama componentes Analizador

Para el Parser se ha utiliza un recurso externo, que su función es realizar un análisis de la oración y devuelve la categoría gramatical de las palabras a parte de otro tipo de información como son el lema y los rasgos de las palabras(en el caso de los verbos informa sobre su conjugación verbal).

Este recurso se llama Grampal(14), pertenece a la UAM(Universidad Autónoma de Madrid). Se ha integrado en el proyecto mediante una petición GET realizada desde el servidor para obtener el código fuente y la instrucción es la siguiente:

$html = file_get_contents($url);

Donde la variable url es la ruta de la web, más los elementos de la oración:

Un ejemplo de una posible petición es "quiero saber cómo te llamas " la url sería: "http://cartago.lllf.uam.es/grampal/grampal.cgi?m=etiqueta&e=quiero%20saber%20como%20te%20llamas"

Y su resultado es:

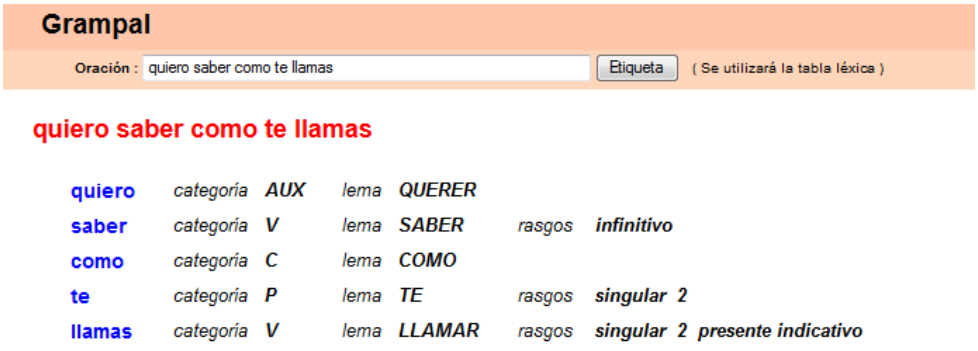

Figura 5.3: Ejemplo gráfico Grampal

La clase se encarga de procesar la información del código fuente y obtener los nombres (N) y verbos (V)

Ejemplo de parte del código fuente obtenido:

```
<td style="font-size:110%;color:blue;font-weight:bold">saber</td>

<td style="font-style:italic;">categoría   <b>V</b></td>


<td style="font-style:italic;">lema   <b>SABER</b></td>


<td style="font-style:italic;">rasgos   <b> infinitivo </b></td>


</tr>


<tr>


<td style="font-size:110%;color:blue;font-weight:bold">como</td>

<td style="font-style:italic;">categoría   <b>C</b></td>


<td style="font-style:italic;">lema   <b>COMO</b></td>


<td></td>


</tr>


<tr>


<td style="font-size:110%;color:blue;font-weight:bold">te</td>

<td style="font-style:italic;">categoría   <b>P</b></td>


<td style="font-style:italic;">lema   <b>TE</b></td>


<td style="font-style:italic;">rasgos   <b> singular  2 </b></td>


</tr>
```

Figura 5.4: Ejemplo código fuente Grampal

Respecto a la base de conocimiento de las palabras clave, esta implementada mediante una sencilla base de datos en MySQL.

Que contiene las palabras relacionadas con una dirección url, donde se redirigirá al usuario y una palabra clave AIML, que será la palabra enviada al módulo del Chatbot para dar respuesta.

Ejemplo tabla palabras:

id	palabra	direccion
1	gobierno	3
3	organización	4
4	introducción	5
5	resumen	5
6	localizacion	6
7	autobus	6
8	metro	6

Figura 5.5: Ejemplo DB palabras

Ejemplo tabla direcciones:

id	url	respuesta
1	http://informatica.ucm.es/	ucm
3	http://informatica.ucm.es/gobierno	gobierno
4	http://informatica.ucm.es/departamentos_1	organizacion
5	https://informatica.ucm.es/presentacion	presentacion
6	https://informatica.ucm.es/localizacion	localizacion
7	https://informatica.ucm.es/futuros-estudiantes-1	futuro
8	https://informatica.ucm.es/excelencia-1	excelencia

Figura 5.6: Ejemplo DB direcciones

La función _getInfoWord:

```
function _getInfoWord ($txt)
{
        _db_connect();
        $query = "call getInfoWord('{$txt}')";
        $campo= _db_query($query);
        $filas= _db_num_rows($campo);
        for($i= 0 ; $i<$filas ; $i++){
            $res[$i]=_db_fetch_array($campo);
        }
        _db_close();
        if(0< $filas){
        return $res;
        }
        else{ return null;}
}
```

Figura 5.7: Código DB query

Realiza una llamada a un procedimiento definido en la Base de datos que devuelve la URL y la palabra aiml.

El módulo del Analizador está programado en PHP, y necesita como dato de entrada una oración, y dependiendo del componente (Parser o BaseConocimiento) se obtendrá un resultado u otro.

En el caso de la Base de conocimiento el resultado será un JSON con los siguientes campos:

Respuesta: Palabra clave que será analizada por el módulo Chatbot
Url: Redirección al usuario
Palabra: Pablara clave localizada en la base de datos

Un ejemplo de ejecución de esta parte sería:

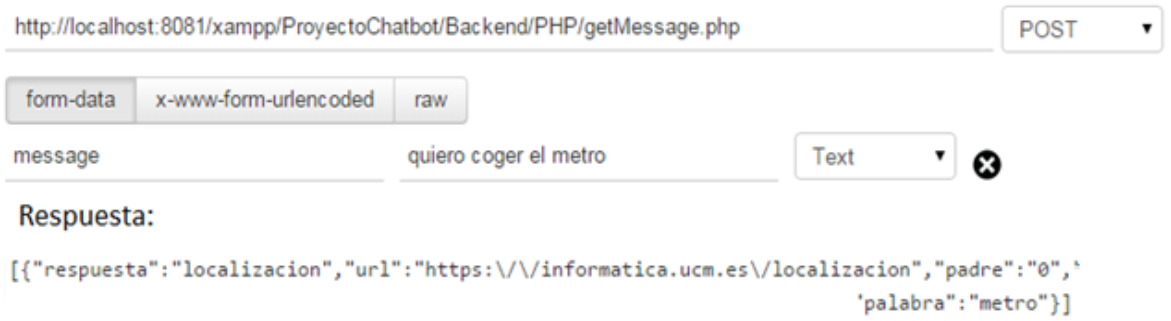

Figura 5.8: Ejemplo petición DB

En el caso de no que no interviniera la base de conocimiento, únicamente el Parser, la respuesta sería un JSON con el siguiente formato:
Respuesta: que será igual a “null”
Elementos: contiene los lemas de las palabras y en el caso de los verbos les elimina la terminación, añadiendo un “*”, se explicará en el Buscador

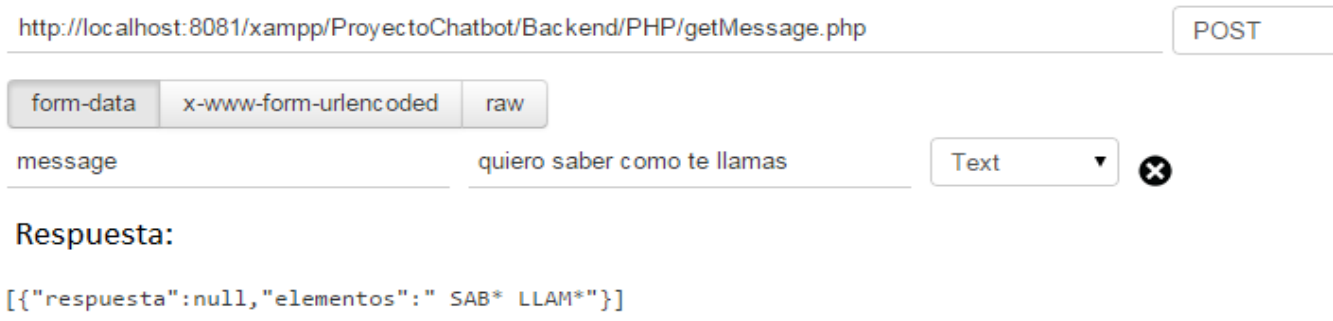

Figura 5.9: Ejemplo petición Parser

Chatbot

La funcionalidad del módulo es realizar la conexión con el servlet de Tomcat que se llama ProgramD, que es un intérprete de AIML realizado en Java a través del cual se obtendrán las respuestas definidas en las reglas de los archivos ".aiml"

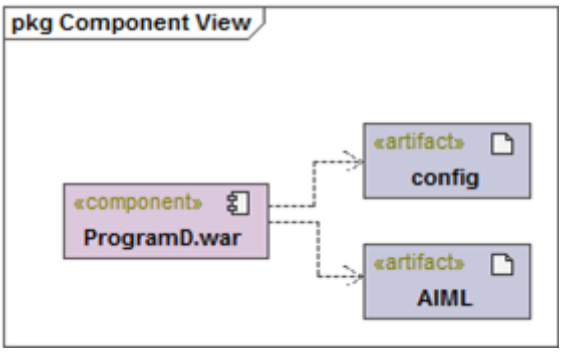

Figura 5.10: Diagrama componentes Chatbot

El servlet está compuesto principalmente por:

- **Archivos de configuración**:
    - properties.xml: que contiene los atributos que identifican al chatbot(nombre, edad, fecha de cumpleaños, etc.).
    - substitutions.xml: en este archivo se puede añadir alguna sustitución como por ejemplo, coloquialmente a veces cuando se escribe se dice: ¿xq no has comido?, con este archivo podemos transformar "xq" por->"por qué".

- **Conocimiento del Chatbot:**
    - Se localiza en los archivos ".aiml" (Explicado en el apartado 4.1.1de las tecnologías) que son el corazón del módulo. Es el encargado de responder al usuario según los patrones insertados.
      Se reutilizaron algunas respuestas adaptadas de algunos chatbots ya creados como son Sara (15) y Galaia(16).

busqueda.aiml	12/05/2015 17:42	Archivo AIML	1 KB
sql.aiml	24/04/2015 17:14	Archivo AIML	1 KB
respuestaGenerales.aiml	24/04/2015 13:03	Archivo AIML	11 KB
Saludos.aiml	22/02/2015 22:03	Archivo AIML	5 KB
curiosidades.aiml	22/02/2015 16:37	Archivo AIML	1 KB
estadoAnimo.aiml	09/02/2015 19:24	Archivo AIML	129 KB
infoUsuario.aiml	08/02/2015 20:31	Archivo AIML	1 KB

Figura 5.11: Archivos AIML

El módulo Chatbot tiene tres funciones:

- inicioProgramD(): Devuelve una cookie necesaria para interactuar con el chatbot.
- senDProgramD($string,$session):Envía una petición http a ProgramD, y los parámetros necesarios son:
  - string: La oración enviada al chatbot
  - sesion: La variable que identifica la sesión a la que está conectado el usuario, que debe de haber sido inicializada con la función previa

  El resultado obtenido es la respuesta del chatbot.
- proccessResult($string):Transformas las tildes y otros caracteres extraños en el formato de código UTF-8.

Ejemplo:

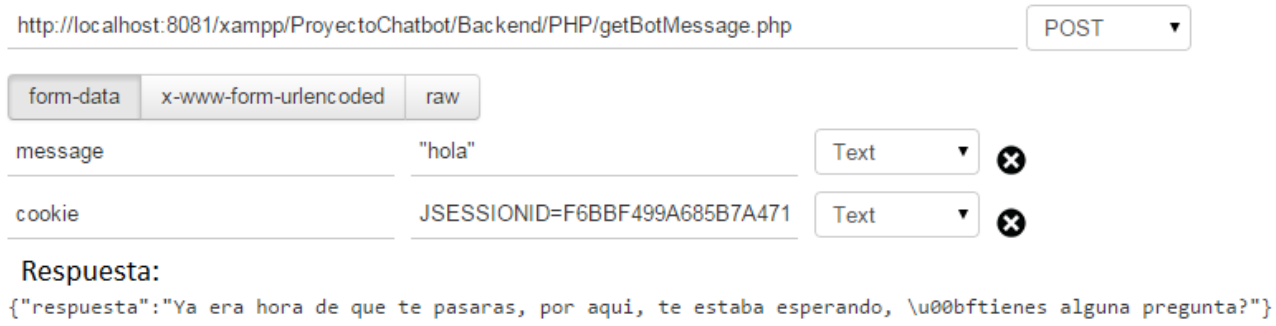

Figura 5.12: Ejemplo 1 petición ProgramD

Se añadió una funcionalidad en los archivos de AIML, y en algunas respuestas tienen codificadas una dirección web y estarían identificadas por la precedencia de " +url+":
Ej: {"respuesta":"Ejemploestudiar+url+http:\/\/informatica.ucm.es\/estudiar"}

En el caso de no encontrar ninguna coincidencia devolverá false:

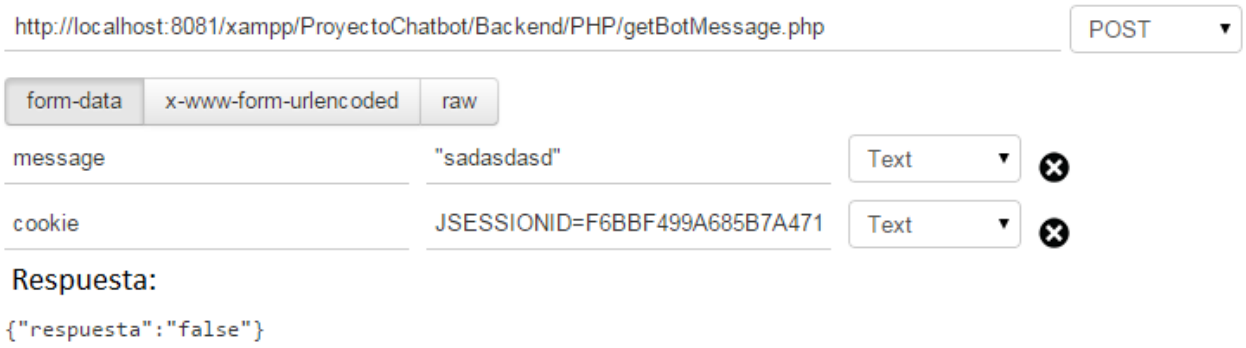

Figura 5.13: Ejemplo 2 petición ProgramD

Buscador:

Se encarga de realizar las peticiones HTTP al servlet Solr de Tomcat y mostrar la información en una página clonada de la página web de la facultada, a partir del JSON devuelto por la petición.
Únicamente se buscan las palabras clave, que pasaron previamente con el Parser, en un ejemplo que se vio la "Figura 5.9: Ejemplo petición Parser" el resultado fue "elementos: SAB* LLAM*" el símbolo especial "*" se utiliza para que el buscador también detecte palabras que compartan la raíz, en este caso buscaría palabras como: Llamada, Sabiduría, Llamamiento, etc.
Un ejemplo de resultado devuelto para la palabra exámenes sería:


```
{
  "responseHeader":{
    "status":0,
    "QTime":1,
    "params":{
      "indent":"true",
      "q":"examenes",
      "wt":"json"}},
  "response":{"numFound":10,"start":0,"docs":[
      {
        "content":"Facultad de Informática. Universidad Complutense de Madrid Navegar identificado
Presentación Gobierno Organización Más... Estudiar Grado Máster Doctorado Títulos propios Más... Es
docente Movilidad Más... Servicios Biblioteca Campus Virtual Sede Electrónica Más... Investigación
Estudiantes  »  Planificación docente  » Exámenes Exámenes Exámenes por curso y grupo Exámenes por
Contacto Aviso Legal Sede Electrónica Fundación General Servicios UCM en línea Información a Profes
        "title":"Facultad de Informática. Universidad Complutense de Madrid",
        "segment":"20141022212006",
        "boost":"0.0372678",
        "digest":"14ec36e07cddf7ad733f9b47b3836281",
        "tstamp":"2014-10-22T19:25:09.357Z",
        "id":"http://informatica.ucm.es/informatica/examenes",
        "url":"http://informatica.ucm.es/informatica/examenes",
        "_version_":1482695123479298049},
      {
        "content":"Facultad de Informática. Universidad Complutense de Madrid Navegar identificado
Presentación Gobierno Organización Más... Estudiar Grado Máster Doctorado Títulos propios Más... Es
docente Movilidad Más... Servicios Biblioteca Campus Virtual Sede Electrónica Más... Investigación
Servicios  »  Laboratorios FdI  » Exámenes en Laboratorios Exámenes en Laboratorios  Localización
Servicios UCM en línea Información a Profesores Laboratorios FdI Portal FDI Sugerencias y Quejas",
        "title":"Facultad de Informática. Universidad Complutense de Madrid",
        "segment":"20141022212006",
        "boost":"0.030429032",
        "digest":"0ccb13be36f0ec289b08fb3d300cf6d4",
        "tstamp":"2014-10-22T19:26:25.883Z",
        "id":"http://informatica.ucm.es/examenes-en-laboratorios/",
        "url":"http://informatica.ucm.es/examenes-en-laboratorios/",
```


Figura 5.14: Ejemplo petición Solr

Interfaz

Para interfaz se han utilizado dos elementos principales:
-Un Iframe que muestra la web de la facultad. Esta solución nos permite visualizar la página y poder interactuar con ella sin tener que tocar el código fuente principal.
-Y un widget, que es el asistente virtual, contiene imágenes, dos cuadros de diálogos, y un botón que es el encargado de enviar la petición a la clase Asistente Virtual.

5.2 Visión dinámica

Una vez visto cuál es la funcionalidad de cada módulo, es el momento de explicar cómo interaccionan entre ellos, primero se dará una visión global del sistema y posteriormente se hará uso de diagramas de secuencia para explicar el funcionamiento utilizando algunos de los escenarios.

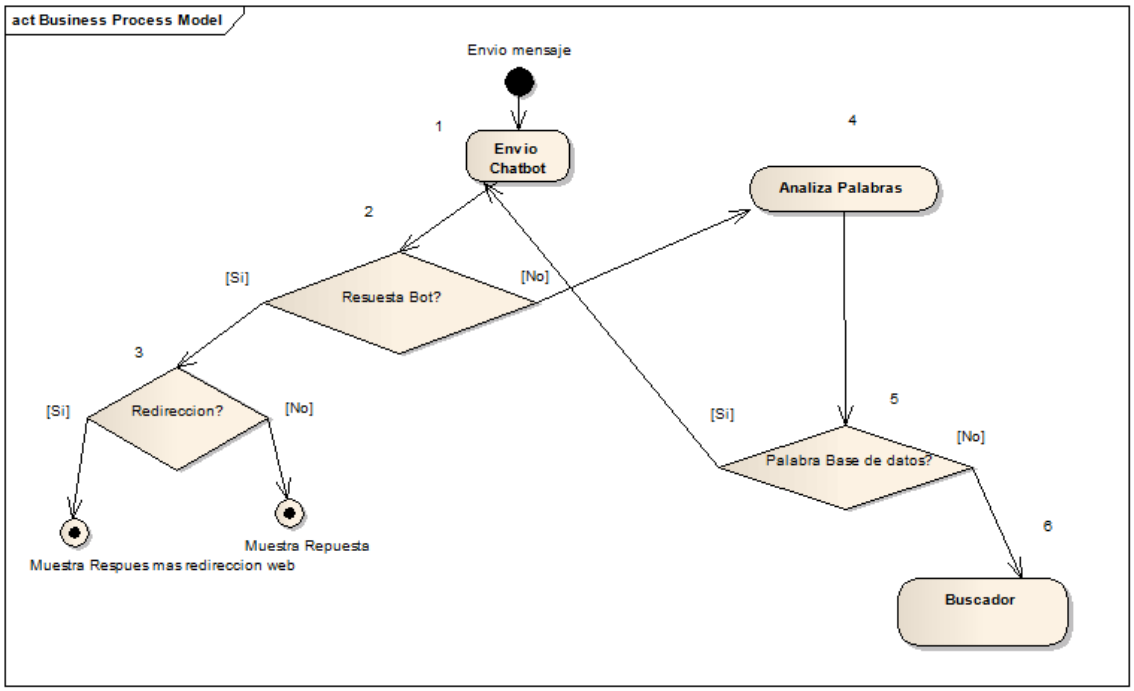

Figura 5.15: Diagrama de flujo global.

Cuando el visitante de la web realiza una petición el flujo de trabajo es el mostrado en la figura:

1. Envía la petición al módulo Chatbot [1], que se encarga de realizar la consulta vía HTTP a ProgramD , en el caso de obtener una respuesta distinta a “false” se mostrará la información al usuario, y en el caso de de existir una redirección de página web[3] se haría.
2. Si la respuesta del Chatbot fue “false”, se procederá a utilizar el módulo del Analizador, primero actuará el componente Parser y posteriormente la búsqueda de las palabras clave en la base de datos.
    a. Si se encuentra en la base de conocimiento enviaría la palabra aiml especial al módulo del Chatbot[1].
    b. En el caso de no existir ninguna palabra clave, se haría uso del buscador, que mostrará al usuario los resultados encontrados.

A continuación utilizaremos algunos de los escenarios ya mencionados:

Escenario 1: Saludo inicial

.Buenos días.
-Hola, bienvenido al asistente virtual de la facultad de informática
.¿Cómo te llamas?
-Me llamo Nerdi ,¿En qué puedo ayudarte?

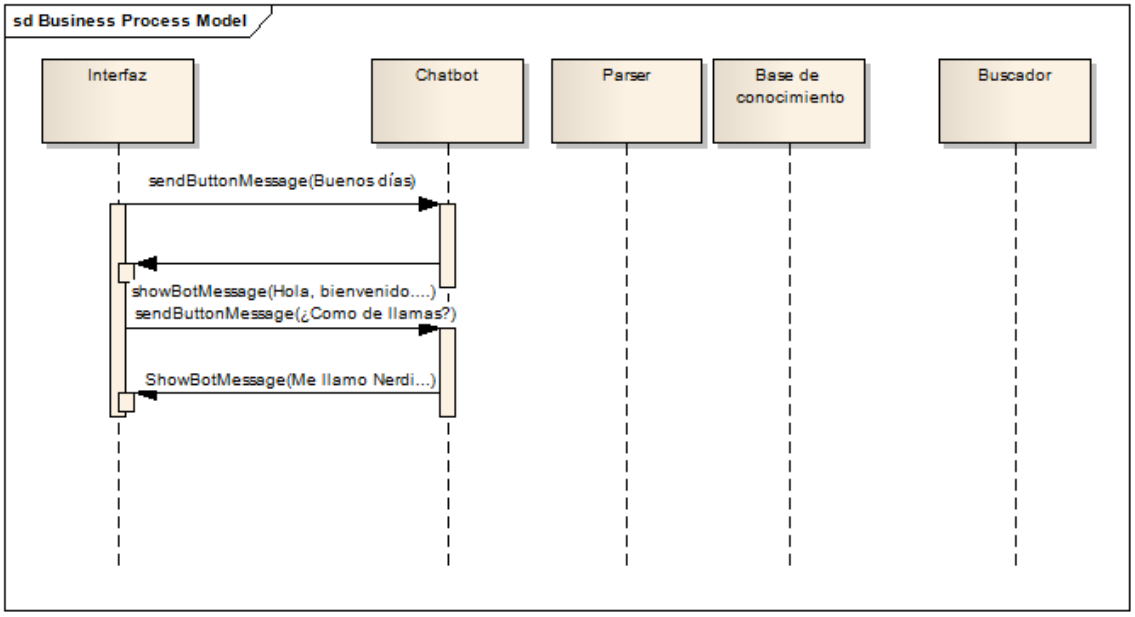

Figura 5.16: Diagrama de secuencia Escenario 1.

Este escenario es el más básico de todos, únicamente actúa la parte del Chatbot, que será el encargado de ofrecer las respuestas.

Escenario 8: Búsqueda Decano

.Me gustaría hablar con el Decano.
-Te voy a facilitar la información que conozco del Decano de la Facultad. (Redirección información Decano)

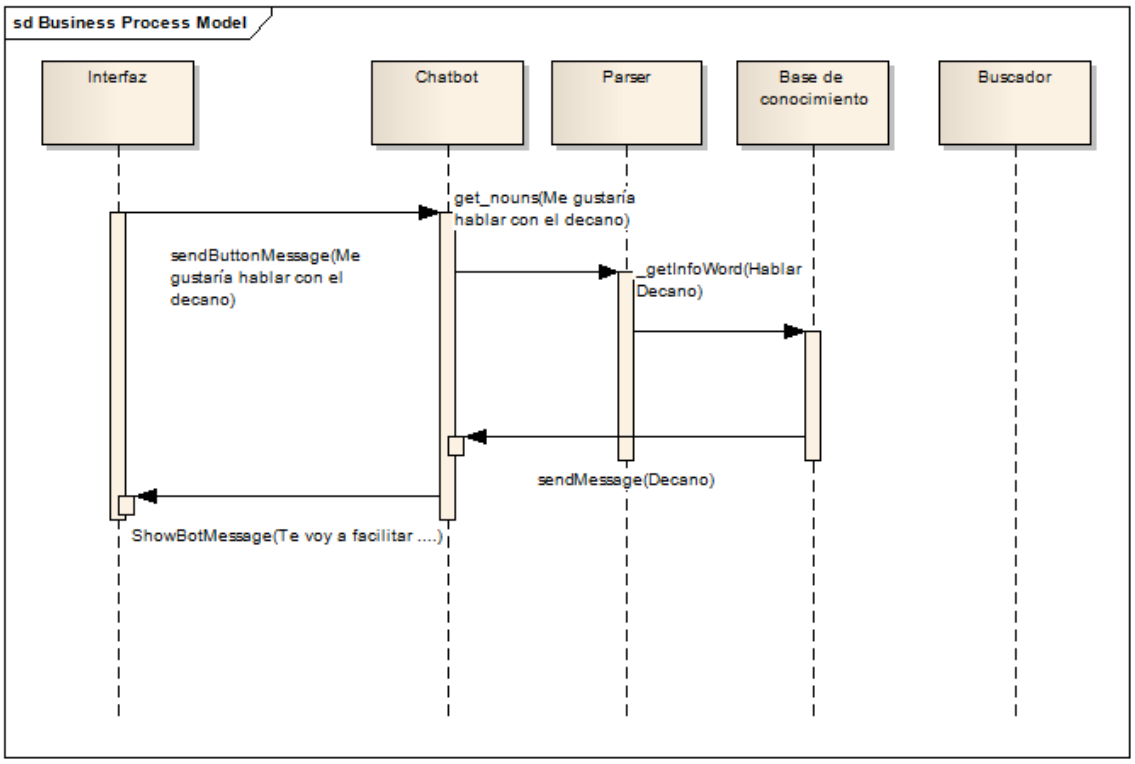

Figura 5.17: Diagrama de secuencia Escenario 8.

El flujo de trabajo ha sido el siguiente:

1. Envío de información al Chatbot, al no tener ningún patrón que encaje con “Me gustaría hablar con el Decano” devolverá “false”
2. Comenzará con el análisis de la oración. Primero a través de Grupal se identifican las palabras importantes(nombres y verbos), que son “Hablar Decano”
3. Se comprueba si existen en la base de datos, en este caso la palabra Decano está reconocida como palabra clave, por lo que se obtendrá una url y la palabra clave aiml.
4. Envío Chatbot palabra clave aiml “Decano”, que tiene predefinidas unas respuestas por ejemplo:“Te voy a facilitar la información que conozco acerca del Decano de la Facultad.”.
5. Devuelve el mensaje y cambia la dirección web a la url obtenida en la base de datos.

Escenario 10: Búsqueda

-Me gustaría buscar una palabra
.Dime
-MIGS
.He encontrado unos resultados quizá puedan ayudarte.

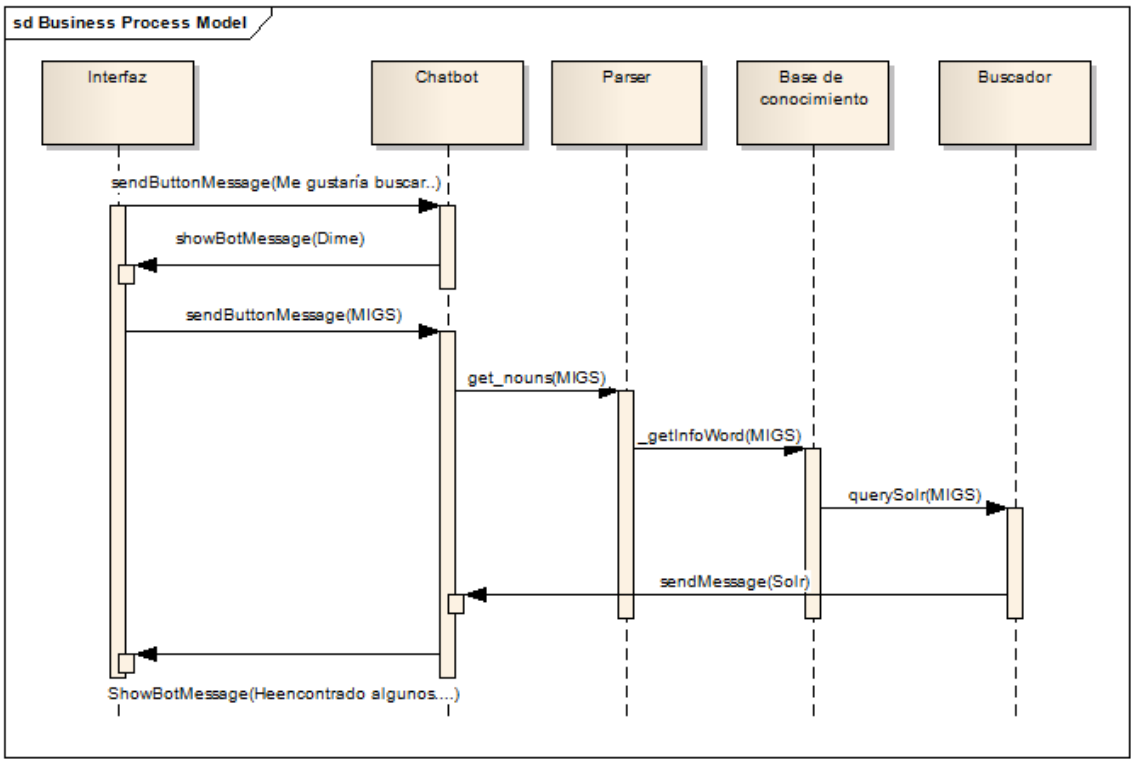

Figura 5.18: Diagrama de secuencia Escenario 10.

En la segunda petición intervienen todos los componentes:

1. Envío de información al Chatbot, al no tener ningún patrón que encaje con "MIGS" devolverá "false"
2. Comenzará con el análisis de la oración. Primero a través de Grupal se identifican las palabras importantes(nombres y verbos). Al ser únicamente una palabra está programado para identificarla como importante.
3. Se comprueba si existe en la base de datos, que en este caso no está añadida.
4. Uso del buscador para ver si aparece en algún sitio del dominio web.
5. Envío Chatbot palabra clave "Solr" que significa que se ha utilizado el buscador y tiene unas respuestas tipo como por ejemplo:"-He utilizado el buscador mira a ver si te interesa alguno de los resultados."
6. Devuelve el mensaje y muestra la web de búsquedas.

6. Evolución del proyecto

A continuación se describe el avance del proyecto en cada uno de los sprints que se han ido realizando, la fecha que aparece es la de finalización de sprint.

El proyecto da comienzo el día 1/10/2014.

Sprint 1 (Buscador) 9-11-2014

El primer objetivo marcado fue el funcionamiento del motor de búsqueda. Para este objetivo se estimó un tiempo de tres semanas y para poder llevarlo a cabo se tenía que abordar previamente las siguientes tareas previas:

- Instalación y configuración de Tomcat.
- Indexación del dominio.

Estos primeros pasos no fueron tarea fácil principalmente por dos motivos:

- Exigió un fuerte aprendizaje de nuevos conceptos: *Crawler*, motor de búsquedas, funcionamiento de Tomcat.
- Realización de las distintas configuraciones del entorno: variables, configuración de Java, compatibilidades de versiones, etc.

Tras 3 semanas se logró configurar Solr y por fin se realizó la conexión con-Tomcat a través del puerto 8080.

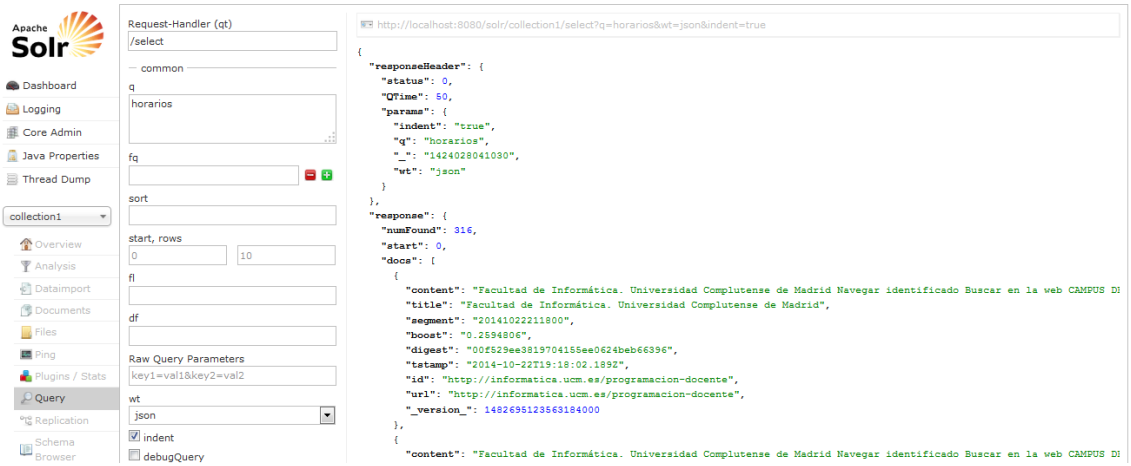

Figura 6.1: Ejecución Solr

A partir de este momento se podrían realizar pruebas de búsquedas sobre nuestro contexto.

Sprint 2 (Integración Buscador con Java) 16-11-2014

El siguiente objetivo marcado fue la integración con Java a partir de un prototipo básico que se comunicara con el servidor de búsquedas.

Para la integración se utilizaron las librerías de Solr, que mediante peticiones se obtenían los resultados de las búsquedas.

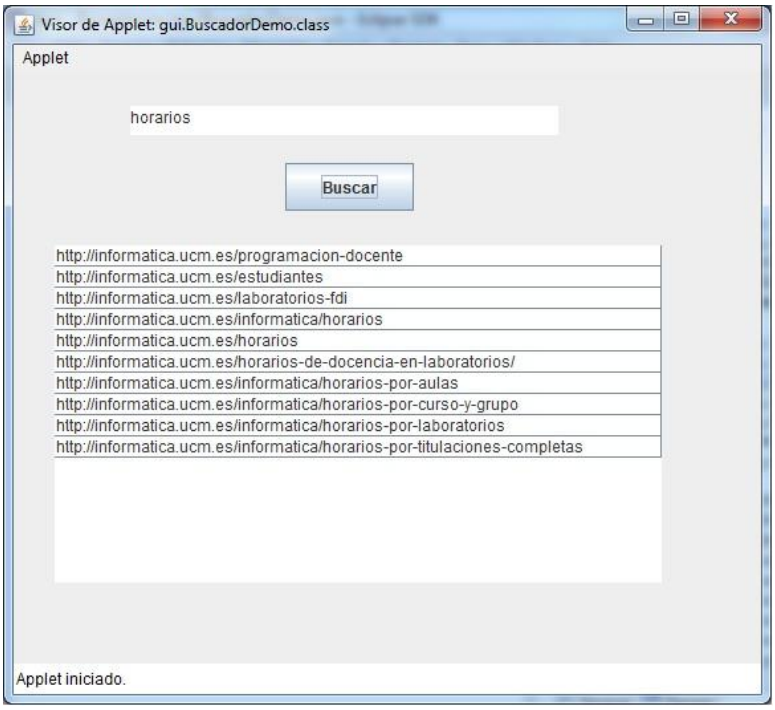

Figura 6.2: Prototipo Buscador

Una vez comprobado el correcto funcionamiento del buscador y su integración con la aplicación, el siguiente paso que se fijó fue la integración del chatbot.

Sprint 3 (Añadir chatbot) 23-11-2014

Para llevar a cabo esta integración se utilizó un entorno de desarrollo llamado *Pandorabots*. Este entorno de desarrollo es un servidor que ofrece a los usuarios cargar los ficheros AIML (conocimientos del chatbot) y realizar las peticiones vía HTTP. El problema de estas ejecuciones es que están limitadas y por ese motivo más adelante se decidió cambiar la plataforma para tener el control 100% de la ejecución del chatbot.

Antes de realizar el cambio de la plataforma se realizó un prototipo que integraba la búsqueda y el robot.

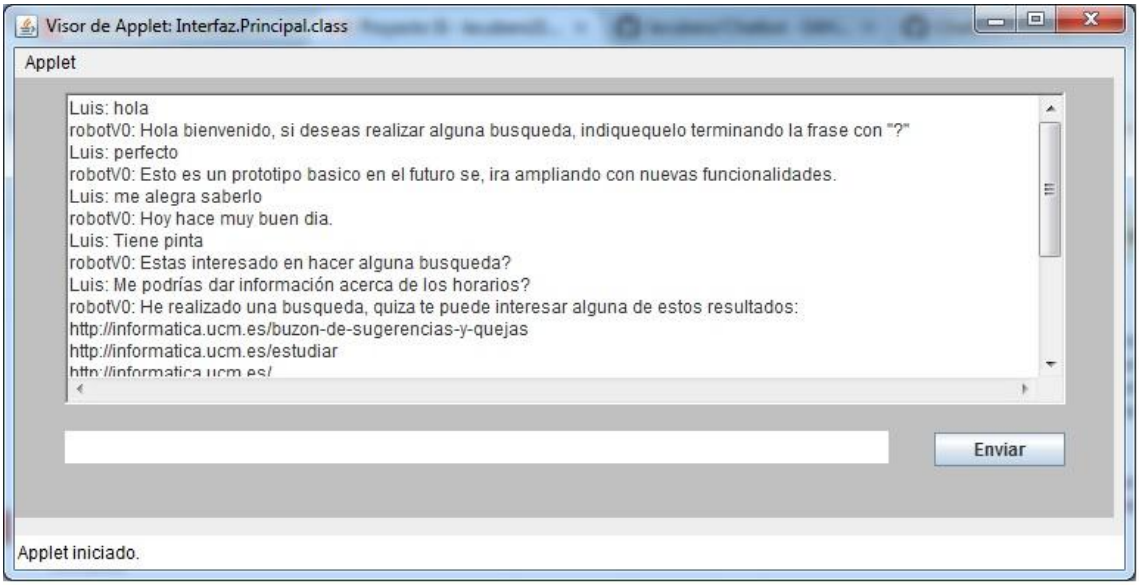

Figura 6.3: Prototipo 1.0

Este prototipo es bastante básico y presenta muchas limitaciones, entre las más importantes destacan:

- No era capaz de distinguir una pregunta de un comentario. Las búsquedas se realizaban cuando el usuario terminaba la frase con el signo de interrogación.
- El conocimiento del robot se reducía a 3 frases.

Sprint 4 (Distinción de preguntas)7-12-2014

Vistas las carencias del anterior prototipo, el objetivo principal pasó a ser la identificación de preguntas.

En este momento se toma la importante decisión de intentar extraer los sustantivos de la oración y, a partir de estos, realizar la búsqueda. En el caso de no encontrar resultados se considerará como una “conversación”.

Se decidió utilizar un analizador desarrollado para Java que ofrecía una interfaz sencilla y se basaba en un perceptrón.
Un perceptrónes una red neuronal artificial capaz de realizar una tarea a partir de un entrenamiento previo.
Una vez integrados los tres elementos (Analizador, Buscador y Chatbot) se empezaron a realizar distintas pruebas. Estas pruebas certificaron las limitaciones ofrecidas por *Pandorabots* y, por tanto, se tuvo que buscar una solución alternativa.

Sprint 5 (Cambio plataforma chatbot) 21-12-2014

La alternativa a *Pandorabots* fue ProgramD. ProgramD se introdujo en nuestro proyecto mediante el despliegue y configuración de un fichero war.

Figura 6.4: Chatbot ProgramD

Una vez comprobado su funcionamiento se integró con el *Applet* mediante peticiones http.

Todo el desarrollo del proyecto se estaba realizando en el entorno de desarrollo Eclipse. En el momento en que se exportó el proyecto a un archivo ".jar" para la ejecución desde una web(hasta ese momento se estaba trabajando en local) varias cosas dejaron de funcionar. Por consiguiente se tuvo que reconfigurar parte del proyecto hasta que finalmente se consiguió su ejecución.

Sprint 6 Prototipo final applet.(Presentación febrero) 8-2-2015

A continuación se mostrarán las imágenes de una demo realizada para la presentación de febrero del applet donde iremos viendo y comentando las deficiencias que tenía.

La primera carencia surgía en el momento de cargar el applet, tardaba aproximadamente 50 segundos en descargar todos los elementos necesarios para la ejecución.

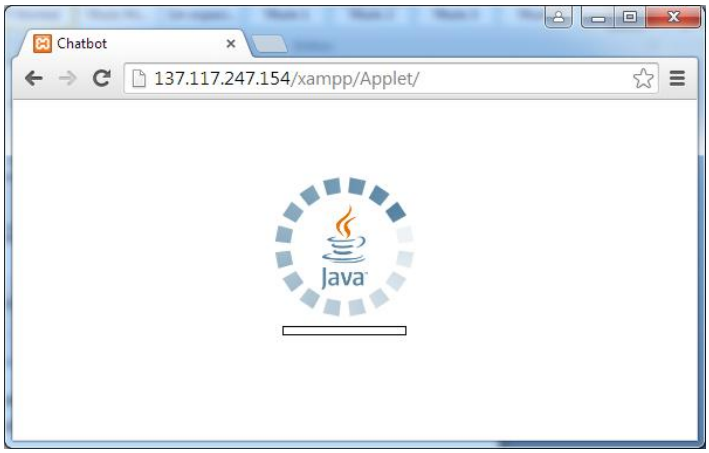

Figura 6.5: Carga Java

Una vez cargado, había que aceptar un mensaje para permitir la ejecución del applets incluso estando firmados.

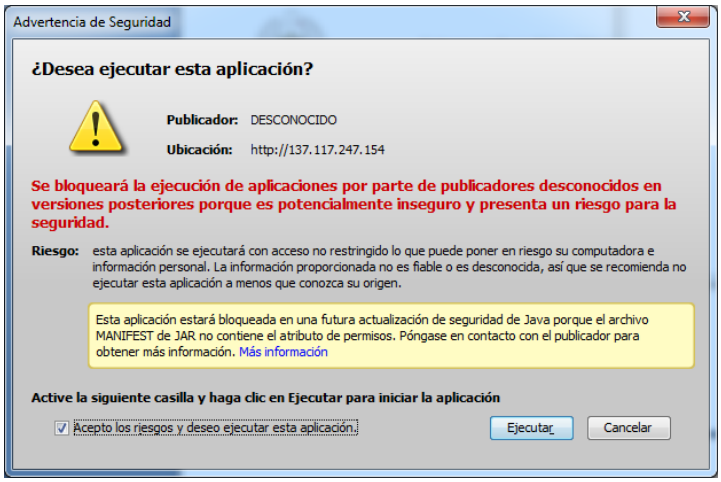

Figura 6.6: Permitir Ejecución

Una vez cargada la aplicación aparecería la interfaz del programa y podremos empezar a conversar con el asistente virtual, se muestran algunos ejemplos de conversación.

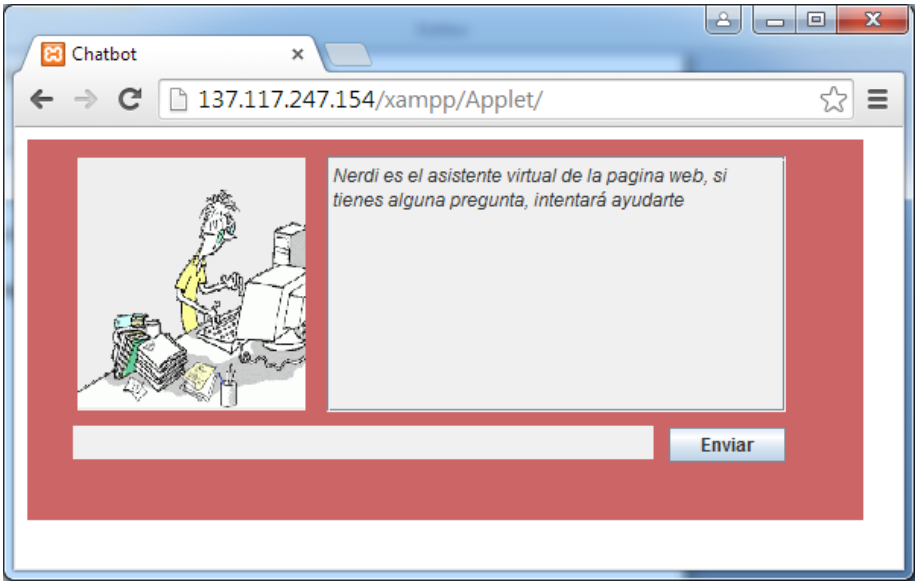

Figura 6.7: Prototipo 2.0

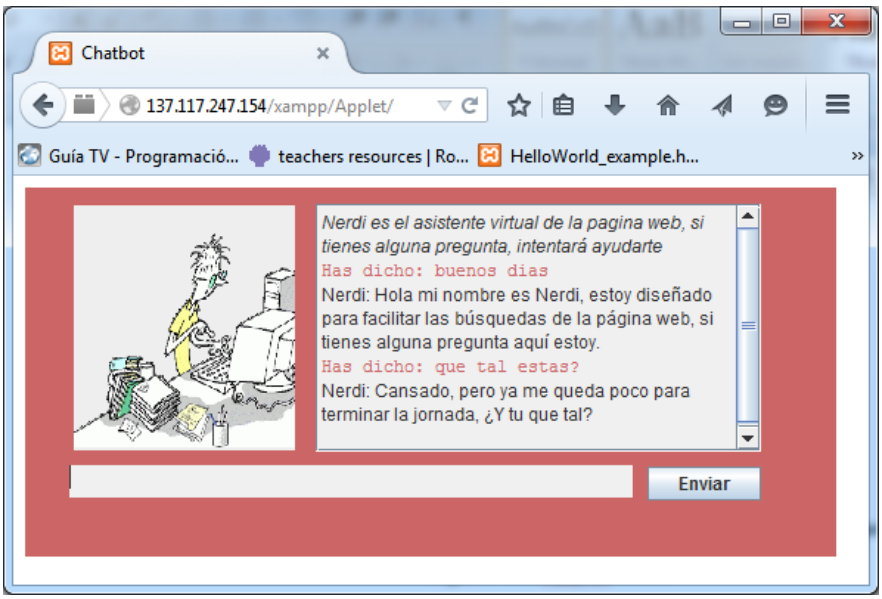

Figura 6.8: Applet Saludo Inicial

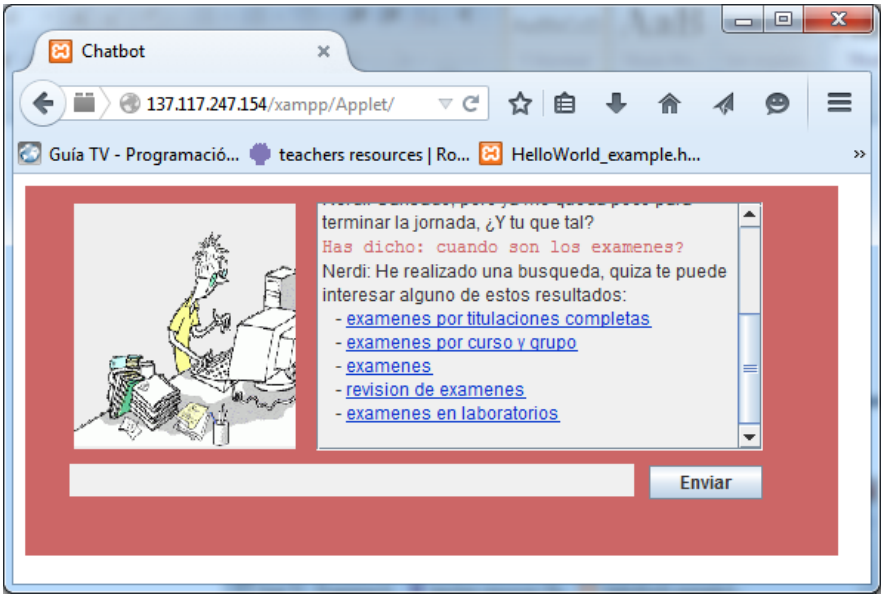

Figura 6.9: Applet Búsqueda

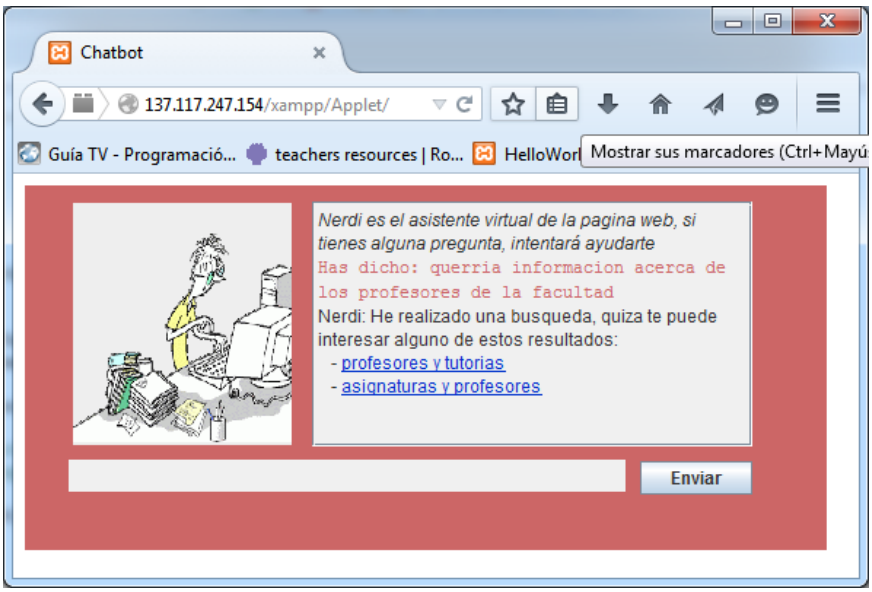

Figura 6.10: Applet Búsqueda 2

Como se pueden ver en los ejemplos el applet no está integrado con la web, la máxima interacción alcanzada fue que recomendaba algunas búsquedas y que abriera una pestaña en el navegador redirigiendo a la web referenciada y en temas de rendimiento los tiempos de respuesta eran bastante elevados.

No habiendo alcanzados los objetivos propuestos, se tuvo la necesidad de realizar un cambio que afectó a la tecnología utilizada, se pasó de un applet a un widget(es un pequeño programa que en este caso está incorporado en la web de la Facultad, con el que se puede interactuar) realizado en JavaScript que pudiera ser integrado con HTML. A pesar del cambio se intentó utilizar todos los elementos desarrollados para el applet.

Sprint7 (Pruebas HTML)15-2-2015

Se empezaron a hacer pruebas de interfaz con HTML y JavaScript, el prototipo consistía en dos cuadros de texto y un botón, y la función que realizaba era escribir el texto de un cuadro al otro cuando se pulsaba el botón.

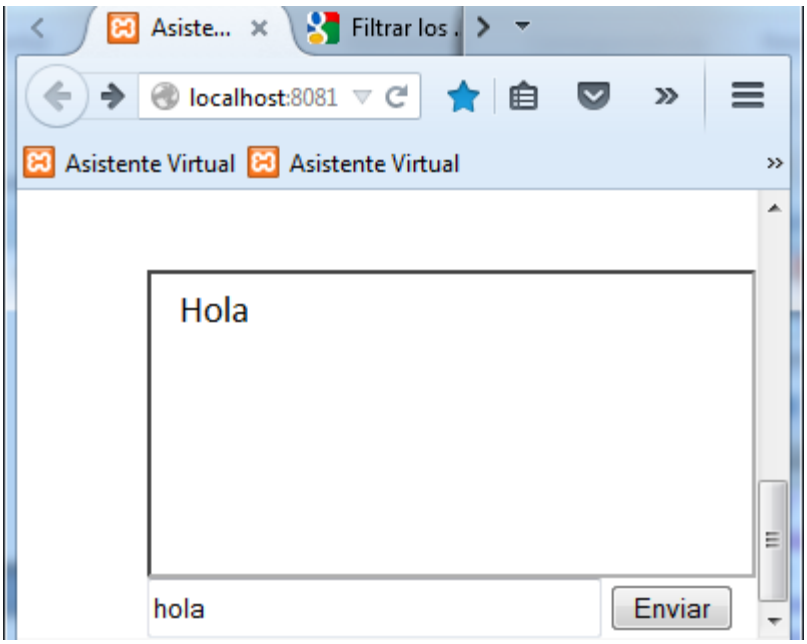

Figura 6.11: Prototipo HTML y JavaScript

Sprint 8 (Pruebas HTML 2) 22-2-2015

El siguiente objetivo fue integrarlo con la página web de la Facultad mediante el uso de Iframes(Complemento para HTML, en el que se puede insertar otro documento HTML anidado).

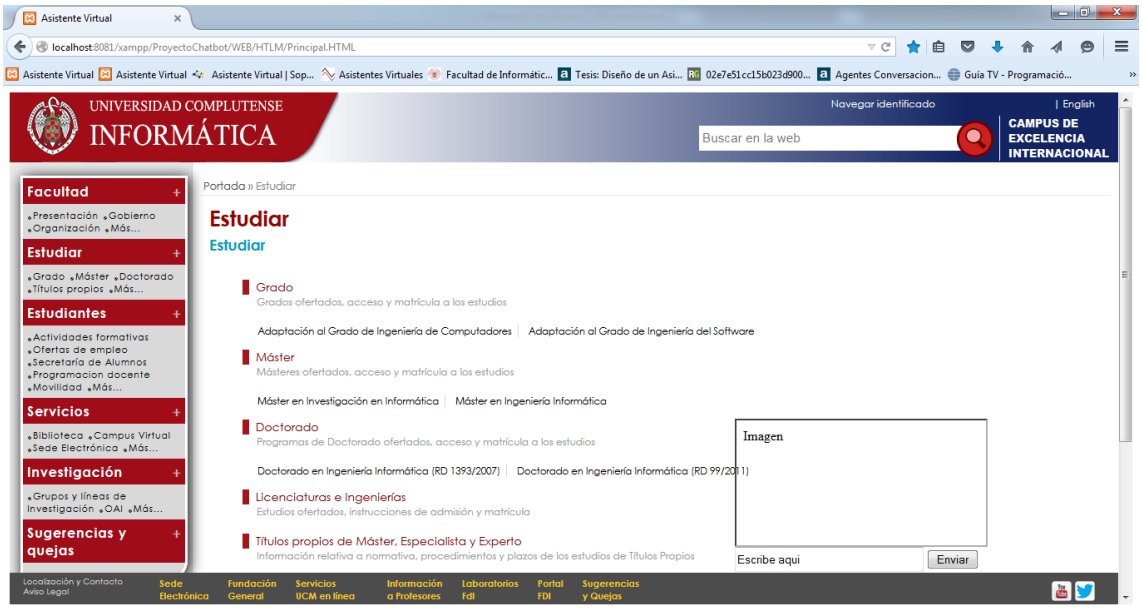

Figura 6.12: Prototipo HTML, JavaScript y Web

Sprint 9 (Integración elementos del Applet) 5-4-2015

Este fue el sprint más largo de todos, duró un mes y medio, estaba definido cuál era el objetivo, y se estuvo desarrollando durante semanas, se debió a que fue necesario la adquisición de nuevos conocimientos tanto como de HTML, JavaScript, PHP, CSS y no se pudo reciclar siempre el trabajo previo realizado en el Applet, en el caso del perceptrón era imposible integrarlo, por lo que se tuvo que buscar otro módulo que tuviera la misma funcionalidad, que fue el explicado en la arquitectura del sistema.

Sprint 10 (Resolución de errores + Base de datos) 26-4-2015

Una vez incorporado todos los elementos se fueron puliendo algunos problemas que habían surgido como la compatibilidad entre navegadores, el inicio de sesión y se añadió la inexistente base de datos

Sprint 11 (Mejora interfaz) 10-5-2015

Durante dos semanas se estuvo diseñando la interfaz del asistente virtual, para que fuera más vistoso, y encajara con la web de la Facultad, puesto que hasta la fecha existían dos cuadros de texto y un botón.

7. Instalación y ejecución

En este apartado se explican los pasos necesarios para desplegar el Asistente Virtual en un host.

El primer paso será la instalación del software necesario. En el apartado 4.5se expusieron las versiones que se han utilizado para desplegar el Asistente Virtual tanto en Windows 7 como en Windows Server 2008 R2.

Configuración Buscador:

Una vez instalado correctamente Solr de deberá poder realizar la conexión en "http://localhost:8080/solr/#/" en el proyecto se usa el puerto 8080 que es el utilizado por defecto en Tomcat, el siguiente paso es indexar de la web de la facultad:

1. Instalación Cygwin, su función es simular la ejecución de un entorno UNIX en un sistema operativo Windows, se utiliza para recompilar el código fuente del crawler.
2. Configuración:
    a. Semillas: Se debe crear el fichero "seeds.txt" en la ruta "C:\apache-nutch-2.2.1\runtime\local\bin\urls\seed.txt", este fichero contiene las url´s que se van consultar, en el proyecto se ha utilizado únicamente el dominiohttp://informatica.ucm.es/.
    b. Filtros de direcciones o documentos: el archivo localizado en "C:\apache-nutch-2.2.1\conf\regex-urlfilter.txt" se pueden filtrar las direcciones o los tipos de documentos que se quieren añadir a la base de conocimiento, se utilizó la siguiente configuración:"+^http://([a-z0-9\-A-Z]*\.)*informatica.ucm.es/([a-z0-9\-A-Z]*\/)*" que indica que procesará todas las direcciones que comiencen por "http" y contengan "informatica.ucm.es/", si no se configura procesará todas las url`s que cuelguen del archivo de las semillas.
3. Ejecución crawler: ejecutar Cygwin y situaremos la ruta sobre los archivos principales y se introduce el siguiente comando "nutch crawl bin/urls -solrhttp://localhost:8080/solr/collection3" en el que se indica la localización de las semillas y donde se generará la base de datos, en este caso en Solr. Una vez ejecutado, comenzará la indexación, por defecto está limitado la profundidad de la búsqueda a 5 niveles, hay que añadir el parámetro "–depth" + "numero de la profundidad" que se considere necesaria:

```
Luis@Luis-PC /cygdrive/c/apache-nutch-2.2.1/runtime/local
$ nutch crawl bin/urls  -solr http://localhost:8080/solr/collection3
```

Figura 7.1: Comando ejecución crawler

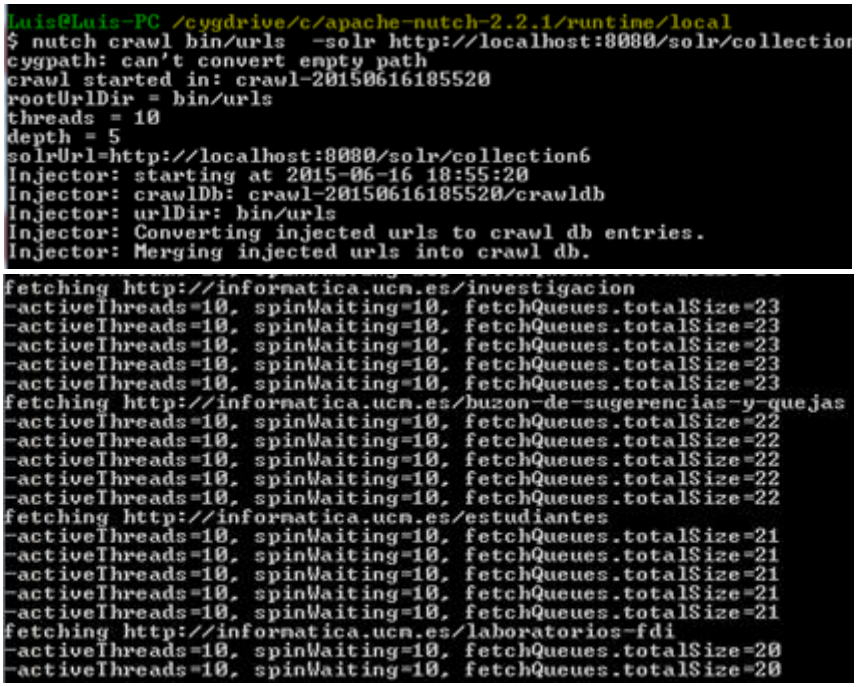

Figura 7.2: Proceso ejecución crawler

Una vez terminado el proceso comprobamos su correcta ejecución en la dirección "http://localhost:8080/solr/#/collection3":

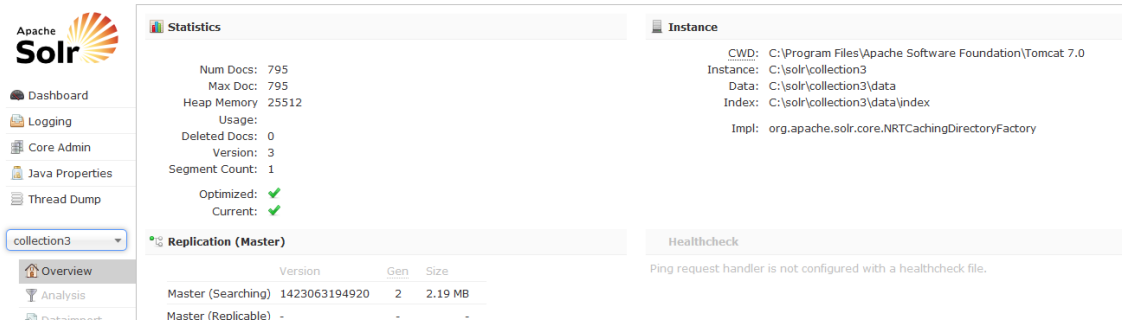

Figura 7.3: Resultado ejecución crawler Solr

En este caso se han indexado 795 documentos, que ya están disponibles para ser consultados.

Configuración ProgramD:

1) Para instalar ProgramD lo primero que se hará será desplegar archivo ".war" en el servidor Tomcat, para ello colocaremos el fichero "programd.war" en la ruta de los webapps, y reiniciaremos el servicio de Tomcat, una vez levantado el servicio nos creará una estructura de carpetas. Se puede comprobar si se accede correctamente en la dirección "http://localhost:8080/programd/":

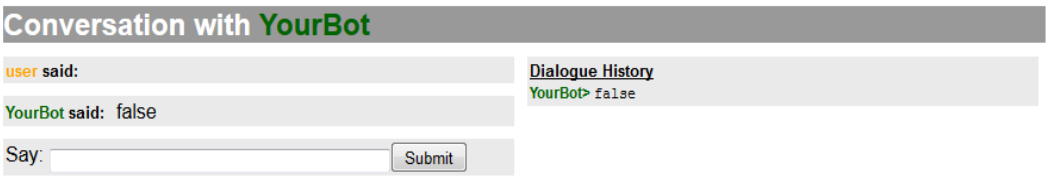

Figura 7.4: Inicio PogramD

2) Configuración:
   a) Archivos AIML: la base de conocimiento se localiza en "Tomcat 7.0\webapps\programd\resources\testing" donde se copiarán los archivos ".aiml".
   b) Configuración chatbot: en la ruta "programd\conf" se encuentran los archivos de configuración:
      (i) "bots.xml" indica el origen de los archivos ".aiml" que se van a compilar:
      "<learn>../resources/testing/*.aiml</learn>"
      (ii) "properties.xml": son los atributos del robot, como nombre edad, sexo…:
      "<property name="botname" value="Nerdi"/>"
      (iii) "substitutions.xml" se configuran las sustituciones que van a realizar como el ejemplo de sustituir "xq" por "por qué":
      "<substitutefind="\bxq" replace=" por qué"/>"

Configuración base de datos:

Habrá que exportar la base SQL actual a un fichero ".sql" que se ejecutará para crear la nueva instancia en el servidor objetivo.

Configuración aplicación:

Una vez que están funcionando todos los módulos por separado, habrá que añadir al servidor web los archivos del programa, que están distribuidos en dos carpetas:

- Backend: que contiene los archivos que realizan la conexión con la base de datos y los PHP que se ejecutan en el servidor.
- Web: donde se encuentran los documentos HTML y JavaScript que se ejecutan en el cliente.

Una vez conectados y configurados todos los elementos se podrá a empezar a ejecutar el programa, a continuación se mostrarán algunos ejemplos de la aplicación:

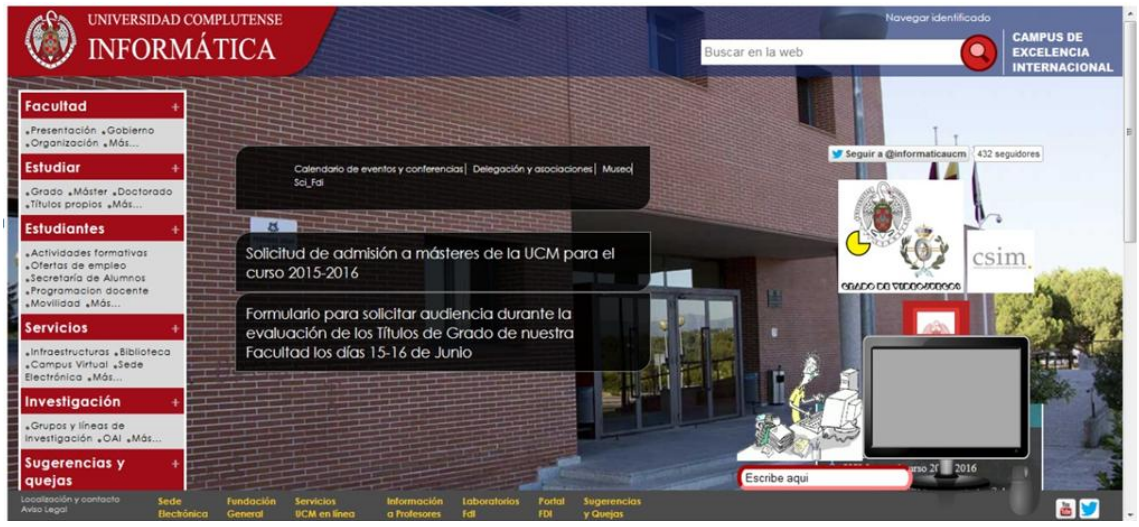

Figura 7.5: Interfaz principal

Una funcionalidad añadida es la posibilidad de desplazar el widget a través de toda la página, por si en algún momento molestara mientras se navega.

Figura 7.6: Desplazamiento widget

Existen dos posibilidades de enviar la información una es pulsando la tecla “Enter” y la otra es haciendo clic en la imagen del ratón.

Demo conversación:

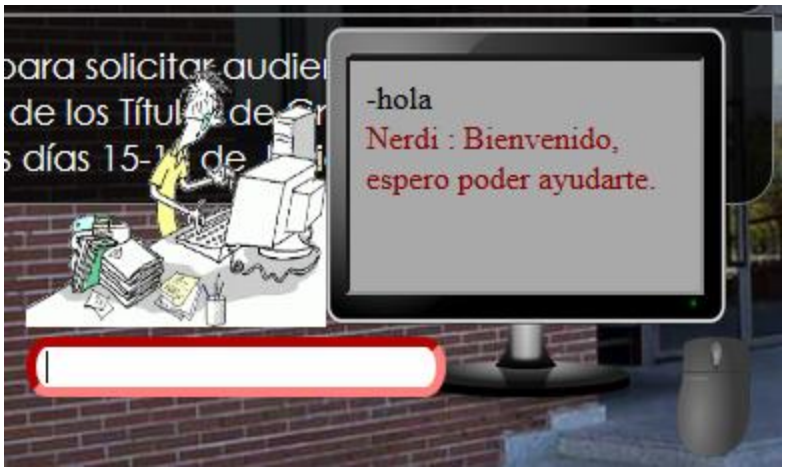

Figura 7.7: Conversación 1 AIML

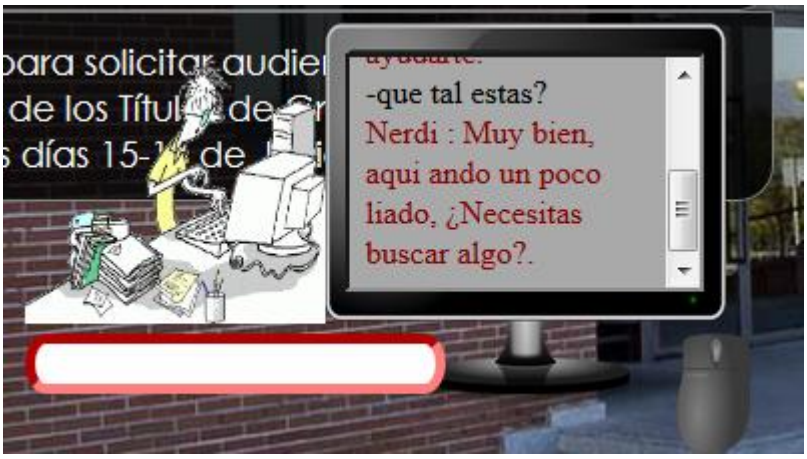

Figura 7.8: Conversación 2 AIML

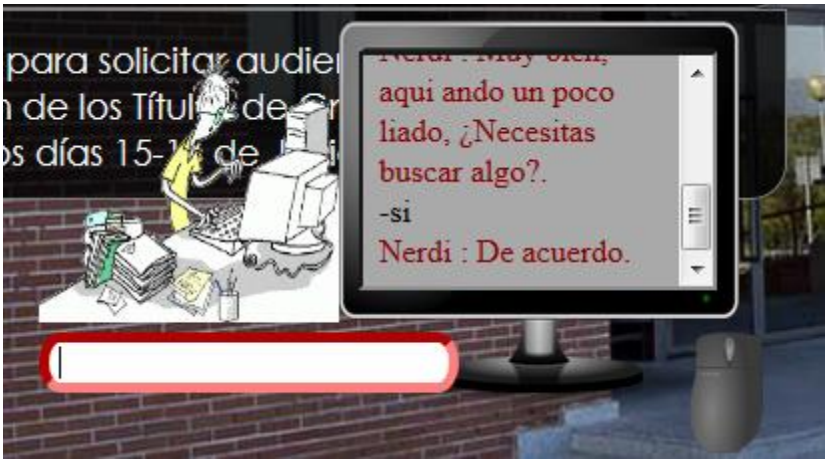

Figura 7.9: Conversación 3 AIML

Ante la pregunta: ¿Donde está el autobús para ir a la universidad?

El asistente virtual redirige a otra url y responde: A ver si te aclara algo este plano.

Figura 7.10: Conversación palabra clave DB

Otro tipo de petición podría ser cuando el usuario utiliza el asistente como buscador:

En este caso el usuario escribe: “MIGS”

La aplicación cambia la url y muestra el clon de la página web con los resultados de la búsquedas, en este caso encuentra una dirección donde se hace referencia, si se hace *clic* en “Otros-servicios” el navegador abrirá la nueva ubicación donde efectivamente se puede comprobar que aparece la palabra.

Figura 7.11: Conversación Búsqueda Solr

Figura 7.12: Comprobación búsqueda

8. Conclusiones

En mi opinión he podido participar en una rama de la informática que ha cobrado mucha importancia en los últimos años y en la que grandes compañías nacionales e internacionales están invirtiendo muchos recursos para poder adquirir estos servicios. Por tanto, el poder desarrollar este proyecto ha supuesto una enorme oportunidad para adquirir una gran variedad de conocimiento y aplicar otros muchos adquiridos durante la carrera.

El proyecto utiliza muchas tecnologías y lenguajes de programación diferentes, algunos de ellos completamente nuevos para mí, como AIML, JavaScript, y otros que había aprendido durante la carrera HTML,CSS,PHP,MYSQL, JAVA.

Uno de los conocimientos adquiridos más importantes son el uso de *widgets* con JavaScript y los Applets, al haber trabajado con los dos he visto cuales son las ventajas de cada uno, cómo funcionan, y cuál es su proceso de publicación, en el caso de Java es necesario compilar el proyecto, firmar la aplicación y las librerías existentes para poder ser ejecutado.

Otro concepto nuevo adquirido es el de servlet, hasta la fecha no sabía de su existencia, y mucho menos sobre su funcionamiento, gracias al proyecto he aprendido cómo trabajan y como se realizan las conexiones.

8.1 Trabajos Futuros

Es un proyecto que se puede mejorar constantemente. Como ya se mencionó en la memoria, los módulos están preparados para poder ser sustituidos por otros, siempre que utilicen la misma interfaz.

Alguna de las mejoras más interesantes en las que se podría seguir desarrollando el asistente son las siguientes:

- Guardar el contexto de la conversación e ir acotándolo.

  El proyecto únicamente utiliza algunas funciones de AIML para ver que ha dicho el usuario previamente, pero realmente no va acotando la búsqueda, podría suponer una gran mejora si los diálogos con los usuarios son largos, y sus preguntas no son muy concretas.

- Insertar un rostro animado con expresiones faciales.

  Se ha realizado algún prototipo de rostro animado con "CrazyTalkAnimator" una herramienta para crear gifs.

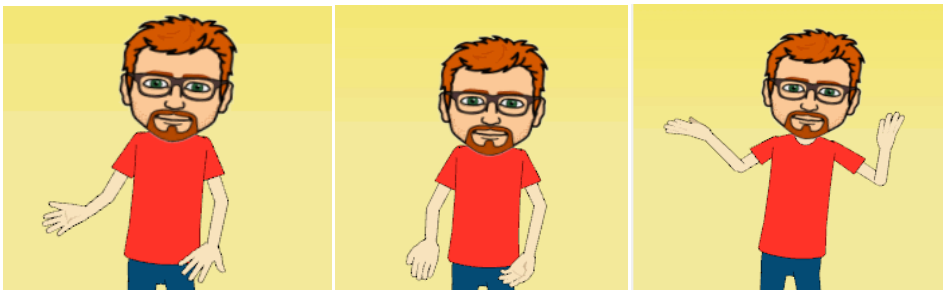

Figura 8.1: Prototipo gif animado

- Mecanismo "Text To Speech":

  Una opción interesante que se comentó en el punto 1.1era que algunos asistente virtuales tenías esta funcionalidad, que consistía en complementar la respuesta escrita con una respuesta auditiva.

- Búsquedas por voz:

  Android tiene una funcionalidad que es una interfaz vocal, el concepto es el inverso al "Text To Speech", es transformar la voz en texto, y poder recoger de esta forma las peticiones del usuario, muy parecido al sistema Siri de Apple.

- Automatización semanal crawler:

  Esta mejora sería la más sencilla de todas, consistiría en la programación semanal de un script que indexara la nueva información del contenido web de la facultad.

- CBR (Case Base Reasoning)

  Razonamiento basado en casos, es un proceso que consiste en resolver problemas haciendo uso de soluciones anteriores, esta incorporación podría suponer una importante mejora, puesto que el sistema podría ir adquiriendo conocimiento automáticamente. También se podría orientar de otra forma, que sería un módulo dónde se sugirieran algunos resultados que se cree que pudieran resultar de interés para el usuario.

Bibliografía

1. **Jesús, Raymundo Domínguez.** Tesis: Diseño de un Asistente Virtual con Diálogo Emocional. [En línea] Mayo de 2011 (consultado el 04/11/2014). http://www.academia.edu/1188981/Tesis_Dise%C3%B1o_de_un_Asistente_Virtual_con_Di%C3%A1logo_Emocional (consultado el 10/05/2015).

2. Eliza Chatbot. [En línea] http://nlp-addiction.com/eliza/ (consultado el 22/1/2015).

3. **Gómez, Juan.** Joseph Weizenbaum, pionero de la informática. [En línea] 2008. http://elpais.com/diario/2008/04/09/necrologicas/1207692001_850215.html (consultado el 22/1/2015).

4. The Loebner Prize in Artificial Intelligence. [En línea] http://www.loebner.net/Prizef/loebner-prize.html (consultado el 22/01/2015).

5. Virtual Solutions. [En línea] http://solucionesvirtuales.es/ (consultado el 10/05/2015).

6. Renfe Chatbot. [En línea] http://consulta.renfe.com/renfe0/index.jsp (consultado el 12/03/2015).

7. Ikea Chatbot. [En línea] http://www.ikea.com/es/es/ (consultado el 12/03/2015).

8. CocaCola Chatbot. [En línea] http://www.coca-colacompany.com/contact-us/ (consultado el 12/03/2015).

9. Windstream Chatbot. [En línea] http://www.windstream.com/Meet-Wendy/ (consultado el 12/03/2015).

10. Elvira Chatbot. [En línea] http://tueris.ugr.es/elvira/ (consultado el 07/05/2015).

11. GoogleBot. [En línea] https://support.google.com/webmasters/answer/182072?hl=es (consultado el 12/06/2015).

12. **Pavón, Juan.** Herramientas de recuperación de información . [En línea]

13. Imagen arquitectura servlet. [En línea] https://parasitovirtual.wordpress.com/page/7/ (consultado el 04/06/2015).

14. **Antonio Sandoval, Moreno y Guiarao Miras, Jose Ma.** Grampal. [En línea] http://cartago.lllf.uam.es/grampal/grampal.cgi (consultado el 12/06/2015).

15. Sara Alicebot. [En línea] http://www.alicebot.org/downloads/sets.html (consultado el 22/01/2015).

16. Chatbot Universidad de Vigo[En línea] http://papa.det.uvigo.es/~galaia/ES/home/bots/ (consultado el 16/12/2014).

17. Pandorabots. [En línea] http://www.pandorabots.com/(consultado el 10/12/2014).

18. Tuturial instalación Nutch. [En línea] https://wiki.apache.org/nutch/NutchTutorial (consultado el 05/12/2014).