UNIVERSIDAD CARLOS III DE MADRID

ESCUELA POLITÉCNICA SUPERIOR

INGENIERÍA TÉCNICA DE TELECOMUNICACIÓN
TELEMÁTICA

PROYECTO FINAL DE CARRERA

DESARROLLO DE UNA INTERFAZ EN GOOGLE WAVE PARA UN ROBOT CONVERSACIONAL DE UNA PLATAFORMA PUBLICITARIA

AUTOR: FÁTIMA DE LA OSA BARRIGA

TUTOR: JESÚS ARIAS FISTEUS

Mayo de 2011

Título: *DESARROLLO DE UNA INTERFAZ EN GOOGLE WAVE PARA UN ROBOT CONVERSACIONAL DE UNA PLATAFORMA PUBLICITARIA.*

Autor: *FÁTIMA DE LA OSA BARRIGA*

Tutor: *JESÚS ARIAS FISTEUS*

La defensa del presente Proyecto Fin de Carrera se realizó el día 18 de Mayo de 2011; siendo calificada por el siguiente tribunal:

Presidente: *Norberto Fernández García*

Secretario *Jaume Barcelo Vicens*

Vocal *Katrin Achutegui Roncal*

Habiendo obtenido la siguiente calificación:

Calificación:

Presidente **Secretario** **Vocal**

Agradecimientos

En primer lugar deseo expresar mi agradecimiento al tutor de este proyecto, Jesús, por su interés y su disposición. Desde el primer día que le presenté el proyecto, lo acogió con buena voluntad, y desde entonces, ha mostrado una dedicación digna de alabar, fundamental para finalizarlo.

En segundo lugar, a mi grupo de trabajo en Telefónica I+D. En especial a Juan y a Torru, que son las personas que han estado desde el principio, dirigiendo mi proyecto y ayudándome con dedicación y esfuerzo en todas las dificultades que nos hemos ido encontrando.

Como no, agradecer a todos mis compañeros de la Universidad Carlos III, su apoyo, confianza y dedicación. En lo profesional, el trabajo en equipo siempre nos ha caracterizado, y sin su ayuda, hubiera sido imposible escribir este proyecto. En lo personal, he vivido una gran experiencia lejos de casa con ellos, y todos me han acogido con cariño y respeto. Muchos, ya se han quedado por el camino, aunque nunca olvidaré los momentos que pasamos, pero muchos otros, siguen siendo mis amigos, y con los que mantengo una especial amistad.

A mis amigas del pueblo, que desde que me vine, me han estado apoyando y han seguido manteniendo su amistad conmigo.

Por último, y el agradecimiento más importante es para mi familia y mi novio. A mi familia, por todo el esfuerzo económico y moral que ha tenido que realizar para que yo estudiara lejos de casa. A mi novio, por su apoyo, su comprensión y su confianza incondicional desde el principio. Ójala que todo este esfuerzo les sea recompensado para volver a estar juntos de nuevo.

Resumen

Las nuevas necesidades del mercado hacen que los usuarios que se conectan a Internet usen las redes sociales como principal canal de comunicación con sus amigos. La búsqueda de nuevos espacios publicitarios da origen a crear una aplicación que aproveche las capacidades de la red de un operador, para ofrecer al usuario interacciones útiles como el envío de SMS o MMS gratuitos a cambio de la recepción de publicidad.

En Telefónica I+D ya existe un robot capaz de mantener conversaciones con los usuarios que lo añaden como contacto a la plataforma de mensajería MSN. Este robot permite a los usuarios enviar SMS. Tiene ciertas limitaciones como el envío de MMS, o la incrustación de publicidad, ya que no se pueden añadir imágenes ni vídeos como elementos de una conversación.

Google Wave ofrece una plataforma de mensajería en tiempo real, con una de las interfaces más flexibles del mercado actual, ya que permite insertar y compartir conversaciones, imágenes, vídeos o páginas XHTML con otros contactos. El objetivo de este proyecto será adaptar el robot de Telefónica I+D para que interaccione con el usuario a través de la plataforma de Google Wave. Para ello, se creará un nuevo Robot según el API de Google Wave, que le de la funcionalidad al usuario de enviar SMS y MMS gratuitos a cambio de recibir en sus conversaciones con el robot, publicidad orientada a su perfil.

Índice general

Lista de Figuras

Lista de Tablas

Capítulo 1

Introducción

En este proyecto se realizará un estudio sobre la posibilidad de desarrollar una aplicación de mensajería en tiempo real, capaz de comunicarse con un usuario a través de Inteligencia Artificial, ofreciéndole la flexibilidad de adjuntar textos, imágenes y vídeos para permitirle el envío de SMS y MMS gratuitos a sus destinos preferidos, a cambio de realizar unos cuestionarios publicitarios sobre su perfil, gustos y preferencias.

1.1. Motivación del proyecto

Las redes sociales, la mensajería instantánea, y el correo electrónico son las principales herramientas de comunicación de la sociedad mundial. A estos usuarios no les importa visualizar mensajes publicitarios si a cambio la interfaz web de comunicación a la que acceden les permite comunicarse con sus allegados.

Las posibilidades que nos ofrecen estas tecnologías son infinitas. Podemos intercambiar archivos, vídeos o conversaciones con cualquier persona que esté en cualquier país del mundo, en tiempo real.

Pero, ¿Por qué tener tres interfaces, si podemos unificar todas en una?

¿Qué nos ofrece Google Wave?

1. Aplicación web de mensajería colaborativa en tiempo real, con la posibilidad de unificar los servicios que a día de hoy ofrece Google (chat, email e intercambio de archivos entre múltiples usuarios) en una única interfaz.

2. Esta interfaz es flexible y completa siendo capaz de detectar eventos como la adicción de imágenes o interacción con contenidos dinámicos, que nos permitirán el envío de MMS.

3. API de robots que permite tener una entidad capaz de interactuar con el usuario, así como con otras aplicaciones externas. Gracias a este punto, este proyecto podrá conectarse con otras dos plataformas ya creadas.

   ¿Qué nos falta que tengamos ya disponibles?

4. Un robot que sirva Inteligencia Artificial, es decir, que sea capaz de conversar con el usuario, independientemente de la hora o el país en el que esté. Gracias a este, el usuario podrá enviar SMS y MMS en el momento que quiera.

5. Una plataforma que nos permita el envío de esos mensajes a un terminal móvil.

6. Unos proveedores de publicidad que nos financien el envío de los mensajes a cambio de conocer el número de teléfono y los datos de cualquier usuario que se conecte a la aplicación.

Todo esto engloba a una aplicación capaz de:

- Comunicarse con el usuario y conversar con él que a través de cuestionarios, se recopilarán sus preferencias, y se almacene su perfil.
- Dependiendo de esas preferencias se incrustará cada cierto tiempo contenidos publicitarios, y así el usuario podrá conseguir puntos.
- Con estos puntos, el usuario podrá enviar gratuitamente MMS y SMS a los destinos que quiera.

1.2. Objetivos

El principal objetivo de este proyecto es la creación de una aplicación web dinámica, flexible e interactiva, a través de la plataforma de Google Wave, capaz de gestionar la comunicación del usuario con el robot de Inteligencia Artificial, conversar con él, y enviar SMS y MMS a distintos destinos. La aplicación web hará de intermediaria a través de Google Wave entre el usuario, el robot y la plataforma de envío de mensajes.

Este objetivo será cumplido si el prototipo es usado por todos los usuarios que utilizan a menudo Internet para comunicarse con sus amigos y compartir archivos. Estos usuarios, necesitan

de una aplicación que englobe todos los sistemas de comunicación que usan a menudo, en uno único y propio que unifique las redes sociales, la mensajería por correo electrónico, la mensajería por terminal móvil en tiempo real.

Si a esto le añadimos que el usuario podrá enviarles mensajes gratis, a cambio de dar su opinión acerca de ciertos anuncios publicitarios que a él le interesen podemos crear una aplicación práctica e interesante.

Para conseguir el objetivo principal se han tenido que abordar los siguientes temas:

1. Estudiar y elegir la tecnología más adecuada para desarrollar el proyecto sobre la plataforma de Google Wave, Java o Python.

2. Decidir una interfaz interactiva, flexible e intuitiva de cara al usuario que quiere enviar un mensaje e implementar la solución.

3. Planificar e implementar el protocolo de comunicación con las otras dos plataformas a usar en el prototipo, la Inteligencia Artificial y el envío de mensajes a móviles.

4. Elaborar la memoria y explicar los conceptos que se han tenido que abordar para realizar el proyecto.

1.3. Plan de Trabajo

Concretamente, las tareas que se desean cubrir con este proyecto son:

- Primera etapa: creación de una entidad (robot) que responda ante ciertos comandos de entrada del usuario.

- Segunda etapa: crear un rpbot que sea capaz de almacenar información en una base de datos, como es el teléfono y el perfil. Para ello, habrá que crear una serie de estados, según el usuario vaya contestando a las preguntas necesarias para la creación de su perfil.

- Tercera etapa: diseño, creación e implementación de un protocolo para que el robot se comunique con la plataforma de envío de SMS y MMS. Una vez que el usuario ya se ha autenticado, puede hablar con el robot.

- Cuarta etapa: creación de un *Gadget* para almacenar las preferencias del usuario, con este perfil, el usuario ya puede enviar mensajes.

- Quinta etapa: comunicación mediante AIML 2.4 con el robot. En esta fase, se creará un protocolo de interacción con esta interfaz externa, y realizará toda la lógica para la comunicación con la Inteligencia Artificial.
- Sexta etapa: incrustación en la interfaz web de vídeos e imágenes para la realización de cuestionarios y opiniones.
- Séptima etapa: creación de un *Gadget* que muestre los vídeos almacenados dependiendo de las preferencias del usuario.
- Octava etapa: envío de SMS y MMS, creación de un *Gadget* para la petición de los datos del MMS al usuario. Estos datos son: el número del destino, el texto y la imagen a insertar. Una vez realizado, hay que crear toda la lógica para el envío de esta imagen al servidor y la comunicación con la plataforma de mensajería.
- Novena etapa: gestionar el recuento de puntos dependiendo del número de veces que el usuario responda a preguntas de contenidos publicitarios. Según vaya respondiendo podrá conseguir nuevos puntos para enviar SMS y MMS gratuitos. Sin estos, solo podrá conversar con el robot de Inteligencia Artificial.

1.4. Organización de la memoria

Esta memoria se divide en los siguientes capítulos:

- En el capítulo *Estado del Arte* se hablará sobre el entorno de alojamiento de aplicaciones Appengine, así como el API de robots y *Gadgets* y la aplicación servidora, consistente en un robot que sirve Inteligencia Artificial y una plataforma para el envío de SMS y MMS. Se detallarán las características y oportunidades que ofrece cada uno.
- En el capítulo *Funcionamiento de la aplicación* se explicará de forma práctica los pasos que debe usar cualquier usuario en el caso de querer usar esta aplicación, así como las entidades que en cada momento resuelven sus peticiones.
- En el capítulo *Especificación de Requisitos* se detallarán todas las necesidades del sistema,
- En el capítulo *Arquitectura del sistema* se detallarán todas las entidades que forman el prototipo, que engloba aplicación web y servidora, así como sus interconexiones. Se mostrarán

en detalle las conectividades y la arquitectura del sistema

- En el capítulo *Diseño del prototipo* se explicarán todas las decisiones involucradas en la creación de los distintos protocolos de comunicación con las interfaces externas, así como todas las decisiones clave para el intercambio de información de distinto tipo.
- En el capítulo *Diseño del prototipo* se explicará la aplicación de las distintas tecnologías para el intercambio y uso de la información.
- En el capítulo *Implementación* se detallarán los algoritmos más importantes utilizados en el desarrollo de la aplicación.
- En el capítulo *Pruebas* se realizará un estudio sobre los resultados obtenidos de las pruebas realizadas, así como de la robustez de las distintas partes que engloba el sistema. Se crearán distintos test, uno de validación de datos de entrada por parte del usuario y otro de pruebas de carga sobre todas las entidades del sistema.
- En el capítulo *Conclusiones y Trabajos Futuros* se repasarán los objetivos logrados pero también aquellas cosas que son objetivamente mejorables en el proyecto. También habrá una sección sobre trabajos futuros donde se propondrán diferentes aplicaciones para desarrollar.
- En el apéndice *Manual de instalación, configuración y mantenimiento* se explicarán las herramientas necesarias para poder programar las aplicaciones y para poder ejecutarlas, así como su soporte. También se expondrá qué configuraciones son necesarias para poder ejecutar el prototipo.
- En el apéndice *Manual de usuario* se expondrán los pasos que un usuario tiene que realizar para llegar al objetivo de enviar un SMS o MMS. Éstos, compondrán las acciones que el usuario realiza desde que se identifica en la aplicación hasta que envía el mensaje

Capítulo 2

Estado del arte

En este capítulo se hablará sobre las múltiples tecnologías que se han empleado para desarrollar el prototipo explicando aquellos aspectos que han resultado más relevantes en la implementación de las dos aplicaciones: Robot Wave con Gadgets y Adaptación del Robot Movistar. Para ello, primero se hablará sobre el servidor de aplicaciones App Engine que es el único compatible con el API de Google Wave y después se entrará en detalle en el API de Robots y el de Gadgets. Por último, se explicarán brevemente las distintas tecnologías usadas en la aplicación tanto para la representación y el almacenamiento de datos, como para la comunicación de las dos aplicaciones.

2.1. Google App Engine y su entorno de desarrollo

Google AppEngine [5] permite el alojamiento y ejecución de aplicaciones web en la infraestructura de servidores de Google, sin la necesidad de instalar o configurar el servidor de aplicaciones web.

Por tanto, se pueden crear aplicaciones web, con soporte para persistencia, ejecución de contenido dinámico, e interacción con otras aplicaciones. Para ello se establecen unos límites de ancho de banda y almacenamiento. En el momento en que la aplicación necesite ampliar esos límites, deberá pagar por el incremento.

Se ofrece una infraestructura segura para las aplicaciones. Soporta balanceo de carga y escalabilidad automática. Permite además la ejecución en dos entornos de programación: Java y Python, y ofrece protocolos y tecnologías comunes para desarrollar en cualquiera de estos entor-

nos.

A continuación se detallan las características de App Engine.

2.1.1. Elección del servidor de aplicaciones

Con App Engine se consigue que la aplicación esté desplegada en un entorno escalable. En el caso en el que se quieran ampliar sus capacidades o que se reciban un gran número de peticiones no habrá problema de robustez del sistema, ya que App Engine, balancea esas peticiones a la aplicación, y se encarga de gestionar el espacio del sistema, tanto de almacenamiento como de memoria.

Google App Engine es lo más cercano a un ideal de entorno web: fácil de desarrollar, fácil de desplegar y fácil de depurar, aportando una página web de gestión de la aplicación donde se pueden visualizar en tiempo real los registros, las entidades de la base de datos, las estadísticas en rendimiento y el versionado de la aplicación.

Por tanto, las principales ventajas de App Engine frente a un alojamiento tradicional son las siguientes:

- Se encarga del balanceo de peticiones. Si se optara por un alojamiento tradicional, se tendría que configurar el servidor de aplicaciones para realizar esta tarea.
- Actualmente sólo se permiten desplegar los Robots Wave subidos a AppEngine, ya que es necesario comunicarse con su API a través del protocolo Wave.
- Es escalable ante futuras ampliaciones del robot o ante un gran éxito de la aplicación.
- Almacena hasta 10 versiones de la aplicación desplegada, siendo posible cambiar de versión en el momento en que se quiera.
- Tiene una herramienta Web de gestión de la aplicación donde se pueden visualizar: estadísticas de rendimiento, entidades de persistencia, registros, configuración de la aplicación...

En conclusión, AppEngine nos ofrece una manera rápida y fácil de desarrollar, desplegar y depurar.

2.1.2. Persistencia: JPA o JDO

El API de persistencia del SDK de App Engine [1] está orientado a la creación de aplicaciones web, que hagan uso de una base de datos ligera para las consultas y modificación de las entidades que la componen. El API del SDK ofrece una capa de abstracción entre la base de datos y las entidades creadas por el usuario delegando a AppEngine las tareas la gestión y administración de los accesos a las entidades.

Se entiende por entidad un objeto JAVA que tiene propiedades y métodos de acceso a ellas. Pueden existir propiedades que sean referencias a otras entidades, pero AppEngine no se basa en un esquema relacional. AppEngine hace uso de una arquitectura distribuida para el almacén de las entidades que provoca que cualquier base de datos declarada en la aplicación web, sea escalable.

Para ello, incluye en su SDK los APIs de Objetos JDO [1] y de interfaces JPA [2], ofrecidas por el proyecto DataNucleus[3] que integra las dos APIs para la creación y consultas de entidades en la base de datos.

Para identificar las propiedades, la clave primaria y las consultas, se hace uso de las anotaciones Java. Existen diferentes tipos de anotaciones. Simplemente se escriben antes de la propiedad o del método que identificar. Las consultas y modificaciones de esas propiedades se realiza mediante los métodos *Getter* y *Setter* de los objetos JAVA.

Cada consulta o modificación se establece en el contexto de una transacción. Esto garantiza que no haya problemas debidos a concurrencia en las consultas o modificaciones de las propiedades de los objetos. La gestión y administración de las transacciones son abstractas para el programador, de estas tareas se encarga el API de persistencia del servidor.

AppEngine establece unos límites en el tamaño de la base de datos. Estos límites afectan al número máximo de entidades, al tamaño máximo de cada entidad o el número máximo de resultados obtenidos de una consulta. El programador puede consultar estas cuotas en la pantalla de administración, explicada en la sección 2.1.6 de AppEngine, y puede aumentarla pagando el precio correspondiente.

Para ver en más detalle las cuotas disponibles para la base de datos, consulte la sección Almacén de datos [14] de la página que describe las cuotas para AppEngine.

[1] http://www.oracle.com/technetwork/java/index-jsp-135919.html(10/10/2010)
[2] http://www.oracle.com/technetwork/articles/javaee/jpa-137156.html(10/10/2010)
[3] http://www.datanucleus.org/(10/10/2010)

2.1.3. Memcaché

Memcaché [4] es un servicio que ofrece App Engine a las aplicaciones que se almacenan en su infraestructura, para recopilar, y almacenar datos en memoria que puedan ser utilizados de manera frecuente por solicitudes consecutivas. Cuando varias peticiones preguntan por un dato que ya ha sido pedido, se almacenan los resultados, y se devuelven desde la memoria de Memcaché en lugar de obtenerlos de la aplicación web. Para peticiones sucesivas de un mismo dato, pueden solicitarse a Memchaché solo si los datos no han caducado o permanecen invariables. Por tanto, es común almacenar en Memcaché los archivos estáticos o los resultados de consultas a la base de datos. Por defecto, el tiempo de caducidad de los datos es ilimitado, hasta que la memoria se va llenando, y por tanto, se van expulsando los datos que menos se han utilizado recientemente. Cada petición a Memchaché se va contabilizando, existiendo una cuota máxima para Memcaché. Para consultar en detalle las cuotas máximas, consulte la sección cuotas.

2.1.4. Contenido estático

Como se explica en la sección de archivos estáticos de Google AppEngine [14], se ponen a disposición pública todos los archivos estáticos que contenga el Robot Wave. Estos archivos son los correspondientes a: imágenes, *Gadgets*, vídeos, hojas de estilo CSS [4], librerías JS [5]... Para que todos estos archivos se visualicen públicamente, deben estar contenidos dentro de la carpeta WAR de la estructura del proyecto, a excepción de los JSPs y los archivos que estén dentro del directorio WEB-INF, que seguirán teniendo acceso privado.

2.1.5. Entorno de ejecución JAVA

AppEngine ejecuta aplicaciones con la Máquina Virtual de Java 1.6. De todas formas, es compatible con cualquier compilador del SDK de AppEngine que utilice Java 5 o versiones posteriores a él. Se utiliza el descriptor de implementación *web.xml* para definir el mapeo de peticiones hacia cada Servlet, y para definir los parámetros iniciales de la aplicación. Para la

[4] Hojas de Estilo en Cascada (Cascading Style Sheets), es un mecanismo simple que describe cómo se va a mostrar un documento en la pantalla, o cómo se va a imprimir, o incluso cómo va a ser pronunciada la información presente en ese documento a través de un dispositivo de lectura

[5] Extensión de ficheros con contenido JAVASCRIPT

ejecución de Servlets y JSPs, Appengine utiliza el estándar Java Servlets [6]. Existirán ciertas restricciones al API de la librería de ejecución de Java. Para consultar en detalle las clases permitidas, compruebe la lista de accesos permitidos a la JRE [7].

2.1.6. Panel de Control de AppEngine

El Panel de Administración de AppEngine proporciona acceso completo a todos los aspectos y características de la aplicación almacenada en la infraestructura de AppEngine. Para ello, solo es necesario acceder a través de el enlace `https://appengine.google.com/` y autenticarse en la cuenta de Google. Este panel se divide en diferentes secciones, que son las siguientes:

- *DashBoard*: Describe el uso de los parámetros de arquitectura *hardware* de los servidores que alojan la aplicación. Ejemplos de estos parámetros son: ancho de banda saliente-entrante, tiempo de procesamiento, almacenamiento utilizado, número de peticiones recibidas...

- Detalles de Cuotas: muestra el uso de los diferentes parámetros de cuota de los distintos servicios que ofrece AppEngine, y avisa de si alguno ha superado los valores máximos. Estos servicios son los siguientes: memcaché, base de datos, peticiones HTTP/HTTPs, email, extracción de URL, manipulación de imágenes..

- Instancias: se refiere a todas las tablas que existen para la base de datos de la aplicación. Muestra tanto las columnas, como los valores de los atributos de cada entidad. Ofrece la posibilidad de editar los valores de la tabla, así como borrar filas o añadir nuevas.

- Registros: contiene una tabla de filas con todos los registros que han sido creados para la aplicación. Ofrece una búsqueda avanzada, para seleccionar registros por tipo de nivel: “Debug”, “Error”, “Warning”, “Request”..., o por período de fechas.

- Tareas CRON [8]: Se muestra una tabla con la configuración de las tareas programadas por la aplicación. Pueden ser modificadas, añadidas, o borradas.

[6] `http://www.oracle.com/technetwork/java/index-jsp-135475.html` (07/03/2011)

[7] `http://code.google.com/intl/es/appengine/docs/java/jrewhitelist.html` (07/03/2011)

[8] cron es un administrador regular de procesos en segundo plano que ejecuta procesos o guiones a intervalos regulares (por ejemplo, cada minuto, día, semana o mes)

- Administración de la Base de datos: se pueden consultar tanto las entidades como las estadísticas de la base de datos, o el directorio de *Blobs* [9] creados.
- Administración de la aplicación: permite visualizar o cambiar parámetros como el nombre de aplicación, los servicios habilitados, la expiración de *cookies*, la habilitación o deshabilitación de la base de datos o la aplicación.
- Permisos: permite habilitar cuentas de Google que pueden acceder a la aplicación.
- Versiones: permite activar las diferentes versiones que existen de la aplicación. Google permite hasta un máximo de 10 versiones por aplicación. En cualquier momento, se puede seleccionar la versión que se quiere mostrar de la aplicación, y ésta se desplegará automáticamente.
- Registros de administrador: representa todos los registros sobre la subida y gestión de la aplicación.

2.1.7. Soporte de tareas programadas mediante CRON.

AppEngine ofrece un API para la creación y configuración de tareas programadas [2] mediante CRON. Gracias a ello, se pueden crear tareas que se ejecuten en un determinado momento automáticamente. Esto se resuelve en que se puede configurar una tarea que realice una petición HTTP al robot con una serie de parámetros. Cuando el robot reciba la petición determinará si es una petición a través de CRON, y ejecutará el manejador correspondiente para este tipo de acción. La configuración de tareas de AppEngine permite establecer los siguientes parámetros:

- Fecha y hora de envío de la petición HTTP.
- Descripción de la tarea.
- URL a la que se realiza la petición HTTP.

La configuración de tareas se especifica en un archivo con formato XML, dentro del contexto de la aplicación. Se puede visualizar el estado de las tareas así como el tiempo y el número de veces que se ha invocado en el Panel de Control de AppEngine 2.1.6.

[9] Binary Large OBjects, objetos grandes binarios son elementos utilizados en las bases de datos para almacenar datos de gran tamaño que cambian de forma dinámica. No todos los Sistemas Gestores de Bases de Datos son compatibles con los BLOB.

2.1.8. Sistema propio de registros

Google AppEngine tiene un sistema propio de registros que permite crearlos, configurarlos y visualizarlos. Para ello, se basa en "java.util.logging" [15], un API de Java que permite la creación y gestión del sistema de registros de una aplicación. Sólo es necesario definir la configuración de registros en un archivo denominado "log4j.properties", que irá almacenado dentro del contexto. Una vez que la aplicación está subida en el entorno de AppEngine, estos registros se podrán monitorizar a través de la Consola de administración 2.1.6. Los niveles que se pueden establecer y visualizar son los siguientes:

- DEBUG: nivel interesante para el programador.
- INFO: nivel que informa sobre distintos comportamientos de la aplicación.
- WARN: nivel que aviso de estados inesperados pero no erróneos.
- ERROR: nivel que indica errores de la aplicación.
- CRITICAL: nivel que indica errores graves de la aplicación.

2.1.9. Conexiones HTTP/HTTPS

Google AppEngine permite realizar conexiones externas HTTP y HTTPs a otros recursos o aplicaciones de entornos externos. Para ello, ofrece un API que implementa la interfaz "java.url.connection" [6] que ofrece la funcionalidad para realizar conexiones HTTP entrantes y salientes. El Robot Wave, por tanto, nunca realizará conexiones directas a aplicaciones externas, para garantizar la seguridad y la robustez del entorno de aplicaciones. Los puertos de estas conexiones son estándar y deben ser el 80 para HTTP y el 443 para HTTPs, nunca se podrán realizar conexiones a otros puertos que no sean estos. Los métodos permitidos para estas peticiones son GET, POST, PUT, HEAD y DELETE. Además se podrán enviar cabeceras o parámetros en las solicitudes, existiendo algunas limitaciones en cuanto a las cabeceras que pueden ser enviadas por motivos de seguridad. Por ello, no se podrán modificar las siguientes:

- Content-Length: indica el tamaño del cuerpo del mensaje y por tanto, el tamaño reservado en memoria para la recepción de los datos. Si se modifica puede producir "desbordamiento" en los "buffers" de recepción de la máquina que recibe la petición.

- Host: indica la dirección y puerto de la máquina a la que se le envía la petición.
- Referer: permite definir al cliente, para el beneficio del servidor, la dirección del recurso desde la cual se realiza la petición. A la hora de realizar conexiones externas, esta dirección es necesaria para que AppEngine gestione las conexiones externas, en el caso de modificarse, podría suponer atentar contra el protocolo Wave de AppEngine.
- Vary: define qué cabeceras de consultas debería tener en cuenta un mecanismo de caché a la hora de crear su clave. En el caso de modificarse, se estaría atentando con el servicio de Memcaché de Google AppEngine 2.1.3
- Via: cabecera añadida por los proxys y nodos intermedios para indicar el camino de una petición y así conocer el camino de vuelta. En el caso de modificarse, podrían provocarse direcciones inválidas para el entorno de AppEngine.
- X-Forwarded-For: dirección del proxy intermedio entre cliente y servidor.

2.2. Protocolo Wave

La aplicación desarrollada del prototipo para Google Wave se basa en un robot, que actúa como cualquier contacto normal de la interfaz de mensajería, diferenciándose sólo de un contacto normal, en que éste se comunica con el Servidor Wave para el envío de eventos, y los manejadores para controlar las acciones que generan esos eventos. Google Wave denomina a este protocolo de comunicación robot-Servidor Wave: “Wire”. La comunicación entre las dos entidades se realiza a través de peticiones HTTP en formato JSON, donde se incluye la información de los eventos y las operaciones. El API de Robots es “activa”, es decir, el robot recibe peticiones de eventos del usuario que le envía el Servidor, y el Robot Wave realiza operaciones ante ciertos eventos en los que ha expresado su interés de escucha, es decir, para que el robot reciba eventos de un determinado tipo, éste debe declararlos en un archivo de configuración, y crear una acción para ese evento. Se distinguen por tanto dos tipos de comunicación:

1. JSON: mensajes que envía el Servidor Wave al Robot Wave con eventos que el usuario ha realizado.
2. JSON-RPC: mensajes que envía el Robot Wave al Servidor Wave, con operaciones a realizar por el Servidor como respuesta a los eventos enviados.

La información incluida en los mensajes del Servidor Wave al robot es la siguiente:

- "Events": que contiene información relacionada con el evento.
- "Blips": indica los datos referentes a la ventana de conversación donde se ha producido el evento.
- "Wavelet": incluye la información de la conversación completa como el usuario o el creador.
- "Metadatos": vinculados al evento, normalmente información sobre el propio robot, como robotAddress.

Para más información sobre el contenido de mensajes del protocolo Wave, consulte la bibliografía [10] o [11].

2.3. Google Wave: API de Robots y *Gadgets.*

El API de Google Wave [7] pretende ofrecer a los desarrolladores las herramientas para extender la interfaz del cliente Wave, y proporcionar nuevas funcionalidades al usuario.

La programación efectiva mediante las API de Google Wave requiere la comprensión de algunos conceptos básicos de las Waves. Se entiende como Wave, a una conversación encadenada que mantiene el usuario tanto con participantes humanos, como con robots. El elemento "Wave", mantiene información sobre los participantes y sobre el estado de la conversación. Gracias e ello se pueden insertar dinámicamente nuevas ventanas en una conversación o elementos como botones, imágenes... El Servidor Wave reacciona ante estos eventos, y los envía al resto de participantes. Todos los usuarios que pertenecen a una "Wave", pueden modificarla en tiempo real, y por tanto el Servidor hará llegar las modificaciones en tiempo real al resto de usuarios. Una misma conversación puede encadenar varias conversaciones, denominadas óndulas, es decir, se puede crear un día una "Wave", y dos semanas después añadir una nueva conversación a esa "Wave", o modificar el estado de las conversaciones anteriores.

Aparece otra entidad que se denomina documento. Un documento es el componente que contiene los mensajes que se transmiten en una conversación, así como ciertos "metadatos", que contienen información sobre esos mensajes.

El "Api de Wave" se compone por tanto de varias APIs, dependiendo del tipo de extensión que se quiera crear. Hay dos tipos de modelo:

1. Extensiones: es el API que aporta las herramientas para la manipulación de "Waves". Gracias a ellas, por ejemplo, se pueden crear robots que cambien el color de la letra mientras el usuario está escribiendo, o que traduzca en tiempo real una conversación que el usuario esté manteniendo con un usuario de otro país, o que corrija las faltas ortográficas en tiempo real del usuario.

2. Inserciones: los desarrolladores pueden incrustar el contenido de Google Wave, directamente en su aplicación o página web, blog o foro, haciendo que la aplicación interactúe con el API de Wave.

En este proyecto se usará el API de extensiones para crear un robot que simule a un usuario real, y por tanto una conversación. Para más información sobre todo lo que ofrece Google Wave, véase el vídeo de la presentación de Google Wave [10].

2.3.1. Robots

En esta sección se explican los conceptos relativos a un Robot del API de Extensiones de Wave [12], así como la manera en la que interactúan con el protocolo y se identifican en la interfaz de Google Wave.

Identidad del robot

Las bibliotecas cliente Java y Python permiten diseñar un robot sin necesidad de tener que gestionar el envío y recepción de peticiones con el Servidor Wave, o el procesamiento de los datos, gracias al protocolo Wire, que resuelve los eventos y las peticiones, y el acceso al Servlet que contiene la implementación del Robot.

Google Wave solo interactúa con robots creados por el API de AppEngine. Se espera que en etapas posteriores, el API de Wave se haga escalable y permita cualquier robot desplegado en cualquier Servidor Web. AppEngine, identifica a las aplicaciones a través de una URL con el siguiente formato: *<idAplicacion>.appspot.com*. El protocolo Wave, resuelve esa URL a IP, y se comunica con el Robot a través de ella, mediante peticiones HTTP.

[10] http://www.youtube.com/watch?v=v_UyVmITiYQ(07/03/2011)

Perfil del Robot

Los robots pueden proporcionar de manera adicional metainformación [11], para que puedan ser identificados por otros usuarios (y por otros robots). La información contenida en este perfil se muestra en una ficha de perfil al hacer clic en el avatar del robot:

Los robots pueden tener asociada metadatos acerca de su perfil, para que puedan ser buscados y añadidos por otros usuarios como contactos de su plataforma de mensajería. Por ello, el API de robots proporciona una implementación sobre el perfil, que contendrá la siguiente información:

- Name: indica el nombre del robot en la plataforma que se muestra cuando se añade como contacto.
- Imagen: corresponde a la imagen que se muestra para el contacto.
- URL: es una dirección web hacia la página que contiene la información del perfil. Para ello, se devuelve un JSON con todos los metadatos del robot.

2.3.2. *Gadgets*

Se entiende como *Gadget* una miniaplicación de fácil acceso y que ofrece contenido dinámico al usuario. Son ejecutados por un motor de *Gadgets* y son de libre acceso por todos los usuarios. Se ofrecen gratuitamente en Internet. Google Wave tiene un API especial para *Gadgets* [13], que permite incrustarlos directamente en una conversación sobre la interfaz Wave e interactuar con ellos. Además, se pueden insertar *Gadgets* creados a través de otros entornos. Las ventajas que ofrece crear un *Gadget* a través del API de *Gadgets* de Google Wave son las siguientes:

- Permite un acceso más directo a la gestión de estado de un *Gadget*, es decir, el ciclo de vida de este.
- Se puede obtener y establecer información de los participantes de una conversación sobre la interfaz Wave.
- Se pueden pasar parámetros y realizar peticiones directamente al Robot.
- Los *Gadgets* del API de Wave pertenecen a la conversación a la que se han añadido. Por ello, toda la información asociada a él se almacena en la "Wave". Esto no ocurre con los

[11] datos que constituyen el perfil

Gadgets creados por otro Contenedor, donde la información pertenece al creador de la página donde está alojado el contenedor.

El contenido de un *Gadget* está basado en un fichero XML con distintas etiquetas que componen su estructura y contienen la información básica de cualquier página de tipo XHTML.

Una de las características más importante de los *Gadgets* del API de Wave es que el ciclo de vida de este es compartido por todos los participantes de la Wave con la que están interactuando. Por ello, si un usuario cambia su estado, el resto de participantes lo verán reflejado. El estado se basa en un Mapa con pares nombre-valor, donde todos los participantes pueden acceder a esos valores, modificarlos o borrarlos. Si dos participantes acceden a la vez a propiedades del estado de un *Gadget*, el API de Wave se encarga de gestionarlo sin que se produzcan problemas de concurrencia.

2.4. O2BOT y la Lógica AIML.

El Robot Movistar es un agente autónomo que permite a sus usuarios descubrir y acceder a servicios Movistar desde un cliente Window Live Messenger.

El Robot utiliza el lenguaje AIML[12] con el fin de dar al usuario la apariencia de mantener una conversación real con un humano. En la actualidad dispone de una base de conocimiento de más 5000 expresiones, clasificadas según temática: servicios Movistar, entretenimiento, saludos, etc.

El objetivo del Robot Wave será el de sustituir el cliente Window Live Messenger por el cliente de Google Wave. Para ello, necesitaremos añadir un nuevo módulo al Robot Movistar, que desvíe las peticiones entrantes desde Google Wave, a un módulo encargado de recibirlas, procesarlas y comunicarse con la IA del Robot Movistar y con la plataforma de mensajería.

Los servicios Movistar accesibles desde el Robot son los siguientes:

- Descargas de juegos, tonos y logos disponibles en el Catálogo Movistar.
- Envío de SMS.
- Guiado del usuario hacia Canal Cliente ante consultas de saldo, recargas, etc.

[12] http://www.alicebot.org/aiml.html(07/03/2011)

- Descubrimiento de servicios y aplicaciones de Movistar (ej. El Robot proporciona el enlace de descarga de la aplicación ante consultas relacionadas con el escritorio Movistar).
- Ventana de actividades.
- Integración con Emoción Web.
- Servicio demostración de localización de la posición del usuario ("localízame").

Asimismo se dispone de un mecanismo de invitación de nuevos usuarios. Por ejemplo, si se escribe "invitar amigo@hotmail.com" el Robot automáticamente añadirá al nuevo usuario a su lista de contactos.

2.5. MIB: envío de SMS y MMS.

El MIB es la plataforma de mensajería de Telefónica. Actualmente soporta las siguientes funcionalidades:

- Mensajería corta (SMS): UPC/EMI y SMPP
- Mensajería Multimedia (MMS): EAIF/MM7
- Envío de WAP-Push: PAP
- Mensajería integrada (MM7+): Permite el envío de mensajes cortos, multimedia, WAP-Push desde un único protocolo unificado.

2.6. El esquema UriData

Este esquema es una especificación estándar de la RFC2397 [13] que trata de definir el protocolo para inyectar datos que corresponden a contenidos binarios en el campo URL de determinados elementos HTML, como si fueran obtenidos de una fuente con almacenamiento externo. Para ello, el formato que debe tener este campo debe ser el siguiente:

data:[<mediatype>][;base64],<data>. Donde:

- Mediatype: corresponde al tipo de contenido (image/jpeg, text/plain, pplication/vnd-xxx-query)

[13] http://www.rfc-editor.org/rfc/rfc2397.txt(07/03/2011)

- Base64: corresponde a la codificación de los caracteres del contenido.
- data: "array de shorts" correspondiente a los datos binarios de contenido.

Las ventajas y desventajas que nos ofrece el uso de este protocolo son las siguientes:

1. Se decrementa el número de conexiones externas. Por tanto, existe menor sobrecarga en la red.
2. El contenido binario, puede ser obtenido de distintas fuentes, ya sean archivos dentro de la propia aplicación o Blobs almacenados en una base de datos.
3. Hay que tener especial cuidado con la codificación del contenido binario. Ya que para ser inyectado, debe estar codificado en base64. Por tanto, hay que tener la lógica para codificar y decodificar "array de bytes".
4. La codificación en Base64 aumenta en una media del 30 % el tamaño de los datos.
5. Las URLS de los elementos HTML con contenido inyectado en URI data suelen ser muy extensas. Por ello, existen limitaciones que no permiten incrustar contenidos superiores a un determinado número de "bytes".
6. Puede haber tipos de contenido no soportados por navegadores.

En este punto hay que tener en cuenta que el elemento "data" debe tener una extensión menor que la establecida por la especificación de cada navegador.

2.7. Anotaciones

Cada ventana de una conversación de Google Wave contiene un "Documento" donde se pueden incrustar diversos elementos, desde el más básico: texto, hasta el más complejo: un *Gadget*. Por ello, el API de Google Wave ofrece un mecanismo que permite insertar etiquetas en ese documento. Las etiquetas permiten dar formato al texto que va insertado en el documento, definiendo un punto inicial del documento donde comenzar el formato y un punto final. El API de Google Wave permite además crear anotaciones personalizadas a usar en la aplicación.

Las anotaciones que existen y que pueden ser utilizadas, se muestran en la tabla 2.1

Nombre de la anotación	Descripción	Ejemplo
style-color	Estilo, color de texto	style-color=rgb(150,75,0)
style-fontFamily	Estilo, nombre de fuente	style-fontFamily=arial
style-fontSize	Estilo, tamaño de fuente (en puntos)	style-fontSize=18
style-fontStyle	Estilo (cursiva)	style-fontStyle=italic
style-fontWeight	Estilo, grosor de la fuente (negrita, cursiva)	style-fontWeight=bold
style-textDecoration	Estilo	style-textDecoration=none
style-verticalAlign	Estilo, alineación vertical	style-verticalAlign=center
link-manual	Hipervínculo	link-manual=http:–www.google.com-
link-Wave	ID de Wave	link-Wave=googleWave.com!w+d7NJm4nWF
lang	Idioma del texto incluido	lang=en
conv-title	El título de la óndula (normalmente la primera frase, a menos que se especifique aquí explícitamente)	conv-title=Introduction

Tabla 2.1: *Anotaciones de la interfaz Wave*

2.8. JSON

JSON especifica un formato estándar y ligero para el intercambio de datos entre diferentes fuentes. Gracias a su carácter "ligero" permite al desarrollador leerlo y generarlo fácilmente y a las máquinas, procesarlo e interpretarlo. Es independiente del lenguaje de programación, por lo que es un formato flexible y portable a cualquier entorno o aplicación. Sólo es necesario utilizar las librerías necesarias dependientes del lenguaje de programación para su procesamiento y creación y además, éstas son muy pequeñas y ligeras para la aplicación. Un JSON se basa simplemente en un conjunto de caracteres que representa pares nombre/valor, donde el valor, puede respresentar: objetos, arrays, listas, mapas... con los caracteres codificados en Unicode.

En este proyecto, se hace uso de la librería GSON [14], que implementa la funcionalidad para procesar una cadena de caracteres JSON, y convertirla a un objeto JAVA.

2.9. Trabajos relacionados y Competidores.

En este apartado se hace una comparativa con proyectos y plataformas similares a Google Wave, y qué ventajas e inconvenientes ofrece frente al resto.

Google Wave intenta inventar un nuevo modo de comunicaciones y colaboración on-line:

[14] http://code.google.com/p/google-gson/(07/03/2011)

la mensajería instantánea en tiempo real que incluye en una interfaz común: email, IM[15], wiki, SNS [16], y que ofrece una gran experiencia de usuario. Por ello, reúne las siguientes características claves:

- Mensajería instantánea en donde las conversaciones que mantiene el usuario se visualizan con el resto de usuarios que pertenecen a la "Wave" en tiempo real, es decir, si uno de ellos escribe en una ventana y el resto de usuarios están añadidos y conectados a la conversación, podrán visualizar en tiempo real (letra por letra) lo que el usuario escribe. Si no lo están, podrán ver los mensajes cuando se conecten a la plataforma.

- Las interacciones en tiempo real permiten que haya contactos que realmente sean desarrollados con el API de Wave, que cambien el estado de los documentos que hay dentro de una conversación: imágenes, textos, vídeos .... Esto permite desarrollar múltiples aplicaciones que tendrán un gran mercado. Por ejemplo: se puede crear una aplicación que traduzca en diferentes idiomas en tiempo real lo que escribe un usuario a otro.

- Colaboración con una interfaz flexible en la que se pueden realizar las siguientes tareas con el resto de usuarios: nuevas planificaciones de eventos, edición compartida de documentos, edición de mapas, encuestas, manteniéndose el estado compartido de los componentes, es decir, mientras un usuario está añadiendo una imagen a la conversación, otro puede estar modificando su nombre.

- Protocolo y tecnologías abiertas que hacen que cualquier desarrollador pueda crear una aplicación para Google Wave, y compartirla con sus contactos, en este proyecto, el Robot Wave, que permite interactuar con otras aplicaciones de la red externa, como es el caso del Robot Movistar.

- Correo electrónico: todas las conversaciones creadas, se mantienen almacenadas en la interfaz. En cualquier momento un usuario puede abrir una conversación anterior, visualizarla y modificarla. Los nuevos mensajes e interacciones le llegarán al resto.

Por ello, analizando lo que aporta Google Wave surge la necesidad de explicar similitudes y diferencias con las siguientes aplicaciones que son competidoras:

[15]IM: Instant Messagging
[16]Social Network Service

- Toqbox: es una plataforma muy similar a Google Wave. Uno de sus puntos fuertes es que el chat se convierte en el principal modo de comunicación con los usuarios. Permite tener varios hilos y compartir contenidos como imágenes, vídeos o llamadas en tiempo real. Una de sus desventajas es que es una plataforma privada que no ofrece un API para que se puedan crear aplicaciones sobre ella, lo que la limita a que los contactos no puedan interactuar o utilizar aplicaciones externas diversas según sus necesidades, solo las que le ofrece la propia plataforma.

- Gmail: ofrece una interfaz que contiene *chat* y *email* en la misma, para los contactos que pertenezcan a ella. Las limitaciones que ofrece frente a Google Wave son las siguientes:

  1. No permite enviar elementos dinámicos a través del *Chat* como imágenes, mapas, vídeos o Gadgets.
  2. No permite colaboración en tiempo real con otros usuarios: edición de documentos, planificación de eventos...
  3. No permite añadir contactos que sean aplicaciones como el Robot Wave.
  4. Es una interfaz orientada a enviar, recibir y almacenar, no a compartir y colaborar modificando o añadiendo contenidos en tiempo real.

  Similitudes a Google Wave son las siguientes: se pueden añadir varios contactos a una conversación y se pueden compartir elementos como imágenes o vídeos, pero solo a través de *Email*.

- Google Buzz: es una herramienta incluida dentro de Gmail inspirada en Google Wave, que ofrece colaboración en forma de red social con los contactos de la red. Las desventajas que ofrece Buzz es que no tiene chat en tiempo real, por lo que las notificaciones se reciben vía email, tampoco permite colaboración en tiempo real, por lo que si solo se quieren compartir fotos, y no editar documentos o planificar eventos, Google Buzz es mejor opción.

- ShareFlow: este servicio ofrece una plataforma de colaboración y de edición de documentos pero que no está bien integrado con el chat. Esto provoca grandes limitaciones al usuario, ya que para compartir cualquier contenido con otro usuario, debe usar el email. Otra de las desventajas, es que es una plataforma privada, que no ofrece la participación a otros

usuarios para crear su implementación o conectarse a otras aplicaciones, como es en nuestro caso, el Robot Movistar.

- SAP-Constellation: esta plataforma se basa en la compartición de decisiones y contenidos sobre un proyecto, la colaboración entre usuarios para la toma de decisiones pretende ser un sustituto del Email, y por ello, es un fiel competidor de Google Wave. Una de las limitaciones es que no está integrado con el chat, lo que no le permite compartir verdaderamente contenidos con otros usuarios en tiempo real, está orientada a ser una plataforma de almacenamiento de contenidos, para que el resto de usuario puedan observar lo que ha hecho uno de ellos, debe introducirse en la plataforma y abrir la actividad. Otro de los problemas, es que se basa en código no abierto, en beta.

- SNS como Facebook o Tuenti: ofrecen una interfaz que incluye *Chat* y almacenamiento de contenidos que pueden ser visualizados por el resto de contactos. También ofrece correo electrónico, pero muy reducido.

  1. Ninguna de las SNS, permiten enviar elementos dinámicos a través del *Chat* como imágenes, vídeos o Gadgets.
  2. Tampoco permiten mostrar en verdadero tiempo real lo que el usuario está escribiendo.
  3. El corro electrónico, los comentarios a publicaciones, imágenes o contenidos que el usuario publique, están limitados a un máximo de 140 caracteres, mientras que Google Wave es ilimitado en este aspecto.
  4. Google Wave, desde el comienzo, ofreció protocolo y tecnología abiertos, para que cualquier desarrollador pudiera crear sus propias aplicaciones, mientras que las SNS ofrecen un API limitado, y protocolo y tecnologías menos abiertas al desarrollo de nuevas aplicaciones.

  Similitudes a Google Wave son las siguientes: los usuarios pueden contactar con aplicaciones e interactuar con ellas, pero para ello serán redirigidos a la página de la aplicación y por tanto están fuera de la interfaz de la SNS.

  Una de las ventajas que tienen es que están orientadas a compartir, y por ello, reciben notificaciones de los nuevos cambios de sus contactos aunque ellos no interactúen con ese usuario.

- ccBetty, Etacts, Xobni, Rapportive: son herramientas desarrolladas para mejorar las plataformas de mensajerías de otros fabricantes como Gmail o Outlook. El motivo por el que no son competencia para Google Wave, es que ellos dependen de cómo esas tecnologías están desarrolladas para incluir las herramientas. Si quisieran participar en el proyecto de Google Wave, sólo tendrían que crear una extensión que ofreciera la funcionalidad a través del protocolo abierto de Wave.

Capítulo 3

Funcionamiento de la aplicación

En este capítulo se abordará en primer lugar el funcionamiento del sistema de forma general, explicando todas las acciones que un usuario puede realizar sobre la aplicación web, y cómo interactúa esta para mostrarle los resultados al usuario final.

Se va a detallar por tanto, el proceso completo desde que un usuario de Google Wave, se identifica en la plataforma, hasta que envía MMS y MMS a un destino.

3.1. Acceso a la plataforma

Para acceder a la plataforma de Google Wave [1] se necesita un usuario y contraseña del grupo "premium" de desarrolladores. Será necesario pedir una invitación.

Una vez que el usuario entra en la plataforma aparece la interfaz completa. Esta engloba varias partes: zona de contactos y navegación, zona de mensajería, y zona de nueva creación de mensajes. Todas estas, se muestran en la Figura 3.1

La descripción de cada zona es la siguiente:

- Zona de contactos y navegación: corresponde al área donde se puede agrupar y ordenar los mensajes según unos filtros determinados como: entrada, enviados, con ajustes. Tiene además un panel para buscar mensajes según los patrones elegidos.

- Zona de mensajes: es correspondiente a todas las conversaciones llevadas a cabo por el usuario, es decir, cuando se elige una de ellas, se despliega una lista anidada de mensajes

[1] https://Wave.Google.com/a/Wavesandbox.com/(07/03/2011)

Figura 3.1: *Zonas de la Plataforma Web de Google Wave.*

escritos por él, y en los que participan el resto de usuarios. Los *Gadgets* se vuelven a ejecutar en su punto de inicio. Incluye también un conjunto de opciones propias de mensajería de email como mover a otra carpeta, borrar, enviar a spam....

- Zona de creación de mensajes: es la parte correspondiente a la creación de una nueva ola y la adición de contactos a ella. Al inicio está vacía, solo aparece un botón para abrir una nueva conversación. Cuando se pulsa sobre él aparece una nueva ventana tipo mensajería instantánea con opciones como: cambiar color, insertar adjunto, insertar contacto... Si se elige la opción de añadir contacto se despliega automáticamente una lista con todos los usuarios de Google Wave que tienes añadidos. Si no está en la lista el usuario que buscas, se puede escribir su dirección en una ventana específica para ello.

3.2. Adición del robot

El Robot Wave englobará la aplicación cliente del sistema y será el intermediario entre el usuario y el servidor sobre la plataforma de Google.

La primera acción que se puede realizar sobre él es invitarlo a una nueva conversación. Para añadir a cualquier contacto, seleccionamos “Add Contact”, y después para que se abra una conversación con él, se pulsa sobre “New Wave”. El robot, la aplicación cliente del prototipo, se comporta por tanto, como un contacto más.

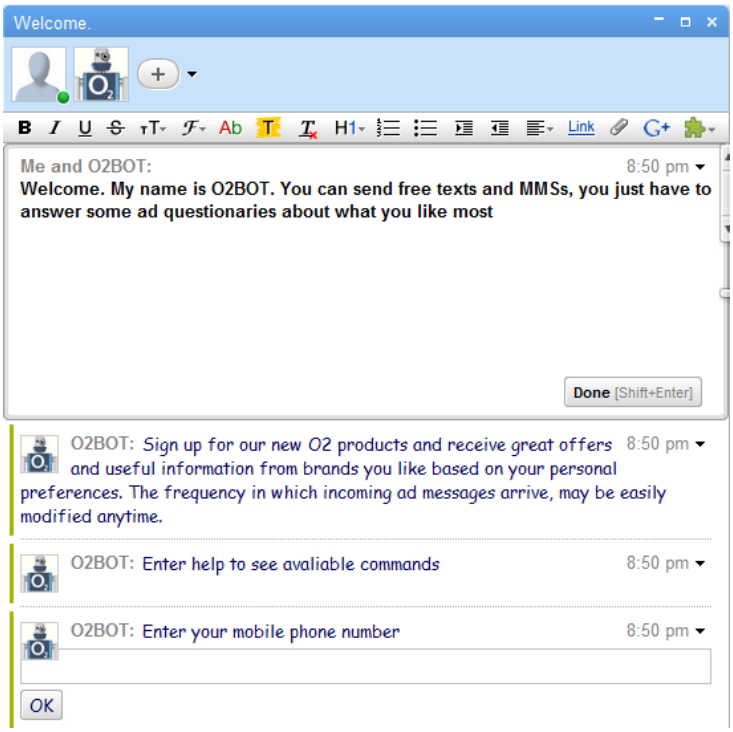


Figura 3.2: *Registro en la aplicación cliente.*

3.3. Registro del usuario

Si es la primera vez que el usuario lo añade o nunca se ha identificado en la aplicación cliente, el robot, le dará la bienvenida a la aplicación, le explicará en qué consiste, y le mostrará el formulario de registro, como se muestra en la figura 3.2.

Una vez que se ha insertado el móvil, el robot se comunicará con la aplicación servidora para que le envíe un mensaje al usuario con la contraseña. Cuando el usuario la reciba, debe insertarla en el campo "contraseña", correctamente. En este momento, el usuario ya puede conversar con el robot de Inteligencia Artificial, como se muestra en la Figura 3.3

3.4. Conversación con el robot

A partir de este momento, el usuario podrá mantener una conversación completa con el robot. Éste será capaz de mantener el estado de la conversación y tener "memoria", esto es, responder a las siguientes preguntas teniendo en cuenta las respuestas anteriores.

Si el usuario está registrado en la aplicación podrá en cualquier instante, cerrar la ventana del

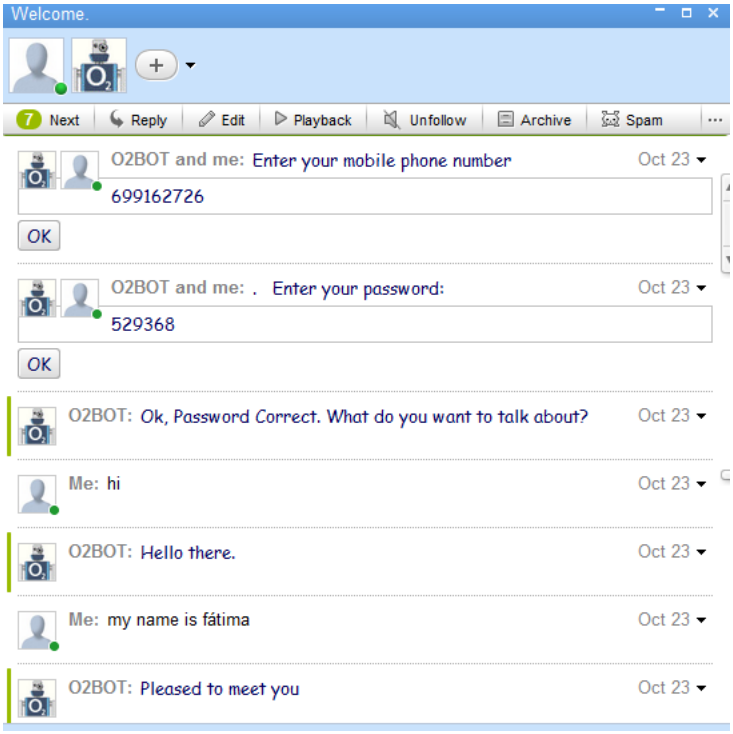


Figura 3.3: *Inserción de la contraseña.*

robot, y abrirla en el momento que quiera. Una vez que lo añade, éste detecta que está registrado en la aplicación, le da la bienvenida, y le pregunta en qué puede ayudarle. Este es un ejemplo de una conversación 3.4

3.5. Envío de MMS

En el momento en que el usuario quiera enviar un MMS solo tiene que escribirlo. El funcionamiento es el siguiente:

- El usuario escribe una sentencia con la palabra MMS o TXT. Por ejemplo: "I want to send a MMS"
- El Robot Wave detecta la palabra y si es el caso, reemplaza MMS por TXT.
- Si el estado del usuario es sólo registrado se le muestra el *Gadget* que pregunta por sus preferencias, que se explicará más detalladamente en la Sección 3.5.
- Si el usuario ya ha respondido al *Gadget* de perfilado anterior se le muestra un cuestionario que va creando dinámicamente la Inteligencia Artificial según las respuestas del usuario

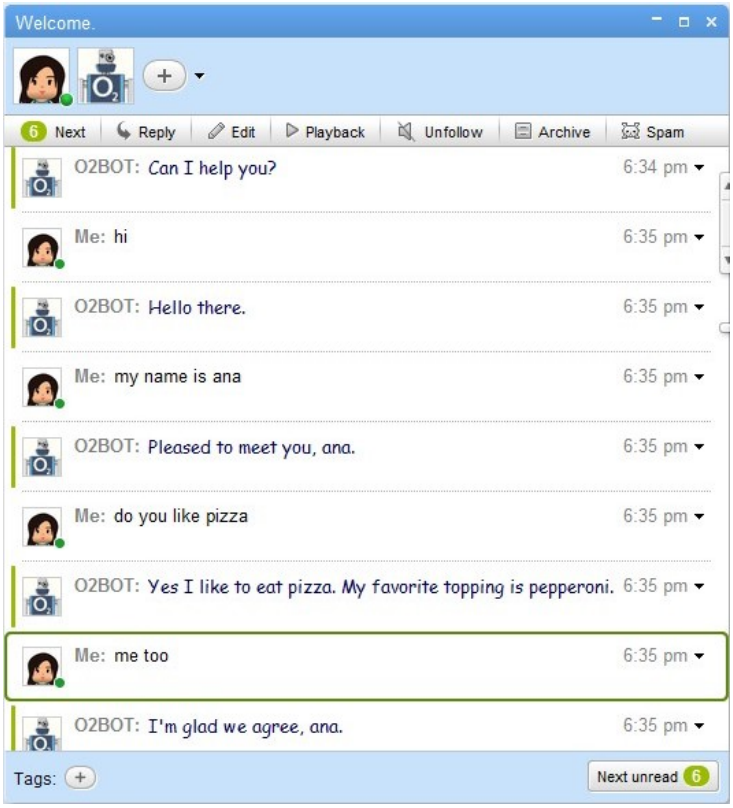


Figura 3.4: *Conversación de un usuario registrado.*

Figura 3.5: *Elección de las preferencias.*

y sus preferencias anteriores. El tema del cuestionario estará definido por defecto y será "Deportes" Se pueden consultar más detalles sobre este tema en la Sección 3.5.

- Una vez rellenados estos dos, se guardará este estado, el usuario tendrá 5 puntos, y podrá enviar el SMS, escribiendo: "SMS 99999999 texto".

- Cuando el usuario haya respondido los dos cuestionarios anteriores, en cualquier momento, ya sea en la misma ventana de conversación o en otra distinta, para enviar un mensaje solo tendrá que escribir la frase anterior, y si no tiene puntos suficientes, se le preguntará sobre un vídeo que tendrá que ver con sus preferencias, explicado con más detalle en la próxima Sección 3.5.

Gadget de Perfilado

Este *Gadget* está implementado bajo el API de *Gadgets* de Google, explicado en la Sección 2.3. Es el encargado de pedirle información al usuario de una forma dinámica mediante el uso de javascript.

El objetivo de la primera pantalla del *Gadget* es preguntar al usuario sobre sus preferencias ante cuatro temas establecidos. Se le muestran por tanto, en contenido XHTML-JAVASCRIPT, cuatro imágenes, cada una, reflejará un tema. Los temas definidos por defecto son: comidas y bebidas, deportes, películas y juegos. Se le pide al usuario, que elija varias de estas opciones. En el momento en el que selecciona una, aparece el nombre del tema elegido en la imagen, y al final del *Gadget*, se le muestra el nombre en color rojo. La imagen 3.5 representa este ejemplo.

Una vez que el usuario elige las preferencias y pulsa continuar se le muestra una pantalla distinta con un cuestionario sobre la frecuencia con la que quiere recibir mensajes de publicidad,

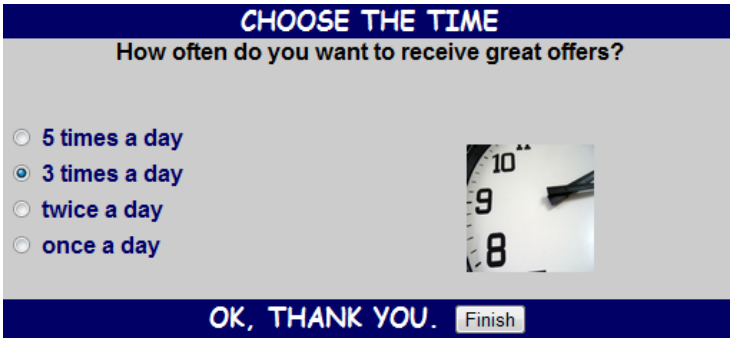


Figura 3.6: *Elección de la frecuencia.*

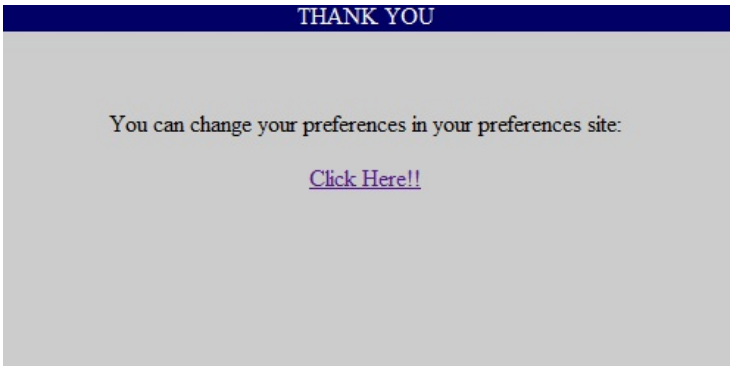


Figura 3.7: *Link de cambio de preferencias.*

y por tanto, conseguir puntos. Este asunto será interesante para una de las futuras evoluciones del robot, que consistirá en insertar publicidad mediante el método *PUSH* en lugar de *PULL*, descrito detalladamente en la Sección 9.2.2.

Este caso es ejemplificado en la Figura 3.6

Se le mostrará por tanto un conjunto de frecuencias definidas por defecto que serán: una vez al día, dos, tres o cinco. Cuando el usuario termine este cuestionario le llevará hasta una página final que contendrá un enlace a una web que contiene otro cuestionario para cambiar las preferencias y la frecuencia elegidas en cualquier momento, como podemos ver en la Figura 3.7

El formulario para el cambio de las preferencias se muestra en la Figura 3.8

Cuestionario sobre su perfil

Una vez contestado el *Gadget de preferencias*, si el usuario escribe MMS se le mostrará un cuestionario sobre preguntas referentes al tema de "Deportes". Las preguntas serán seleccionadas por la Inteligencia Artificial, dependiendo de la respuesta a la anterior pregunta, así se conseguirá

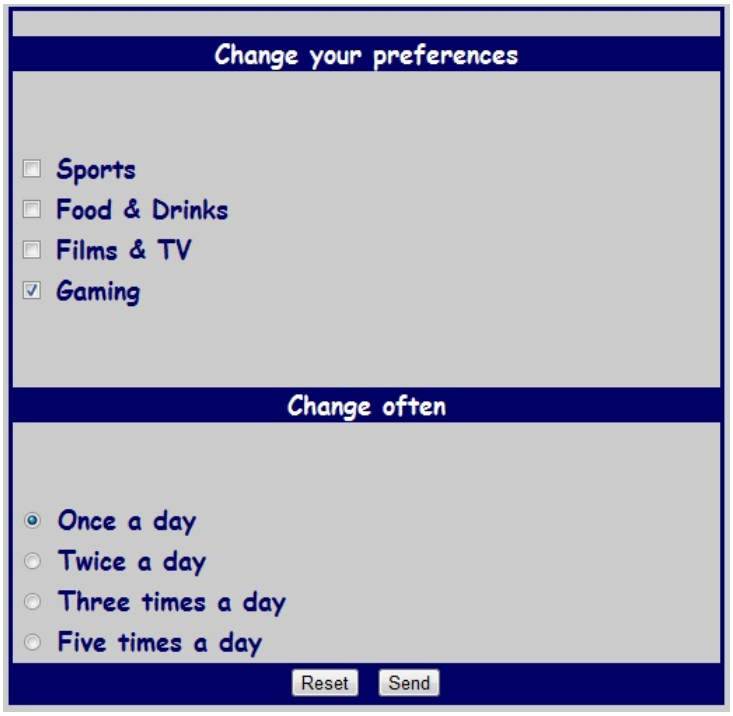


Figura 3.8: *Formulario de cambio de preferencias.*

un perfilado más completo del usuario. Las respuestas del usuario además se almacenarán en formato de registros en el servidor, y se construirán estadísticas del perfil del usuario.

Veamos un ejemplo de cuestionario en la Figura 3.9

Gadget-Vídeo sobre una de sus preferencias

Si el usuario ha contestado tanto el *Gadget de perfilado* como el cuestionario de la Inteligencia Artificial y escribe una sentencia con el texto mensaje, lo primero que hace el Robot Wave es comprobar el número de puntos que tiene el usuario y si éstos son suficientes para enviar un MMS o MMS. Para cualquier tipo de mensaje, el número de publi-puntos necesario es 5. Si el usuario tiene menos de 5 puntos tendrá que contestar su opinión sobre un vídeo incrustado en un *Gadgetde Google Wave*.

Esta pantalla contendrá un reproductor de vídeo flash procedente de una página que transmite vídeos mediante *Streaming*. En este caso, usaremos Youtube, explicado detalladamente en la Sección 7.4.4.

El vídeo que el Robot Wave muestra al usuario va a representar una de las preferencias que el usuario ha elegido en el cuestionario. El diseño y el protocolo de este punto se explica

Figura 3.9: *Cuestionario de perfilado.*

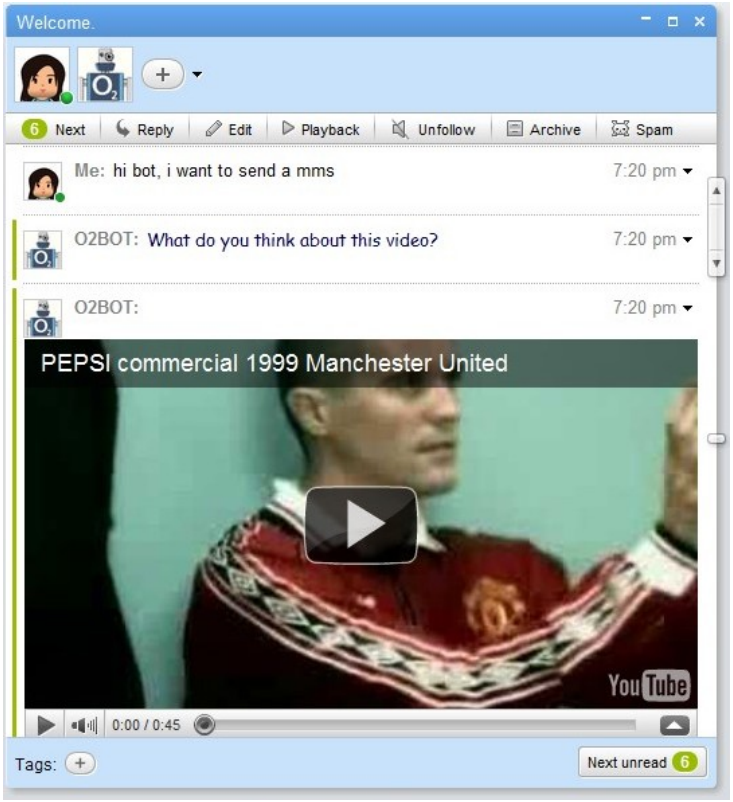


Figura 3.10: *Vídeo de Opinión I.*

detalladamente en la Sección 3.

Cuando el usuario responda con su opinión en la ventana de conversación del vídeo se le añadirán 5 publi puntos a su cuenta y ya podrá enviar MMS y MMS.

Veamos un ejemplo en la Figura 3.10

3.6. Envío de MMS

Cuestionario para el envío del MMS

Para enviar un MMS, el usuario, tenga puntos o no deberá responder con su opinión al vídeo explicado anteriormente en la Sección 3.5 y cuando conteste, se le mostrará un cuestionario incrustado en las ventanas de conversación, pidiéndole los datos necesarios para el envío. Éste será parecido al cuestionario de tipo registro y estará basado en dos pasos. El primero de ellos, es de la petición de los datos de envío: teléfono y texto. El segundo contendrá un *Gadget* que tendrá el botón de petición de un fichero en XHTML.

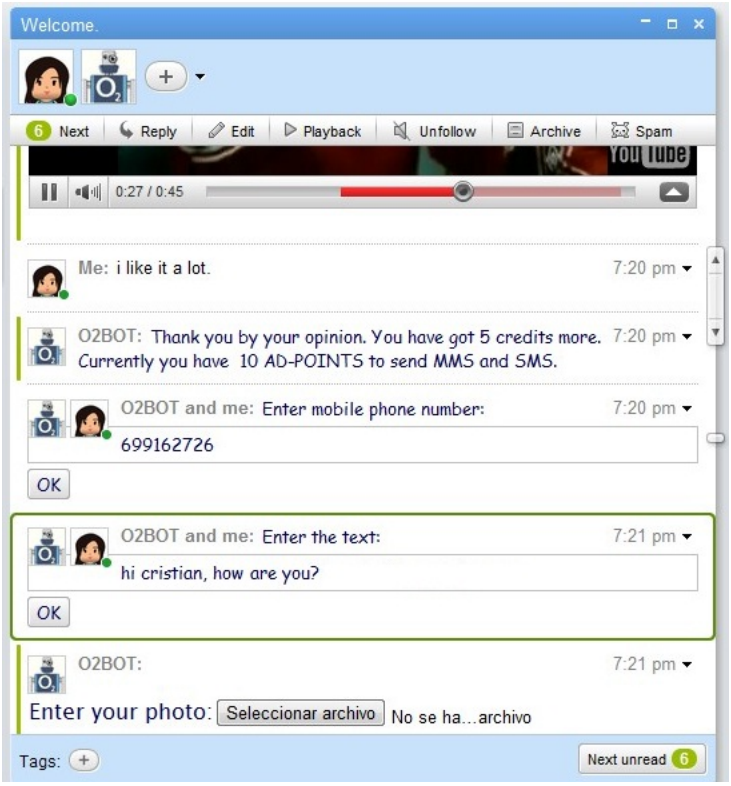


Figura 3.11: *Vídeo de Opinión II.*

El momento en que el usuario visualiza el vídeo y contesta con su opinión se puede ver en la Figura 3.10. Y el momento en que el usuario rellena los datos de envío del MMS, en esta figura 3.11

Cuando el usuario introduzca la imagen sobre el *Gadget* se le mostrará en HTML una vista previa del MMS a enviar que contendrá, los datos introducidos (texto y número) y la imagen. Si esta imagen es demasiado grande en tamaño, se reescalará manteniendo las proporciones para que quepa en la pantalla del *Gadget*.

La vista previa del mensaje se puede ver en la Imagen 3.12

Si el usuario pulsa enviar se enviará el MMS al destinatario y se mostrará un mensaje de confirmación sobre la misma pantalla, como se puede ver en la Figura 3.13

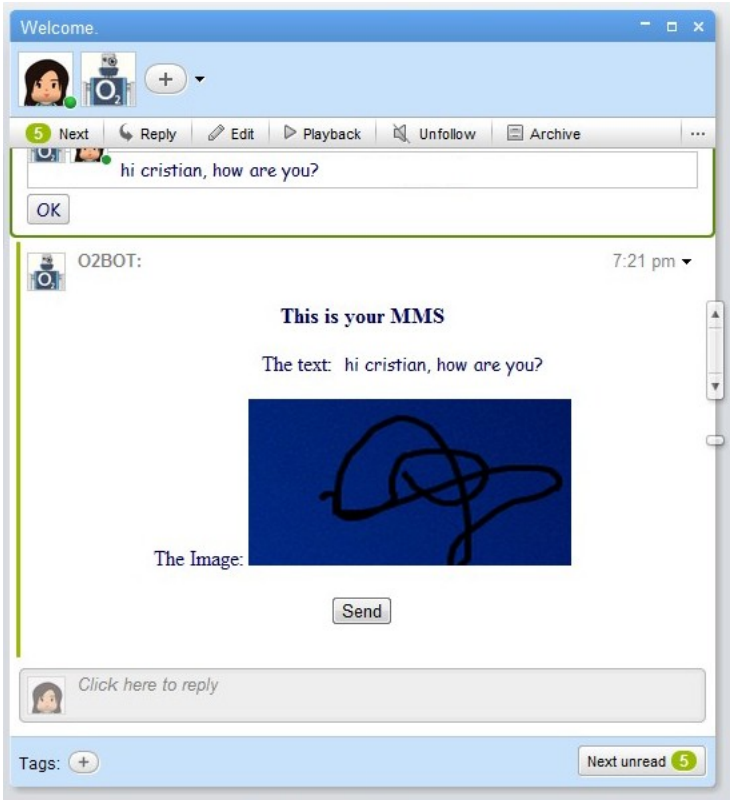


Figura 3.12: *Vísta previa del MMS.*

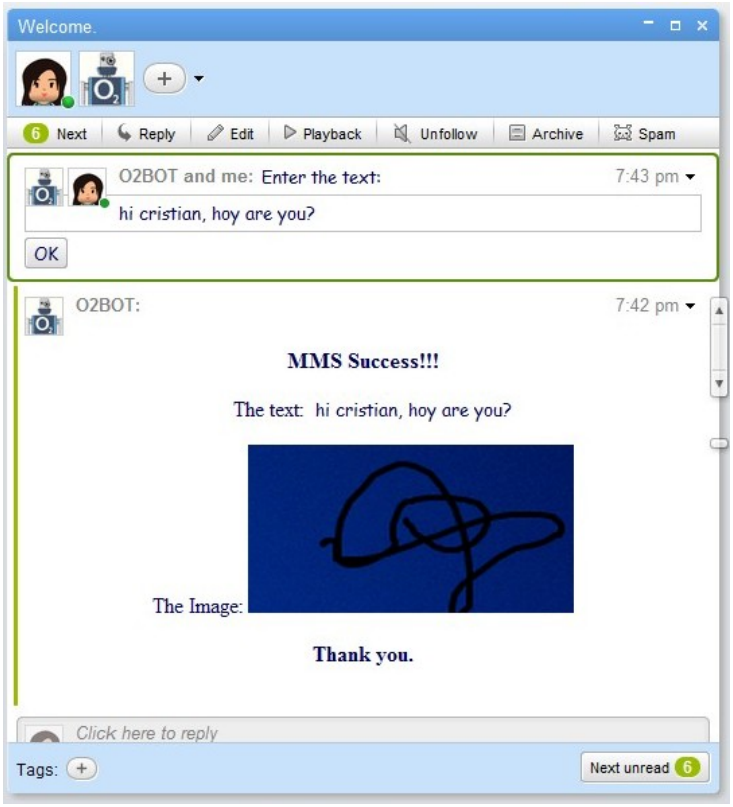


Figura 3.13: *Confirmación del MMS.*

Capítulo 4

Especificación de requisitos

4.1. Servidor de aplicaciones web AppEngine

La aplicación cliente del sistema, es decir, el Robot Wave necesita interactuar con el API de Google Wave para ofrecerle al usuario la interfaz de acceso al robot, capaz de interpretar e interactuar con sus acciones. En esta versión del API de robots es necesario que la plataforma cliente sea desplegada en el ámbito de AppEngine. Para futuras versiones del API de robots esta arquitectura podrá ser desplegada en cualquier servidor de aplicaciones que ejecute una Máquina Virtual de Java. Por tanto, sin el entorno de aplicaciones AppEngine, será imposible lanzar el robot. Para más información, consultar la referencia bibliográfica [8].

4.1.1. La aplicación web

El Robot Wave no tendrá la misma estructura que una aplicación J2EE, ya que éste se crea a través de la especificación del "Framework GWT" [1]. El uso de GWT junto al complemento de Eclipse para el desarrollo de aplicaciones de AppEngine facilitará la ardua tarea de desarrollar el proyecto del Robot Wave, así como su depuración y su subida al servidor de aplicaciones.

Para la comunicación con el Robot Movistar se necesitará que el servidor de aplicaciones pueda compilar *Servlets* o intercambiar contenido con páginas HTML. Este es el caso de acceder al formulario de cambio de preferencias 3.8. AppEngine utiliza el estándar Java Servlet Technology [2] para realizar esta tarea y por lo tanto, compilar JSPs, o acceder a archivos estáticos como en

[1] http://code.google.com/intl/es-ES/webtoolkit/(10/10/2010)

[2] http://www.oracle.com/technetwork/java/index-jsp-135475.html(07/03/2011)

cualquier aplicación J2EE.

4.1.2. Gestión de sesiones

En el caso en que el Robot Wave necesite el soporte de sesiones en la aplicación, o el paso de atributos entre Servlets, Google AppEngine permite su gestión de sesiones, activando esta propiedad en uno de sus archivos de configuración.

4.2. Servidor de aplicaciones Apache

La aplicación servidora del prototipo creada por Telefónica I+D fue diseñada para ser robusta, escalable y reusable sobre cualquier ámbito. Al ser modular ofrece la posibilidad de extenderse hacia cualquier tipo de cliente, en este caso, de Google Wave. Por tanto, si se quiere dar soporte a peticiones procedentes de esta plataforma, lo único necesario será crear un nuevo módulo encargado de recibirlas, interpretarlas y procesarlas para devolver una respuesta igual que se hace para un cliente de Messenger.

Este módulo será compilado y desplegado junto con la aplicación; sólo será necesario añadir las librerías que necesite y cambiar las órdenes del ANT [3] para que se compile con las nuevas librerías.

4.3. Almacenamiento

En esta sección se hablará sobre las necesidades de almacenamiento que necesita tanto el Robot Wave.

4.3.1. Base de Datos

En el caso en que el Robot Wave necesite almacenar datos de forma persistente se deberá usar la tecnología de Persistencia del API de Google Wave, ya que AppEngine no permite conexiones con base de datos externas, ni bases de datos ligeras como “Sqlite” dentro del contexto de la aplicación, por motivos de seguridad.

[3] Apache Ant es una herramienta usada en programación para la realización de tareas mecánicas y repetitivas, normalmente durante la fase de compilación y construcción (build)

4.3.2. Almacenamiento de contenido estático

Esta sección se refiere a todos los archivos estáticos necesarios que estarán almacenados en el contexto de la aplicación. La estructura de GWT tiene una carpeta denominada "WAR" que corresponde a la parte pública de la aplicación. Todos los archivos que estén localizados en ella, podrán ser vistos desde el exterior de la aplicación.

Los contenidos que pueden ser necesarios a almacenar en el contexto del Robot Wave son los siguientes:

- Imágenes: puede haber casos en que se necesiten mostrar imágenes en la interfaz.
- Páginas XHTML: se refiere a los formularios que puede necesitar el Robot Wave para la autenticación y la recogida de datos por parte del usuario.
- Almacenamiento de *Gadgets*: se refiere al contenido XHTML-JAVASCRIPT a incrustar en la interfaz de Google Wave, y que interactúa con el robot. Aunque estén colgados de la parte pública del contexto de la aplicación, solo podrán ser ejecutados e interpretados en un entorno de ejecución de *Gadgets* de AppEngine, ya sea en la interfaz Web o el entorno de Pruebas de *Gadgets* de AppEngine.

4.4. Conexiones síncronas externas HTTP

AppEngine limita el acceso a redes externas ya que puede perjudicar la seguridad del entorno, y el de la propia aplicación. Por ello, solo permite conexiones externas HTTP a equipos que escuchen en el puerto 80 o HTTPS a los que escuchen en el 413, y solo a través del API de conexiones de Java *Java.net* [6].

Se necesitará tener acceso a conexiones externas en los siguientes casos:

- Se necesitará el soporte de conexiones HTTP salientes y entrantes desde IPs públicas de Internet, para mostrar contenidos externos como vídeos o imágenes.
- Conexión al Robot Movistar: permitirá la comunicación con el prototipo creado por Telefónica I+D para el envío de SMS y comunicación con la IA.
- Conexión con los *Gadgets* y formularios a crear dentro de la aplicación del Robot Wave para la interacción con el usuario.

4.4.1. Interfaz web

Uno de los objetivos del Robot Wave es que ofreciera al usuario una interfaz más dinámica e interactiva que el MSN, para conversar y enviar MMS al robot O2 Movistar. Esto se consigue gracias a su API, que permite la inserción de imágenes, elementos de formulario, y código HTML en cada ventana de conversación. Gracias a esta interfaz, se podrán enviar imágenes en un SMS o mostrar formularios de publicidad con vídeos e imágenes de las marcas. Por tanto, la interfaz de Google Wave nos proporciona:

- Soporte de formularios: se refiere a todos los elementos que se pueden incrustar en la ventana de la interfaz Wave como "Button", "RadioButton", "Input", "TextArea", "Image"...
- Ejecución de *Gadgets*: la ejecución de este tipo de complemento, proporcionará al robot una manera más dinámica y vistosa de interacción, ya que un *Gadget* es prácticamente una página XHTML-JAVASCRIPT que se ejecuta en tiempo real. Además los *Gadgets* creados según el API de Wave, permiten que el usuario interactúe con la conversación en la que está involucrado, y por tanto, que tenga acceso a todas las capacidades del robot.
- Ejecución de vídeos por STREAMING, a través de conexiones HTTP.

4.4.2. Control de errores y seguimiento de peticiones

Este es uno de los puntos más críticos de la aplicación ya que permite en tiempo real, visualizar el estado de la aplicación, así como realizar un seguimiento de todas las peticiones. GWT nos proporciona una configuración básica para los registros de una aplicación AppEngine. Una vez que la aplicación se despliegue se podrán visualizar desde el Panel de Control de la aplicación.

La configuración de los registros es flexible; se pueden configurar los distintos niveles que se quieren mostrar así como la especificación de cada registro. De todas formas, si se elige la configuración básica, desde el panel de registros de AppEngine, se pueden filtrar o seleccionar el nivel que queremos visualizar. La figura 4.1 muestra un ejemplo de la visualización de los registros con los siguientes filtros:

- Nivel: "Debug".
- Palabras contenidas: "image".
- Fecha inicial: "Hora actual".

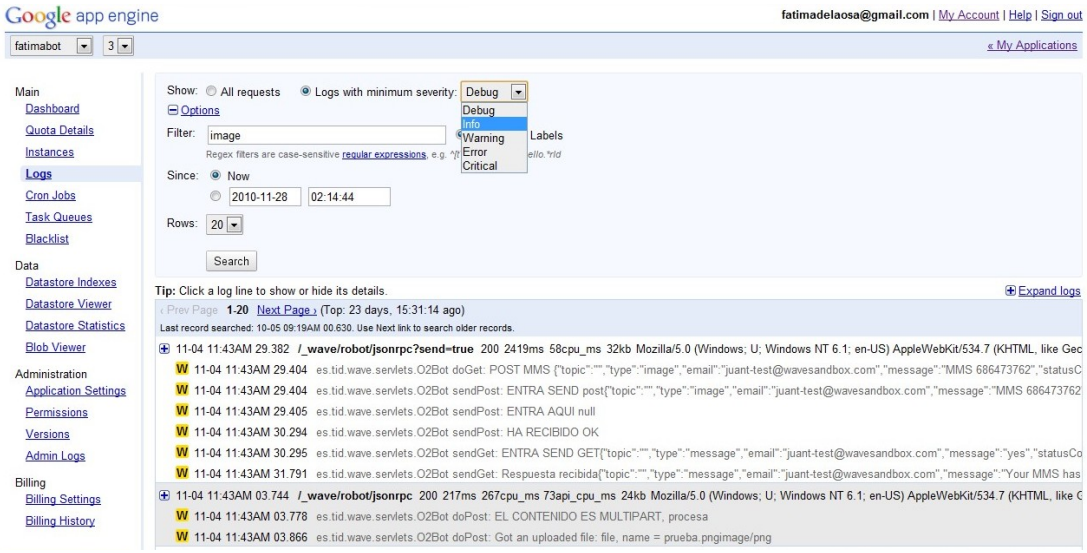


Figura 4.1: *Vista de los registros desde el Panel de Control.*

- Número de registros: “20”.

Capítulo 5

Arquitectura del sistema

En este capítulo se detallará la arquitectura del prototipo desde tres puntos de vista diferentes: software, hardware y red. En cada punto de vista se profundizará en las distintas partes que conforman la arquitectura, cliente y servidor, y se explicarán en detalle, cada una de las interfaces que hacen posible la comunicación entre ambas. Concretamente se describirán en profundidad las partes que se han diseñado y desarrollado en este proyecto, además de explicar cómo interactúan con el prototipo ya creado por Telefónica I+D: El Robot Movistar. Por tanto, las entidades que se han diseñado y desarrollado en el proyecto son:

1. Robot Wave: se ha diseñado la aplicación completa que interactúa con el API de Google Wave, y con la interfaz. Es la encargada de recibir los eventos que el usuario realiza, procesar las acciones correspondientes, comunicarse con el robot Movistar para enviar SMS, MMS, y obtener respuestas a las preguntas del usuario a través de la IA.

2. Gadgets: son los encargados de ofrecer dinamismo a la aplicación. Para ello, se han creado tres *Gadgets* del API de Google Wave Gadgets que ofrecen al usuario formularios e interactúan con él, y se comunican con el Robot Wave.

3. Adaptación del Robot Movistar a la interfaz Wave: el prototipo creado por Telefónica I+D estaba doseñado para detectar los eventos desde la interfaz de Microsoft Messenger. En este proyecto, se ha tenido que diseñar y desarrollar un módulo que permite integrar este robot con el robot de Google Wave.

4. El Modelo AIML para el Robot Wave: la Inteligencia Artificial creada por Telefónica I+D permitía responder a las preguntas del usuario ante una conversación normal o sobre temas

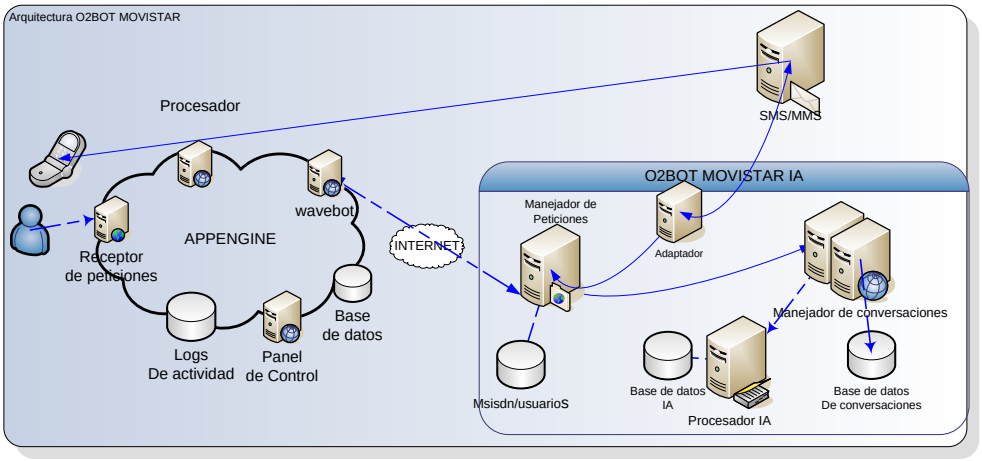


Figura 5.1: *Arquitectura SoftWare del sistema.*

de Movistar Emoción, esto se sigue manteniendo con la interfaz Wave, pero en este proyecto se ha tenido que diseñar y desarrollar un nuevo módulo, para el formulario de publicidad personalizada que se le muestra al usuario, donde se le pregunta sobre marcas de publicidad, dependiendo de sus gustos y preferencias, como se muestra en la figura 3.4

5.1. Arquitectura software

La figura 5.1 representa la arquitectura software del sistema, que engloba los siguientes elementos principales:

- Receptor de peticiones: se refiere a la interfaz web de Google Wave que es la que interactúa con el usuario escuchando, atendiendo eventos y enviándoselos al Procesador de peticiones.
- Procesador de peticiones: es el encargado de recibir los eventos e interpretarlos según el API de robots de Wave. Una vez procesados hará peticiones al Robot Wave, pasándole el evento y sus propiedades.
- El Robot Wave: es el encargado de crear el manejador para cada evento, así como enviar las peticiones que convengan al Robot Movistar, y mostrar una respuesta en la interfaz web.
- Panel de Control: corresponde a la interfaz web de administración de Appengine. Gracias a esta, se pueden consultar los registros de actividad, en modo “Debug”, “Error”, “Warning” o

de peticiones, además de habilitar o deshabilitar la aplicación, cambiar de versión, consultar la cuota disponible, y programar eventos CRON.

- Registros de actividad: es una de las secciones del panel de control. Se configuran en la aplicación y se pueden visualizar en la interfaz gráfica del portal.

- Base de datos Wave: es el elemento de la arquitectura que provee persistencia a la aplicación. Pertenece a la arquitectura de Appengine, que nos ofrece una capa de abstracción intermedia para acceder a ella.

- Receptor de peticiones de Google Wave: es el módulo creado específicamente para recibir y manejar las peticiones que proceden de la interfaz de Google Wave. Este módulo será añadido a la aplicación Robot Movistar de Telefónica I+D. En el momento que recibe una petición, debe procesarla y redirigirla al robot, para que la maneje y elabore una respuesta, ya sea comunicándose con la IA o con la MIB. Una vez realizadas las acciones correspondientes, la respuesta será analizada por el módulo para enviársela al robot de Google Wave, con un protocolo creado para ello.

- Robot Movistar: es la aplicación creada por Telefónica I+D, que consta de los siguientes elementos:

  - Receptores de peticiones: son los módulos encargados de recibir peticiones. En esta aplicación existe el creado por Telefónica I+D para recibir eventos de la interfaz MSN o el nuevo módulo añadido en este proyecto, para la recepción de peticiones del Robot Wave.

  - Manejador de peticiones: es el módulo encargado de asignar el manejador adecuado a una petición que provenga o del Receptor de peticiones de Google Wave o del receptor de la interfaz MSN. Si la acción está asociada con la IA, procesa y redirige la petición hacia ella. Si la petición está asociada con el envío de MMS o SMS, la redirige hacia el adaptador correspondiente.

  - Procesador IA: Es el encargado de encontrar la respuesta ante un patrón determinado. Se comunica con su base de datos para obtener esas respuestas.

  - Adaptador. Es el manejador de peticiones de tipo SMS o MMS. Se encarga de comunicarse con la MIB, a través del protocolo de comunicación asociado.

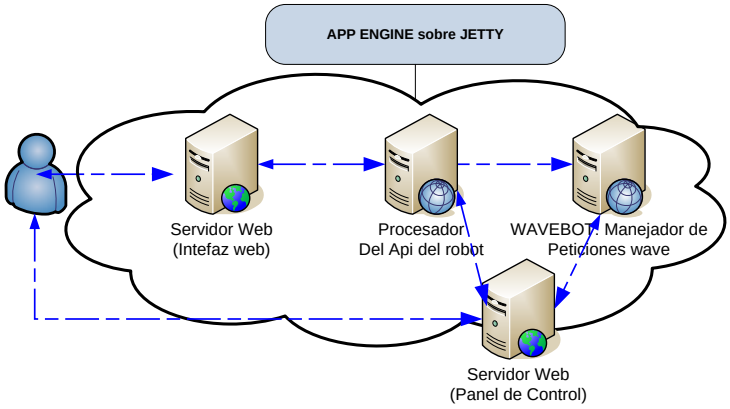


Figura 5.2: *Arquitectura del Robot Wave dentro de Appengine.*

Por tanto, podemos identificar tres entidades globales en el sistema:

- Aplicación cliente-servidor Wave: es la referida al robot de Google Wave. Esta aplicación estará desplegada sobre el servidor de aplicaciones Appengine versión 1.2.6. Para ello, solo será necesario desplegar y subir la aplicación usando el SDK de Google en local bajo la estructura que crea GWT. AppEngine se encargará de recibir, gestionar y balancear las peticiones entrantes de Google Wave. El sistema Appengine usa el servidor de aplicaciones Jetty, que aporta escalabilidad en el número de aplicaciones, mejora los tiempos de procesamiento, y ofrece eficiencia en el uso de la memoria del sistema, memoria de aplicación y memoria de disco duro. Si se quiere monitorizar el estado de la aplicación Google ofrece un portal de control web, para gestionar la aplicación, así como cambiar de versión o desactivarla. La Figura 5.2 muestra un ejemplo de esta arquitectura.

  La estructura del Robot Wave desplegado sobre AppEngine difiere de la de una aplicación típica. Google lo aísla de la arquitectura física, restringiendo el acceso a puertos, conexiones externas, lanzamiento o parada de procesos, para garantizar que la seguridad de la aplicación, y la del sistema, no se vea alterado [5]. En la figura se muestra este esquema 5.3

- Aplicación servidor O2Bot: el resultado de este sistema es una aplicación J2EE empaquetada en un WAR que será compatible con cualquier servidor de aplicaciones. Esta aplicación fue creada por Telefónica I+D, pero para poder comunicarse con el robot de Google Wave, se tuvo que añadir un nuevo módulo que recibiera las peticiones, las procesara para poder ser manejadas por el Robot Movistar y comunicarse con la Inteligencia artificial o con

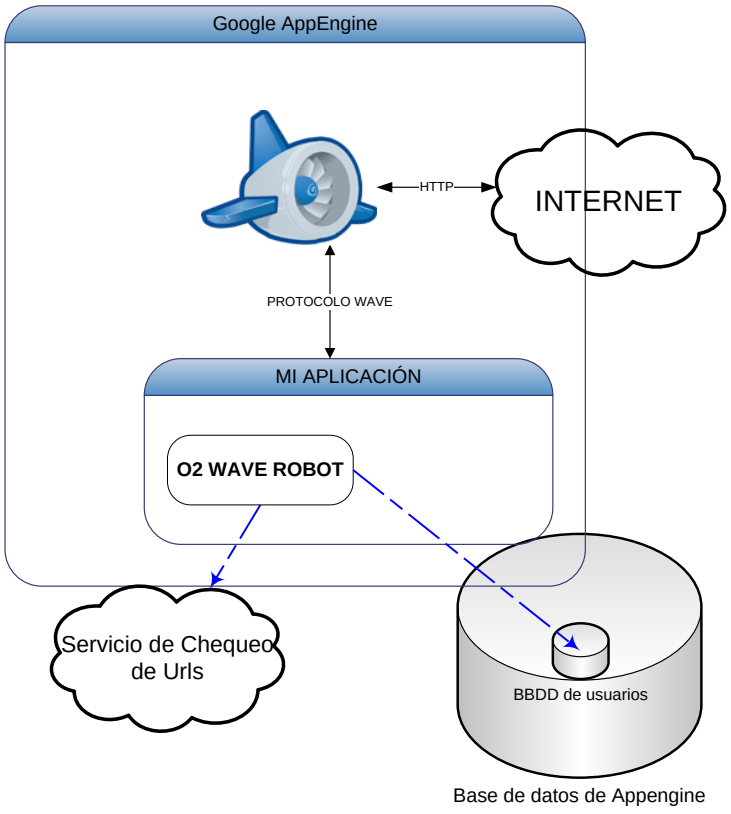


Figura 5.3: *Arquitectura O2BOT dentro de Appengine.*

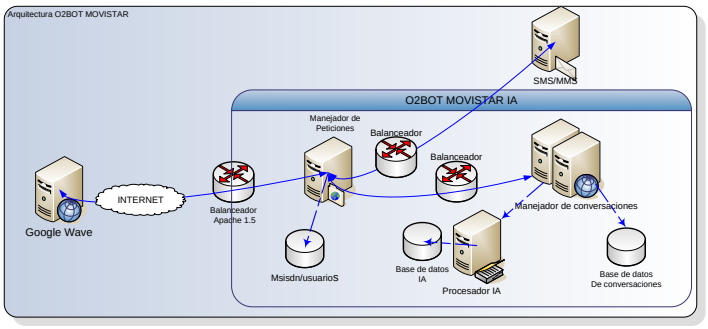


Figura 5.4: *Arquitectura software O2BOT.*

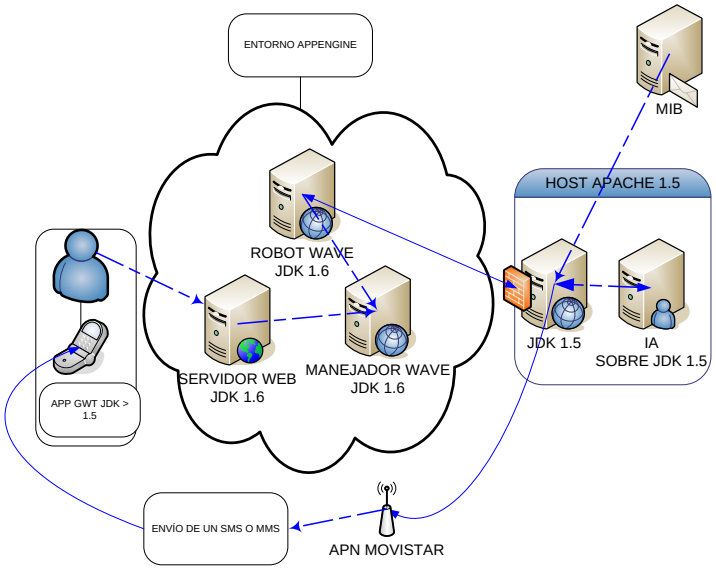


Figura 5.5: *Arquitectura hardware del sistema.*

la plataforma de mensajería. Esta aplicación estará desplegada sobre un Apache Tomcat versión 1.5.3. En la figura 5.4 se muestra el esquema.

5.2. Arquitectura hardware

La figura 5.5 representa la arquitectura hardware del sistema.

- Aplicación cliente-servidor Wave:

  - Java: para el despligue de la aplicación en local solo será necesario de un sistema operativo que soporte la máquina virtual de Java, a partir de la versión 1.5. La aplicación

será creada bajo el entorno GWT [1]. Una vez subida, mediante comandos o mediante el *Plugin* de Eclipse, se alojará en uno de los servidores de Google Wave. El Appengine ejecuta las aplicaciones en un entorno 1.6, pero es compatible con 1.5

- Hilos: la gestión de los hilos es transparente al desarrollador. El Appengine, a través de un middleware, se encargará de esta tarea.

- Aplicación servidor O2Bot: esta plataforma será el resultado de una aplicación J2EE [2] estándar. Por ello, será necesaria una máquina que permita unos mínimos requisitos de almacenamiento, CPU, memoria y soporte para J2EE. Solo es necesario un sistema operativo capaz de ejecutar una máquina virtual de Java versión 1.5. En la misma máquina estará la IA [3] junto con los manejadores del robot. En otra máquina, estará la MIB. Durante el desarrollo, la aplicación será probada en un sistema "x86 bajo CentOS 5.2 (RedHat Linux Enterprise libre)".

  La plataforma será diseñada para soportar diferentes entornos, dependiendo del uso y la demanda del sistema. Por ello, puede ser desplegada en las siguientes arquitecturas:

  - Despliegue centralizado: aplicación y base de datos ejecutadas en la misma máquina.
  - Despliegue distribuido: aplicación y base de datos ejecutadas en distintas máquinas.

5.3. Arquitectura de red

La figura 5.6 muestra la arquitectura de red general del sistema.
En el sistema existen las siguientes redes:

- Red interna AppEngine: engloba el conjunto de servidores que necesita la aplicación para funcionar. Para ello, el usuario se conecta al dominio de la aplicación, que se resolverá mediante una IP pública, y tendrá asociado el Robot Wave.

- Red interna de Telefónica: engloba el Robot Movistar, la IA y la plataforma de mensajería. Todas ellas se comunican mediante IPs internas, salvo la interfaz de entrada del Robot Movistar, que deberá tener una IP Pública para recibir peticiones desde Google Wave, al puerto 80.

[1] http://code.google.com/intl/es-ES/webtoolkit/(10/10/2010)
[2] Java 2 Platform Enterprise Edition
[3] Intelligence Artificial

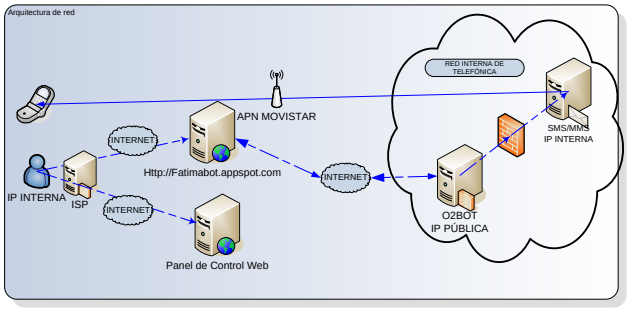


Figura 5.6: *Arquitectura de red del sistema.*

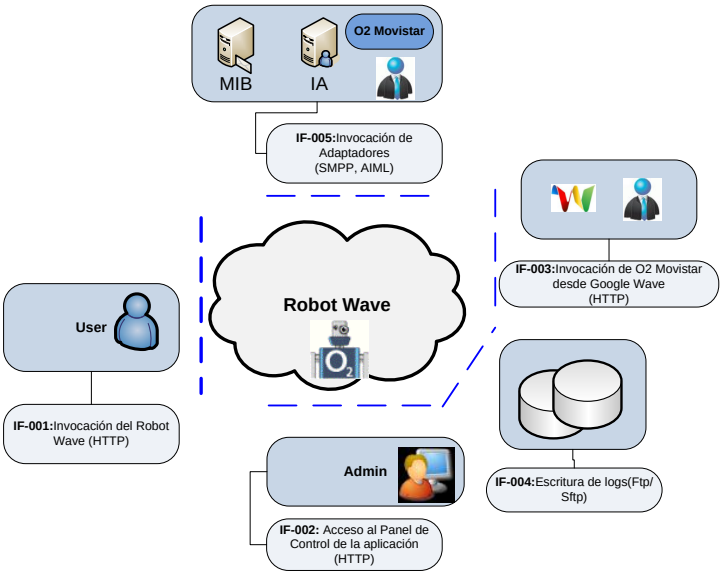


Figura 5.7: *Interfaces de la plataforma*

- Red interna del usuario: El usuario tendrá una IP interna asociada a su red local. El proveedor de servicios le dará una IP externa para comunicarse con Google Wave, a través de internet.

- Panel de Control: es la aplicación web de administración y gestión de las aplicaciones subidas en Google AppEngine. Tendrá una IP pública, y se accederá a ella, identificándose mediante usuario y contraseña de la cuenta de Google para crear aplicaciones.

5.4. Interfaces

La Figura 5.7 muestra el conjunto de interfaces que la plataforma tiene que manejar:

- IF-001 Invocación del Robot Wave: interfaz usada para invocar todas las funcionalidades que ofrece el robot de Google Wave dentro de la infraestructura de Appengine.
- IF-002 Invocación del Panel de Control: interfaz usada por el administrador de la aplicación para obtener todas las características sobre el funcionamiento del Robot Wave.
- IF-003 Invocación del Robot Movistar: interfaz encargada de recibir y analizar las peticiones que provengan de Google Wave, teniendo en cuenta el protocolo creado para la comunicación.
- IF-004: Invocación de los adaptadores: interfaz encargada de comunicarse con las diferentes entidades de la aplicación, es decir la IA y la MIB.
- IF-005 Registros: interfaz usada para crear informes de la actividad del usuario y de la aplicación.

5.4.1. IF-001 Invocación del Robot Wave

Esta interfaz define como los usuarios invocan la plataforma de Google Wave. El único modo de interacción que existe para acceder a la interfaz web de Google Wave, y tener conexión con el robot, es registrándose en la plataforma de Google Wave, y añadir al robot como un contacto más. El usuario, al interactuar con el robot, estará invocando una serie de eventos que serán manejados por el robot, cuyo objetivo será interpretar la acción y mostrarle una respuesta al usuario.

La figura 5.8 representa el protocolo Wave.

5.4.2. IF-002 Invocación del Panel de Control

Esta interfaz define cómo el usuario que es administrador de la aplicación, invoca la plataforma de Control de aplicaciones de AppEngine. Para ello, el administrador, desde su cuenta de Google, debe seleccionar el nombre de la aplicación y la versión, y en ella consultar todos los registros de actividad de la aplicación, así como errores, estadísticas o flujo de peticiones.

5.4.3. IF-003 Invocación del Robot Movistar

Esta interfaz define la comunicación entre el Robot Wave y el Robot Movistar, cuando el usuario que está interactuando con el Robot Wave, realiza una acción correspondiente a la

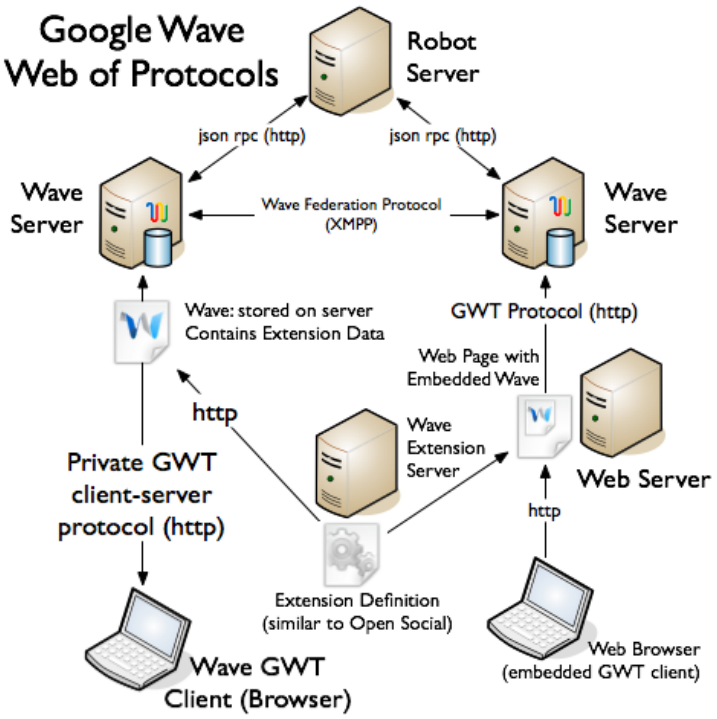


Figura 5.8: Protocolo Wave [3]

invocación de la IA o de la MIB. Estas acciones ocurren cuando el usuario está conversando con el Robot Movistar o cuando quiere enviar un SMS o MMS. El módulo receptor de las peticiones, debe ser capaz de convertirlas en el protocolo asociado al Robot Movistar, para que los manejadores de esas peticiones las entiendan y sean capaces de elaborar una respuesta.

La figura 5.8 describe el protocolo de comunicación en una conversación que incluye el envío de un SMS, entre el usuario, Robot Wave y el Robot Movistar.

5.4.4. IF-004 Invocación de Adaptadores

Esta interfaz define el protocolo asociado a la comunicación entre el Robot Movistar y la MIB o la IA. La IA es una plataforma independiente que está incrustada en la aplicación del Robot Movistar, y la MIB, es otra plataforma independiente situada dentro de la red interna de Telefónica.

5.4.5. IF-004 Registros

Esta interfaz define el registro de actividad correspondiente al módulo de recepción de peticiones desde Google Wave.

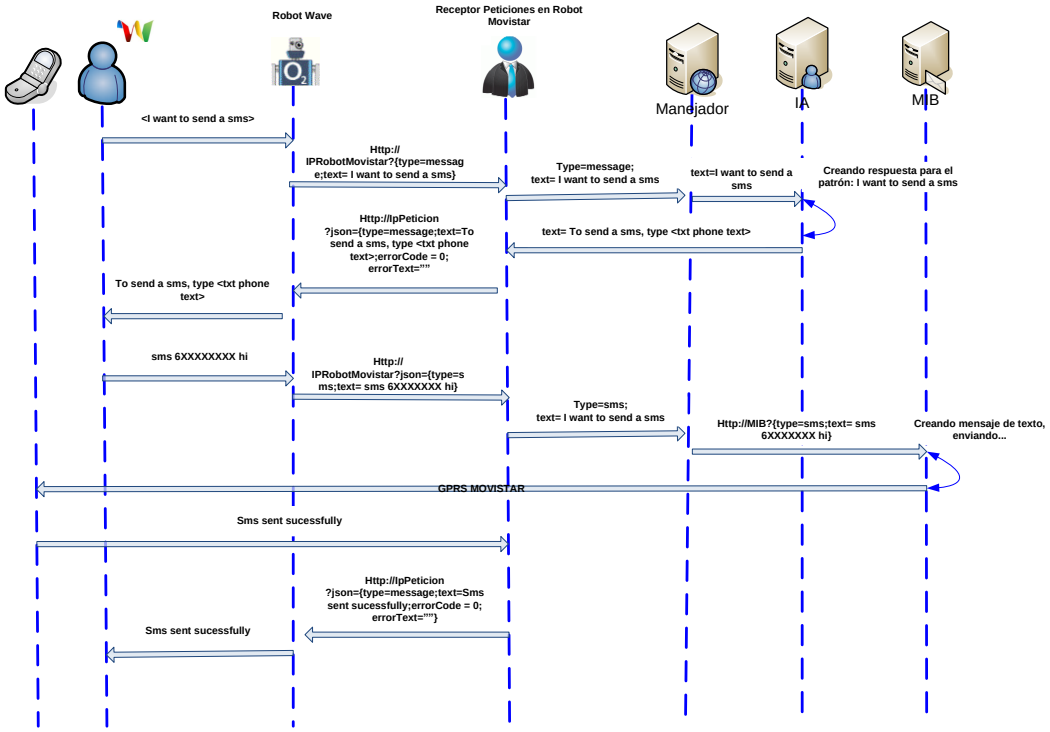


Figura 5.9: *Protocolo de comunicación entre robots*

5.5. Portabilidad del sistema

El Robot Wave es portable a cualquier sistema ya que se despliega sobre los servidores de AppEngine, únicamente es necesario crear la aplicación en local, en un sistema operativo que tenga una JDK 1.5 o mayor, y bajo el entorno GWT, subirla, o mediante el *Plugin* de Eclipse.

Para que se conecte al Robot Movistar necesita conocer la IP Pública. Por tanto, para que el Robot Wave pueda conversar con el Robot Movistar, solo se necesita que la aplicación empaquetada en un WAR esté desplegada en un Apache Tomcat versión 5.5 bajo JDK 1.5.

Para que el Robot Movistar pueda enviar mensajes necesita tener acceso a la MIB, ya que su conectividad es interna a la red de Telefónica. Por ello, deberán especificarse las reglas de rutas en el correspondiente *router* de la red. Por tanto esto hace que el Robot Movistar no sea portable a otra red externa, sin cambiar las reglas del *router*.

Capítulo 6

Diseño

6.1. Elección del lenguaje de programación

La creación de robots para Google Wave necesita de una biblioteca cliente que se integre con Google App Engine. Hay dos bibliotecas disponibles dependiendo del lenguaje de programación: Java o Python.

El Robot Wave de este proyecto está realizado con la biblioteca de Java. Los motivos que llevaron a tomar esta decisión son los siguientes:

- La creación del Robot Wave es motivada por sustituir al cliente de MSN que se conectaba con el Robot Movistar. Este cliente tenía limitaciones debido a que no se podían agregar imágenes o vídeos y por tanto, no se pueden financiar los SMS o MMS de los usuarios a través de publicidad. El Robot Movistar estaba implementado en lenguaje Java incluído en una aplicación típica J2EE. Por tanto, resulta lógico implementar el Robot Wave en lenguaje Java para actuar como cliente de la aplicación J2EE ya creada.

- Este proyecto está encapsulado en un grupo de programadores para Telefónica I+D cuyo desarrollo es principalmente Java. Crear el Robot Wave en Python exige por tanto el aprendizaje de una nueva tecnología que lleva tiempo y coste para la empresa.

- El API que ofrece Java y Python son equivalentes en cuanto a lo que proporcionan, por tanto, no hay ninguna ventaja respecto al API, que aporte Python frente a Java. Los dos

son compatibles con todos los módulos que incluye AppEngine. Sólo cambia la manera de desarrollar la aplicación, desplegarla y ejecutarla. Para más información sobre cada ámbito, consultar la referencia de Entornos de Ejecución en App Engine [9].

6.2. Adaptación del Robot Movistar

El objetivo de este proyecto es reutilizar la aplicación creada por Telefónica I+D, consistente en un robot con Inteligencia Artificial que conversa con el usuario, y que permite el envío de SMS y MMS, para cambiar el cliente de MSN donde solo se puede incluir texto, a un cliente Wave, que posibilita la inclusión de imágenes, vídeos y formularios HTML. Esto hace posible que se pueda enviar un MMS o que se puedan incluir anuncios publicitarios en la página, que antes era imposible debido a las limitaciones de la interfaz.

La motivación que lleva la incrustación de anuncios es que los SMS y MMS puedan ser financiados por agencias publicitarias a cambio de que el usuario visualice anuncios o se interese por alguna campaña.

Por tanto, hay que reutilizar la arquitectura cliente-servidor del Robot Movistar, para añadir un nuevo cliente, que será el Robot Wave. Esto llevará algunos cambios en el Robot Movistar. La adición del nuevo módulo al modelo, se explica en los siguientes pasos:

- Se crea una nueva interfaz receptora de peticiones, ya que antes, esa interfaz implementaba varios listeners que escuchaban las acciones de un cliente de MSN. Ahora, solo necesitaremos de un Servlet que reciba peticiones HTTP GET y POST, con las acciones que vaya realizando el usuario.

- Se crea un conversor que reciba el JSON incluido en la petición, y procese los parámetros que necesita el Robot Movistar para comunicarse con la Inteligencia Artificial o la MIB.

- Se crea un conversor de la respuesta que procese el resultado que le devuelve la Inteligencia Artificial o la MIB para enviarlos al Robot Wave.

6.3. Diseño del Robot Wave

En esta sección se va a explicar el diseño de la aplicación Wave correspondiente al proceso desde el cual el usuario realiza acciones en la interfaz de Mensajería de Google Wave, hasta el

procesamiento por parte de la aplicación para crear una respuesta. Como ya se explica en la sección del capítulo Estado del arte 2.2, la interfaz Wave está orientada a eventos. Los tipos de eventos que la interfaz puede detectar son los que se muestran en la tabla 6.1.

Tipo de Evento	**Descripción**
WAVELET BLIP CREATED	El usuario ha creado una ventana de conversación.
WAVELET PARTICIPANTS CHANGED	Se han añadido o eliminado participantes de la conversación, por ejemplo el Robot Wave.
WAVELET TITLE CHANGED	El título de la conversación se ha modificado.
BLIP SUBMITTED	El usuario ha pulsado el botón de Enviar de la ventana con la que estaba interactuando (Escribiendo texto, añadiendo imágenes, rellenando un formulario)
DOCUMENT CHANGED	Los elementos que estaban incluídos en la ventana han cambiado, por ejemplo, el botón de un formulario se ha pulsado, o se ha cambiado el valor de un control *Radio-Button*.
FORM BUTTON CLICKED	El botón de un formulario se ha pulsado.
WAVELET SELF ADDED	Se ha abierto una nueva conversación, o una anteriormente creada.

Tabla 6.1: *Eventos del Protocolo Wave*

Por ello, el diseño y funcionamiento de la aplicación Wave, se basa en los siguientes componentes:

1. Servlets: son encargados de recibir, procesar peticiones y devolver una respuesta en función del evento y la acción creados por el usuario, así como comunicarse con el Robot Movistar, para el envío de mensajes y la creación de respuestas procedentes de la IA.

   - Servlet *robotWave*: Es el *Servlet* principal de la aplicación que se encarga de recibir las peticiones que provienen del usuario a través del protocolo Wave. Extiende de la clase "AbstractrobotServlet", del API de robots de Google Wave. Esta clase a su vez, extiende de "HttpServlet", del API de Servlet 2.5. Por ello, este robot, se compone de los siguientes métodos:

     - Método *processEvents*: es el método implementado de la clase abstracta "AbstractrobotServlet". Es llamado por el protocolo Wave cuando se detecta un evento de los explicados anteriormente por parte del usuario. Este método, tiene un objeto JAVA que es el manejador del evento y que contiene toda la información necesaria sobre él. Por ello, dependiendo del tipo de evento solicitado, se creará una acción

u otra en respuesta al usuario. La implementación para cada evento, se explica más detenidamente en la sección Manejo de Eventos del capítulo Implementación 7.4.1.

- Métodos *doGet* - *doPost*: son los métodos sobrescritos por la clase "AbstractrobotServlet" de la clase padre: "HttpServlet" y son los encargados de recibir las peticiones del Robot Wave, procesar la información y crear el objeto que se le pasa al llamar a "processEvents". En el Robot Wave, se necesitará sobrescribir estos métodos para la comunicación con el Robot Movistar o con los *Gadgets* de la aplicación a través de HTTP. En el caso en que las peticiones no provengan del Robot Movistar o de los *Gadget*, se llamará a los métodos doGet y doPost de la clase "padre" para que ejecute la lógica correspondiente. Esto se explica más detalladamente en la sección 7.4.2

- Servlet *Profile*: extiende de la clase ProfileServlet y es el *Servlet* encargado de almacenar la información sobre el Robot Wave. Este robot, se utiliza a través de la interfaz de Mensajería de Google Wave, como un contacto más de la agenda, de este modo, para conversar con él, solo es necesario añadirlo a la conversación. Por ello, es útil almacenar información sobre él como el nombre, la dirección de contacto o la dirección de la imagen de su perfil. Cuando el protocolo Wave, necesita mostrar el nombre o la dirección de este contacto, porque algún usuario lo quiere añadir a su agenda, hace una petición GET a este Servlet y obtiene la información que necesite a través de los métodos "Getter" del Servlet. Un ejemplo de este *Servlet*, se muestra en el capítulo de Implementación 7.4.3.
- Servlet *ImageProcessor*: es el encargado de la manipulación de imágenes para su procesamiento y el envío hacia el Robot Movistar. Como la imagen se introduce a través del Gadget, es necesario un Servlet intermedio que redirija la petición al Robot Wave.

2. *Gadget*: la mayoría de *Gadgets* que existen en Internet pueden ejecutarse sobre la interfaz de Google Wave, pero Google, ha creado un API especial para *Gadgets* sobre Wave, que permite que los *Gadgets* creados sobre este API, interactúen con los elementos que componen la conversación, y se pueda modificar el estado de ella, así como pasar parámetros entre el robot y el Gadget. En este caso, para su creación, se ha utilizado el API de *Gadgets* de Wave, explicado más detenidamente en el capítulo de Estado del arte 2.3.2. Para este

proyecto, se han creado los siguientes *Gadgets*:

- *VideoGadget*: se basa en una página XHTML con un objeto perteneciente al API de Google Wave *Gadget*, que permite la incrustación de contenido FLASH.
- *ProfileGadget*: está formado por una página XHTML con tres vistas que contienen formularios. Cuando el usuario rellena la información de una página, pulsa el botón "Continue" que le llevará al siguiente formulario. En este caso, se oculta el formulario anterior, y se muestra el nuevo, dando la sensación de que el usuario ve tres páginas. Un ejemplo de visualización, se podrá consultar en el capítulo de Funcionamiento 3.5.
- *PhotoGadget*: está formado por una página XHTML que contiene un formulario de tipo Multipart Form-data, que contiene un "input" de tipo "file" para insertar un fichero. Cuando el *Gadget*, detecta el evento "onchange", se procesa el contenido de la imagen y se le muestra al usuario mediante Uri-Data, para más información consultar la sección 2.6.

Para visualizar un ejemplo de código de cada *Gadget*, véase capítulo de Implementación 7.4.4.

6.4. Protocolo de comunicación mediante JSON

Para el diseño del intercambio de comunicación entre el Robot Wave y el Robot Movistar será necesario el envío de información, así como su lectura e interpretación tanto en cliente como en servidor. En este caso, se usa JSON 2.8 que nos ofrece varias ventajas, una de ellas, es que podemos reutilizar ese objeto tanto en cliente como en servidor, y todas las órdenes que se transmiten entre ellos, ser procesadas a ese bean.

Las alternativas que existen para el intercambio de información son infinitas. Una de ellas, es el paso de parámetros en la petición Get, la otra, es el envío de parámetros en el cuerpo de la petición Post. Éstas dos alternativas tienen las siguientes desventajas:

- En cada petición hay que reescribir la URL y pasarle los parámetros que se necesitan dependiendo de la acción. Con JSON, simplemente se llama a los métodos "Setter" del objeto, estableciéndole las propiedades que se quieren, se convierte el objeto a una cadena de caracteres y se le pasa en la URL. El código de este va a ser reutilizado en cada acción, simplemente se le pasa un objeto nuevo.

- La lógica de la interpretación de las peticiones cambia dependiendo de los parámetros que llegan en la petición. Con JSON, la interpretación de la información tanto en cliente como en servidor, es reutilizada por todas las peticiones, y por tanto independiente de los parámetros que se envía. Solo hay que llamar a los métodos "Getter" del objeto.

- Los parámetros tienen que estar codificados en el Sistema de Caracteres UTF-8. Con JSON no existe este problema, ya que al convertirse el objeto a la cadena de caracteres, todos los caracteres se codifican automáticamente.

- El envío de parámetros siempre tiene que ser de la forma nombre-valor. Con JSON, se pueden enviar estructuras de datos como *arrays*, dentro de una propiedad.

Después de comprobar las ventajas que ofrece JSON se va a proceder a explicar cómo se aplica en este proyecto:

- Almacenamiento de imágenes: para el envío de un MMS es necesario que el usuario inserte una imagen y la envíe al servidor para que la almacene y así pueda enviarla a la MIB. Esta acción es distinta del resto de interacción del usuario con el Robot Movistar, ya que no se basa en un intercambio de conversaciones, sino que es una parte inherente a la acción del envío de un MMS. Una vez que el usuario selecciona la imagen, se detecta el evento, se muestra mediante el esquema URI-DATA y se crea un objeto tipo MMSWave, que contendrá los campos necesarios para el envío de un MMS, que son los siguientes:

  - Número de teléfono: destino del MMS.
  - Content-type: "Mime-Type" de la imagen, necesario para luego poder reconstruirla.
  - texto: texto del MMS.

  Una vez construido el JSON se envía una petición POST que contiene en el cuerpo del mensaje, el JSON y el contenido de la imagen codificada en Base 64. En el servidor, simplemente habrá que obtener el cuerpo del mensaje, e identificar las dos partes sabiendo que el JSON termina con el carácter "}".

- Intercambio de información: este punto corresponde al resto de las acciones que son propias del envío de información al servidor procedentes de las interacciones del usuario con el Robot Wave. Para ello, se creará un bean, que contiene campos como el tipo de acción que

se realiza, el mensaje del usuario... Después, se crea una petición GET, y se convierte el objeto a una cadena de caracteres que irá introducida en la URL de la petición. El servidor, sólo tendrá que recuperar el JSON de esa URL, e interpretar los parámetros que se envían dentro de él.

6.5. Tecnología de Persistencia

El API de AppEngine ofrece implementaciones de dos APIS de persistencia de Java: JDO y JPA. Ambas son bastante parecidas, la diferencia está en que JPA extiende las capacidades de JDO ya que proporciona un esquema relacional entre entidades de una base de datos.

El Robot Wave solo necesita de una entidad, que se denominará "Perfil", y que almacenará aquellos datos, que son necesarios persistir para diferentes sesiones del usuario con el Robot Wave. Esta entidad no se comunicará con otras tablas, por lo que se en esta aplicación se usará el API JDO. No existe otra alternativa de persistencia a esta, ya que AppEngine no permite conexiones JDBC a otras bases de datos de otros sistemas.

EL API JDO se encarga de crear, acceder y gestionar las entidades de la base de datos, así como administrar el ciclo de vida de las transacciones que se manejan, haciendo más fácil la configuración de una base de datos en la aplicación. Para más información sobre el API JDO, consultar la sección 2.1.2 del Capítulo de Estado del Arte.

Un ejemplo de la tabla de Perfil del Usuario que se puede visualizar en la sección: "Visor de Datos" del Panel de Control de AppEngine es la que se muestra en la figura 6.1.

Figura 6.1: *Tabla de Perfil de Usuario.*

Los datos que se necesitarán almacenar en la aplicación son los siguientes:

- *ID/Name*: representa el identificador de usuario dentro de la red de Google Wave.
- *Frecuence*: representa la frecuencia elegida por el usuario en el formulario de perfil, donde define el número de veces que quiere recibir publicidad.

- *Opinion*: muestra el texto escrito por el usuario como opinión al vídeo. Gracias a esta opinión se le podrá informar a las empresas de publicidad del éxito de los vídeos.
- *Phone*: es el número de teléfono introducido por el usuario en el momento de la autenticación en el sistema.
- *Preferences*: representa los gustos elegidos por el usuario en el formulario de perfil. Hay cuatro tipos: comida/bebidas, deportes, películas y videojuegos. Gracias a ella, se mostrará publicidad personalizada.
- *Publipoints*: es el número de puntos acumulados que lleva el usuario. Estos puntos se consiguen respondiendo a formularios o vídeos de publicidad, y gracias a ellos, podrá enviar MMS y SMS gratuitos.
- *Status*: almacena el estado del usuario en la aplicación: autenticado, cuestionado...

6.6. Elección del componente

El API de Google Wave ofrece dos posibles desarrollos para la interacción con el usuario: inserciones y robots. Los robots son tareas automatizadas que reciben peticiones en la interfaz de Google Wave, y las inserciones, son tareas automatizadas pero que son incrustadas en una página Web. Para más información consultar la sección de Estado del arte 2.3

La decisión de elegir el robot, y por tanto la interfaz Wave, son las siguientes:

- Se buscaba algo similar a la interfaz de MSN pero que añadiera nuevas funcionalidades, como la inserción de imágenes o contenido dinámico. La interfaz de Google Wave es parecida, el usuario se autentica en la aplicación, y añade a un contacto, en este caso del robot. En la figura 6.2, se muestra una comparativa entre la interfaz de Microsoft Messenger MSN (a la izquierda) y la interfaz Google Wave (derecha).
- Se necesita autenticación del cliente para tenerlo registrado en la base de datos, es decir, cuando un usuario quiere comunicarse con el Robot Movistar, desde la interfaz Wave, primero debe identificarse en la pantalla de autenticación de Google Wave, y después desde ese usuario, añadir al robot. Es en este momento cuando el Robot Wave, identifica al usuario. Esto es imposible realizarlo desde una página Web, a no ser que creemos un espacio de validación, es decir que creemos una interfaz parecida a la de Google Wave.

Figura 6.2: *Comparativa de las interfaces MSN-WAVE.*

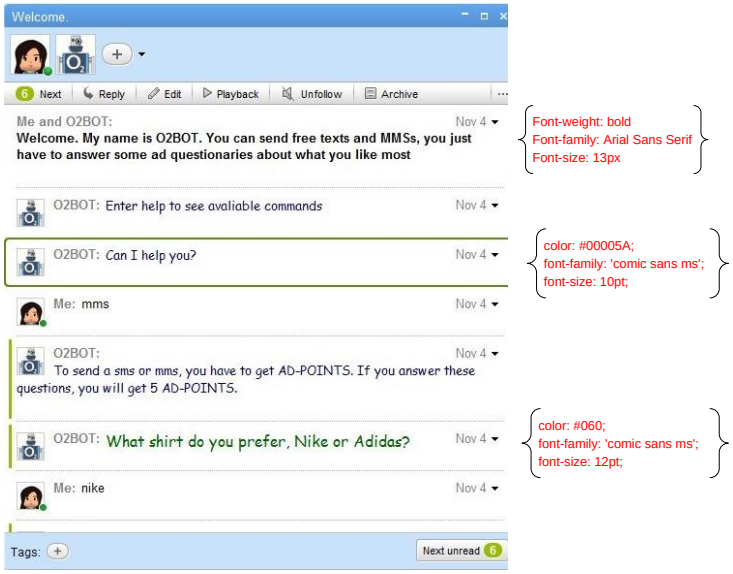


Figura 6.3: *Uso de Anotaciones en la interfaz de Google Wave.*

6.7. Interfaz Web: Uso de Anotaciones

En cada ventana de conversación de la interfaz de Google Wave se permite incrustar contenido HTML con CSS, para cambiar los estilos de la página, y crear ventanas de conversación más atractivas. El Robot Wave utiliza un mecanismo de Google Wave denominado Anotaciones, que permite, mediante métodos del API, cambiar los estilos de una ventana, sin llenar de código XHTML el código JAVA existente. Se puede cambiar desde el tipo de letra hasta el tamaño. Esto aporta gran facilidad para diferenciar los tipos de contenidos que se muestran en la interfaz Wave. Por ejemplo, para mostrar las preguntas del formulario, se cambia el color de la letra, para el título de la ventana de conversación el grosor, y para el resto de conversación, los estilos son los de la especificación de Telefónica I+D. En la figura 6.3 se muestran ejemplos de su uso:

Para más información sobre la implementación de Anotaciones, consultar la sección 2.7.

Capítulo 7

Implementación

En este capítulo se detallarán los algoritmos y protocolos implementados en las acciones más decisivas del prototipo, que han sido descritas previamente en el capítulo de Diseño de la aplicación, véase el capítulo 6.

7.1. Intercambio de datos a través de JSON

Como ya se explica en la sección 6.4 se usa la estructura de JSON para el envío de datos entre cliente y servidor. Por ello, habrá que elaborar un protocolo de intercambio de datos entre las dos entidades del sistema. El JSON, será el resultado de convertir un *bean* Java a "String". Las propiedades que tiene este objeto, así como los métodos, dependen del tipo de mensaje que se quiera enviar. La manera de determinar el tipo de mensaje dentro del JSON, vendrá establecido por una propiedad denominada "type".

En la comunicación con el servidor van a existir tres tipos de mensajes:

1. Tipo Mensaje: se refiere a las oraciones que el usuario introduce a través de la interfaz Wave correspondientes a las propias de una conversación general, es decir, que no contengan la palabra "SMS" o "MMS". El servidor detecta el tipo de acción que le llega e invoca el manejador correspondiente para la acción. Un ejemplo del intercambio de mensajes ante la introducción de una oración por parte del usuario se muestra en la figura 7.1:

2. Tipo Contraseña: este tipo de mensaje se produce cuando el usuario se autentica por primera vez. En este momento, introduce el móvil, y se envía una contraseña a su terminal para que la introduzca. Esta contraseña es generada aleatoriamente por el Robot Wave, y

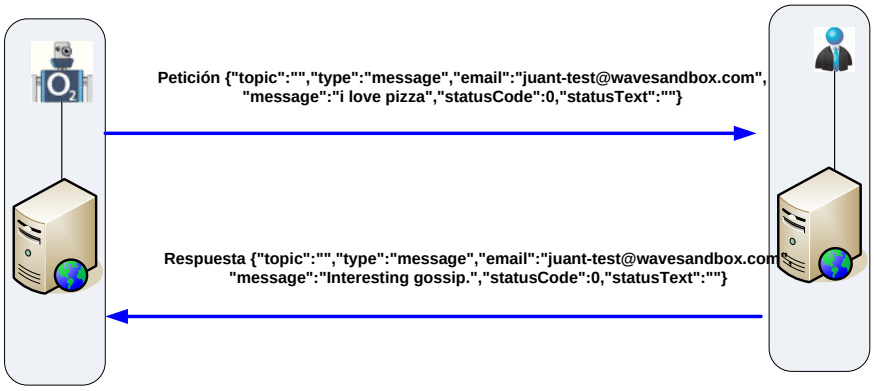


Figura 7.1: *Protocolo de envío de un mensaje.*

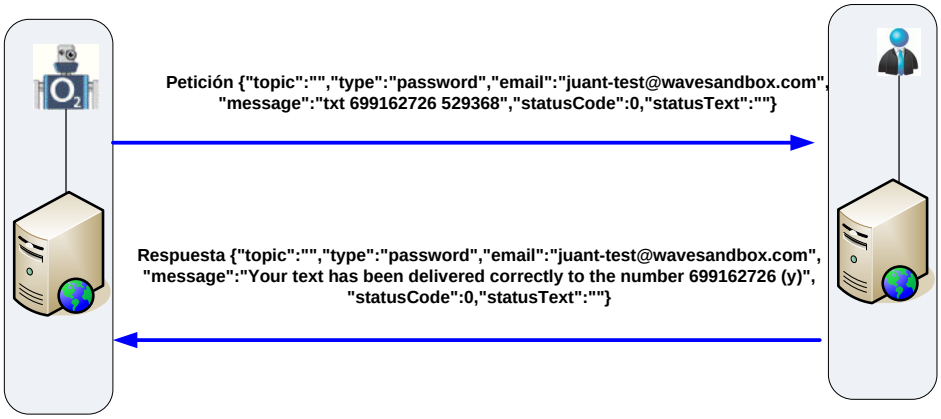


Figura 7.2: *Protocolo de envío de un mensaje tipo contraseña.*

enviada al servidor en un tipo de mensaje "password", para que el Robot Movistar envíe un SMS a través de la plataforma MIB. Una vez que el usuario la recibe, la debe introducir, y él Robot Wave debe comprobar que sea la misma que él ha generado, si no es así, se la pide hasta que la introduzca bien. Un diagrama de flujo de este tipo de comunicación es el de la figura 7.2.

3. Tipo Question: se refiere al intercambio de información del cuestionario proporcionado por la Inteligencia Artificial. En esta primera versión, se elige como tema del cuestionario por defecto, el de "Deportes", pero para próximas versiones, el tema elegido dependerá de las preferencias seleccionadas por el usuario en el *Gadget de Perfilado*. El cuestionario se inicia cuando el usuario elige sus preferencias y escribe la palabra "SMS". Es en este momento, cuando el Robot Wave inicia el cuestionario mostrándole siempre la misma pregunta inicial: *What shirt do you prefer, Nike or Adidas?*. Una vez respondida esta pregunta

por el usuario, se hace una petición con la respuesta escrita de tipo “question” y tema “WaveSports” hacia el Robot Movistar. En este momento, se establece el tema actual de conversación a “WaveSports”, y la Lógica Artificial redirigirá todas las preguntas al archivo que contiene los patrones para este tema respondiendo con la siguiente pregunta del cuestionario, que dependerá de la respuesta que le ha llegado anteriormente. La figura 7.3 ilustra el flujo del cuestionario.

Figura 7.3: *Protocolo de envío de un mensaje tipo cuestionario.*

En la figura 7.3, se puede observar los siguientes detalles:

- El tema “WaveSports” solo se envía una vez, ya que es el Robot Movistar el que gestiona el tema de la conversación cada vez que el usuario va respondiendo una pregunta. Es una manera de ir anidando las preguntas de la conversación.
- En las preguntas que la Inteligencia Artificial responde se pueden ver parámetros que empiezan por el carácter: “#”, y que identifican a un tipo de contenido a mostrar en el formulario. El formato del envío de contenidos es el siguiente: “#tipo=valor”. Existen dos tipos de contenidos, el primero, es el denominado “media” y se refiere a

las imágenes que se van a mostrar en el cuestionario o a los vídeos. El segundo, se denomina "radio", y se refiere a las posibles respuestas para cada pregunta. En esta versión no se incluyen ni vídeos ni imágenes, pero la solución está implementada para futuras versiones.

- Cuando el Robot Wave recibe un contenido que es un vídeo obtiene la URL y se la pasa al *VideoGadget*, que se explica detalladamente en la sección 3.5. Éste se conecta "Youtube", e incrusta el vídeo en la interfaz Wave.

- La última petición contiene el mensaje "ENDPOLLING", y es iniciada por el cliente Wave. Cuando la Inteligencia Artificial recibe esta cadena de caracteres, cambia el tema general de la conversación a "vacío" y finaliza el cuestionario. Si a partir de ahora el cliente Wave le envía una oración, él la interpretará como una conversación de carácter general.

7.2. Implementación de la lógica AIML

La inteligencia artificial del Robot Movistar, utiliza el lenguaje AIML para crear la lógica de una conversación. Para información detallada sobre este lenguaje, consulte la sección 2.4.

De manera simplificada, el proceso desde que el Robot Movistar recibe una respuesta del usuario hasta que la Lógica Artificial la procesa y crea una respuesta es el siguiente:

1. El Servlet encargado de recibir la petición la procesa y establece en un "bean" Java, la propiedad "tema" equivalente a "Deportes", y el mensaje del usuario.

2. Se le pasa este objeto a una clase que se encarga de obtener el mensaje, y guardarlo en un archivo.

3. El manejador va analizando el archivo que contiene la lógica de la conversación, con las posibles preguntas y respuestas, para ver si algún patrón es "parecido" al del archivo donde se ha guardado el mensaje. Se dice "parecido", porque en el archivo, se pueden especificar varios patrones para una misma entrada. En el siguiente ejemplo, se utiliza el guión, como sustituto de cualquier palabra que vaya delante de "MADRID" 7.4

```
<category><pattern>REAL MADRID</pattern><template><srai>REAL MADRID</srai></template></category>
<category><pattern>REAL MADRID _</pattern><template><srai>REAL MADRID</srai></template></category>
<category><pattern>_ MADRID _</pattern><template><srai>REAL MADRID</srai></template></category>
<category><pattern>MADRID _</pattern><template><srai>REAL MADRID</srai></template></category>
<category><pattern>MADRID</pattern><template><srai>REAL MADRID</srai></template></category>
<category><pattern>REAL MADRID</pattern><template><srai>REAL MADRID</srai></template></category>
```

Figura 7.4: *Ejemplo de una categorización de entrada AIML.*

4. La figura 7.4 viene a decir que si introducimos variedades de la palabra "Real Madrid" tanto detrás como delante o escribimos "R. Madrid" todas se transforman a la palabra "REALMADRID".

5. Una vez que se tiene la palabra transformada, se va analizando el archivo con las librerías de JDOM de Java [1], que busca si existe el patrón "REALMADRID". Si se encuentra se obtiene la respuesta para esa pregunta y se devuelve al Servlet inicial, si no, se lleva a una lógica que devuelve una respuesta por defecto.

6. Cada pregunta-respuesta se localiza en un ámbito al que solo se puede acceder si antes se ha establecido ese ámbito. Esta es la manera de hacer que las preguntas y respuestas estén anidadas. Cuando se responde por "NIKE" se establece un nuevo ámbito para que la próxima respuesta se busque solo en ese nuevo ámbito. Un ejemplo de esto se muestra en la figura 7.5

7. La explicación de la figura 7.5 es la siguiente:

   - Se recibe una petición con tema "WAVESPORTS" con el patrón "ADIDAS" o "NIKE" y se busca en ese ámbito la palabra que el usuario ha introducido para ver si es acorde al patrón. Si es así, se establece el nuevo ámbito al número 101 si la respuesta ha sido "NIKE" o al ámbito 102 si la respuesta ha sido "ADIDAS". Entonces se le devuelve al usuario la pregunta correspondiente a ese patrón. La próxima vez que el usuario responda a una de las dos opciones enviadas, la IA solo buscará las opciones en los ámbitos 101 o 101 respectivamente.
   - Además existirá un patrón por defecto que será analizado cuando el usuario introduce una de las opciones que no están contempladas. En este caso hacemos uso de un

[1] http://www.jdom.org(07/03/2011)

```
<topic name="WAVESPORTS">

<category><pattern>_ NIKE _</pattern><template><srai>NIKE</srai></template></category>
<category><pattern>_ ADIDAS _</pattern><template><srai>ADIDAS</srai></template>
...

<category><pattern>NIKE</pattern><template>What sportsman do you prefer, Cristiano or
Nadal?#radio=Cristiano#radio=Nadal#media=cristiano.jpg#media=nadal.jpg#

<think><set name="topic">WAVESPORTS101</set></think>

</template></category>

<category><pattern>ADIDAS</pattern><template>What sportsman do you prefer, Kobe Bryant
or Jenson Button?#radio=Kobe Bryant#radio=Button#media=kobe.jpg#media=button.jpg#

<think><set name="topic">WAVESPORTS102</set></think>

</template></category>

</topic>

<topic name="WAVESPORTS101">

<category><pattern>_ CRISTIANO _</pattern><template><srai>CRISTIANO</srai></
template></category>
<category><pattern>_ NADAL _</pattern><template><srai>NADAL</srai></template></
category>


<category><pattern>CRISTIANO</pattern>
<template>Do you prefer Real Madrid or Manchester City?#radio=Real
Madrid#radio=Manchester C.#media=realmadrid.jpg#media=manchester.jpg#
<think><set name="topic">WAVESPORTS201</set></think></template></category>


<category><pattern>NADAL</pattern>
<template>Do you prefer tennis or
paddle?#radio=Tennis#radio=Paddle#media=tennis.jpg#media=paddel.jpg#
<think><set name="topic">WAVESPORTS203</set></think></template></category>
...
</topic>

<topic name="WAVESPORTS102">

<category><pattern>_ KOBE _</pattern><template><srai>KOBE</srai></template></category>
<category><pattern>_ JENSON _</pattern><template><srai>BUTTON</srai></template></
...
<category><pattern>KOBE</pattern>
<template>Do you prefer NBA or
EUROLEAGUE?#radio=NBA#radio=EUROLEAGUE#media=nba.jpg#media=euroleague.jpg#
<think><set name="topic">WAVESPORTS202</set></think></template></category>


<category><pattern>BUTTON</pattern>
<template>Do you prefer Ferrari or
McLaren?#radio=Ferrari#radio=McLaren#media=ferrari.jpg#media=mclaren.jpg#
<think><set name="topic">WAVESPORTS204</set></think></template></category>

<think><set name="topic">WAVESPORTS102</set></think></template></category>
...

</topic>
```

Figura 7.5: *Ejemplo de anidación de preguntas AIML.*

wildcard [2]. Como respuesta, la IA le pregunta al usuario si quiere seguir la encuesta o no. Si es que sí, se establece el tema al actual, le vuelve a realizar la misma pregunta, y la respuesta será analizada en el ámbito actual. Sin embargo si es que no, se cambia el tema a general, y se finaliza el cuestionario. Un ejemplo de ello para un ámbito determinado se muestra en la siguiente figura 7.6

7.3. “Representación” de imágenes

7.3.1. Codificación Base64

Cuando el usuario quiere enviar una imagen dentro de un MMS se le muestra un *Gadget* que contiene un elemento HTML para la entrada de un fichero. Cuando el usuario selecciona la imagen, se lanza el evento “onchange” del API JAVASCRIPT, y se envía una petición “multipart-form/data” a un *Servlet*, que obtiene el contenido de la imagen, y se lo pasa al Robot Wave, codificado en Base 64, que es el formato adecuado para el envío de datos binarios en una comunicación entre cliente-servidor. Para más información, consultar la referencia [16]. Este tipo de codificación resultará útil para luego mostrar la imagen en el esquema URI.

7.3.2. Representación de la imagen mediante URI Data

A la hora de enviar un MMS, el usuario debe seleccionar una imagen. Para ello, se le muestra un *Gadget* con un formulario. Una vez, que el usuario la selecciona, se detecta ese evento, y se le pinta la imagen en la pantalla del *Gadget*. Para pintar la imagen dinámicamente, sin almacenarla en ningún sitio, existen actualmente dos maneras:

- Esquema URI, explicado en la sección 2.6 consiste en obtener el contenido binario de la imagen, el Mime-Type, y la codificación y mostrarla dentro de un elemento HTML de tipo imagen.

- Canvas es una nueva nueva etiqueta HTML que permite la creación de objetos JAVASCRIPT “Image” a partir de contenido dinámico.

En un primer momento se optó por Canvas, pero la interfaz de Google Wave, por seguridad no permitía recuperar el contenido del campo “Input” del usuario, y por tanto mostrar contenido

[2] cualquier caácter que representa un conjunto de elementos

```
<topic name="WAVESPORTS202">

<category><pattern>_ NBA _</pattern><template><srai>NBA</srai></template></category>
<category><pattern>_ EUROLEAGUE</pattern><template><srai>EUROLEAGUE</srai></
template></category>


<category><pattern>NBA</pattern>
<template>What do you think about this video?#media=http://www.youtube.com/v/
Q2J2tMsVtn0#<think><set name="topic">WAVESPORTS301</set></think></template></
category>

<category><pattern>EUROLEAGUE</pattern>
<template>What do you think about this video?#media=http://www.youtube.com/v/
hRkE1r9YHho#<think><set name="topic">WAVESPORTS301</set></think></template></
category>


<category><pattern>YES</pattern>
<template>Do you prefer NBA or
EUROLEAGUE?#radio=NBA#radio=EUROLEAGUE#media=nba.jpg#media=euroleague.jpg#
<think><set name="topic">WAVESPORTS202</set></think></template></category>


<category><pattern>NO</pattern>
<template>#type=cancelPoll#You have cancelled the poll, try it again to send sms and get ad-
points.
<think><set name="topic"></set></think></template></category>


<category><pattern>*</pattern>
<template>This option is not right. Do you want to continue this poll and receive more ad-points,
yes or no?
<think><set name="topic">WAVESPORTS202</set></think></template></category>

</topic>
```

Figura 7.6: *Ejemplo de replanteo del cuestionario AIML.*

```
<html>
<head></head>
<body>
<img
src="data:image/png;
base64,

iVBORw0KGgoAAAANSUhEUgAAADAAAAArCAYAAAA+EwvfAAADOklEQVR42u2Zy0s6URTHJ01rUYGC
CC6ECoIKeggRlKQrI4KEgtpEWYsQgghbhPkH2KIWJQTiSlwpYm2CCFqFULRoFwUVPaUHrRQzX+fHOeA
QP5+/c4Mv7lwYbj3DJzPzLnnnO8MAwIfjAggApQe6XQastmsMAESiQSMjIyAzWaDTCYjPAC7
3Q4Mw4DBYIBkMiksgJubG5DL5QRwfn4uvBAKhULkZXV1fcAuDrHx4eJmfu7u7Y
ooRzeXk5703gWZmdnaX9xcVF7nuhx8dHepo9PT0QjUZp7f7+ng5nLqS+vr7y7HEP7TgH2N3dJWfm
5+chlUqx6wcHB5Qeca+9vR0+Pj5ofW9vj9aGhoaoB+IUAB0eHR0lh7a3t/P2Pz8/YWBggA2pi4sL
UCgUdL2zs8N9O41POedcsfH+/k5N2u9D3tvbyw89YLVaySGHw1HpVFJeF5yk1Ov1FD7d3d3C08Svr68UOghweHjlP4BS
VIxOT08rst/a2gKj0fhPFbtqAE9PTyCVSkGpVFJeF5yk1Ov1FD7d3d3C08Svr68UOghweHjlP4BS
4hrHyckJ1NfXQ2trK/++SmBbMD09XdIGxQc+/fX1dX4BrK6usrl6bm4O4vF4vF4nU5yGjNPc3MzXbvd7pp/gskD
ODo6YgVHMBgseuPb2xuJj1zmGRwcLFvoqg6A2WR8fBxaWlpgijS0R7bYi6G+INDBBABRAAR
4P8G+AOURmS0BsUvewAAAABJRU5ErkJggg==" >

</img>
</body>
</html>
```

Figura 7.7: *Representación de una imagen con URIDATA.*

Canvas. Por tanto, se optó por el esquema URI. El procedimiento consistía en enviar la imagen como campo de un formulario a un servlet, obtener los bytes de la imagen, y mostrarlos en el *Gadget* dentro de un elemento de tipo imagen.

La especificación URI DATA se basa en la creación de un objeto HTML tipo "img" donde el atributo "src" contiene el contenido binario de la imagen, junto a otras propiedades como la extensión del fichero, la codificación y el contenido binario de la imagen. Un ejemplo de una página simple HTML con una imagen es el que muestra la figura 7.7.

7.4. Implementación del Robot Wave

La siguiente figura 7.8 muestra la estructura general de la aplicación. Se compone por tanto de:

- Carpeta src: contiene tanto los Servlets como las clases Java de utilidad necesarias para el desarrollo del robot.

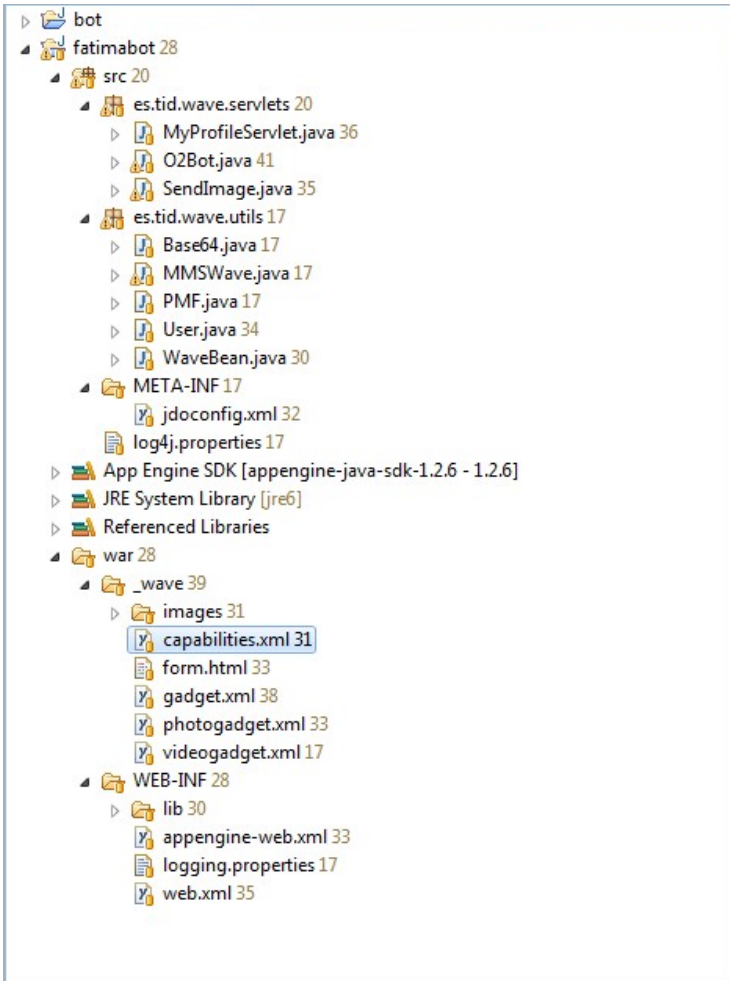


Figura 7.8: *Estructura de la aplicación Robot Wave.*

- Carpeta war: contiene la zona pública del robot. Se compone de:

  - Carpeta wave: contiene las imágenes, el archivo capabilites.xml que contiene la información respecto a eventos y los *Gadgets*, así como el formulario para el cambio de preferencias.

  - Carpeta web-inf: contiene los archivos de configuración del servidor, como el web.xml (definición y mapeo de servlets), la configuración de logs en logging.properties, y los parámetros de configuración de la aplicación, en el archivo appengine-web.xml, donde se habilitan propiedades como: sesiones, número de versión, el nombre de la aplicación. También contiene la carpeta con las librerías necesarias para la aplicación.

- Carpeta meta-inf: contiene un fichero con la configuración de la tecnología de persistencia JDO.

7.4.1. Definición de eventos

Para que el protocolo Wave detecte eventos hay que especificar el nombre de los eventos que se quieren detectar en un archivo de la aplicación denominado "capabilities.xml". A su vez, hay que crear la lógica para cada evento, y si añadimos nuevos eventos a la aplicación, hay que incrementar el número de versión de la aplicación, para avisar al protocolo Wave, de que se incluyen nuevos eventos, y que debe detectarlos y progresarlos al Robot Wave. Al subir la aplicación al entorno AppEngine, si el número de versión es mayor que el anterior, procesa el archivo "capabilities.xml" y habilita la detección de los nuevos eventos definidos.

7.4.2. Servlet Robot Wave

La siguiente figura 7.9 muestra la estructura del Servlet principal de la aplicación.

7.4.3. robot Profile

El robot Profile es el encargado de almacenar la información del robot como el nombre, imagen de perfil, descripción .Un ejemplo de este Servlet, es el de la figura 7.10

```
package es.tid.wave.servlets;
//imports

public class O2Bot extends AbstractRobotServlet {

     //variables estáticas.

   public void doPost(HttpServletRequest request, HttpServletResponse response)
   throws java.io.IOException {
      if (request.getRequestParameter("form") != null) {...}
      else if {...}
      else if {...}
      else super.doPost(request, response);
   }
   public void doGet(HttpServletRequest request, HttpServletResponse response)
   throws IOException {
      if (request.getParameter("send") != null){...}
      else if {...}
       else super.doGet(request, response);
   }

   public void processEvents(RobotMessageBundle robotMessageBundle) {
      wavelet  = robotMessageBundle.getWavelet();
     userId = wavelet.getCreator();

      for (Event event : robotMessageBundle.getEvents()) {

         Blip blip = event.getBlip();
         EventType tipo = event.getType();
         switch(tipo) {
               case WAVELET_SELF_ADDED:
                  blip.setTitle(WELCOME_TEXT);
               break;

               case BLIP_SUBMITTED:

                  if (!isRoot(blip, wavelet)) {
                     text = blip.getDocument().getText();

                     if (text.equalsIgnoreCase("deleteUser"))
                        this.deleteUser();

                     else if((isSms(text) || isMms(text))&& userPerfiled()){
                        blip = wavelet.appendBlip();
                        sendText(blip, "\nTo send a sms or mms, you have to get AD-
                          POINTS."+
                        "If you answer these questions, you will get 5 AD-POINTS.\n");
                        {...}
                     }

               break;
               case FORM_BUTTON_CLICKED:
                  if (event.getButtonName().equals("okMobile")){...}
               break;
         }

      }

    }

//Sección de métodos de utilidad para el Robot Wave.

}
```

Figura 7.9: Estructura del Servlet del Robot Wave

```
package es.tid.wave.servlets;

import com.google.wave.api.*;

public class MyProfileServlet extends ProfileServlet {

   public static String ID = "fatimabot";
   public static String IMAGES_URL = "http://"+ID+".appspot.com/_wave/images/";
   public static String IMAGE_AVATAR_URL = IMAGES_URL+"o2_bot.gif";
   public static String NAME ="O2BOT";
   public static String PUBLIC_URL = "http://"+ ID +".appspot.com/_wave/";
   public static String ROBOT_URL = "http://"+ ID +".appspot.com/_wave/robot/
jsonrpc";

   public String getRobotName() {
      return NAME;
   }

   public String getRobotAvatarUrl() {
      return IMAGE_AVATAR_URL;
   }

}
```

Figura 7.10: Estructura del Servlet del Perfil del Robot Wave

7.4.4. Gadgets

En esta sección se mostrarán ejemplos e implementación de los Gadgets del prototipo. La estructura de un *Gadget*, es común a todos ellos y es como se muestra en la siguiente figura 7.11

Intercambio de información entre los *Gadgets* y el Robot Wave

La comunicación y el paso de parámetros entre el robot y el Gadget se realiza mediante peticiones GET o POST. Para ello, se sobrescribe el método "doGet" y "doPost" del API de robots de Google Wave. Estos dos métodos extienden a su vez del API de Servlet 2.5 de Java. Dependiendo del *Gadget* empleado, se utiliza un parámetro para marcar el *Gadget* desde el cual se envía la petición. Los métodos "doGet" y "doPost", analizan ese parámetro y en función de él, realizan las operaciones necesarias, como son el envío de parámetros al servidor o el acceso de base de datos. Desde los métodos citados anteriormente no se tiene acceso a la conversación, ya que el acceso al método solo se lanza a través de eventos del usuario, por tanto, no se podrá mostrar información o abrir nuevas ventanas de conversación como respuesta al *Gadget*. Para mostrar una respuesta que de continuidad a la interacción del usuario, cada vez que se quiere mostrar

```
<?xml version="1.0" encoding="UTF-8" ?>
<Module>
   <ModulePrefs title="ROBOTPREFERENCES"
            height="250"
            author="FÁTIMA DE LA OSA"
            author_email="fatimadelaosa@hotmail.com">
          <Locale lang="en" country="uk" />
   </ModulePrefs>

   <Content type="html">

      <![CDATA[

            <style type="text/css">
            //CÓDIGO CSS
            </style>

            <script language="javascript">
            //CÓDIGO JAVASCRIPT
            </script>

           //CODIGO XHTML
      ]]>
   </Content>
</Module>
```

Figura 7.11: Estructura de un Gadget

```
function sendPhoto(v){
      document.getElementById('form').submit();
}
<form id="form" action="http://fatimabot.appspot.com/_wave/robot/jsonrpc"
method="post" enctype="multipart/form-data">

   <div lang="en-US" class="letter" id="gral">
      Enter your photo:<input name="file" lang="en-US" type="file"
      onChange="sendPhoto()" />
   </div>
</form>
```

Figura 7.12: Ejemplo de código de PhotoGadget

una respuesta en la vista, se pinta el contenido en la salida de la clase HttpServletResponse del API de robots Wave. Esto produce que se pinte la salida, en la ventana del *Gadget*, pero nunca en la ventana de conversación.

PhotoGadget

El *Gadget* utilizado para que el usuario introduzca la imagen lo único que tiene es un formulario, con un campo para entrada de un fichero. Cuando el usuario selecciona la imagen, se realiza una petición "multipart/form-data" a un Servlet, que obtiene el contenido y redirige la respuesta al Servlet del Robot Wave. Todavía la imagen no se ha almacenado en servidor, ya que el usuario aún no ha pulsado sobre el botón "Enviar". Es en este punto cuando se crea un objeto de tipo MMS, con un campo donde se envía codificado el contenido de la imagen junto a otros parámetros como la extensión o la codificación. Este se convierte a JSON, y se envía en una petición GET como es normal. El servidor, analiza la petición, obtiene los campos y almacena la imagen en una carpeta destintada para ello, a través del API de DiskFileUpload [3] con un nombre único compuesto de la fecha en la que recibió la imagen y el nombre de ella que viene incrustado en el JSON.

Un Ejemplo del formulario para la inserción de la imagen se muestra en la figura 7.12

[3] http://tomcat.apache.org/tomcat-6.0-doc/api/org/apache/tomcat/util/http/fileupload/DiskFileUpload.html(10/10/2010)

```
var div = document.getElementById('content_div');

function stateUpdated() {

  //obtengo la url del Robot, si el parámetro es null, obtengo un vídeo por defecto
   var url = wave.getState().get('urlV','http://www.youtube.com/v/vKU90BSHqHk');

      gadgets.flash.embedFlash(url, div, {
                swf_version: 6,
                id: "flashid",
                width: 300,
                height: 250
    })
```

Figura 7.13: Ejemplo del código VideoGadget

VideoGadget

Como se ha dicho anteriormente, desde que se muestra un *Gadget* no se tiene acceso a la conversación del Robot Wave hasta que el usuario no interaccione con él. Por este motivo, cuando se muestra el *VideoGadget* que es el vídeo que reproduce una URL de "Youtube" acorde a las preferencias del usuario, se utiliza una variable para marcar el estado de la interfaz a "Waiting". Cuando el usuario escribe una oración justo después del vídeo, el Robot Wave entiende que es la opinión a ese vídeo, la almacena en base de datos, y ya tiene acceso a la conversación respondiéndole con una nueva ventana de conversación.

Un ejemplo de código de la inclusión de un vídeo externo en el *Gadget* es el que se muestra en la figura 7.13

ProfileGadget

El funcionamiento de este *Gadget* es más complejo, ya que se basa en mostrar y ocultar vistas dependiendo de las acciones del usuario. La descripción de la implementación de cada vista es la siguiente:

1. Primera Vista: se compone de un div, que a su vez contiene las cuatro imágenes que muestran las preferencias a elegir por el usuario. Cada imagen tiene asignado el evento onClick. Cuando una imagen es seleccionada por el usuario, se le muestra otra imagen de

mayor tamaño. Cuando el usuario selecciona una imagen, se almacena el id de la imagen en un array.

2. Segunda Vista: en el momento en que el usuario pulsa sobre Continuar de la primera vista, se oculta el div que contiene la primera vista, y se muestra el de la segunda. Este contiene un formulario con la frecuencia que el usuario quiere recibir publicidad. Cuando pulsa sobre continuar, se envía la frecuencia, y las preferencias de la vista 1 al Robot Wave que las almacena en Base de datos.

3. Tercera Vista: esta vista la pinta el Robot Wave en respuesta a la recepción de parámetros, por ello, se obtiene la respuesta HttpResponse del Servlet, y se pinta en ella, el contenido XHTML de la vista. Un ejemplo de código es el siguiente 7.14

```
public void doGet(HttpServletRequest request, HttpServletResponse
response) throws IOException {
…

    if (request.getQueryString().contains("FormGadget=")){

        often = request.getParameter("often");
        preferences= request.getParameter("preferences");

        response.getWriter().println("<div id='global2' style='background-
        color:#CCC; width:100%;; text-align:center;'><div id='title2'
        style='background-color:#006; color:#FFF; width:100%; text-
        align:center; font-family:'Comic Sans MS', cursive; font-size:18px;
        font-weight: bold;'>THANK YOU</div>"+
        " <p> </p>"+
        “<p>You can change your preferences in your preferences site: </
        p>"+
        "<p><a href="'+MyProfileServlet.PUBLIC_URL+"form.html'
        target='blank'>Click Here!!</a><p> </p>"+
        "<p> </p><p> </p><p> </p></div>");

    }
    else ...
}
```

Figura 7.14: Ejemplo del código Respuesta al Gadget de Perfil

Capítulo 8

Pruebas

En este capítulo se mostrarán las pruebas que se han realizado en las diferentes aplicaciones que componen el prototipo, aplicación web, plataforma de mensajes, y protocolo con la Inteligencia Artificial. Estas pruebas pretenden medir la robustez de las aplicaciones y su comportamiento en situaciones concretas.

8.1. Pruebas de la interfaz web

Este test se basará en comprobar el correcto funcionamiento de la interfaz web, ante la interacción del usuario. Para ello, se dividirá la prueba en varias fases dependiendo del estado de autenticación en el que se encuentre el usuario.

Validación del login

Cuando un usuario entra por primera vez en la aplicación necesita identificarse para poder conversar con el robot. En este momento, el usuario no podrá comunicarse con el robot ni enviar mensajes. Por tanto, se creará un validador que compruebe:

- Estado no autenticado: cuando el usuario añade al robot se mostrará la pantalla de identificación, y el usuario no podrá realizar ninguna acción. Si escribe al robot, este le mostrará el siguiente mensaje: "You have to sign up to talk to me."
- Número de Teléfono: se comprobará que contenga menos de 11 dígitos, y que todos ellos sean números. Si no es así, le avisará del error, y se lo pedirá de nuevo, hasta que introduzca un número válido.

- Número de Teléfono en la base de datos: se comprobará que el número de teléfono queda almacenado en el correspondiente usuario.

- Contraseña: una vez que el usuario escribe el número de teléfono y acepta su envío, se le enviará una contraseña aleatoria al móvil con un mensaje de texto y el usuario tendrá que escribir este mismo. Por ello, se creará un validador que chequee que las contraseñas coinciden. Si no es así, se le volverá a pedir la contraseña, hasta que el usuario la introduzca correctamente.

- Estado Autenticado: una vez completado el paso anterior, se comprobará en la base de datos, que el estado del usuario ha cambiado a "Autenticado". En este punto, ya podrá conversar con el robot.

Perfil del usuario a través de *Gadget*

- Estado de autenticación: desde el momento en el que el usuario está identificado en la aplicación, ya podrá hablar con el robot, pero nunca enviar mensajes. Si en alguna de las conversaciones escribe la palabra "MMS", el robot le mostrará un *Gadget* preguntándole por sus preferencias.

- Estado Perfilando: en este momento, se comprueba que el nuevo estado el usuario a pasado a ser "Perfilando".

- *Gadget de Perfil I*: en la primera página se comprobará que cuando el usuario elige sus preferencias entre 4 por defecto: comida, juegos, películas y deportes y pulsa sobre alguna de ellas, la imagen, debe cambiar a la misma pero con el nombre elegido mostrado sobre ella y los subtítulos del *Gadget* pasarán a ser rojos, solo en los valores que se han pulsado. Si se pulsa de nuevo, la imagen vuelve a su estado inicial, y el color cambia al de por defecto. Todo esto se muestra en la Figura 3.5.

- *Gadget de Perfil II*:en la segunda página se comprobará que la frecuencia con la que el usuario quiere recibir publicidad, queda persistida en la base de datos, una vez enviada. Un ejemplo de esta acción se puede visualizar en la Figura 3.6.

- *Gadget de Perfil III*:cuando el usuario acepta la frecuencia le lleva a una última página que contiene un enlace a un formulario para cambiar las preferencias elegidas. Si en el

formulario se cambian, también se tienen que cambiar en la base de datos. Si todo ha ido bien hay que comprobar que tanto la frecuencia como las preferencias seleccionadas, han quedado persistidas y que el estado a pasado a ser Perfilado. A partir de este momento el usuario puede enviar SMS y MMS.

Perfil del usuario a través de cuestionario AIML 2.4.

Una vez que el usuario se ha perfilado, para enviar un MMS o SMS necesita conseguir puntos. Los puntos mínimos para enviar mensajes son 5. Por tanto, antes de enviarlo se le va a realizar un cuestionario con la lógica AIML. La respuesta de cada pregunta será la base para la siguiente cuestión, esto es así porque es una forma de conocer profundamente las preferencias de cada usuario. Por ello, en esta prueba, se necesitará:

- Validación de las respuestas. Este aspecto lo detallaremos en la próxima Sección 8.1.3
- Comprobación del estado: una vez que el usuario ha terminado el cuestionario se comprobará en base de datos que el nuevo estado es "Cuestionado", y que el usuario tiene 5 puntos para enviar mensajes. Por tanto, si escribe el texto <SMS numero texto>el robot enviará un mensaje al destino especificado y se comprobará que el usuario tiene 5 puntos menos.
- Envío de MMS: con este estado el usuario ya puede enviar MMS. Si en el momento que escriba MMS no tiene puntos, se le debe mostrar un vídeo acorde a sus preferencias para que el usuario de su opinión. Por tanto, habrá que comprobar en este punto, que el vídeo que se muestra es acorde a las preferencias del usuario, y que una vez que ha contestado con su opinión, se le muestre el *Gadget* para el envío de MMS.

Envío de MMS.

Esta prueba consistirá en comprobar que el texto MMS se detecta correctamente en el robot y que el mensaje llega correctamente al usuario. Por tanto las pruebas serán las siguientes:

- Detección del texto: si el usuario escribe TXT o MMS, pero no escribe los demás campos, es decir número de móvil destino y mensaje, el robot le contestará con este mensaje: "Para escribir un MMS tienes que escribir *<txt numero texto>*"
- Comprobación de llegada: una vez que el mensaje llega al destino, se muestra un mensaje de entrega satisfactoria. Las pruebas de envío de MMS se detallan en la Sección 8.1.2

- Comprobación de publi-puntos: en este punto, hay que comprobar que el usuario tiene 5 puntos menos.

Envío de MMS.

Esta prueba consistirá en comprobar que el texto MMS se detecta correctamente en el robot, que la imagen que introduce el usuario se muestra en pantalla, y que el mensaje se envía correctamente al usuario.

- Detección del texto: si el usuario escribe MMS, y tiene puntos para enviar el MMS, se le pedirá en un formulario el número de teléfono del destino, el texto y la imagen.
- Introducción de la imagen: cuando el usuario seleccione la imagen, se le mostrará automáticamente una vista previa de cómo quedará el mensaje. La validación de la imagen, se realizará en servidor, y se detallará en la Sección 8.1.1.
- Vista Previa: ésta, debe contener, el texto con Sistema de codificación de caracteres correcto, la imagen con un tamaño que no sobrepase el máximo de 50 de ancho, y por tanto, deberá mantener la relación ancho x alto, y el número de teléfono, que debe ser el mismo que el introducido por el usuario. Además aparecerá un botón para enviar la imagen, si el usuario lo pulsa, se mostrará otra pantalla, con la vista previa del mensaje más un texto de entrega satisfactoria.
- Comprobación de llegada: una vez que el mensaje llega al destino, se muestra un mensaje de entrega satisfactoria. Las pruebas de envío de MMS se detallan en la Sección 8.1.2
- Publi-puntos: en este momento, los puntos del usuario han decrementado en 5.

8.1.1. Pruebas del servidor

En este apartado se explicarán las pruebas correspondientes del servidor, es decir, la aplicación que recibe la conversación del usuario y por tanto el envío de mensajes.

El protocolo de comunicación entre cliente y servidor, explicado en la Sección 2.8 se basa en el envío de un JSON para el intercambio de información. Por tanto, será importante comprobar:

- Sistema de codificación de caracteres: que la codificación del texto tanto enviado como recibido, sea UTF-8, y que caracteres como la ñ o las tildes se muestran correctamente.

- Control de errores: En este apartado, se prueba la robustez del servidor, por ello, se enviarán peticiones con JSON corruptos, para comprobar que siempre recibo una respuesta. Si en algún punto el servidor falla, responderá con el siguiente mensaje: "En este momento no puedo escucharte. Inténtalo de nuevo más tarde".

- Pruebas de conexión: en este punto, se comprobará, que si la conexión cliente-servidor falla, existe un control de temporizador. Es decir, si después de cierto tiempo, el servidor no responde, se muestra el mismo error que en el punto anterior 8.1.1.

- Envío de imágenes : este es uno de los puntos más críticos de la comunicación, ya que para mostrar la imagen en el cliente, y poderla enviar luego en un MMS, se debe almacenar en el servidor, y después enviársela al cliente para que la muestre. Para ello, se envía la imagen codificada en Base64 dentro del contenido del JSON. Por tanto, habrá que comprobar:

  - Nombre de la imagen: que cada imagen, aunque sea igual a otra introducida, se almacene con un nombre distinto.
  - Codificación de la imagen: que la imagen enviada y la recibida, sean equivalentes.
  - Tamaño de la imagen: si la imagen supera un tamaño máximo de fichero, se devolverá un mensaje de error, ya que Google Wave tiene restricciones de tamaño.
  - Ancho y alto de la imagen: si supera un ancho y alto, deberá ser transformada para ser mostrada en el cuadro de conversación sin pasar los límites, y manteniendo la relación de aspecto original.

8.1.2. Pruebas de la plataforma de mensajería

En esta sección se realizarán pruebas de carga sobre la MIB. Para ello, se tendrán en cuenta, los siguientes aspectos:

- Validación del teléfono destino: debe ser un número menor de 11 cifras, y correspondiente al territorio español.

- Validación del patrón recibido: debe coincidir con <txt número texto>

- Envío al destino: si el destino no existe, debe devolver un mensaje de error.

- Comprobación del Sistema de codificación de caracteres: hay que chequear que el texto recibido en el móvil es el mismo que el enviado por el usuario, incluyendo los caracteres extendidos del código ASCII.
- Control de errores: si alguno de los anteriores apartados es inválido, se devolverá al cliente web, el mensaje de error por defecto, visto anteriormente 8.1.1.

8.1.3. Pruebas de la Lógica AIML

Este test está orientado a detectar posibles errores del protocolo con la Inteligencia Artificial. Se comprobarán por tanto distintos estados en los que el usuario está comunicándose con la Lógica AIML. Estos, son los siguientes:

Conversación con el robot

En este punto es cuando el usuario realiza una conversación normal con la lógica. Se comprobarán

- Respuestas: para un mismo patrón de entrada, habrá distintas respuestas.
- Almacenamiento del estado anterior: Esta prueba consistirá en crear un hilo de conversación sobre un tema, se debe comprobar que el robot tiene "memoria" de las preguntas y respuestas anteriores.
- Uso de caracteres extendidos del código ASCII: se enviarán distintas frases que contengan acentos o signos extendidos. El robot, debe contestar correctamente a esas preguntas.
- Uso de frases anidadas: la lógica del robot cuando el usuario envía varias frases seguidas de punto seguido, es la de separar cada frase, y construir una respuesta conjunta para todas ellas. Por tanto esta prueba consistirá en comprobar que ante la entrada de un párrafo, el robot, contesta con sentido a todas ellas.

Cuestionario de perfilado

Aparte de los puntos citados anteriormente, en una conversación tradicional con el robot, se añadirán otras pruebas en el momento en el que el usuario está respondiendo al cuestionario de perfilado, necesario para enviar un MMS.

En este punto habrá que validar:

- Respuesta incorrecta: Si de las dos respuestas posibles a una cuestión, el usuario escribe otra, el robot debe responder con el siguiente mensaje: "La respuesta es incorrecta. ¿Quieres cancelar la encuesta, sí o no?". Si el usuario responde que sí, se le tiene que volver a mostrar la misma respuesta, si responde con un no, automáticamente se debe cancelar el cuestionario, y el usuario volverá a un estado anterior.

- Patrón: Una respuesta puede contener varios patrones de entrada, por ejemplo, antes la respuesta "Real Madrid", el usuario puede contestar con distintas palabras: "R. Madrid", "Madrid"... Por tanto, se realizarán varias pruebas, enviando distintas respuestas que se refieren a la misma palabra.

- Cancelación de la encuesta: Si el usuario cancela la encuesta, o sale de la página en el momento en el que está contestando al formulario, el estado del usuario pasará de "Cuestionando" a "Perfilado".

Capítulo 9

Conclusiones y trabajos futuros

En este capítulo se explicarán las conclusiones que se han tomado tras la realización del proyecto, así como los objetivos que se han cumplido desde la idea inicial, y las nuevas necesidades que han ido surgiendo a lo largo de la realización del mismo. En la sección de Trabajos futuros 9.2, se propondrán nuevas funcionalidades o mejoras para una segunda fase de desarrollo del proyecto.

9.1. Conclusiones

La idea inicial del proyecto era crear una nueva interfaz para el Robot de Telefónica I+D, que supliera las limitaciones que ofrecía la interfaz de MSN integrada en él. Estas limitaciones causaban que no se pudiera inyectar publicidad en distintos formatos en las conversaciones entre un usuario y el robot. Este primer objetivo se cumplió gracias a la interfaz de Google Wave, que permite la inserción de imágenes, vídeos, páginas XHTML en cualquier conversación. Además, se mejora la experiencia del usuario a través de cuestionarios realizados con Gadgets que le permiten elegir las respuestas de un modo más interactivo (imágenes o vídeos), o ver la vista previa de un MMS.

El principal objetivo era que la plataforma de Google Wave, se pudiera comunicar con cualquier aplicación de un entorno externo, en este caso, la del Robot de Telefónica. Este objetivo también se cumplió, ya que Google Wave permite enviar peticiones al puerto 80 de cualquier IP, y por tanto, conectarse con la plataforma de mensajería del operador.

Otro de los objetivos era que el usuario recibiera publicidad orientada a él. Esto se consigue, gracias al módulo de Inteligencia Artificial integrado en el Robot Movistar, que en cuestionarios

realizados sobre las preferencias del usuario, es capaz de crear dinámicamente preguntas en base a sus respuestas.

Como conclusión final, se puede decir, que la mayoría de los objetivos iniciales se han cumplido: adaptación del robot de Telefónica a una interfaz más flexible, inyección de distintos formatos de publicidad en la interfaz, interacción del usuario con la Inteligencia Artificial y envío de SMS y MMS. El único objetivo no cumplido ha sido que no se ha podido poner el producto en producción debido al cierre actual de la interfaz de Google Wave.

9.2. Trabajos futuros

9.2.1. Aplicación multiusuario

En el estado actual del prototipo, el Robot Wave solo se puede comunicar con un usuario, es decir, el usuario que añade al robot como contacto, es el que mantiene el estado con el robot y el que puede enviar SMS y MMS. En el caso en que se añadan más usuarios a la conversación, estos solo serán partícipes de ella como lectores.

Se podrían aprovechar las capacidades que aporta la plataforma de Google Wave, para crear grupos, y que la publicidad que se inyecte sea acorde al perfil del grupo. Gracias a ello, se tendrán captados varios perfiles de usuarios como uno único, y por tanto se obtiene mucha más información y más usuarios autenticados en la aplicación. Esto favorece a las agencias de publicidad, ya que un mensaje publicitario puede ser solicitado por varios contactos del grupo, y no por uno solo.

9.2.2. Conversión al método PUSH

La publicidad que se inyecta en la plataforma de Google Wave es originada a través de interacciones del usuario con el robot. Si el usuario no contacta con él no recibirá ningún mensaje publicitario. En el momento en que el usuario contacta con él y dependiendo de si solicita envío de SMS o MMS o no, se le inyecta el mensaje publicitario si no tiene puntos.

Sería conveniente cambiar el tipo de inyección de publicidad, a la estrategia PUSH[1]. Esto quiere decir, que la publicidad se inyecta un determinado número de veces, a determinadas horas del día, independientemente de si el usuario está conectado a la plataforma o está contactando con

[1] http://en.wikipedia.org/wiki/Push-pull_strategy

el robot. Así, obligamos a que el usuario visualice e interaccione con más mensajes publicitarios, generando más beneficios para las empresas de publicidad y por tanto para el operador.

9.2.3. Nuevos modelos de negocio

Se puede sacar más beneficio de las capacidades del operador de Telefónica si se crean nuevas interacciones como la realización de llamadas a usuarios, el envío de eventos a través de VCalendar[2], el envío de VCards[3] o de mapas a los móviles de sus amigos. Ofreciéndole diferentes funcionalidades al usuario se consigue que éste esté más ligado a la aplicación y por tanto a las agencias de publicidad.

[2] http://www.ultraapps.com/app_overview.php?app_id=19

[3] http://es.wikipedia.org/wiki/VCard

Apéndice A

Manual de Usuario

En este capítulo se explican los pasos necesarios para el acceso a la plataforma Wave, la adición del Robot Wave como un contacto, la comunicación con él, algunos comandos útiles para el envío de MMS y SMS y el establecimiento de una conversación con la Inteligencia Artificial del Prototipo de Telefónica.

A.1. Acceso a la plataforma Wave

En los inicios de Google Wave, para acceder a la plataforma, era necesario tener una cuenta proporcionada por Google, en el caso en que el usuario tuviera un perfil de desarrollador y se justificara que trabajaba en una empresa dedicada al desarrollo Web. Para ello, se realizaba una petición expresa mediante e-mail a Google Wave pidiendo acceso para desarrollar aplicaciones para su entorno.

A finales del desarrollo de Google para la plataforma, se permitía que cualquier usuario con una cuenta de Gmail, pudiera acceder e interactuar con los robots existentes.

Por ello, para entrar en la plataforma, solo es necesario ir a la página de `https://wave.google.com/a/wavesandbox.com/`(07/03/2011), e introducir usuario y contraseña, ya sea de desarrollador o de Gmail.

A.2. Adición del Robot Wave

Una vez dentro de la plataforma, para añadir un contacto, o un robot existente con el que queramos interactuar, es necesario conocer su dirección, e ir a la sección de contactos, donde hay

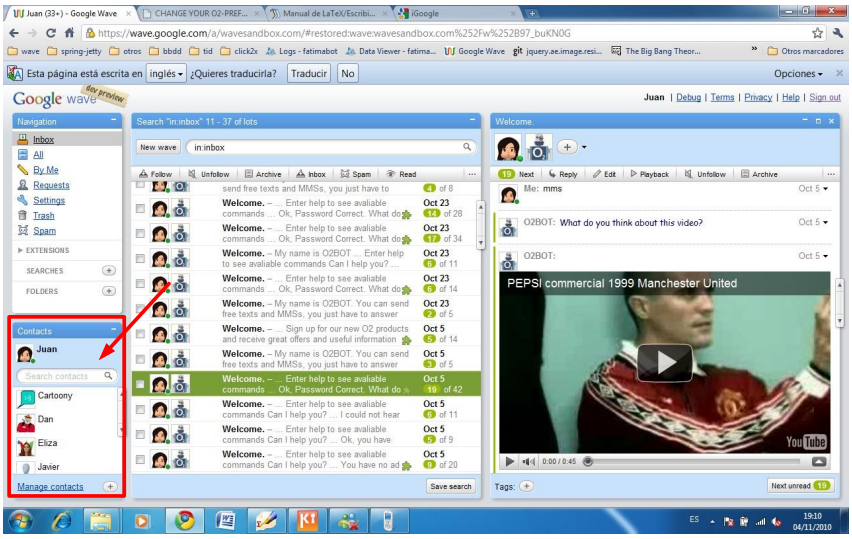


Figura A.1: Contactos

una entrada para escribir la dirección del contacto, o un botón de añadir, para añadirlo. Una vez realizado, tanto si el contacto existe o no, aparece en la lista. La sección de contactos, se muestra *marcada en rojo* en la siguiente figura A.1.

A.2.1. Gestión de Contactos

En la sección de adición de contactos A.1, se puede modificar o rellenar la información completa de uno de ellos, pulsando sobre el enlace "Manage Contacts". Esta página pertenece a la cuenta con la que se ha accedido a la plataforma, y contiene todos los contactos asociados. Se muestra un ejemplo de la página en la figura A.2

A.3. Comunicación con un contacto

Para conversar con un contacto, solo es necesario, añadir una "Wave", en el link "señalado en rojo" que aparece en la figura A.3

En este momento, se abre una pantalla de conversación, y hay que añadir al robot, pulsando sobre el enlace "Add Contact", mostrado en la figura A.4

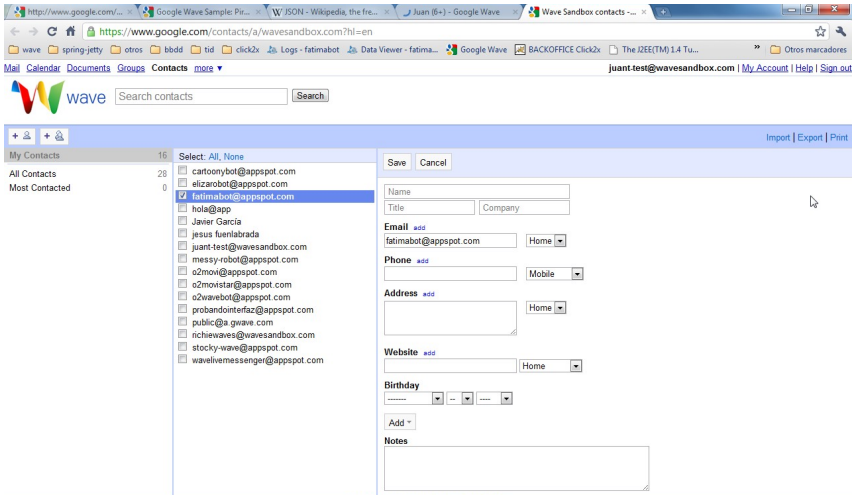


Figura A.2: Modificar Contactos

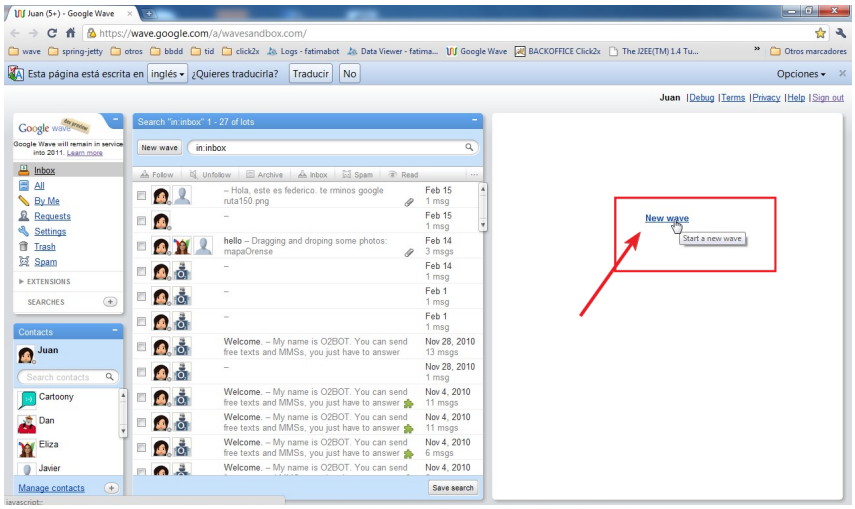


Figura A.3: Nueva Wave

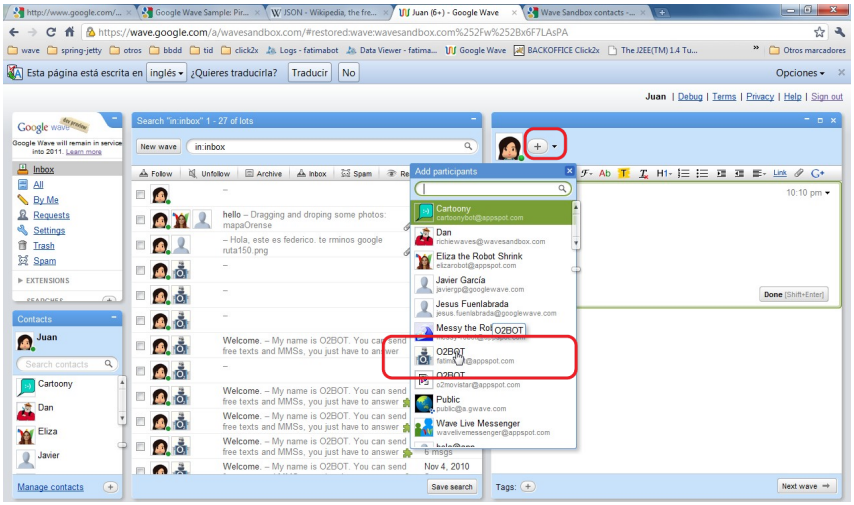


Figura A.4: Añadiendo al Robot

A.4. Autenticación del usuario

En el momento en que se añade al robot, este responde con un mensaje de bienvenida. Si el usuario está registrado en la aplicación, el usuario directamente puede comenzar a dialogar con él, si no es así, el Robot le mostrará un formulario de autenticación. La autenticación está basada en un mensaje que se envía al móvil con la contraseña. Para ello, el usuario introduce el móvil en el formulario, y en este momento se le envía un SMS con el texto de la contraseña. Cuando el usuario reciba la contraseña, debe introducirla, y si es correcta, el usuario ya puede interactuar con el robot. Un ejemplo de autenticación se muestra en la figura A.5

A.5. Formulario de Perfil

Cuando un usuario comience a interactuar con el robot, se le pedirán datos sobre sus preferencias a través de un Gadget. Una vez rellenado, el usuario ya podrá conversar con la Inteligencia Artificial.

A.6. Envío de SMS

Para el envío de un SMS, es necesario que el usuario tenga puntos. Estos puntos se obtienen respondiendo a cuestionarios de publicidad o dando la opinión sobre distintos vídeos que se le muestran acordes a sus preferencias. En el momento en que el usuario quiera escribir un SMS, debe

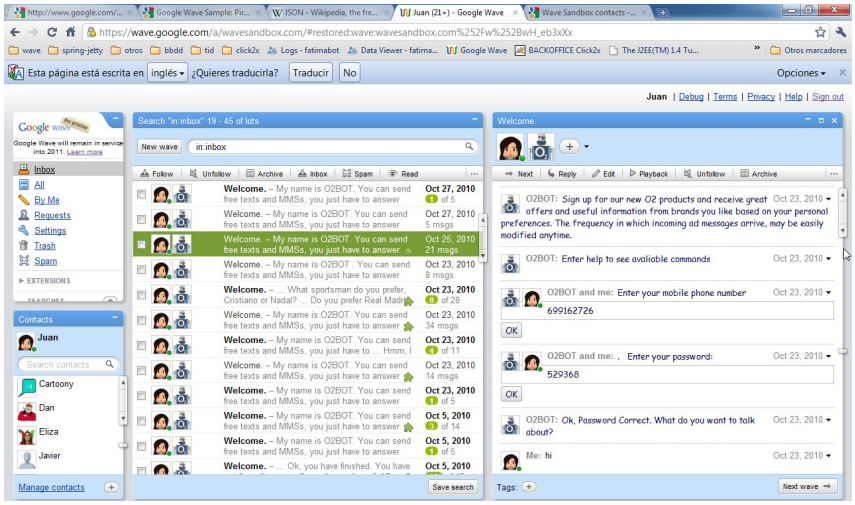


Figura A.5: Autenticación del usuario

escribir : TXT <NUMERO><TEXTO>o SMS <NUMERO><TEXTO>. El usuario también puede escribir la palabra “SMS”, y ya el robot le define la sintaxis correcta.

Si el usuario escribe este comando, pero no tiene puntos, debe completar un cuestionario que le realiza el robot. A través de este cuestionario se obtienen preferencias del usuario. Una vez respondido, el usuario podrá enviar un SMS.

Si el usuario no tiene puntos, y ya ha rellenado el cuestionario, se le muestra un vídeo acorde a sus preferencias, sobre el que el usuario puede obtener 5 puntos equivalentes a un SMS. Una vez responda con su opinión, el robot le informará de los puntos que tiene.

Un ejemplo de los pasos para el envío de un SMS se muestra en la figura A.6

A.7. Envío de MMS

Para el envío de un MMS, el usuario solo tiene que escribir “MMS”. Si el usuario no tiene puntos, se le muestra un vídeo acorde a sus preferencias, sobre el que tiene que responder con su opinión, igual que en el caso del SMS. A continuación, se le muestra un formulario pidiéndole los datos del MMS. Estos datos son: teléfono destino, texto del MMS, y fichero a enviar. Una vez que el usuario introduce la imagen o vídeo a enviar, se le muestra una vista previa del mensaje, y si el usuario pulsa enviar, se le envía al destino. Un ejemplo del formulario de envío de un MMS se muestra en la figura A.7

A.8. Comandos útiles

Si el usuario quiere visualizar los comandos de los que dispone el robot, solo tiene que escribir "HELP". Uno de los comandos de los que dispone es "DeleteUser", que permite borrar al usuario de la aplicación. Así, si este mismo usuario vuelve a abrir una "Wave" y añadir al Robot, le aparecerá el formulario de autenticación.

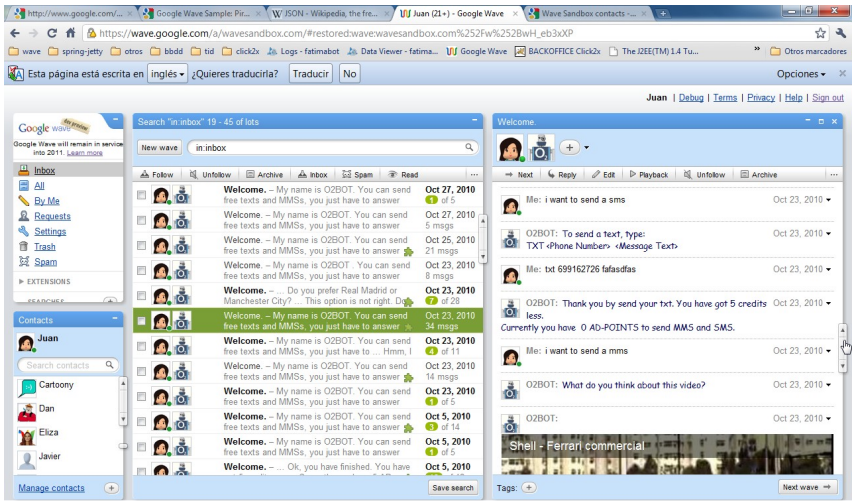


Figura A.6: Envío de un SMS

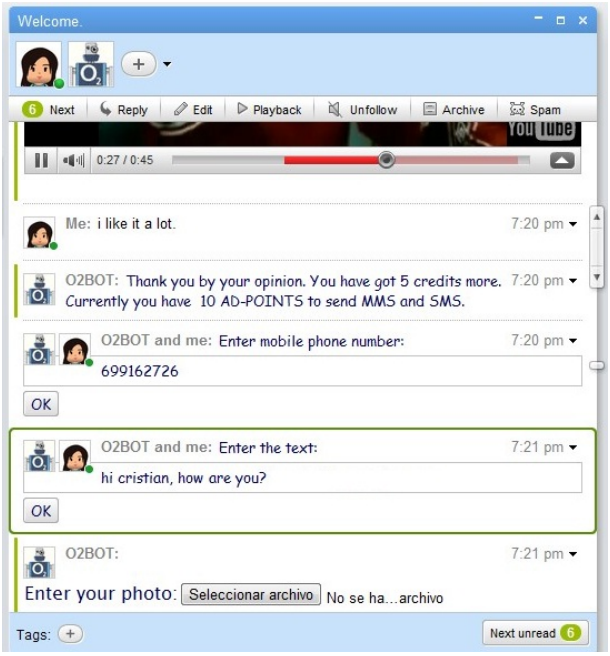


Figura A.7: Formulario de envío de un MMS

Apéndice B

Manual de instalación, configuración y mantenimiento.

En este capítulo se explican detalladamente todos los pasos para crear un robot para Google Wave, así como un Gadget con el API de Gadgets de Wave en el lenguaje de programación Java. Para ello, se detallarán todos los pasos necesarios tanto para la instalación de software como para el desarrollo y mantenimiento de la aplicación a nivel técnico.

B.1. Configuración de la JVM

Antes de crear una aplicación para Google Wave en Java, hay que instalar la máquina virtual de Java en el sistema operativo. AppEngine trabaja con versiones del API de JDK de Java a partir de la versión 1.5.

Para comprobar en Windows o Linux la versión de JDK instalada, hay que ejecutar el siguiente comando: java -version

Si la versión instalada es menor, se puede descargar a través de la página de Oracle [1].

B.2. Instalación del entorno de desarrollo

En principio, se puede utilizar cualquier entorno de desarrollo para crear una aplicación sobre el entorno de AppEngine. Pero Eclipse [2], es compatible con un complemento de Google

[1] http://www.oracle.com/technetwork/java/javase/downloads/index.html(07/03/2011)

[2] http://www.eclipse.org/(07/03/2011)

que facilita la creación de aplicaciones que utilicen AppEngine, ya que incluye el SDK del servidor de aplicaciones y herramientas para crear y subir aplicaciones. Por ello, es necesario descargar e instalar Eclipse. Las versiones de Eclipse con las que el plugin son compatibles son a partir de la versión 3.4. Las páginas de las que se puede descargar el entorno dependiendo de la versión son las siguientes:

1. Eclipse Ganymode 3.4 [3].
2. Eclipse Galileo 3.5 [4].
3. Eclipse Helios 3.6 [5].

El prototipo de este proyecto se ha desarrollado con la versión 3.5 de Eclipse.

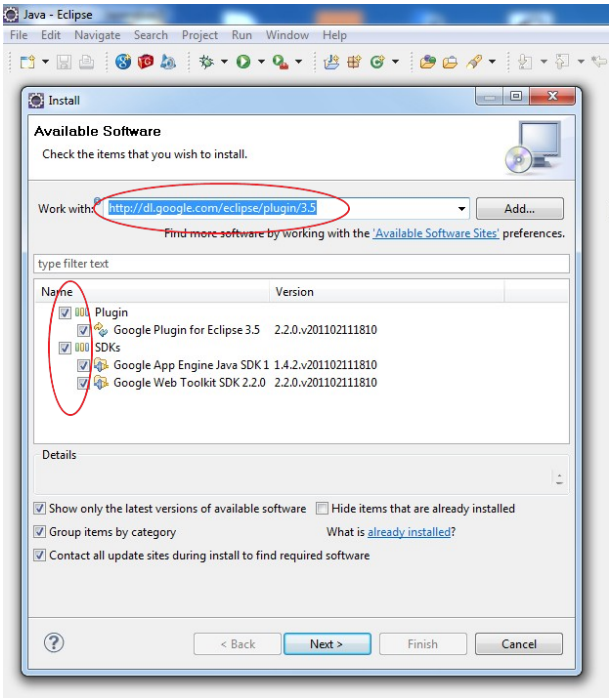
B.3. Instalación del plugin para Eclipse

Una vez instalado el entorno de desarrollo, hay que añadirle el plugin. Se puede añadir a través del centro de actualizaciones del propio Eclipse, o descargándose los paquetes y almacenándolos en la carpeta de plugins correspondiente del entorno de desarrollo. En este caso, se usará el centro de actualizaciones y se tomará como referencia Eclipse Galileo versión 3.5. Para instalar el complemento, hay que realizar los siguientes pasos.

- Abrir Eclipse y seleccionar *Help >Install New Sofware*.
- En el apartado *Work With*, hay que introducir la url `http://dl.google.com/eclipse/plugin/3.5`, y marcar todos los paquetes que aparecen en la pantalla, como se muestra en la figura B.3
- Pulsar en *Siguiente*.
- Revisar los nuevos paquetes y pulsar en *Siguiente*. Pulsar en Finalizar.

Ya está listo el entorno de desarrollo para comenzar a crear una aplicación para Google AppEngine.

[3] `http://www.eclipse.org/ganymede/`(07/03/2011)
[4] `http://www.eclipse.org/galileo/`(07/03/2011)
[5] `http://www.eclipse.org/helios/`(07/03/2011)



B.4. Creación de la aplicación

Para crear una aplicación con el complemento de Google para AppEngine, es necesario seguir los siguientes pasos:

- Crear un nuevo proyecto, seleccionando el plugin de Eclipse, como se muestra en la figura B.1
- Añadir nombre del proyecto y nombre de los paquetes de las clases Java, como se muestra en la figuraB.2
- Como se puede ver en la figura B.3, se crea la estructura de un proyecto para AppEngine.

La aplicación ya está lista para ser desplegada localmente a través de: *Debug As* >Web Application.

B.5. Configuración de AppEngine

En el caso en que la aplicación se quiera desplegar en el entorno AppEngine de Google, hay que realizar los siguientes pasos:

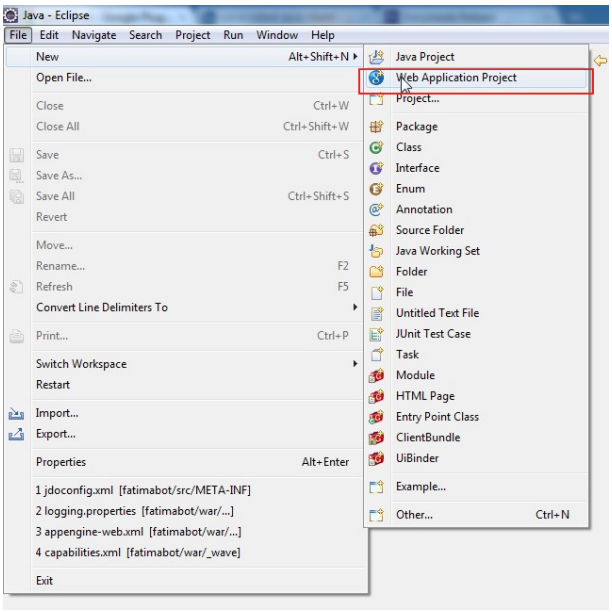


Figura B.1: Creación de un nuevo proyecto

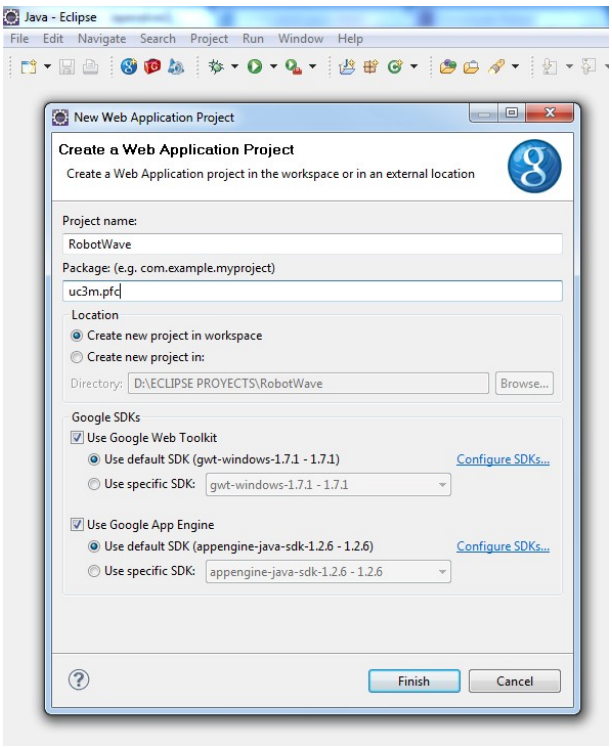


Figura B.2: Configuración del nuevo Proyecto I

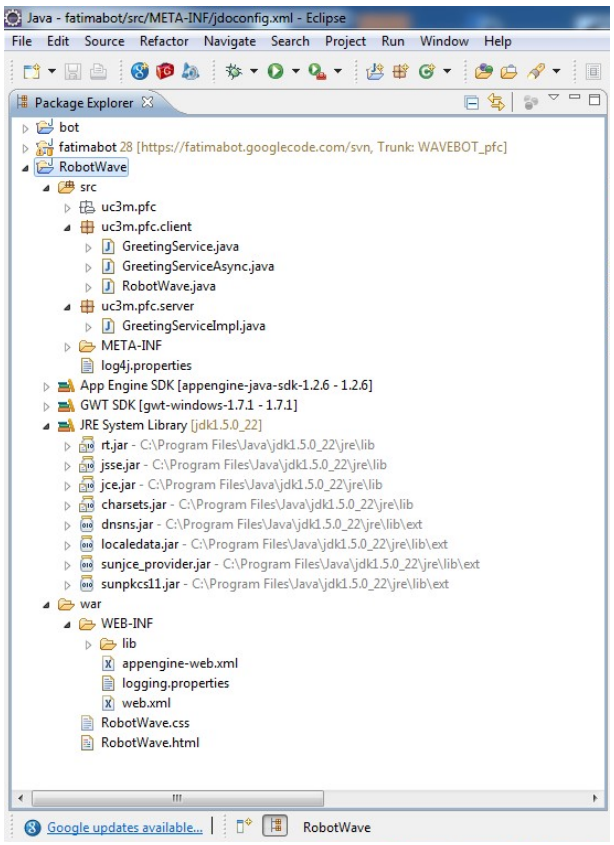


Figura B.3: Estructura de una nueva aplicación

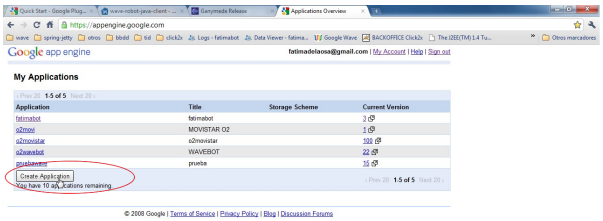


Figura B.4: Nuevo Id de aplicación en AppEngine

- Tener una cuenta en Gmail, y registrar un nombre de aplicación. Para ello, es necesario crearse una cuenta en Google AppEngine [6], y crearse un ID de aplicación como se muestra en la figura B.4

- Asignarle ese ID de aplicación al proyecto que se ha creado en el entorno de desarrollo en: *Proyecto >Google >App Engine Settings*, en el apartado de *Application Id*.

- Para desplegar la aplicación en los servidores de AppEngine: *Google >Deploy to AppEngine*

B.6. Adición de las librerías de robots

Ya tenemos configurada una aplicación propia de AppEngine, pero falta, añadir las dependencias necesarias para crear robots de Google Wave sobre AppEngine. Para ello, es necesario realizar los siguientes pasos:

Para la primera versión de Google Wave, que es la que se usa en este prototipo, hay que descargarse de la página de librerías cliente de Google Wave, los siguientes JAR:

- json(07/03/2011)

- jsonrpc(07/03/2011)

- gson(07/03/2011)

- Biblioteca cliente Google Wave(07/03/2011).

Una vez descargados, hay que copiarlos en la carpeta lib del directorio WEB-INF de la aplicación (war >WEB-INF >lib), y añadirlas al CLASSPATH del proyecto (Propiedades >Add Jars), junto con el resto de dependencias que se necesiten para la aplicación.

[6] `https://appengine.google.com`(07/03/2011)

B.7. Subida de la aplicación a AppEngine

Una vez configurada la aplicación con las dependencias, ya puede ser desplegada en el entorno de AppEngine. Para ello, en propiedades del proyecto sobre Eclipse, seleccione Deploy to AppEngine, configurando antes el Id de la aplicación. En este proceso, se le preguntará por el usuario y contraseña de su cuenta de AppEngine.

B.8. Configuración del versionado

Cada vez que se quiera modificar o añadir un nuevo evento, hay que avisar al servidor de Aplicaciones. Esto se realiza a través de la versión de la aplicación. A la hora de añadir el evento y configurarlo, hay que incrementar la versión del servidor. Esto se configura en las propiedades de la ventana de "Deploy to AppEngine".

Apéndice C

Presupuesto del proyecto

En este apéndice se presentan justificados los costes globales de la realización de este Proyecto Fin de Carrera. Tales costes, imputables a gastos de personal y de material, se pueden deducir de la siguiente figura C.1

UNIVERSIDAD CARLOS III DE MADRID
Escuela Politécnica Superior

PRESUPUESTO DE PROYECTO

1.- Autor:Fátima Montaña de la Osa Barriga

2.- Departamento:Telemática

3.- Descripción del Proyecto:

- Titulo: Desarrollo de una interfaz en Google Wave para un robot conversacional de una plataforma publicitaria
- Duración (meses): **8,2**
- Tasa de costes Indirectos: **20%**

4.- Presupuesto total del Proyecto (valores en Euros): 58.143,68

5.- Desglose presupuestario (costes directos)

PERSONAL

Apellidos y nombre	N.I.F.	Categoría	Meses	Coste mes	Coste (euro)
de la Osa Barriga, Fátima	xxxxxxxx	Ingeniero Junior	8,2	3.000,00	24.600,00
Turrión, Juan	xxxxxxxx	Jefe de Proyecto	2	6.000,00	12.000,00
González, Miguel Ángel	xxxxxxxx	Ingeniero Senior	1,6	4.500,00	7.200,00
Jesús Arias Fisteus	xxxxxxxx	Ingeniero Senior	1	4.500,00	4.500,00
		Hombres mes	**12,8**	**Total**	48.300,00

EQUIPOS

Descripción	Coste (Euro)	% Uso dedicado proyecto	Dedicación (meses)	Periodo de depreciación	Coste imputable [d]
Pc de sobremesa	800,00	100	8,2	60	109,33
Ratón	10,00	100	8,2	60	1,37
Teclado	10,00	100	8,2	60	1,37
Monitor	300,00	100	8,2	60	41,00
				Total	153,07

OTROS COSTES DIRECTOS DEL PROYECTO[e]

Descripción	Empresa	Costes imputable
	Total	0,00

6.- Resumen de costes

Presupuesto Costes Totales	
Personal	48.300
Amortización	153
Costes de funcionamiento	0
Costes Indirectos	9.691
Total	58.144

Figura C.1: Presupuesto

Bibliografía

[1] G. Code. Aspectos generales del api java de almacén de datos. Technical report, 2010. Disponible en http://code.google.com/intl/es/appengine/docs/java/datastore/overview.html(07/03/2011).

[2] G. Code. Tareas programadas mediante cron para java. Technical report, 2010. Disponible en http://code.google.com/intl/es/appengine/docs/java/config/cron.html(07/03/2011).

[3] A. Farr. Wave s web of protocols. Technical report, 2009. Disponible en http://www.cubiclemuses.com/cm/about(07/03/2011).

[4] Google. El servicio memcaché. Technical report, Google Wave Corporation, 2009. Disponible en http://code.google.com/intl/es/appengine/docs/java/memcache/overview.html(07/03/2011).

[5] Google. Google app engine. Technical report, Google Corporation, 2009. Disponible en http://code.google.com/intl/es/appengine/docs/whatisgoogleappengine.html(7/03/2011).

[6] Google. Google app engine. Technical report, Google Corporation, 2009. Disponible en http://code.google.com/intl/es/appengine/docs/java/urlfetch/usingjavanet.html(07/03/2011).

[7] Google. Google app engine. Technical report, Google Corporation, 2009. Disponible en http://code.google.com/intl/es/apis/wave/guide.html(07/03/2011).

[8] Google. Google app engine. Technical report, Google Corporation, 2009. Disponible en http://code.google.com/intl/es-ES/apis/wave/extensions/robots/basics.html#Identity(07/03/2011).

[9] Google. Google app engine. Technical report, Google Corporation, 2009. Disponible en http://code.google.com/intl/es/appengine/docs/whatisgoogleappengine.HTML(07/03/2011).

[10] Google. Google wave. Technical report, Google Corporation, 2009. Disponible en http://code.google.com/intl/es/apis/wave/extensions/robots/protocol.html(07/03/2011).

[11] Google. Google wave. Technical report, Google Corporation, 2009. Disponible en http://www.waveprotocol.org/Home(07/03/2011).

[12] Google. Google wave. Technical report, Google Corporation, 2009. Disponible en http://code.google.com/intl/es-ES/apis/wave/extensions/robots/basics.html#RobotProfiles(07/03/2011).

[13] Google. Tutorial de gadgets de wave. Technical report, Google Wave Corporation, 2009. Disponible en http://code.google.com/intl/es-ES/apis/wave/extensions/gadgets/guide.html(07/03/2011).

[14] Google. Uso de archivos estáticos. Technical report, Google Wave Corporation, 2009. Disponible en http://code.google.com/intl/es/appengine/docs/java/gettingstarted/staticfiles.html(07/03/2011).

[15] L. Vogel. Java api logging. Technical report, 2010. Disponible en http://www.vogella.de/articles/Logging/article.html(07/03/2011).

[16] Wikipedia. Base64. Technical report, 2010. Disponible en http://es.wikipedia.org/wiki/Base64(07/03/2011).