PONTIFICIA **UNIVERSIDAD CATÓLICA** DEL PERÚ

PONTIFICIA UNIVERSIDAD CATOLICA DEL PERU

Facultad de Ciencias e Ingeniería

IMPLEMENTACIÓN DE UNA PLATAFORMA DE MENSAJERÍA UNIFICADA INTEGRADA A UNA APLICACIÓN B2B

Tesis para optar por el título de ingeniero de las Telecomunicaciones

Presentado por

JAMPIER CHRISTOPHER MENDOZA VARAS

Septiembre 2008

Lima – Perú

RESUMEN

En esta tesis se estudio y desarrollo la plataforma de mensajería unificada para luego integrarla a una aplicación B2B existente. Primero se inició con un estudio teórico de las tecnologías involucradas, además fue necesario hacer un análisis de las posibles herramientas a utilizar.

Luego se diseñó e implementó el cliente prototipo que integra los servicios de Voz y de mensajería junto con el correspondiente módulo de registro de todos estos eventos en una base de datos compatible con la aplicación. Una vez terminado el cliente se integró a la aplicación B2B.

Finalmente se muestran algunas pruebas de desempeño, recomendaciones y los avances futuros sobre nuevas implementaciones que se pueden realizar sobre la plataforma.

DEDICATORIA

A mi familia,

Mis amigos de toda la vida.

AGRADECIMIENTOS

A Dios por mostrarme que puede hacer cosas extraordinarias en gente ordinaria, por abrirme las puertas, por haber puesto gente extraordinaria en mi camino y porque creo que en El se pueden hacer grandes proezas.

A mis padres por su enorme esfuerzo por darme una excelente educación y por su confianza durante todo este tiempo.

Al Ing. Enrique Larios, mi asesor, porque hay muchas cosas porque agradecerle y sobre todo por su invaluable amistad.

A los jefes de mi trabajo, al Ing. Jorge Berrocal Pérez Albela y Dn. Gonzalo Alegría Varona por la confianza y el apoyo puesto en mí y por la satisfactoria experiencia que me brindaron.

A todos mis amigos de quienes aprendí mucho, aunque algunos ya no los veo, los tengo siempre presentes.

A la comunidad desarrolladora por compartir las herramientas fundamentales para la implementación de mi tesis.

Y si me olvido de alguien sin querer, pues gracias también.

ÍNDICE

LISTA DE FIGURAS

LISTA DE TABLAS

INTRODUCCIÓN

La fusión de grandes empresas tecnológicas es algo que ahora ocurre muy a menudo y más allá de un beneficio económico, el fin de todo esto es unir tecnologías e infraestructuras que acelerarán o mejorarán el control sobre algún grupo de servicios, trayendo consigo mayor productividad a las empresas y usuarios que se vean afectados por esta fusión.

Actualmente la convergencia de tecnologías es importante ya que impulsa un desarrollo tecnológico tomando lo mejor de cada uno para generar un único producto más fácil y rápido de usar. En comunicaciones, la convergencia de servicios nace de la necesidad de poder tener el control y gestión de nuestros diferentes servicios de comunicaciones desde un mismo punto o dispositivo de telecomunicaciones.

La plataforma de mensajería unificada para aplicaciones business to business (B2B) que se desarrollará en esta tesis, está teniendo gran demanda actualmente por parte de las empresas que requieren tener un mejor control sobre las comunicaciones que realizan, en vez de tener un dispositivo diferente para cada servicio que hayan contratado (teléfono, computadora, teléfono IP, etc.) Es mas rápido acceder a todos ellos mediante un único producto, además de esto tener también la posibilidad de saber los detalles de estas comunicaciones para poder verificar la productividad de los empleados dentro de una empresa.

En esta tesis se describe los conocimientos y destrezas necesarias y en base a ellos los pasos a seguir para la satisfactoria implementación de esta plataforma mediante el uso de software libre, además de los avances futuros a desarrollarse sobre esta plataforma.

Capítulo 1

Estudio teórico y descripción de tecnologías involucradas en la aplicación.

En este capítulo se exponen los conceptos de Voz sobre IP, mensajería instantánea y comunicaciones en la red conmutada básica (PSTN) y su relación de estos con la aplicación B2B.

1.1 Voz sobre IP

1.1.1 Protocolos de señalización

En Voz sobre IP se debe entender por señalización como la función de ejercer control sobre el establecimiento, duración y finalización de una comunicación entre dos entidades. Es decir, poder determinar si ambas entidades pueden establecer una comunicación, y si se establece, asegurar que el intercambio de información fluya correctamente y finalmente poder terminar dicha comunicación apropiadamente. Los protocolos que se expondrán a continuación son SIP e IAX.

1.1.1.1 SIP

Es un protocolo de inicio de sesión multipropósito, es decir, puede ser usado para iniciar una sesión entre dos entidades que deseen transmitir cualquier tipo de datos tipo multimedia. Este protocolo fué creado en 1996 por la IETF (engineering task force) y está descrito por la última RFC 3261 (año 2003) y nace en respuesta a la complejidad presentada por el protocolo H323.

SIP a diferencia de su antecesor H323 es mucho más sencillo y muchas veces es comparado con HTTP y SMTP, ya que transmite la información en modo texto lo que permite una interpretación más fácil y rápida, lo cual hasta ahora le ha dado el primer lugar como protocolo de señalización de voip más usado.

Este protocolo utiliza otros protocolos adicionales RTP (protocolo de transporte en tiempo real), RCTP (protocolo de control de transporte en tiempo real), UDP (protocolo de diagrama de usuario) y SDP (protocolo de descripción de sesión). Los paquetes SIP se envían mediante UDP por el puerto 5036 para establecer una sesión, RTP es usado para el envío del flujo de datos multimedia en tiempo real, RTCP es para controlar este flujo de datos y SDP es para poder saber la descripción de la sesión, es decir sobre las entidades participantes en la comunicación como por ejemplo la dirección IP, el puerto a usar, el tipo de datos a enviarse y el códec a negociar.

En Internet las entidades que desean establecer una comunicación generalmente se encuentran detrás de entornos NAT (traductores de dirección de red), SIP posee campos dentro de su cabecera que le permiten saber de que IP publica vienen las entidades participantes, sin embargo RTP/RCTP no los posee, mas bien este último se vale de la información que le provee SDP el cual se ejecuta en la estación cliente que en entornos NAT posee una dirección IP privada, entonces el resultado es que RTP apuntará a direcciones privadas que no están mapeados en Internet por lo tanto se podrá establecer una sesión entre ambas entidades pero no se podrá transmitir ningún tipo de dato.

Para solucionar este problema existen varias técnicas, que se basan en determinar cual es el par dirección IP pública y puerto con la que la Internet ve al usuario, de esta manera esta información es usada por RTP que transmitirá

los datos hacia direcciones públicas que si están mapeadas en la Internet. Entre los más populares se encuentran los servidores STUN, que no funcionan en NAT simétricos.

1.1.1.2IAX

IAX significa Inter Asterisk Exchange, es un protocolo de señalización y a su vez también fue creado especialmente para la transmisión de datos de audio y video. Fue creado por Digium para hacer la interconexión entre servidores Asterisk más eficiente en lo que al uso de ancho de banda se refiere, ya que una de sus principales características es la reducción de la cabecera de los paquetes que van hacia el mismo destino. Además IAX transmite la información codificada en modo binario, lo que lo hace más rápido y fácil de procesar. Este protocolo aún no tiene RFC y se encuentra en su versión más reciente IAX2 que fue lanzada en el año 2005. En lo que refiere al resto del documento cada vez que se mencione IAX, este será en su versión más reciente.

IAX como se adelantó anteriormente, solo utiliza un único puerto UDP, el 4569 tanto para transmisión de datos y para señalización, ahora para la transmisión de datos utiliza unas tramas llamadas “miniframe” que son mas pequeñas que las tramas “fullframe”, estas últimas son usadas para la señalización ya que pueden llevar información consigo acerca del inicio, duración y finalización de la sesión. Fue creado especialmente para optimizar la transmisión de datos tanto en uso de ancho de banda como en el establecimiento de la misma. Entonces al viajar tanto la señalización como los datos por un mismo canal, IAX puede atravesar fácilmente entornos de NAT simplificando aun más su funcionamiento.

1.1.1.3Comparativa de ambos protocolos

Para comparar ambos protocolos nos basaremos en cuatro puntos específicos [FOR2006]:

- Ancho de banda: ambos protocolos pueden trabajar con los mismos códecs, IAX consume menos ancho de banda que SIP, debido a que

reduce la cabecera hacia paquetes que van al mismo destino, dicho destino puede ser un proveedor de voip.

- NAT: a diferencia de SIP, IAX envía tanto el flujo de señalización como el flujo de datos por el mismo canal, por tanto atraviesa fácilmente entornos NAT muy comunes actualmente.

- Puertos y flujos de audio: SIP ocupa mayor cantidad de puertos para la comunicación que IAX que usa solo uno, ya que utiliza un puerto para la señalización y otros dos para el RTP / RTCP.

- Estandarización y uso: SIP es el protocolo más usado actualmente por lo que podemos encontrar más equipos y empresas que ofrecen terminaciones de voip basadas en SIP, en cambio IAX es un protocolo nuevo pero que esta teniendo cada vez mas aceptación en el mercado.

1.1.2 Códecs

Abreviación en inglés de Compressor / Decompressor. Algoritmo que se encarga de comprimir y descomprimir un flujo de datos normalmente relacionados con multimedia, es decir, audio o video

Estos códecs nos permiten un ahorro en la transmisión de información ya que la comprime pero esto afecta a la calidad dependiendo de cuanto se ha comprimido. En la tabla 1.1 se mencionan los tres códecs a analizar.

TABLA 1.1 CÓDECS PARA VOIP
Fuente: [COD2006]

Códec	Frecuencia de muestreo(khz)	Ancho de banda (kbps)	Licencia	Ventajas	Desventajas
G711	8	64	No	Diseñado para entregar calidad	Con la sobrecarga de la cabecera

				máxima de voz. Muy bajo consumo de cpu	consume mas de 64kbps y esto aumenta si es que estamos en entornos half duplex, ya que se requerirían 128kbps tanto para bajado como para subida
G729	8	8	patentado	Uso excelente del ancho de banda sin afectar en mucho la calidad de voz	Se requiere pagar licencia para su uso
GSM	8	13	No	Uso eficiente del ancho de banda	

1.1.2.1GSM (RTE-LTP)

El GSM (Global System for Mobile communications) usa un códec cuyo nombre original es “Regular Pulse Excitation Long-Term Prediction” (RTE-LTP) pero comúnmente es llamado GSM. Este códec utiliza las muestras previas para predecir las actuales. Esto lo hace dividiendo la señal de voz en bloques de 20ms, cada bloque es codificado generando una taza de 13 kbps con bloques de 260bits. [GSM2004]

1.1.2.2 G711

Este códec no utiliza compresión alguna para la señal de voz, por lo que ofrece la mejor calidad de voz y la menor latencia. Este códec genera una taza de 64kbps ya que codifica la señal mediante modulación de código de pulso (PCM) de 8bits por muestra y 8KHz de muestreo y es usado por la red RTC o PSTN y las líneas ISDN. [G712007]

1.1.2.3G729

Este códec comprime la voz haciendo uso de la predicción lineal con excitación por código algebraico de estructura conjugada, es decir se aprovecha de que las muestras consecutivas de la señal de voz no varían mucho y solo codifica algebraicamente las diferencias, esto genera una taza de 8Kbps [QUI2006]. El inconveniente con este códec es que no es gratuito tiene un coste por canal concurrente [G722006] y además no se usa para el envió de tonos de multifrecuencia (DTMF).

1.2 Mensajería instantánea

La mensajería instantánea es una forma de comunicación en tiempo real que permite el intercambio de texto e imágenes emotivas (emoticonos) entre dos usuarios. Es una herramienta ampliamente conocida y muy difundida tanto así que se ha convertido en una herramienta de trabajo imprescindible para las personas y empresas, independientemente si mejora o empeora la productividad. Los servicios mas usados actualmente son Instant Messenger de AOL, MSN Messenger de Microsoft, Yahoo! Messenger e ICQ (que utilizan protocolos propietarios) y Google Talk (que usa el protocolo abierto Jabber).

1.2.1 Protocolo Jabber

Extensión del protocolo XMPP, es un protocolo libre para mensajería instantánea y está gestionado por la XMPP Estándar Fundation, estandarizado bajo la RFC3921 [JAB2007].

Entre sus principales características destacan [AST2006]:

- Abierto: este protocolo al ser abierto goza de todas las ventajas del software libre, como el soporte que brinda la comunidad desarrolladora, además, el código está abierto y disponible para ser modificado sin pagar ninguna licencia comercial, etc.

- Extensible: al estar basado en XMPP y este último en XML, se pueden añadir mejoras y funciones personalizadas sobre el protocolo, pero para

que estas sean interoperables son manejadas por la Jabber Software Foundation.

- Descentralizado: un usuario puede descargar e instalar su propio servidor con la opción de integrarlo a la extensa red de servidores basados en jabber.

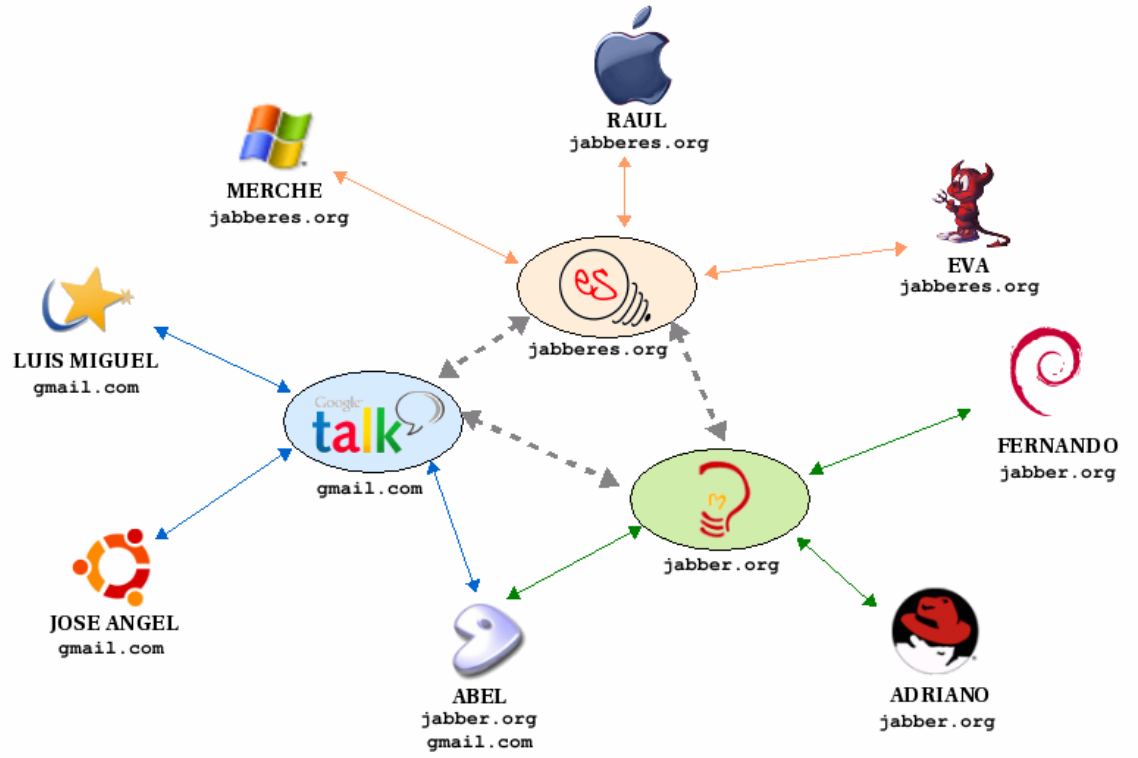

FIGURA 1.1 DESCENTRALIZACIÓN ENTRE SERVIDORES DE MENSAJERÍA INSTANTÁNEA BAJO EL PROTOCOLO JABBER
Fuente: [JAB2007]

- Seguro: Cualquier servidor de Jabber puede ser aislado de la red pública Jabber, cualquier implementación del servidor usa SSL para las comunicaciones cliente-servidor y numerosos clientes soportan PGP-GPG (cifrado asimétrico) para encriptar las comunicaciones de cliente a cliente. Además, está en desarrollo una seguridad más robusta gracias al uso de SASL y contraseñas de sesión.

- Multiredes: este protocolo posee extensiones que le sirven de pasarela, lo cual le permite comunicarse con otros clientes de diferentes protocolos.

1.2.2 Protocolo XMPP

Protocolo extensible de mensajes y de presencia basado en XML. Fue formalizado por la IETF en 2002 y 2003 y actualmente se encuentra estandarizado bajo la RFC 3920 y 3921 que fueron publicadas en el 2004.

XMPP fue creado para ser el protocolo base de comunicaciones en tiempo real, es decir puede ser usado en diversos tipos de aplicaciones tales como mensajería instantánea, de presencia, de negociación de medios, pizarras virtuales, colaboración, voz sobre IP usando XML y cualquiera que se base en XML, ya que al estar escrito bajo este estándar lo hace extensible e interoperable sobre cualquier red, como la red de servidores jabber, y aplicación que esta bajo XML, debido a que el estándar XML es bastante extendido. [XMM2008]

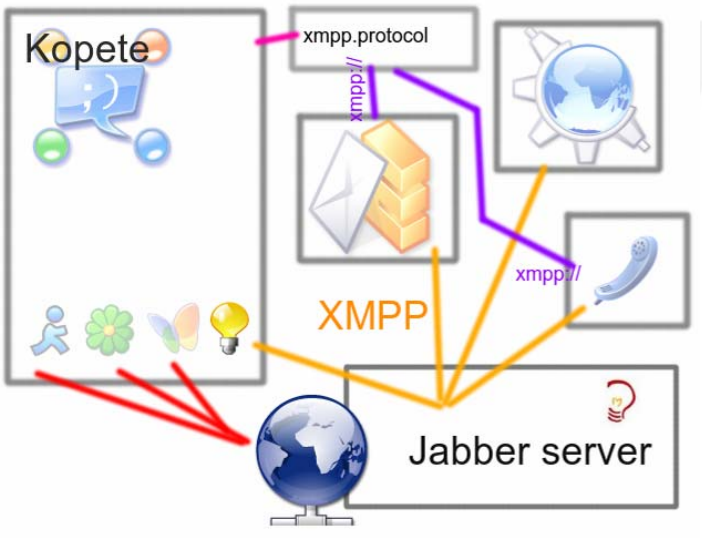

FIGURA 1.2 INTEROPRABILIDAD de XMMP CON OTRAS APLICACIONES
Fuente: [INT2006]

1.3 Red de Telefonía básica (RTC)

La RTC o PSTN por sus siglas en inglés, es una red de telefonía, inicialmente analógica y creada para la transmisión de voz, que es ahora totalmente digital por lo

que puede transportar además de tráfico de voz, también datos, tales como el fax. En esta red dos terminales telefónicos se comunican a través de una central de conmutación mediante un solo canal tanto para recibir como para transmitir voz y datos.

La PSTN es la red de redes públicas de telefonía del mundo, las llamadas se realizan usando el protocolo de señalización SS7. El canal digital básico (BRI) es de 64Kbps debido a que para codificar el audio utiliza PCM de 8 bits por muestra y 8KHz de muestreo (como se mencionó anteriormente en el códec), luego estos circuitos son multiplexados sobre canales de gran capacidad los cuales son llamados troncales. Estas troncales agrupan los canales en cierto número dependiendo si el estándar es europeo, americano o asiático. Por ejemplo, al troncal básico del estándar europeo se le llama E1, el cual agrupa 30 canales digitales básicos (BRI), luego se pueden tener canales de tráfico aun mayores agrupando troncales generando así E4 que agrupa a cuatro E1, E16, y así sucesivamente. Si se tratara del estándar americano estaríamos hablando de un T1 con 24 canales básicos, y por último si fuera el asiático tendríamos una troncal de 31 canales llamado J1. [PST2007]

1.4 Plataforma de mensajería unificada integrada a una aplicación B2B

1.4.1 Definición

La plataforma de mensajería unificada de la cual trata esta tesis, es un sistema basado en software libre que integra diversos servicios de comunicaciones ya conocidos como voz sobre IP y la mensajería instantánea (chat) así como también los convencionales como las llamadas hacia la RTC o PSTN. Así en el sistema final un usuario podrá realizar comunicaciones de voip, chat y realizar llamadas hacia teléfonos convencionales a través de una única interfaz, a su vez estos eventos se registran en una agenda para su posterior uso y aprovechamiento de dicha información por parte de la empresa. Esta plataforma está orientada a una aplicación business to business (B2B), es decir está hecha también para ser usada por pequeñas y medianas empresas, es decir las pymes, que deseen poseer una suite integrada de sus comunicaciones con la posibilidad de controlar y gestionar todo este flujo de información a su preferencia.

1.4.2 Aplicaciones

La posibilidad de integrar la comunicación e información de una empresa facilita mucho la automatización de las rutinas comerciales y además permite extender la cobertura de la empresa mas allá de una oficina o un edificio, es decir podemos estar hablando de agentes, colaboradores internacionales conectados a la plataforma sobre quienes se puede ejercer control a distancia mediante el registro de todos sus eventos de comunicaciones que realicen.

1.5 Tecnologías anexas a la plataforma de mensajería unificada.

1.5.1 PBX

Una PBX (Private Branch Exchange) es una central telefónica que puede ser usada por una empresa para sus comunicaciones internas sin un proveedor telefónico (de ahí el término privado), así como también para las comunicaciones externas entrantes o salientes de la RTC. De esta manera se reducen los costos al tener interconectados a varios usuarios de la organización en vez de contratar al proveedor una línea telefónica por cada uno. [ORT2007]

Este término no sólo lo encontramos en la telefonía analógica, sino también en la telefonía IP, por lo que también se hace referencia a una PBX IP, con diversas alternativas en sus versiones comerciales como en software libre.

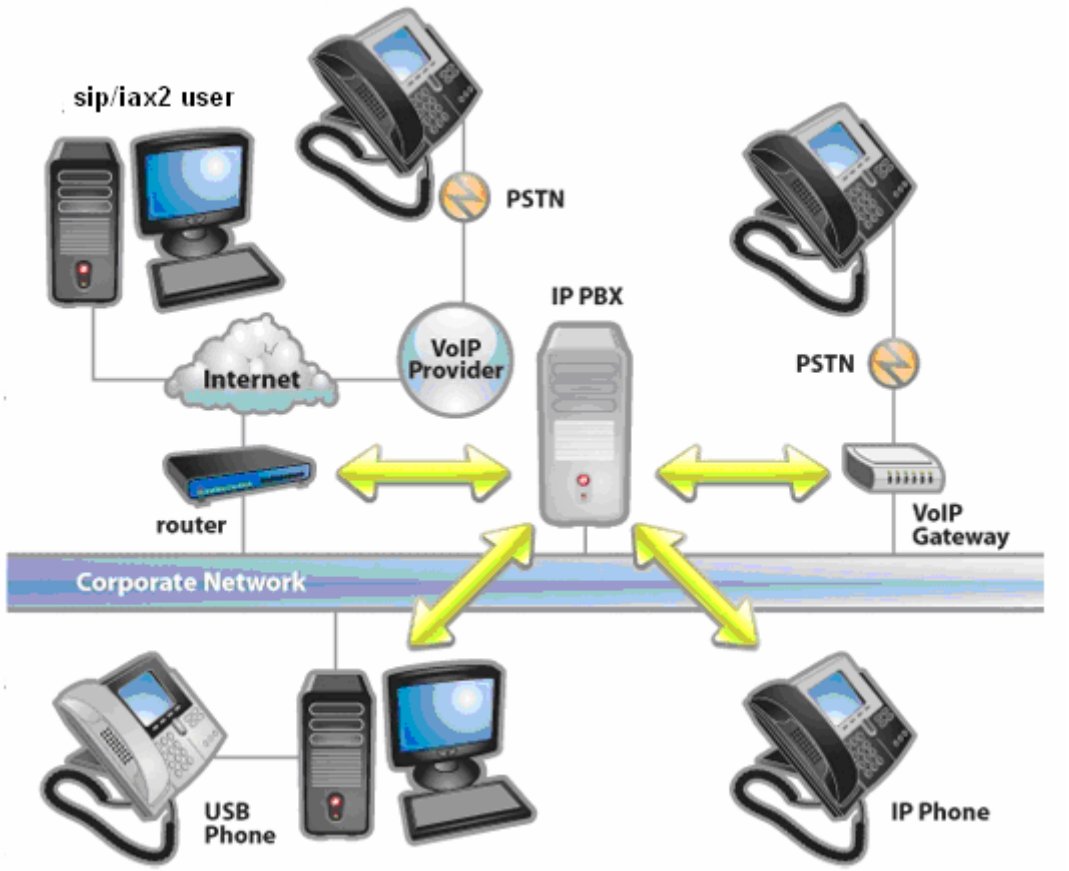

FIGURA 1.3 ENTORNO DE TRABAJO DE UNA PBX

Fuente: [PBX2008]

Algunas de las funcionalidades que ofrecen son:

- Registro detallado de llamadas
- Redirección de llamadas hacia otros números para ser atendidos
- Manejo de troncales hacia la proveedores de voip.
- Sistemas de facturación de llamadas

1.5.2 Proveedor de voip (carrier)

Los proveedores de voip, también conocidos como carriers de voip, son organizaciones que ofrecen terminaciones de llamadas de voip de cualquier usuario, es decir nos permite enviar nuestras llamadas hacia ellas y luego estas organizaciones se encargan de redireccionarlas hacia cualquier destino del mundo desde la RTC o PSTN e Internet.

Generalmente estas organizaciones hacen contratos con usuarios que demandan gran flujo de comunicaciones, debido a que es más barato contratar por cantidad, es por ello que es una muy buena opción en lo que a costos se refiere.

Las terminaciones de llamadas mas conocidas en el mercado están basadas en los protocolos SIP, IAX2 y algunas empresas ofrecen H323.

1.5.3 Sistemas Unificados de Mensajes (UMS)

Estos sistemas unifican servicios de mensajes como fax, correos de voz y correos electrónicos, ofreciendo las funcionalidades de administrar esta información convenientemente desde cualquier dispositivo tales como un teléfono celular o una computadora. [UCM2008]

1.5.4 Sistemas de Comunicación Unificados (UCS)

Estos sistemas ofrecen los mismos servicios que los UMS's pero además le agrega la posibilidad de comunicación en tiempo real, tales como mensajería instantánea y llamadas telefónicas. Para ello estos sistemas son capaces de interconectar usuarios que estén en cualquier ambiente (teléfono o computadora) bajo cierto tipo de red existente. De esta manera se hace la comunicación más dinámica y centralizada. [UCM2008]

1.6 Plataformas de mensajería unificada en el mercado

Existen aplicaciones comerciales de empresas como Cisco o Avaya, destinados al sector empresarial, que dan solución a este mismo problema (integración de la Comunicación en la Información de la empresa). Sin embargo, el costo es demasiado alto para que una empresa pyme pueda pagarlo ya que aparte del costo de la implementación inicial las actualizaciones que se desarrollen y necesiten en el futuro también tienen un costo elevado. Al ser la plataforma de mensajería unificada de esta tesis de libre distribución, se da a las empresas la posibilidad de comunicarse de forma segura, rápida, barata y eficiente, con una gran cantidad de clientes y proveedores potenciales ya que se hace uso de las mejores herramientas de software libre existentes en el mercado.

También existen diversos intentos fallidos o que se quedan en meras declaraciones de intenciones. En la primera categoría estarían las aplicaciones de Empresa de Skype para empresa, que se limitan a ser los mismos productos que para persona física, por ejemplo, de poco le sirve al empresario que le permitan chatear hasta con 100 personas en simultáneo. No obstante, dos de las herramientas que incluye el paquete de empresa, si resultan interesantes: el "Skype in" (tener un número local para recibir llamadas en otro país y redirigirlas al tuyo) y el desvío a teléfono móvil (para poder ausentarse de la oficina) [SKY2007]. En resumen: Skype no ha diseñado una oferta atractiva para las empresas, adaptada a sus auténticas necesidades de planificación y seguimiento.

Otro producto que ha sido lanzado recientemente es el "Unified Communication (UC) Integrated Branch" de la alianza Nortel-Microsoft llamada "Innovative Communication

Alliance“ (ICA) [MIN2007] que es un único dispositivo que facilitará la integración de las comunicaciones empresariales tales como voip, mensajería instantánea, etc., a un precio asequible, con una alta calidad y facilidad de instalación. Esto si es más atractivo para las necesidades de una empresa, lo cual compite directamente con el trabajo propuesto en la tesis, pero el enfoque técnico y comercial es distinto ya que el objetivo de esta plataforma es de llegar a las pymes y a su vez puedan desarrollarse mas funcionalidades adicionales y beneficiarse de ellas sin la necesidad de pagar un alto costo por ello manteniendo la calidad y confiabilidad del servicio.

Capítulo 2

Análisis y arquitectura de la solución

En este capítulo se especifican de forma conceptual las tecnologías a elegir para la implementación de la plataforma, teniendo en cuenta su escalabilidad y compatibilidad para que éstas ínter operen entre sí.

2.1 Análisis de la solución

Como toda implementación de un servicio surge para cubrir la necesidad de alguna insatisfacción de parte del usuario, en este caso como se vio en el primer capítulo algunas soluciones propietarias no contemplan verdaderamente la necesidad de las aplicaciones Business to Business (B2B), Esta es la razón de ser de la plataforma de mensajería unificada de esta tesis, la cual es implementar una solución orientada directamente a las necesidades empresariales haciendo uso de las herramientas mas usadas actualmente en comunicaciones tales como voip, chat o mensajería instantánea y llamadas hacia la PSTN

2.2 Alcances y Limitaciones del Proyecto

- El proyecto consistirá, por el lado del cliente, en un programa cliente implementado en el lenguaje java el cual podrá accederse desde cualquier explorador con soporte para aplicaciones en java o más específicamente soporte para applets. Y por el lado del servidor, se recibirán los reportes de los eventos generados desde los clientes para su correspondiente registro para su posterior manipulación.

- Dicho cliente integrará los servicios de comunicaciones de mensajería instantánea (chat), voip y las llamadas telefónicas hacia la PSTN.

- La mensajería instantánea constará de iconos divididos por categorías que dependerán del tema en discusión con el fin de facilitar las comunicaciones entre usuarios participantes cuyos idiomas no necesariamente sean similares.

- El servicio de voip será accesible desde cualquier cliente del sistema, siendo los parámetros como la calidad y el retardo de la voz dependientes directamente del entorno y características de la red en el que el cliente sea ejecutado.

- Las llamadas telefónicas hacia la PSTN serán efectuadas desde el mismo cliente con una interfaz adecuada para el discado del número a llamar, el resto de funcionalidades son transparentes al usuario.

- Además de integrar y ofrecer estos servicios, este sistema registrará los eventos ocurridos por el uso de estos servicios en una agenda inteligente cuya información podrá utilizarse para diversos fines, como es el control de las actividades del usuario, reporte de datos estadísticos, etc. .

- El objetivo es darle al usuario una interfaz amigable que facilite el acceso a todos los servicios de manera transparente además de que este sistema será parte de una plataforma con más herramientas orientado a aplicaciones B2B.

2.3 Propuesta de la Arquitectura de la Solución

En el primer capítulo se menciono acerca de los diversos protocolos, sistemas, tecnologías involucradas para el desarrollo de la tesis, ahora se describirán la interacción de estas tecnologías para la implementación de la plataforma.

El diseño de esta solución se divide en la parte de los servidores y en la interfaz que ven los clientes. Los servidores que se requieren son una base de datos, una centralita telefónica, un servidor web y un servidor de mensajería. Para la parte de la interfaz del cliente se ha determinado que permitirá al usuario final acceder al servicio desde exploradores web y además sea multiplataforma.

Los elementos de esta arquitectura consisten en los siguientes:

- Servidor de aplicaciones.
- Servidor de mensajería.
- Servidor de voz (PBX).
- Servidor de base de datos y de directorios (LDAP).
- Aplicación B2B

El servidor de aplicaciones, que estará montado sobre un servidor web Tomcat, recibirá los reportes de eventos del servicio de mensajería para almacenarlos posteriormente en una base de datos, y la PBX, además de conmutar las llamadas, también se encargara del registro de eventos correspondiente a los servicios de voip y llamadas hacia la PSTN ya que esta más especializada en este tipo de tareas.

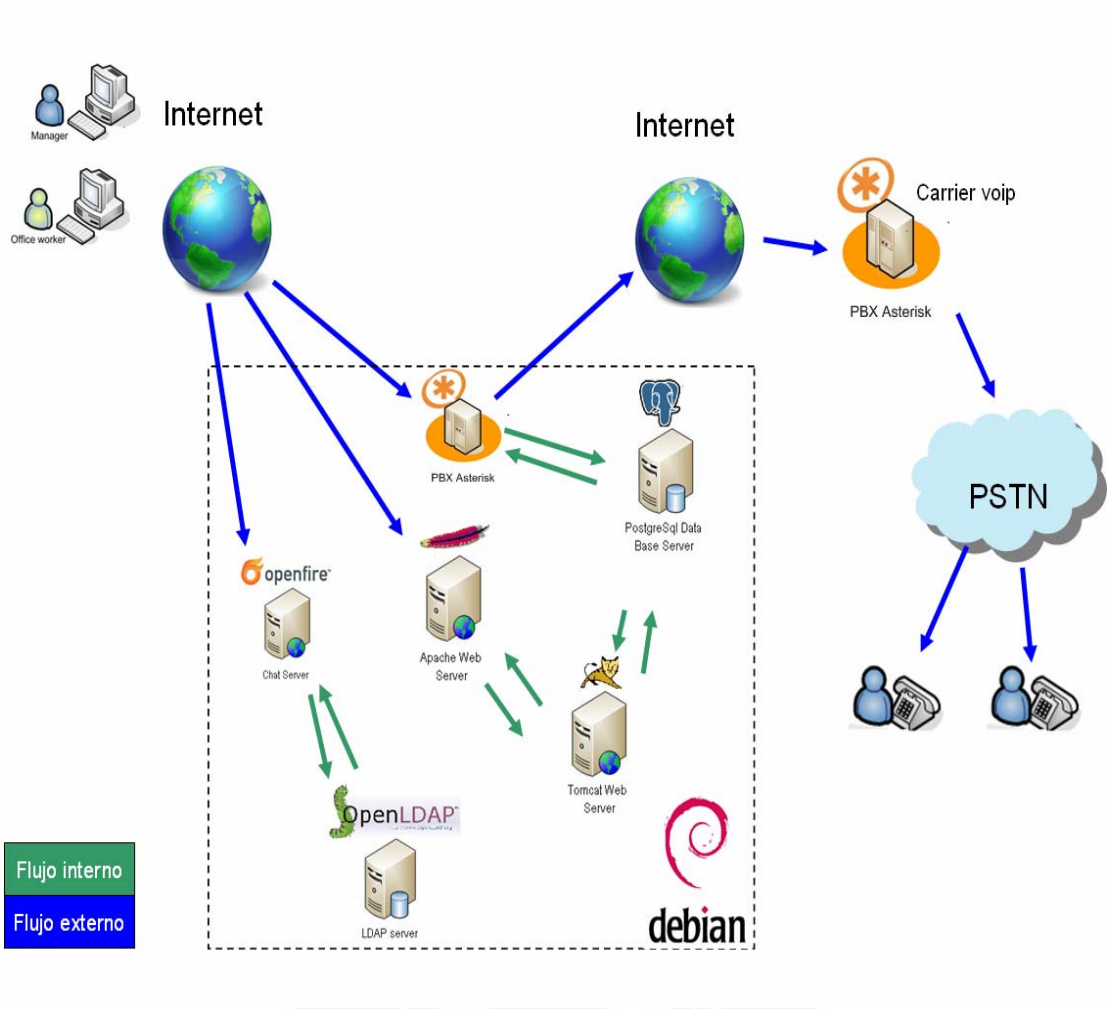

FIGURA 2.1 ARQUITECTURA PROPUESTA

En la FIGURA 2.2 se observa que las peticiones de Internet no los recibe directamente el servidor web Tomcat, sino que son recibidos por el servidor apache, esto facilita la operación del sistema en entornos donde los puertos están muy restringidos.

La aplicación B2B no es parte de la plataforma, sino más bien es un componente externo al cual se deberá integrar. Dicha plataforma será una herramienta de comunicaciones fundamental para la aplicación, que permitirá que los usuarios se comuniquen entre si a través de Internet y la PSTN (ver FIGURA 2.2).

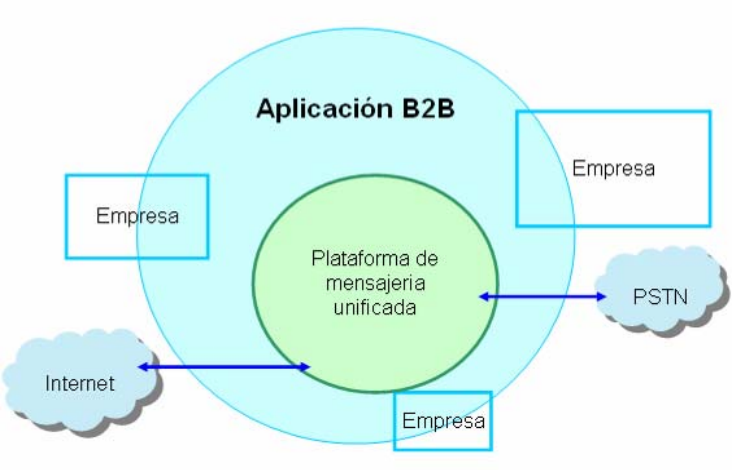

FIGURA 2.2 APLICACIÓN B2B Y LA PLATAFORMA DE MENSAJERÍA UNIFICADA

2.4 Tecnologías de la Solución

Todos los servicios estarán montados sobre la distribución Debian Etch debido a la facilidad que ofrece para la instalación de nuevas funcionalidades y la estabilidad de sus paquetes, esto favorece la operación de los servicios siendo el más crítico la PBX Asterisk. Al mencionar estas características no se intenta afirmar que Debian es la mejor distribución para los servidores que se van a utilizar, sino mas bien cuestión de comodidad y familiaridad con el entorno de desarrollo, por parte de quien implementará el sistema. [CUA2006]

2.4.1 PBX

El PBX a usar es Asterisk 1.4.9 el cual estará corriendo bajo la distribución debian Etch. Asterisk es un software de código abierto que fue creado por Mark Spencer en 1999. [MAR2008]

Este software funciona como una centra telefónica y actualmente maneja muchas funcionalidades y protocolos (entre ellos el IAX2), los cuales le permiten integrarse con otras aplicaciones ya existentes, trabajar con diferentes lenguajes de programación, inter operar entre protocolos de voip distintos, interactuar con hardware de comunicaciones, entre otros.

Una de las funcionalidades de Asterisk es que permite el registro detallado de llamadas en una base de datos como PostgreSQl, lo cual ayudará al registro de

eventos tanto de los servicios de voip y de las llamadas a teléfonos convencionales.

Otras de sus funcionalidades que se usarán en esta tesis es la posibilidad de conectarse con otros servidores asterisk, los cuales pueden ser proveedores de voip (carriers) que nos brinden el servicio de "terminación de llamadas", es decir que cuando un usuario del sistema desee hacer una llamada hacia un teléfono o celular convencional dicha llamada redirigida hacia el servidor del proveedor, quien a su vez se encargará de que esta llamada llegue a su destino pasando por la PSTN.

2.4.2 Servidor de Mensajería Instantánea

El servidor de mensajería será el Openfire 3.3 el cual estará corriendo bajo la distribución debian Etch. Openfire es un software, creado por la comunidad de desarrolladores Igniterealtime, que permite a los usuarios enviar mensajes en tiempo real usando el protocolo jabber, así como también permite que usuarios registrados en distintos servidores jabber interactúen entre si. Se usará la versión libre que es suficiente para los servicios a brindar, salvo que esta versión no permite el registro en base de datos de los eventos generados, pero si permite trabajar con el servidor de directorios Open LDAP para la autenticación de usuarios. [IGN2008]

2.4.3 Servidor de base de datos

El servidor de base de datos constará de un servidor de directorios llamado Open LDAP [OPL2008] y un servidor específicamente para almacenar datos el cual será el PostgreSql 8.1 [POS2008] ambos estarán corriendo bajo la distribución debian Etch.

2.4.4 Lenguaje de programación

Para este sistema se empleó el lenguaje de programación java creado por Sun Microsystem, el cual se caracteriza principalmente por su portabilidad, ya permite trabajar sobre sistemas operativos como Linux, windows y Mac. Algunas de sus características más importantes son [TUJ2007]:

- Es orientado a objetos: java aplica los tres modelos de la programación orientada a objetos: polimorfismo, herencia y encapsulación, donde la unidad de programación es la “clase” que puede ser definida por el propio usuario con atributos y métodos que le permitan programar de manera mas simple y estructurada. Esto conlleva a que las “clases” que generemos pueden utilizarla otros programadores [HAR2004].

- Es multitareas : Java permite la ejecución de mas de una tarea a la vez gracias a su función de Multithread, que consiste en procesos independientes y pequeños más livianos que el proceso principal que los invoca, por lo que su ejecución es más rápida y permite un comportamiento más interactivo en aplicaciones de tiempo real.

- Es robusto: Esto es muy importante ya que java no solo permite la detección de errores de manera detallada en tiempo de compilación, sino que además permite el manejo de excepciones, comprobación del uso de memoria (así se evita la preocupación del desbordamiento de memoria), de esta manera se reduce al mínimo las posibilidades de error en la implementación de una aplicación.

2.4.5 GUI

La interfaz del usuario consistirá en un programa en java que pueda invocarse en una página html, dichos programas reciben el nombre de applets. Para el diseño de esta interfaz se usarán el conjunto de librerías de la *Java Foundation Classes* (JFC), dicha biblioteca, llamada Swing, permite realizar aplicaciones gráficas dotándolas de elementos interactivos tales como botones, cajas de texto, menús desplegables, etc, los cuales serán necesarios para diseñar e integrar las interfaces de los tres servicios que ofrecerá este proyecto, chat voip y llamadas a teléfonos convencionales.

2.4.6 Servidor de aplicaciones

Este servidor de aplicaciones cumplirá las funciones de recibir los eventos de chat de la interfaz del usuario para su respectivo registro en base de datos, además es

desde donde los usuarios podrán descargar la aplicación. Esto se debe a que la versión libre del servidor de mensajería que usaremos no cuenta con un registro de eventos a una base de datos.

Los servidores de aplicaciones elegidos fueron el Apache 2 y el Tomcat 5.5. Específicamente el Apache 2 recibirá todas las peticiones que se reciban del exterior por parte de los usuarios, y delegará estas peticiones al servidor Tomcat en tareas que requieran conexión a base de datos para los registros de eventos entre otros que se puedan implementar a futuro. Esta forma de atender las peticiones son muy convenientes en entornos con puertos muy restringidos ya que el Apache 2 escucha por el puerto 80 que es por donde siempre se navega por internet así que es poco probable encontrar una computadora con acceso a internet que tenga filtrado el puerto 80.

2.4.7 Aplicación B2B

La aplicación B2B depende de dos conceptos importantes: Comercio electrónico y los e-MarketPlaces.

2.4.7.1 Comercio electrónico

El comercio electrónico en sentido restringido es toda aquella operación que se realiza utilizando medios electrónicos con fines lucrativos entre ofertantes y demandantes. Las relaciones comerciales entre demandantes y ofertantes, sean estos consumidores o empresas, dependerá de quien vende a quien, y se clasifican en cuatro grupos: empresa a consumidor (B2C), empresa a empresa (B2B), consumidor a consumidor (C2C) y consumidor a empresa (C2B) [PRO2006].

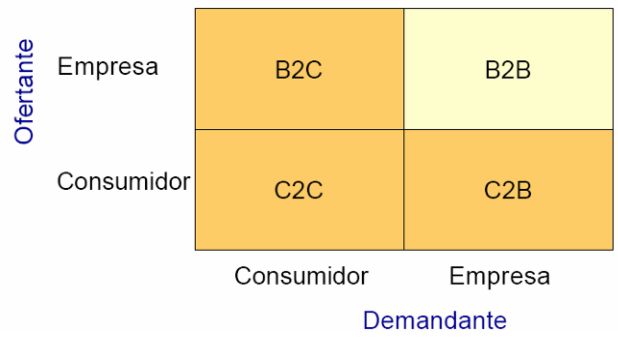

FIGURA 2.3 MODELO DE RELACIONES E-BUSINESS

2.4.7.2 Los e-MarketPlaces

Los e-MarketPlaces son plataformas web con funcionalidades comerciales que brindan una serie de herramientas basadas en tecnologías de información orientadas a facilitar el establecimiento de relaciones comerciales entre varias empresas, consumidores, proveedores, etc., en un solo lugar, rompiendo barreras geográficas, tecnológicas y hasta incluso idiomáticas [PRO2006].

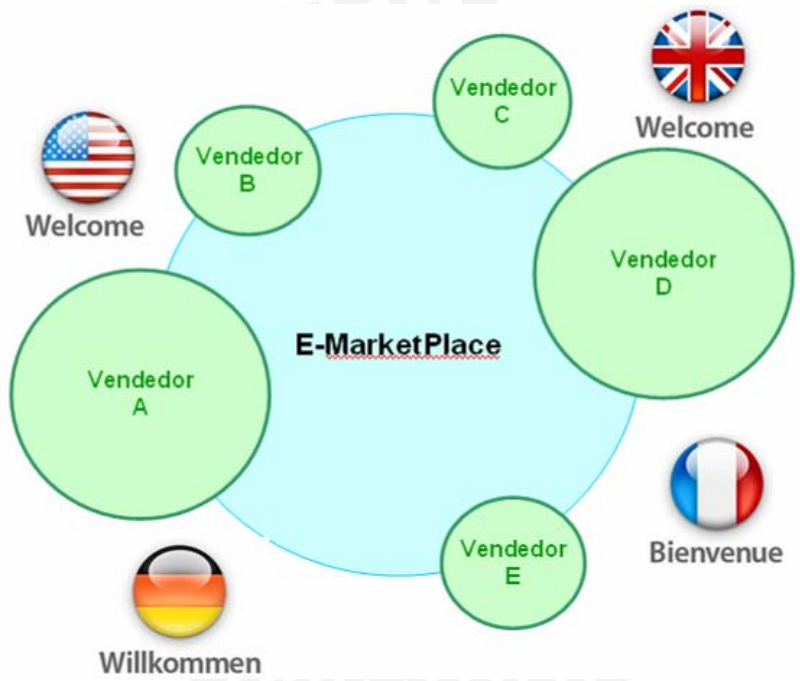

FIGURA 2.4 FUNCIONALIDAD COMERCIAL DEL E-MARKETPLACE

2.4.7.3 Definición de la aplicación B2B

La aplicación B2B es un e-MarketPlace donde los ofertantes y compradores son empresas, y por ello también se dice que es un e-MarketPlace B2B. Esta aplicación ofrece muchos beneficios entre los cuales tenemos:

Para los ofertantes:

- Aumento de la cartera de clientes fijos y potenciales nacionales e internacionales
- Reducción considerable de los tiempos de búsqueda de clientes.

- Facilidad para contactar con los clientes potenciales.
- Reducción de costos de transacción.

Para los compradores:

- Reducción de costos de transacción y carga administrativa.
- Reducción considerable del tiempo de búsqueda de diferentes ofertas.
- Aumento en las opciones de compra debido a la variedad de ofertantes nacionales e internacionales.

Finalmente la plataforma de mensajería unificada dará las facilidades de comunicaciones que la aplicación B2B actual necesita, que es brindar las herramientas de comunicaciones imprescindibles para facilitar las relaciones entre empresas dentro de estos mercados, desde un solo lugar.

Capítulo 3

Diseño de la solución

3.1 Descripción del servicio

Al servicio tendrán acceso todos los usuarios que se encuentren registrados en la aplicación B2B a la cual se encontrará integrada la plataforma es decir que un usuario que desee usar las funcionalidades de la plataforma deberá haberse registrado previamente en la aplicación. El servicio se divide en las siguientes partes funcionales:

3.1.1 Registro

A través de la aplicación B2B, los usuarios llenarán un formulario perteneciente a dicha aplicación, durante este registro al usuario se le genera dos cuentas, una que servirá para el servicio de Chat y otra para el servicio de VoIP y de llamadas a la PSTN, ambas cuentas tendrán el mismo identificador, así el usuario tendrá la sensación de usar una sola cuenta para los tres servicios.

Una vez completado el registro, el usuario puede acceder a la aplicación B2B y al hacerlo se le mostrará una serie de herramientas entre ellas el acceso a la

plataforma, luego el usuario puede acceder a las funcionalidades de dicha plataforma sin la necesidad de volver a registrarse otra vez, ya que esto se realiza de manera automática.

3.1.2 Comunicaciones

El usuario puede seleccionar la plataforma y proceder a su uso, para ello se levantará una interfaz de bienvenida mientras se cargan automáticamente los componentes necesarios y los registros a los servicios de mensajería y de voz. Después de un breve momento la plataforma estará completamente cargada y lista para la jornada de trabajo por parte del usuario. Los tres servicios están listos para ser usados en cualquier momento donde cada evento generado será registrado en una base de datos perteneciente a la aplicación B2B.

3.1.3 Agenda

Los usuarios registrados en la aplicación tendrán la posibilidad de visualizar los eventos dentro de una agenda inteligente, la cual muestra los eventos generados al detalle clasificados por los tres tipos de servicios antes mencionados.

3.2 Diagramas de flujo de la solución.

3.2.1 Chat.

El Chat es la herramienta básica de la plataforma, este es el módulo que carga los contactos de un usuario específico junto con información adicional de los mismos. En la figura 3.1 observamos el diagrama de flujo de esta herramienta, que explica las funcionalidades básicas de esta herramienta: en donde primero es necesario tener contactos agregados para iniciar una sesión de chat con uno de ellos. Finalizada la sesión de chat y si existió transferencia de datos se procede a registrar el evento generado.

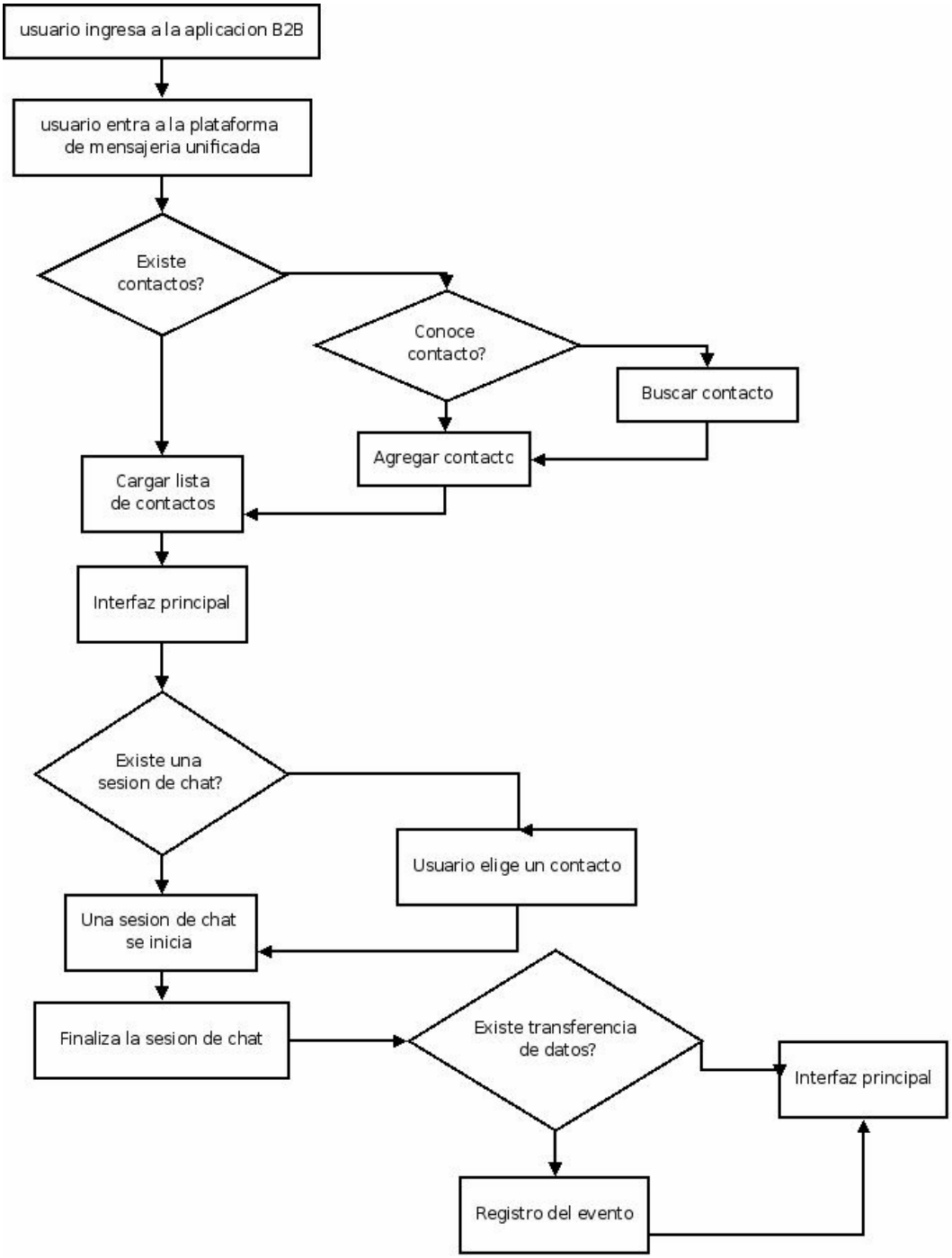

FIGURA 3.1 DIAGRAMA DE FLUJO DEL SERVICIO DE CHAT

3.2.2 VoIP.

En el diagrama de flujo (ver Figura 3.2) se explican las funciones básicas de esta herramienta: primero de la lista de contactos disponibles se debe elegir uno de ellos para llamar, luego emerge una interfaz mediante la cual se realizara la llamada, que una vez terminada, se procede a registrar el evento.

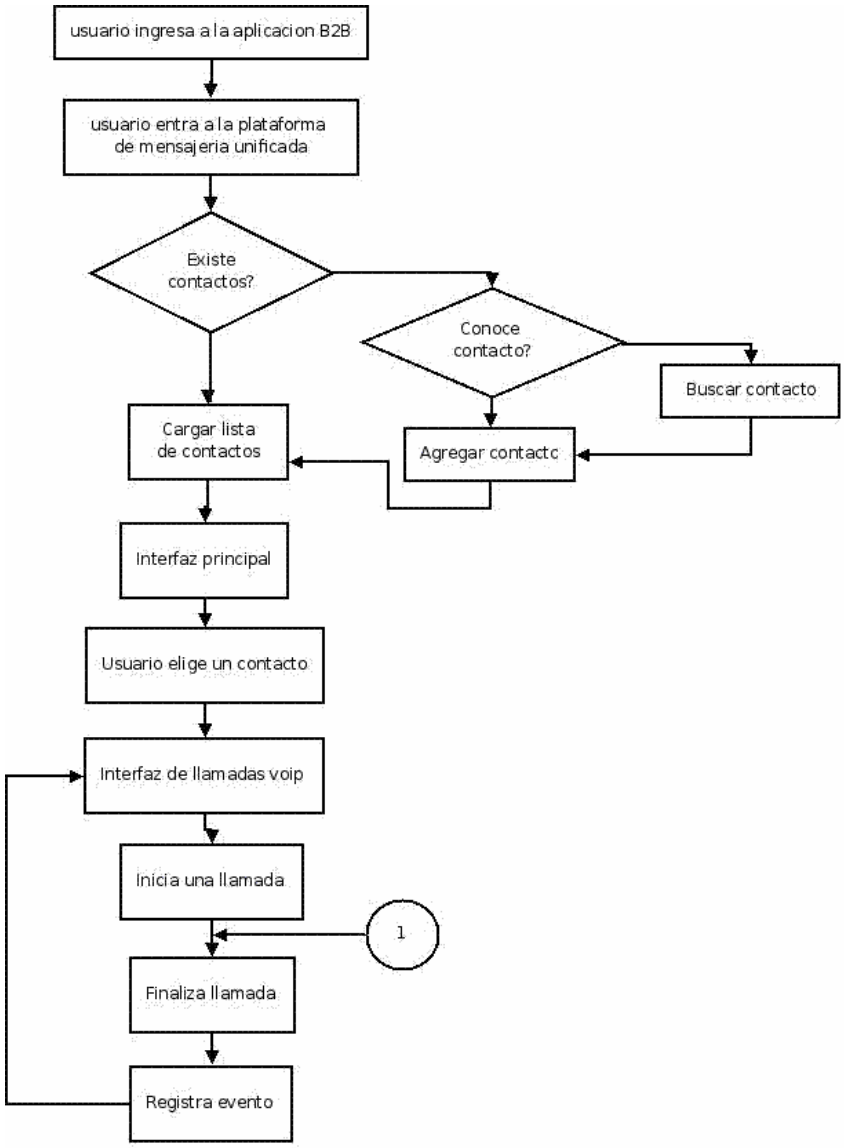

FIGURA 3.2 DIAGRAMA DE FLUJO DEL SERVICIO DE VOIP

3.2.3 Llamadas hacia la PSTN.

Según el diagrama de flujo mostrado en la figura 3.3 este módulo maneja las llamadas al igual que el módulo de VoIP, pero adicionalmente es necesario que el usuario ingrese un número en formato internacional, el cual debe ser validado para que no se ingresen caracteres no numéricos, así se minimiza el riesgo de error en el ingreso del número.

Luego se valida que el número ingresado sea de un mínimo tamaño, en caso contrario se solicita que ingrese el número correctamente. Además el usuario

dispone de un combo desplegable donde se le muestra los códigos y banderas de los países.

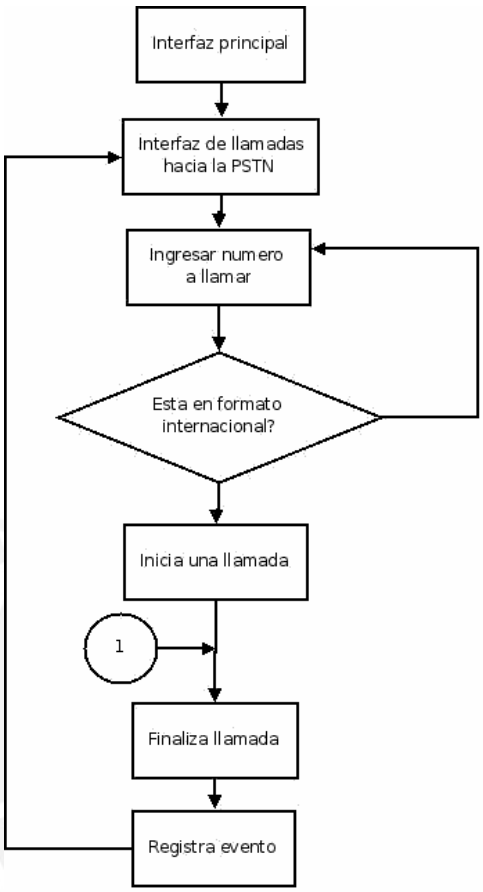

FIGURA 3.3 DIAGRAMA DE FLUJO DEL SERVICIO DE LLAMADAS HACIA LA PSTN.

3.2.4 Registro de eventos.

Para los eventos se buscó la manera de aprovechar alguna característica de los servidores elegidos que nos permita efectuar el registro de eventos, detallando la hora de inicio, hora fin, la duración, el que inicio el evento, el que lo recibió, y el tipo. Específicamente los eventos se registran de dos formas:

Registro mediante el servidor de aplicaciones (1).

Este registro se da para el servicio de Chat mediante el servidor de aplicaciones, ya que el servidor de mensajería no registra eventos (esto se verá

más detalladamente en la parte de implementación de la tesis). El servidor de aplicaciones recibe los eventos del programa cliente y luego de clasificarlos los inserta en una base de datos.

Registro mediante la PBX (2).

El servidor PBX si puede registrar eventos en una base de datos, dichos eventos son conocidos como registro detallado de llamadas o CDR. Por tanto todo el flujo de llamadas que pase por este servidor será registrado en una base de datos.

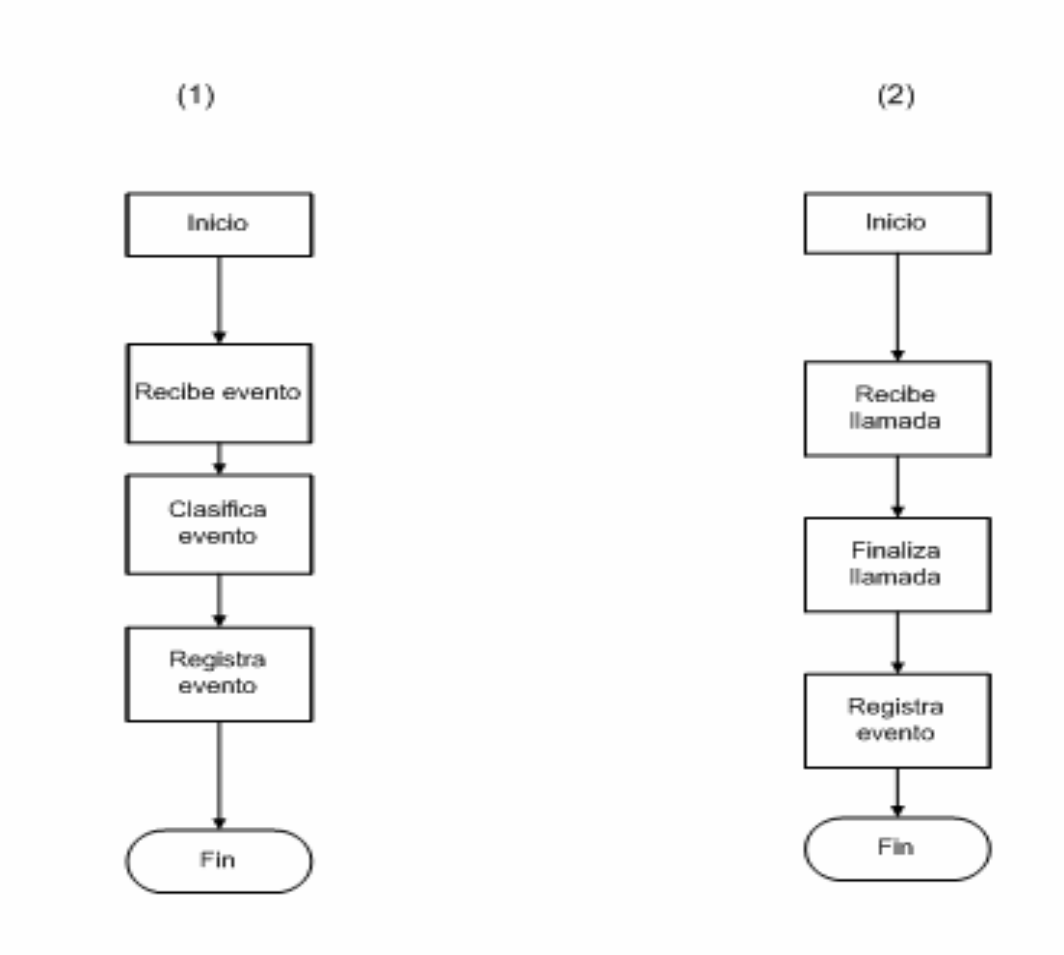

FIGURA 3.4 DIAGRAMA DE FLUJO DEL REGISTRO DE EVENTOS

3.3 Estándares del sistema

A pesar de no ser objetivo, se trató en lo posible seguir ciertos estándares para facilitar la comprensión del código desarrollado y así facilitar futuras implementaciones.

Estándares de diseño de interfaces: todas las interfaces dispondrán de botoneras cuadradas con el mismo tamaño de letras y un fondo blanco.

Estándares de programación: para la creación de las clases, métodos y variables se usaron nombres acordes con las funcionalidades y valores que representan, mas no los tipos.

Estándares usados en base de datos: Para este caso en la creación de la base de datos , para los servicios de voz y chat, no se uso ningún estándar puesto que la estructura de base de datos tiene que ser idéntica a la que proponen los creadores de dichos servidores para su correcto funcionamiento consistencia de datos.

3.4 Diseño de la plataforma de mensajería unificada

Al estar conformada la plataforma por tres servicios distintos, esta se divide en tres módulos los cuales son el módulo de chat, VoIP y de las llamadas hacia la PSTN.

3.4.1 Módulo de Chat

Para este módulo tenemos un servidor de mensajería, y además disponemos de un servidor de aplicaciones en el cual se encuentran alojados en directorios los emoticonos categorizados con los que trabajará el cliente. En este módulo se diferencian dos tareas: Registro y descarga de datos.

3.4.1.1 Registro

Como se mencionó anteriormente los datos como el usuario y la contraseña son provistos por la aplicación B2B en modo de parámetros hacia la plataforma, por tanto el usuario utilizará estos datos para registrarse al servidor de mensajería, el cual hace la correspondiente validación en el servidor de autenticación, en ningún caso el usuario deberá ingresar estos datos, salvo exista algún error en la transferencias de los mismos desde la aplicación B2B hacia la plataforma.

3.4.1.2 Descarga de datos

Antes de la validación, los datos tales como los plugins, librerías, y emoticonos para trabajar, y demás logotipos son descargados desde el servidor de aplicaciones por la plataforma. Una vez descargados estos elementos, se procede al registro del usuario, para luego cargar los siguientes datos que son, sus datos personales que desee publicar, así como los contactos y grupos que posee además de los datos que estos contactos hayan publicado.

3.4.2 Módulo de VoIP

Este módulo depende enteramente del servidor PBX y de la base de datos, así que en este caso el servidor PBX estará escuchando las peticiones de conexión de la plataforma y además la base de datos estará escuchando las peticiones provenientes del servidor PBX cuando este requiera registran un CDR. Podemos dividir este módulo en tres tareas: registro, inicio de llamadas y CDR.

3.4.2.1 Registro

Inicialmente cuando apenas la plataforma es activada se inicia el registro del usuario en el servidor PBX, para lo cual dicho servidor hará la consulta a la base de datos si dicho usuario existe, si el procedimiento anterior es satisfactorio, entonces el usuario ya estará registrado en el servidor PBX y habilitado para recibir y realizar llamadas hacia otros usuarios de la plataforma.

3.4.2.2 Iniciar llamadas

Una vez que el usuario ya esta reconocido por el servidor PBX, puede proceder a realizar una llamada, para lo cual solo deberá seleccionar un usuario de la plataforma desde la interfaz principal y luego iniciar la llamada, en este punto el servidor PBX verificará que el usuario exista, si es que existe entonces verifica que esté registrado en el sistema, en caso de estar registrado se procede a establecer un canal de comunicación entre ambos clientes.

3.4.2.3 CDR

El servidor PBX debe registrar las llamadas que pasan por él en una base de datos donde existen tablas con los campos adecuados para poder alojar estos registros de llamadas.

3.4.3 Módulo de Llamadas hacia la PSTN

Al igual que el módulo de VoIP, este módulo depende del servidor PBX y de una base de datos. Pero además también depende de un elemento adicional: la conexión con el proveedor de VoIP o Carrier. El usuario podrá realizar llamadas hacia la PSTN a través de la interfaz respectiva marcando solamente el respectivo código internacional, código local más el número a llamar. Las tareas involucradas en este servicio son: inicio de llamadas y el CDR.

3.4.3.1 Iniciar llamadas

Las llamadas se inician cuando el usuario marca el número en el formato valido y activa la opción de llamar, ahora bien este número tal y como está no es enviado al servidor PBX, sino que se le adiciona un número mas para luego redirigir las llamadas hacia el Carrier. La razón es la siguiente: el Carrier requiere que los números marcados comiencen siempre con los dígitos “11”, es decir, si se llamase al destino 511995664020 y si se envía tal y como esta al Carrier, este rechazara la llamada, en cambio se sigue formato requerido el número que se le indica al proveedor deberá ser “11511995664020”, lo cual no quiere decir que el usuario debe completar este número agregándole los dígitos adicionales cada vez que desee hacer una llamada, sino que la plataforma lo hace automáticamente.

3.4.3.2 CDR

Al igual que en el caso del servicio de VoIP, aquí no se tiene en cuenta si la fuente de la llamada o el destino existen en la aplicación B2B, simplemente el servidor PBX registra todo el tráfico que fluya por el. Así que cuando

algún usuario realice una comunicación con un número telefónico, este evento quedará registrado sin ningún inconveniente.

3.4.4 Registro de eventos

Los eventos generados por la plataforma tales como las llamadas y las sesiones de Chat deben quedar registrados en la base de datos en un mismo formato. Si alguno de los servidores no realiza este registro de eventos se deberá implementar dicha funcionalidad teniendo en cuenta el único formato requerido para esta tarea.

Capítulo 4

Implementación de la solución

En este capítulo se mostrará la implementación de la plataforma teniendo en cuenta que el objetivo principal es la integración de los servicios de VoIP, llamadas a la PSTN, Chat y el registro de eventos a la aplicación B2B a través de un cliente prototipo y del servidor de aplicaciones. La documentación de las clases generadas para estos desarrollos se encuentra en los anexos 1 y 3. Los servidores que se mencionarán a continuación se encuentran instalados en una PC de las siguientes características: procesador Intel core 2 duo de 2.13GHz, 2GB de memoria RAM y 120 GB de disco duro. Dichos servidores están operando sobre el sistema operativo debian Etch.

4.1 Servidor PBX.

El servidor Asterisk estará interactuando constantemente con una base de datos PostgresSql 8.1 para lo cual del lado del Asterisk se necesitará una configuración de unos archivos en texto plano y por el lado del PostgresSql se deben tener las tablas con los campos adecuados según se detallará más adelante (ver sección 4.4 Servidor de base de datos).

4.1.1 Asterisk – CDR.

Esta tarea consiste en hacer que el servidor Asterisk inserte el registro detallado de llamadas en una base de datos, que para propósitos de la tesis, es una base de datos PostgreSql.

```
; Sample Asterisk config file for CDR logging to PostgresSQL

;/etc/asterisk/cdr_pgsql.conf
[global]
hostname=82.194.85.57
port=5432
dbname=bmundidb01
password=XXXXXXXXXX
user=bmundiusr01
table=cdr_conf          ;SQL table where CDRs will be inserted
```

FIGURA 4.1 ARCHIVO CDR_PGSQL.CONF

4.1.2 Asterisk - Real Time.

Una configuración básica de Asterisk consiste en definir los usuarios dentro de grupos o contextos y luego un plan de discado para cada uno de ellos, para lo cual se utilizan dos archivos el “iax.conf” para los usuarios y el “extensions.conf” para el plan de discado. Estos archivos se encuentran dentro de la carpeta “/etc/asterisk/” y están en texto plano por lo que cada vez que queramos agregar un usuario nuevo con su plan de discado tendríamos que agregarlo manualmente dentro del archivo”iax.conf”, lo cual no es manejable en comparación a que si estos usuarios con sus respectivos planes de discado los pudiéramos agregar en una base de datos y leerlos directamente desde allí. Asterisk – Real Time es funcionalidad que permite al servidor Asterisk trabajar con usuarios definidos en una base de datos, así como también los planes de discado asociados a estos usuarios para el establecimiento de llamadas entre ellos. Esto se logra configurando los siguientes archivos:

- extconfig.conf : En este archivo se direccionan o mapean los archivos “iax.conf” y “extensions.conf” con sus respectivas tablas que va a leer el Asterisk cada vez que alguien efectúe una llamada

en el contexto, el cual se le ha llamado “default” :

```
; /etc/asterisk/extconfig.conf
iaxpeers=>pgsql,bmundidb01,iax_conf
iaxusers=>pgsql,bmundidb01,iax_conf
extensions=>pgsql,bmundidb01,extensions_conf
```

FIGURA 4.2 ARCHIVO EXTCONFIG.CONF

- iax.conf : configuración para Asterisk Real-Time con el protocolo IAX2

```
;/etc/asterisk/iax.conf

rtcachefriends=yes
rtupdate=no
rtautoclear=yes
rtignoreregexpire=yes
```

FIGURA 4.3 ARCHIVO IAX.CONF

- extensions.conf : Configuración de Asterisk Real-Time para el contexto “default”

```
; /asterisk/etc/extensions.conf
[default]

switch=>realtime/default@extensions; nombre del  archivo "extensions.conf"
```

FIGURA 4.4 ARCHIVO EXTENSIONS.CONF

4.1.3 Asterisk - Proveedor de VoIP.

Para la conexión con la PSTN, el proveedor de VoIP provee a los usuarios registrados los códigos de acceso a sus servidores desde nuestro servidor de asterisk local, y en base a ello se deben modificar los archivos que se muestran en las figuras a continuación:

```
;/etc/asterisk/iax.conf
[voipjet]
type=peer
host=east.voipjet.com
username= 17488
secret= XXXXXXXXXXXXX
auth=md5
context=default
deny = all
allow = ulaw
```

FIGURA 4.5 ARCHIVO IAX.CONF 2

```
;/etc/asterisk/extensions.conf
exten => _1NXXNXXXXXX,1,SetCallerID(4153574000); Set your CallerID as a te
exten => _1NXXNXXXXXX,2,Dial,IAX2/17488@voipjet/${EXTEN} ; VoipJet.com NAI
exten => _011.,1,SetCallerID(4153574000); Set your CallerID as a ten digi
exten => _011.,2,Dial,IAX2/17488@voipjet/${EXTEN} ; VoipJet.com WORLD
exten => _011.,3,hangup()
```

FIGURA 4.6 ARCHIVO EXTENSIONS.CONF 2

4.2Servidor de Mensajería Instantánea.

El servidor de mensajería utilizado es el OpenFire 3.3, cuyos detalles de instalación y configuración con servidores Open LDAP y PostgresSql 8.1 se encuentran en [ING2007]. En este servidor no hubo la necesidad de hacer ningún cambio adicional para propósitos de la tesis, solo se siguieron los pasos en la referencia mencionada.

4.3 Servidor de Aplicaciones.

El servidor usado, es la combinación de Tomcat 5.5 con apache2, y su funcionamiento consiste en lo siguiente: el programa cliente de la plataforma envía las tramas al servidor apache con dirección "HTTP://xxx.xxxxxx.xxx/ServidorDeAplicaciones", el cual enviará las tramas, de manera interna, hacia la dirección del servidor Tomcat: "HTTP://xxx.xxxxxx.xxx:8080/ServidorDeAplicaciones". Esto se hace por el motivo de que muchas veces el puerto 8080 no siempre se encuentra disponible para los terminales usuarios, sobre todo en entornos proxy y firewals, pero el puerto 80 del apache2 siempre esta abierto puesto que es necesario para navegar por internet. Así garantizamos que por el lado del cliente los mensajes llegarán al servidor de aplicaciones con menos

probabilidad de pérdida. Los pasos para esta implementación fueron tomados de [IAT2007].

Una vez minimizado el problema del acceso de puertos, se implementó el conjunto de clases apropiado para hacer más manejable las tramas de eventos que se reciban desde el programa cliente de la plataforma. Dichas clases (ver Diccionario de clases) siguen la arquitectura MVC y se encuentran descritas en el diccionario de clases.

Finalmente, una vez recibidas estas tramas se procede con su inserción en la tabla “oficaccionesagenda” de la aplicación B2B.

4.3.1 Diccionario de clases

TABLA 4.1 CLASE SEVENTO

Clase	Atributos	Métodos	Descripción
SEventos		processRequest	Recibe y procesa las peticiones externas al servidor de aplicaciones, para su posterior respuesta
		doGet	Recibe y procesa las peticiones externas sólo del tipo GET al servidor de aplicaciones, para su posterior respuesta
		doPost	Recibe y procesa las peticiones externas sólo del tipo POST al servidor de aplicaciones, para su posterior respuesta

TABLA 4.2 CLASE BEVENTO

Clase	Atributos	Métodos	Descripción
BEventos	String fuente	getFuente	Obtiene el nombre de usuario que origino el evento
	String destino	setFuente	Establece el nombre de usuario que origino el evento
	String inicio	getDestino	Obtiene el nombre de usuario destinatario evento
	String fin	setDestino	Establece el nombre de usuario destinatario del evento
	String conectado	getInicio	Obtiene la fecha de inicio del evento
	String duración	setInicio	Establece el inicio del evento
	String tipo	getFin	Obtiene la fecha de finalización del evento
		setFin	Establece el fin del evento
		getConectado	Obtiene el estado de la conexión del usuario

			destinatario
		setConectado	Establece el estado del usuario destinatario del evento
		getDuracion	Obtiene el tiempo de duración del evento
		setDuracion	Establece el tiempo de duración del evento
		getTipo	Obtiene el tipo de evento
		Insert	Establece el tipo de evento

TABLA 4.3 CLASE DEVENTOS

Clase	Atributos	Métodos	Descripción
DEventos		insert	Registra el evento en la base de datos seleccionada, en base al objeto de la clase BEventos.

4.4 Servidor de base de datos.

La versión de base de datos que se usa es el PostgreSql 8.1, en dicho servidor se han creado las tablas de datos que requieren todos los servidores en misma base de datos. Para el servidor asterisk se crearon las tablas conforme a [WIP2007] y para el servidor Openfire conforme a [ING2007]. De la aplicación B2B sólo se hace uso de la tabla “oficaccionesagenda” (ver TABLA 4.8) para la inserción de eventos.

Para una de las tablas del servidor Asterisk se ha hecho uso de un disparador o trigger [TRI2007] que servirá para trasladar los datos de la tabla cdr_conf que aloja los CDR hacia la tabla “oficaccionesagenda”, la posee todos los eventos en un único formato. Dicho trigger se ejecuta luego de que se ha insertado un nuevo registro en el CDR, y su tarea es extraer los datos relevantes haciendo las transformaciones correspondientes para generar un nuevo registro y luego insertar este registro modificado en la tabla “oficaccionesagenda” (ver TABLA 4.4).

TABLA 4.4 TABLA OFICACCIONESAGENDA

Nombre	Tipo de dato	Descripción
logincreador	character varying	Id del usuario creador del evento
destinatario	character varying	Id del usuario destinatario del evento

asuntoaccion	character varying	Descripción del asunto
descripacionaccion	character varying	Descripción de la acción del evento
estadoaccion	character	Estado de la acción
horainicioaccion	character varying	Hora de inicio del evento
horafinacion	character varying	Hora de fin del evento
loginrealizador	character varying	Id del usuario que realiza el evento
tipoaccion	character	Tipo de acción que genero el evento
codigotipotarea	character	Código de evento deacuerdo al tipo de tarea
nrotarea	numeric	Número de tarea
nroaccion	numeric	Número de acción
fechaaccion	date	Fecha de registro del evento
duracionaccion	character varying	Duración del evento
nombreaccion	character varying	Nombre del evento deacuerdo al tipo
codigoempresausuaria	character	Código de la empresa del destinatario
idsession	character varying	Id de la sesión

4.5 Implementación de la aplicación cliente prototipo.

El cliente prototipo es un applet escrito en java y desarrollado con el IDE netbeans 5.5 utilizando el JDK 1.5, ya que es el más utilizado y estable en la actualidad. Este applet estará embebido dentro de una página escrita en lenguaje PHP 5, la cual es necesaria para el envío de parámetros desde la aplicación B2B, los cuales son la dirección del servidor de mensajería, la dirección del servidor de Voz, el usuario y la contraseña. Dicho cliente se ha separado en módulos deacuerdo a las cuatro funcionalidades descritas anteriormente: módulo de chat, módulo de VoIP, el módulo de llamadas hacia la PSTN y externamente el registro de eventos.

4.5.1 Módulo de Chat

Para la implementación de este módulo se basó en la existencia de un cliente de chat escrito en java [JET2007], dicho cliente es un applet que posee una interfaz gráfica básica la cual, gracias a que el autor provee los códigos fuentes, se puede modificar a nuestro interés. Sobre este cliente se implementó una serie de funcionalidades (entre nuevas clases y métodos para el registro de eventos y emoticonos de trabajo) que permitieron la integración con la aplicación B2B y con el módulo de llamadas de VoIP.

Este módulo es el que envía los mensajes de ocurrencia de eventos hacia el servidor de aplicaciones cuando una sesión de Chat ha finalizado a través de la clase “DataServlet.java” que hace una conexión HTTP con el Servidor de aplicaciones.

Las clases más importantes en este módulos se encuentran en el paquete “nu/jw/jeti/ui” y son

TABLA 4.5 CLASES IMPORTANTES DEL MÓDULO DE CHAT

Paquete	Clase	Descripción
nu/jw/jeti/ui	jeti.java	Este clase inicia los elementos de la interfaz principal, tales como el menú de inicio principal, el árbol de contactos conectados y desconectados así como los grupos en los que se encuentran
nu/jw/jeti/ui	chatwindow.java	Contiene los métodos para la interfaz del módulo de chat y para el manejo de las sesiones de chat, invoca a la clase necesaria para la utilización de los emoticonos, envió y recepción de mensajes y además envía el evento generado a través de la clase DataServlet.java
nu/jw/jeti/ui	SendMessage.java	Se encarga del envío de las tramas en formato XML con los campos y atributos adecuados según el protocolo XMMP para mensajería, entre estos mensajes se encuentran los emoticonos
nu/jw/jeti/ui/plugin/emoticonos	Plugin.java	Mediante esta clase se cargan los emoticonos categorizados desde el servidor de aplicaciones mediante conexión HTTP.
ext/util	DataServlet.java	Posee los métodos necesarios para el establecimiento de la conexión HTTP entre el cliente y el servidor de aplicaciones para el envío de las tramas que contienen los eventos generados

Para este módulo tenemos al servidor de mensajería Openfire que siempre esta escuchando las peticiones de conexiones seguras por el puerto 443, además disponemos de un servidor apache2 en el cual se encuentran

alojados en directorios los emoticonos categorizados con los que trabajará el cliente. Esto se describe mejor en dos tareas distintas: registro y descarga de datos.

4.5.1.1Registro

Los datos como el usuario y la contraseña son provistos por la aplicación B2B en modo de parámetros hacia la plataforma, los que servirán para registrarse al Openfire, el cual hace la correspondiente validación en el servidor Open LDAP, en ningún caso el usuario deberá ingresar estos datos, salvo exista algún error en la transferencias de los mismos desde la aplicación B2B hacia la plataforma.

4.5.1.2 Descarga de datos

Antes de la validación, los datos tales como los plugins, librerías, emoticonos de trabajo (emoticonos para trabajar), y demás logotipos son descargados desde el servidor apache2 por la plataforma, para ello hace uso del protocolo HTTP.

4.5.1.3 Registro de eventos

Cuando finaliza una sesión de Chat, se construye una trama que contiene la hora de inicio, la hora de fin, la duración, la fuente, el destino, el estado de presencia del destino y el tipo de tarea. Luego esta trama es enviada desde este módulo (applet) mediante la clase “DataServlet.java“ hacia el servidor de aplicaciones mediante una conexión entre ambos elementos [APP2002].

4.5.2 Módulo de VoIP

Al igual que el módulo de Chat, para este módulo también se usó un cliente de llamadas de VoIP bajo el protocolo IAX2 escrito en java [JIA2006]. Dicho cliente era un applet que presentaba una interfaz gráfica bastante básica, el cual consistía en una caja de texto, teclado numérico y de control.

Además hubo la necesidad de implementar nuevas clases y métodos que permitieran la integración con el módulo de Chat.

La clase más importante de este módulo se llama IAXTest.java, la cual descarga e instala la librería iaxc.dll en la máquina virtual de java de la computadora cliente, crea las interfaces para este módulo y recibe y permite realizar llamadas de un usuario a otro.

TABLA 4.6 ALGUNOS MÉTODOS DE LA CLASE IAXTEST.JAVA PARA EL MÓDULO DE VOIP

tipo	nombre	descripción
void	call	Inicia una llamada hacia un número o contacto específico
void	openbIp	Muestra la interfaz del módulo de chat
void	initIaxc	Inicia el registro del usuario en el Asterisk y descarga la librería iaxc.dll
void	startIax	Inicia el proceso IAX que recibirá y realizara las llamadas
void	stopIax	Detiene el proceso IAX

Este módulo depende enteramente del servidor Asterisk y la base de datos PostgresSql. El servidor Asterisk estará escuchando las peticiones de conexión por el puerto 4569 que es el puerto estándar para el protocolo IAX2 y además la base de datos PostgreSql estará escuchando por el puerto 5038 las peticiones provenientes del servidor Asterisk cuando se registre un CDR. La implementación de este módulo es como se detalla en las siguientes tareas a continuación:

4.5.2.1 Registro

Inicialmente cuando apenas la plataforma es activada se descargan las librerías necesarias para este cliente desde el servidor apache2, las cuales permitirán a la plataforma trabajar bajo el protocolo IAX2, luego si el procedimiento es satisfactorio entonces se procederá al registro del usuario en el servidor Asterisk, para lo cual dicho servidor hará la consulta al PostgreSql si es que dicho usuario existe, si el procedimiento anterior es satisfactorio, entonces el usuario ya estará

registrado en el servidor de voz y habilitado para recibir y realizar llamadas hacia otros usuarios de la plataforma.

4.5.2.2 Inicio de llamadas

Una vez que el usuario ya esta reconocido por el servidor Asterisk, puede proceder a realizar una llamada, para lo cual solo deberá seleccionar un usuario de la plataforma desde la interfaz principal y luego iniciar la llamada, en este punto el servidor Asterisk verificará que el usuario exista, si es que existe entonces verifica que este registrado en el sistema, en caso de estar registrado se procede a establecer un canal de comunicación entre ambos clientes bajo las especificaciones del protocolo IAX2 que utilizamos [FOR2006].

4.5.2.3 CDR(Call Detail Record)

El servidor Asterisk por defecto siempre registra las llamadas que pasan por él en un archivo de texto plano, con la opción de registrarlo también en una base de datos. Para la tesis se hizo uso de la base de datos PostgreSql donde existen tablas con los campos adecuados para poder alojar estos registros de llamadas. Esta tabla no posee ninguna llave foránea por lo que no se puede determinar si estos datos como la fuente de la llamada o el destino de la misma existen en otras tablas.
Así que por tanto las llamadas pueden tener un destino como un origen que no necesariamente coincida con datos de la aplicación B2B.

4.5.3 Módulo de Llamadas hacia la PSTN

En este módulo, al igual que el módulo de VoIP, también se usó el cliente de IAX2 basado en java [JIA2006]. Aquí fue necesario implementar clases y funciones que permitieran el uso adecuado de este servicio, como el de validar el ingreso de datos numéricos, el tamaño, ingreso de prefijos, entre otros. Este módulo depende en gran parte de la configuración del servidor asterisk ya que deacuerdo al número que llamemos el carrier direccionará la llamada hacia la PSTN.

Este módulo a nivel de programación utiliza la misma clase que el módulo de VoIP, así que su principal clase es IAXTest.java, sólo se diferencia en la interfaz que adicionalmente tendrá una botonera numérica, una lista desplegable para tarjetas prepago en futuras aplicaciones y otro logotipo

TABLA 4.7 ALGUNOS MÉTODOS DE LA CLASE IAXTEST.JAVA PARA EL MÓDULO DE LLAMADAS HACIA LA PSTN

tipo	nombre	descripción
void	call	Inicia una llamada hacia un número o contacto específico
void	openbIp	Muestra la interfaz del módulo de chat
void	initIaxc	Inicia el registro del usuario en el Asterisk y descarga la librería iaxc.dll para windows e iaxc.so para linux.
void	startIax	Inicia el proceso IAX que recibirá y realizara las llamadas
void	stopIax	Detiene el proceso IAX
void	createButtons	Crea los botones numéricos para el marcado de números telefónicos
void	openbPhone	Muestra la interfaz del módulo de llamadas hacia la PSTN
boolean	validarnúmero	Este método verifica que el número ingresado este en el formato correcto, en caso contrario no se podrá efectuar la llamada

Este módulo depende del servidor Asterisk y del PostgreSql. Pero además también depende de un elemento adicional: la conexión con el proveedor de VoIP o Carrier. El usuario podrá realizar llamadas hacia la PSTN a través de la interfaz respectiva marcando solamente el respectivo código internacional, código local más el número a llamar. Las tareas involucradas en este servicio son: inicio de llamadas y el CDR.

4.5.3.1 Inicio de llamadas

Como se mencionó en la parte del diseño, si al número llamado es necesario añadirle un prefijo esta tarea debe hacerse de forma transparente al usuario. El carrier con el que se ha hecho las pruebas

exige que los número con destino a Estados unidos y Canadá se le deben agregar el prefijo “1”, para el resto del mundo se le debe agregar “11”. Esto lo hace el cliente de manera automática: detectando el país de destino y en base a ello agregar los dígitos correspondientes.

4.5.3.2 CDR(Call Detail Record)

Al igual que en el caso del servicio de VoIP, aquí no se tiene en cuenta si la fuente de la llamada o el destino existen en la aplicación B2B, simplemente el Asterisk registra todo el tráfico que fluya por el. Así que cuando algún usuario realice una comunicación con un número telefónico, este evento quedará registrado sin ningún inconveniente.

4.5.4 Registro de eventos

Los eventos generados por la plataforma tales como las llamadas y las sesiones de Chat deben quedar registrados en la base de datos en un mismo formato. En esta tarea es importante recordar que el servidor de mensajería Openfire no registra eventos, y además el servidor Asterisk registra los eventos pero con su propio formato. Por tanto la tarea es hacer que de alguna manera se registren eventos de ambos servicios en una única tabla de manera homogénea así que la solución se abordará por cada servidor.

4.5.4.1 Solución servidor Asterisk

Los registros que se inserten en la base de datos del servidor Asterisk deberán ser transformados a un formato adecuado para luego ser insertado en una tabla que contendrá todos los eventos de la plataforma. Para esta tarea se hace uso de un disparador o trigger, que lanzará una función después de alguna inserción de datos en la tabla CDR del Asterisk. Esta función analizará las tramas que se inserten para poder extraer y modificar las que sean necesarias formando una nueva trama compatible con nuestra única tabla de registro de eventos. Así tendremos ya registrados en una única tabla

los eventos que correspondan a las llamadas de VoIP y las que tengan como destino la PSTN.

Para resumir la idea, lo que el Asterisk inserte el disparador lo moverá a la tabla única de registro de eventos.

4.5.4.2 Solución servidor OpenFire

Esta tarea se deberá hacer por parte de la plataforma de la siguiente manera : el programa cliente cada vez que se inicie una sesión de Chat enviará una trama de este evento (siempre y cuando hay a existido transferencia de datos) indicando datos importantes como el usuario que inició la sesión, el destinatario, la hora de inicio y fin, si el destinatario estaba desconectado o no, y el tipo de tarea que en este caso es una sesión de Chat , para ello se hace uso de un servidor de aplicaciones, que estará montado sobre el servidor web Tomcat, cuya finalidad es escuchar los mensajes que envíe el programa cliente para luego registrarlo en la tabla única de registro de eventos con el formato adecuado.

Dicho servidor de aplicaciones se trata de una aplicación web hecha sobre java que esta bajo la arquitectura modelo-vista-controlador (MVC), aunque en este caso lo único que se necesita es la parte de modelo-controlador, donde el modelo estará conformado por las clases java llamadas Data Access Object (DAO) que nos servirán para insertar información en la base de datos y el controlador estará conformado por servlets que son clases en java que responden y atienden peticiones de información desde las aplicaciones clientes.

Finalmente el programa cliente de la plataforma enviará mensajes al servidor de aplicaciones que insertará las tramas de los eventos en la única tabla de registro de eventos.

4.5.5 Integración de los módulos

La integración de los módulos antes descritos fue facilitada gracias a que todos ellos se encontraban escritos en el mismo lenguaje de programación, pero se vio dificultada por el complejo código fuente que cada uno poseía, entonces tomo tiempo entender los módulos por separado para luego integrarlos en un solo programa, con un mismo diseño, con el fin de que se vea como un cliente unificado y evitar problemas de incompatibilidad entre sí.

La clase principal en la integración fue “Jeti.java” (ver TABLA 4.8) ya que aquí se inicia la carga del applet y la de todos los demás módulos.

TABLA 4.8 CLASE IMPORTANTE PARA LA INTEGRACIÓN DE LOS MÓDULOS

Paquete	Clase	Descripción
un/jw/jeti/applet	Jeti.java	Esta es la clase principal, y la primera en ser llamada, donde se inician todos los módulos y las interfaces, desde esta clase se inicia el registro del usuario en el servidor de voz y en el servidor de mensajería a través de los parámetros enviados por la página que inicio la carga del applet

4.6 Interfaces de la aplicación cliente prototipo

Las tareas mencionadas anteriormente: el Chat, VoIP y las llamadas hacia la PSTN corresponden a las herramientas que la aplicación B2B bChat, bIP y bPhone respectivamente.

La interfaz principal (ver FIGURA 4.7) muestra a todos los usuarios en dos grandes categorías la de “conectados” y “desconectados” y luego vienen las sub-categorías de contactos creados por el propio usuario. Cada usuario tiene una serie de opciones que permitirán al usuario realizar llamadas de VoIP, Chat y llamadas hacia la PSTN.

FIGURA 4.7 PROGRAMA CLIENTE : INTERFAZ PRINCIPAL

4.6.1 Interfaz de servicio Chat.

Esta interfaz tiene los elementos básicos para realizar una sesión de Chat, pero además dispone de un panel de tareas o “acciones” a realizarse durante el transcurso de la sesión. En el caso del modulo de VoIP al seleccionarse aparecerá la interfaz correspondiente con el nombre de usuario preseleccionado, para el caso del modulo de llamadas hacia la PSTN se levantará la interfaz correspondiente sin ningún número preseleccionado.

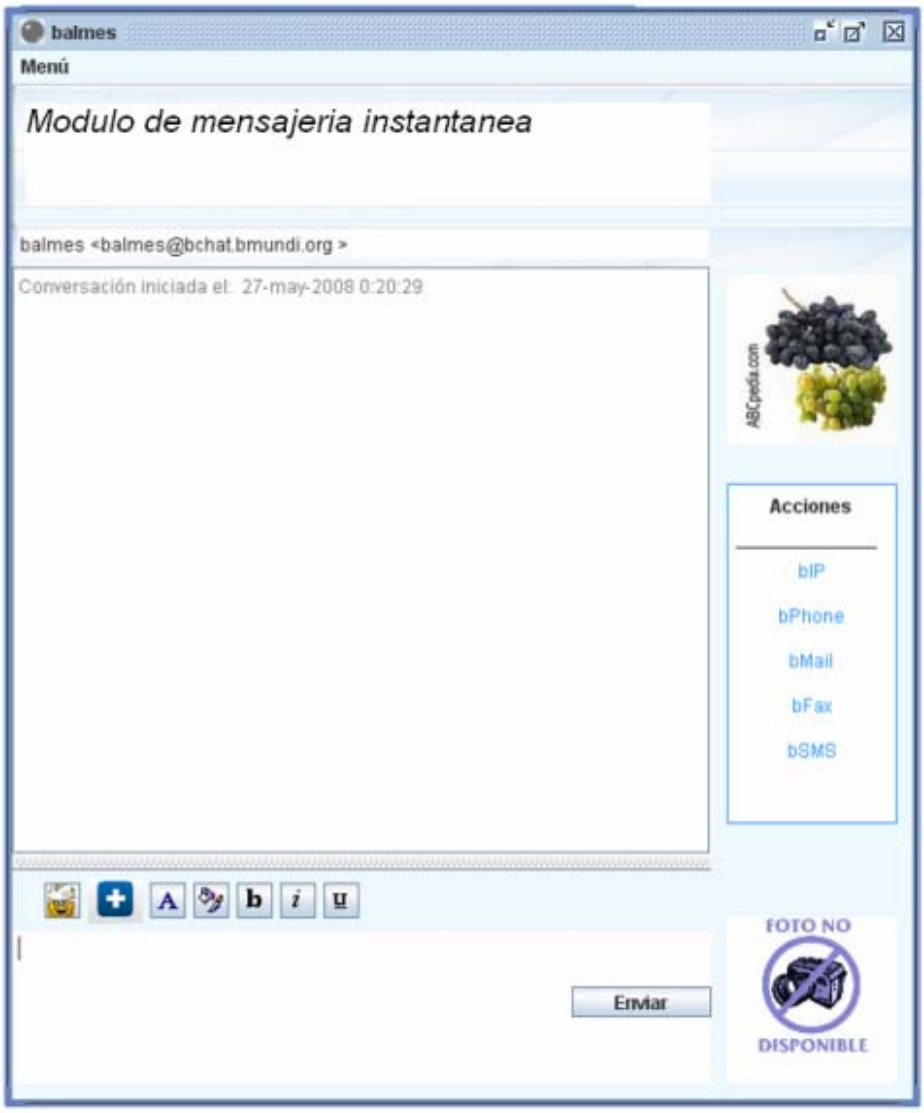

FIGURA 4.8 PROGRAMA CLIENTE : INTERFAZ CHAT

Esta interfaz (ver FIGURA 4.8) dispone de emoticonos de trabajo, los cuales están categorizados dependiendo del ambiente de trabajo, en este caso se han definido siete tipos: tiempo, logística, profesional, comunicaciones, oficina, finanzas y lugares. Estas categorías aparecerán cuando se presione el botón azul con una cruz (ver FIGURA 4.9).

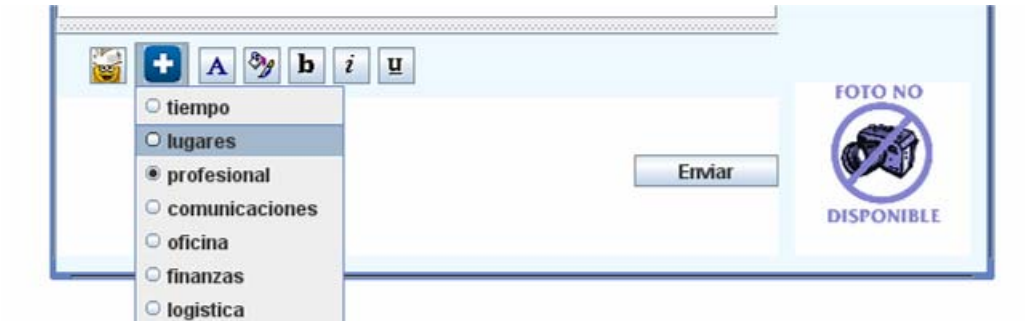

FIGURA 4.9 PROGRAMA CLIENTE : INTERFAZ CHAT 2

Luego de seleccionar la categoría correspondiente, el usuario podrá acceder directamente a todo este grupo presionando el botón de color amarillo y tendrá libre disponibilidad de usar el emoticonos de trabajo que más se adecúe al contexto sobre el que se esta trabajando (ver FIGURA 4.10).

FIGURA 4.10 PROGRAMA CLIENTE : INTERFAZ CHAT 3

La idea aquí es la de minimizar en lo posible, las barreras idiomáticas introduciendo emoticonos de trabajo categorizados y estándares en la plataforma (ver FIGURA 4.11).

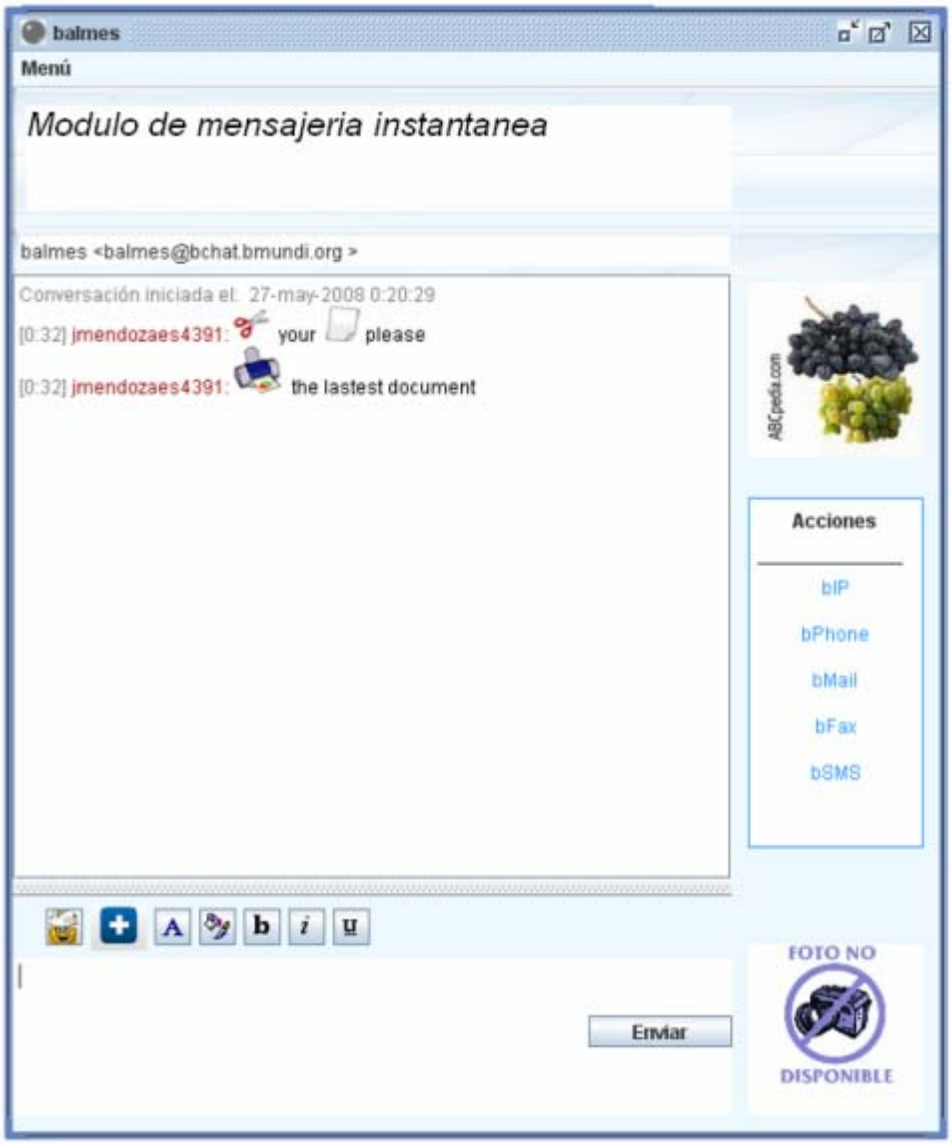

FIGURA 4.11 PROGRAMA CLIENTE : INTERFAZ CHAT 4

4.6.2 Interfaz de servicio de VoIP.

La interfaz de VoIP (ver FIGURA 4.12), consta de una botonera básica la cual permitirá al usuario efectuar llamadas de voz sobre otros usuarios que estén conectados a la plataforma y también modificar el volumen de audio. Además también cuenta con dos botones adicionales: herramientas y restablecer, donde el primero servirá para la configuración del audio y el segundo para reiniciar sólo esta interfaz en caso de algún error.

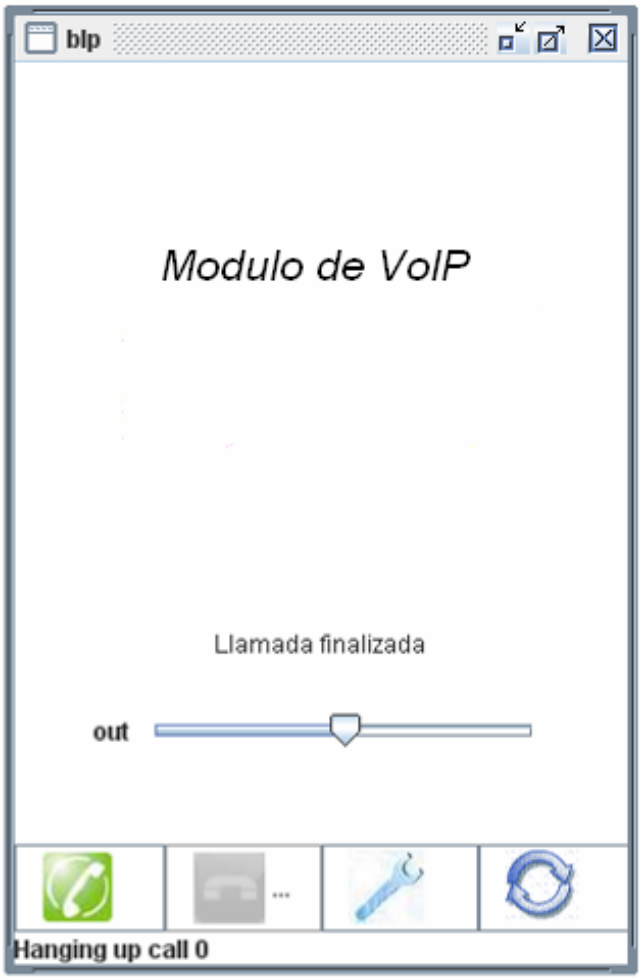

FIGURA 4.12 PROGRAMA CLIENTE : INTERFAZ VOIP

4.6.3 Interfaz de servicio de llamadas hacia la PSTN.

Esta interfaz es una copia de la interfaz anterior (ver FIGURA 4.12), adicionalmente posee una botonera numérica, una caja de texto para ingresar los números a llamar, una lista desplegable de países con sus prefijos y banderas correspondientes y un combo adicional para seleccionar tarjetas prepago, sólo esta última es una implementación pensada a futuro (ver FIGURA 4.13).

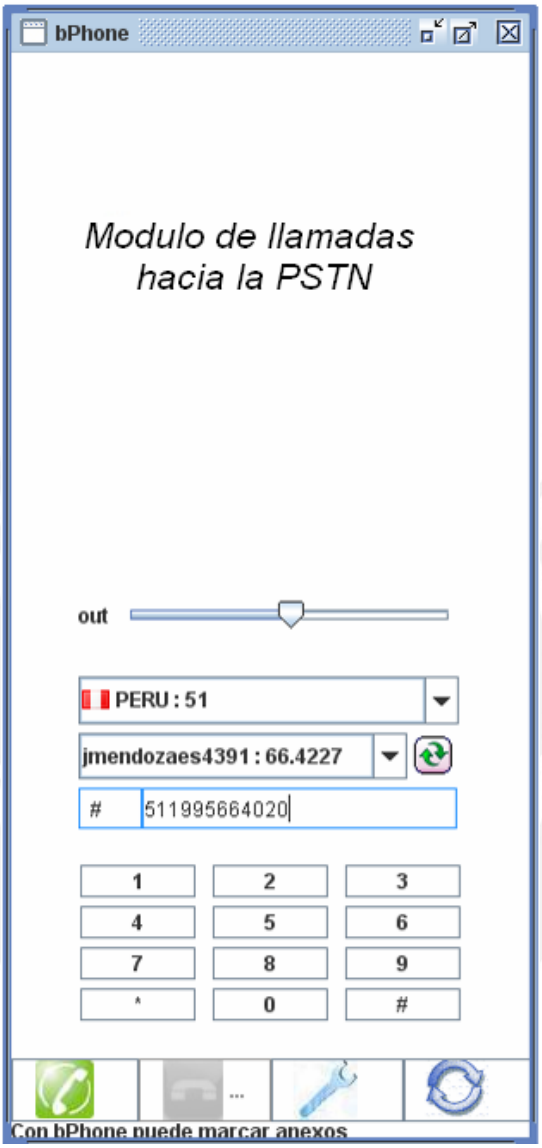

FIGURA 4.13 PROGRAMA CLIENTE : INTERFAZ DE LLAMADAS HACIA LA PSTN

Finalmente el diagrama de la plataforma completa queda como se muestra en la siguiente figura:

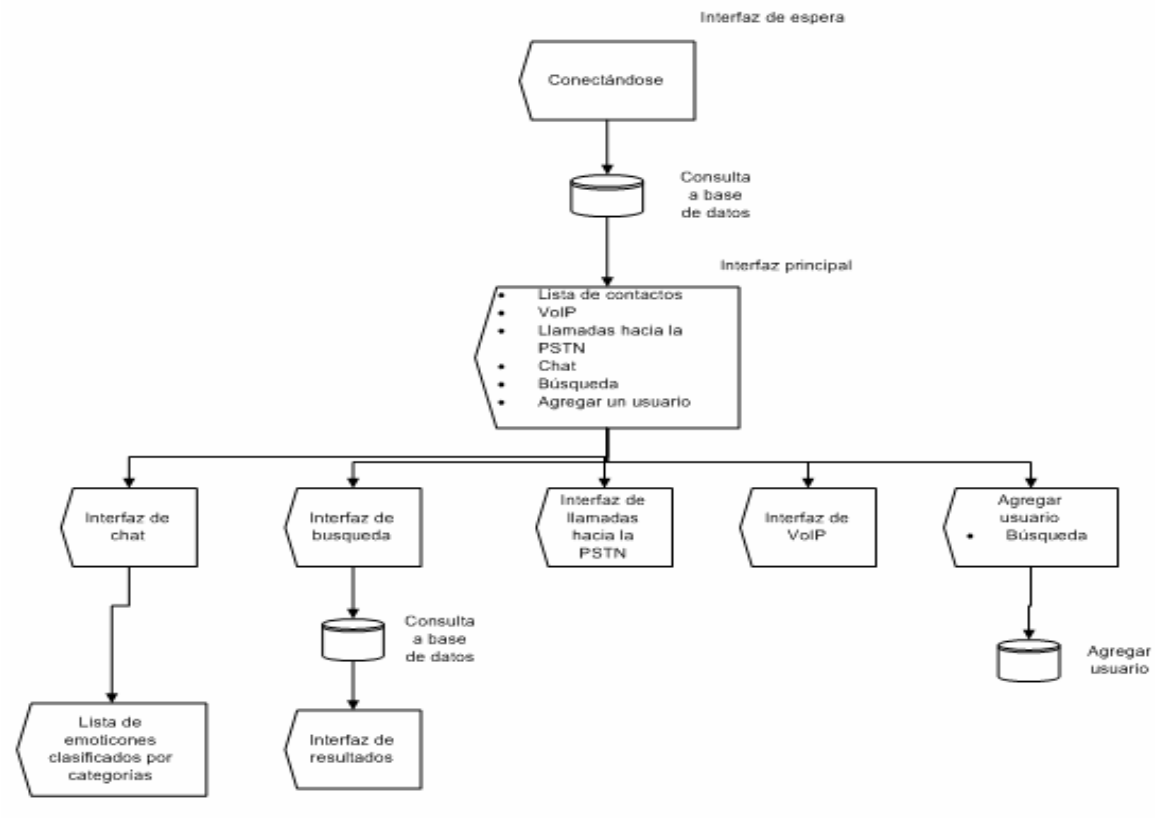

FIGURA 4.14 DIAGRAMA DE LA PLATAFORMA.

4.7 Integración del cliente con la aplicación B2B.

En esta sección se mostrará el acceso a la aplicación B2B y desde ella, el acceso al programa cliente de la plataforma y los eventos que muestra la aplicación B2B.

La siguiente figura muestra el registro a la aplicación B2B con el usuario “jmendozaes4391”.

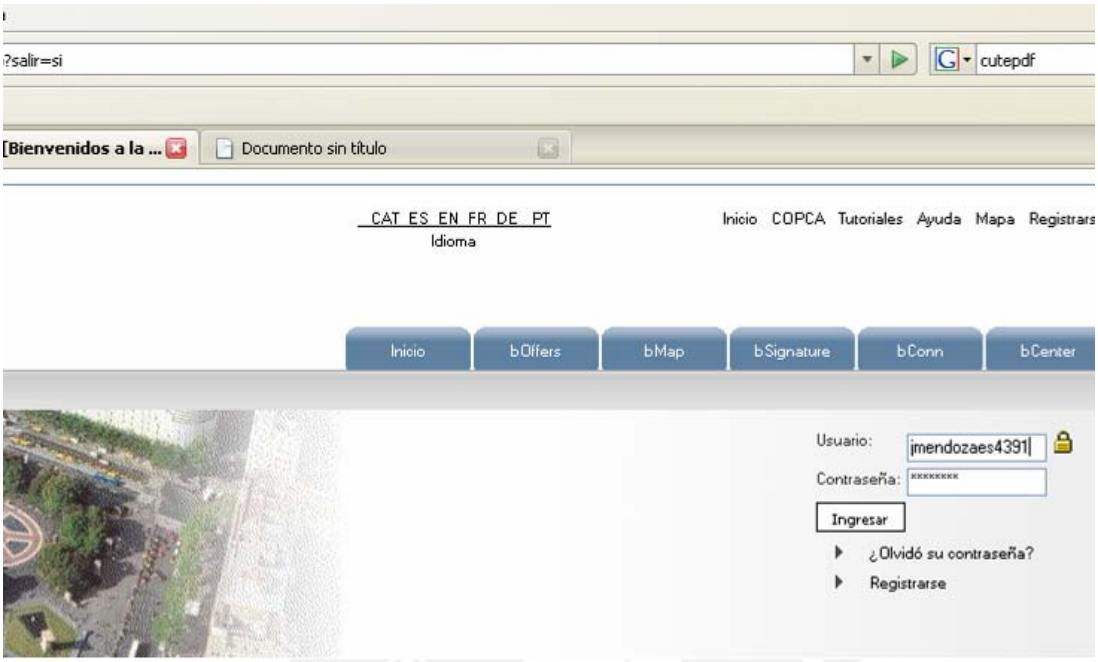

FIGURA 4.15 APLICACIÓN B2B : INTERFAZ DE REGISTRO

Una ves registrado en la aplicación se debe acceder al programa cliente de la plataforma mediante el enlace que dice “bConn” dentro de la pestaña “Comunicación”.

FIGURA 4.16 APLICACIÓN B2B : INTERFAZ DE ACCESO AL PROGRAMA CLIENTE

Luego se levantará una ventana, la cual cargara la aplicación (ver FIGURA 4.17):

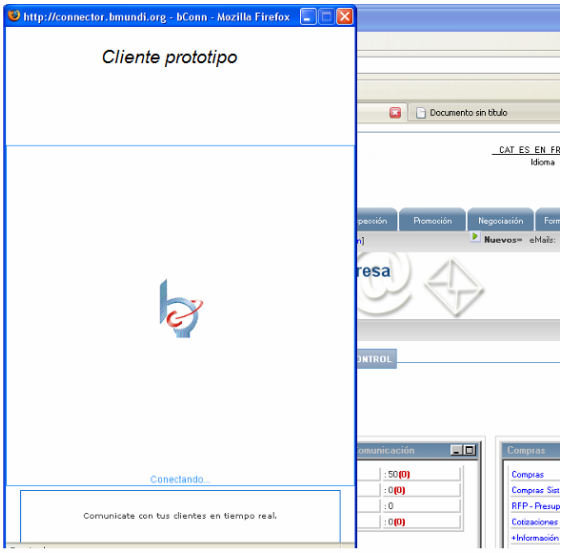

FIGURA 4.17 APLICACIÓN B2B : CARGA DEL PROGRAMA CLIENTE

Pasado un determinado tiempo, el cual depende mucho de la velocidad de conexión, se cargará la interfaz principal antes mostrada (ver FIGURA 4.7).

Para efectos de exposición, se han efectuado las tres tareas de la siguiente forma: se ha hecho una llamada de VoIP desde el usuario “jmendozaes4391” hacia el usuario “balmes”, se inicio una sesión de Chat desde el usuario “jmendozaes4391” con el usuario “balmes” y finalmente se hizo una llamada telefónica hacia el destinatario “(+511)995664020” desde el usuario “jmendozaes4391”.

A continuación se muestra como ingresar a la interfaz que muestra los eventos de las tareas antes mencionadas, dicha interfaz en la aplicación B2B se llama bAgenda y para acceder a ella se debe ingresar al enlace “bAgenda” dentro de la pestaña “organización”:

FIGURA 4.18 APLICACIÓN B2B : ACCESO A LA AGENDA

En la siguiente figura (FIGURA 4.19) se ven las tres tareas efectuadas, cuyo asunto son bChat, PHONE y VoIP

Control de Jornada 27/05/2008

Inicio	Fin	Creador	Ejecutor	Tarea	Tipo	Estado	Destinatario	Asunto
8:05	8:05	jmendozaes4391	jmendozaes4391	Chat - bConn	Emisor	Ejecutada	balmes	bChat
7:36	7:36	jmendozaes4391	jmendozaes4391	Desconectado - bConn	Emisor	Ejecutada	jmendozaes4391	bConn
5:19	5:19	jmendozaes4391	jmendozaes4391	Desconectado - bConn	Emisor	Ejecutada	jmendozaes4391	bConn
08:05:53	08:05:06	jmendozaes4391	jmendozaes4391	bPHONE	Emisor	Ejecutada	11511995664020	Phone
07:55	07:56	jmendozaes4391	jmendozaes4391	conectado	Emisor	Ejecutada	jmendozaes4391	bMundi
07:05:37	07:05:41	jmendozaes4391	jmendozaes4391	Llamada - bConn	Emisor	Ejecutada	balmes	VoIP
05:05:42	05:05:45	jmendozaes4391	jmendozaes4391	bPHONE	Emisor	Ejecutada	11511626****	Phone

FIGURA 4.19 APLICACIÓN B2B : ACCESO A LA AGENDA 2

Capítulo 5

Pruebas de desempeño

La plataforma desarrollada aun no se encuentra en un ambiente de pruebas, por lo que no se poseen resultados propios, sin embargo debido a que usa protocolos bien conocidos jabber basado en XMPP para la mensajería, el que utiliza el servidor de mensajería Openfire, e IAX2 para las comunicaciones, el cual utiliza nuestro servidor Asterisk. Existen pruebas ya hechas sobre estos servidores bajo ciertos parámetros como códecs, protocolos, arquitecturas de procesador, las cuales se mostrarán a continuación. En el caso del servidor Asterisk no se especifica cual es el tipo de hardware usado, pero sólo se tomarán estos resultados como referencia para comparar el desempeño de los códecs cuando hay un aumento de tráfico.

5.1 Servidor Openfire

Como se explico en capítulos anteriores el Openfire usa el protocolo jabber. Dicho servidor ha sido sometido a pruebas de escalabilidad los cuales han sido realizados por los mismos desarrolladores del servidor, dichas pruebas se muestran a continuación:

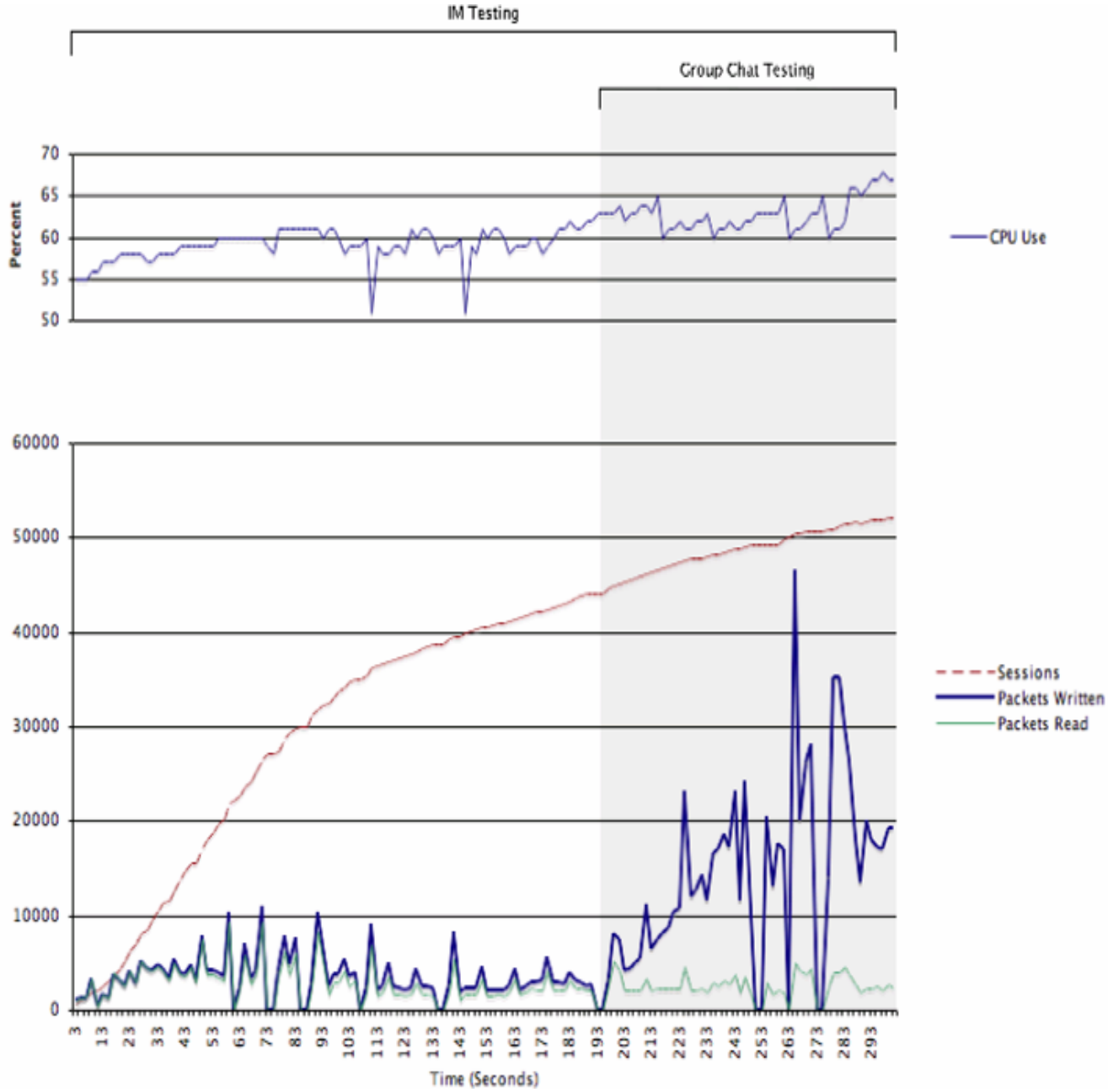

FIGURA 5.1 SERVIDOR OPENFIRE: CARGA DE PROCESADOR [OPS2007]

Aquí vemos dos gráficos, el superior muestra el porcentaje de uso de CPU respecto al tiempo, y el segundo gráfico muestra el número de usuarios que se van conectando respecto al tiempo. El análisis nos muestra que la carga del servidor esta alrededor del 60% cuando tiene cerca de 45000 sesiones de chat abiertas, lo cual indica que se requiere un gran número de usuarios para cargar el servidor. Esta prueba fué hecha sobre un servidor SUN 280R con dos CPU 1.2GHz UltraSparc-lll, 4GB RAM y usando una base de datos MySql para almacenar los datos.

5.2 Servidor Asterisk

Sobre este servidor se han realizado muchas pruebas y estas se pueden encontrar en Internet cuyas fuentes son universidades, empresas, centros de investigación y usuarios que quieren cooperar con la comunidad de desarrolladores y usuarios aficionados de este software. Para propósitos de la tesis solo se mostrará pruebas de rendimiento hechas sobre los siguientes parámetros los cuales se consideran críticos para la puesta en marcha de los servicios cuando la plataforma entera este en producción, los cuales son en base al protocolo de voz usado, el códec de voz, el uso de CPU y el consumo de ancho de banda.

Como se mencionó en los capítulos anteriores, para las comunicaciones de voz de la plataforma se ha usado el protocolo IAX2, con los códecs GSM para las llamadas por VoIP y ulaw (versión Estadounidense de G711) para las llamadas hacia la PSTN, pero para este último sólo se mostrarán pruebas del códec alaw (versión Europea de G711) hechas por la comunidad de desarrolladores de Asterisk. Las pruebas que se mostrarán a continuación se enfocarán en estos parámetros antes mencionados.

5.2.1 Códec vs. CPU

Primero debemos distinguir entre los diferentes códecs existentes, y tener en cuenta que cada uno tiene un tratamiento distinto de la voz y lo principal es que si bien el códec aprovecha muchas propiedades de la voz humana para comprimirla, también le resta calidad por lo que existe un compromiso entre ahorro de información en contra de calidad de voz. Esto se visualiza mejor en la siguiente gráfica:

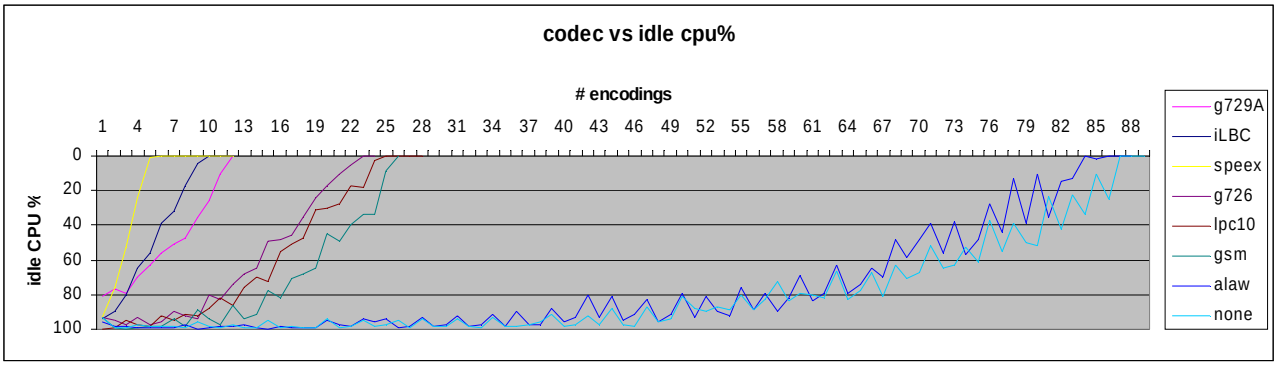

FIGURA 5.2 CÓDEC: CARGA DE PROCESADOR [ASP2004]

De aquí concluimos que el Códec que más consume procesador conforme el número de llamadas concurrentes va en aumento es el GSM respecto del alaw.

5.2.2Códec vs. ancho de banda

En la siguiente tabla (ver TABLA 5.1) se muestra una de las propiedades de IAX2 que es la reducción de cabeceras (trunking), esto quiere decir que este protocolo en vez de enviar cada paquete de voz con su propia cabecera, envía un grupo de paquetes con una sola cabecera, por lo que el ancho de banda consumido se ha ahorrado.

TABLA 5.1 PROTOCOLO IAX2 MODO TRUNKING: USO DE ANCHO DE BANDA [ASB2003]

4.5.4.3 G.711 one call: 164333.75 bps/94.26 pps (82.1 kbps) two calls: 296171.60 bps/101.46 pps (148.0 kbps) Thus: 3.4.2.4 For every additional call: 131837 bps (65.9 kbps) 3.4.2.5 Est. IP/IAX2 overhead (1 call): 32495 bps (16.0 kbps) 3.4.2.6 Raw number of calls per megabit: 15
2.4.8 GSM one call: 70958.16 bps/102.13 pps (35.4 kbps) two calls: 100455.23 bps/102.63 pps (50.2 kbps)

Thus:
3.5 For every additional call: 29497 bps (14.7 kbps)
3.6 Est. IP/IAX2 overhead (1 call): 41461 bps (20.7 kbps)
3.7 Raw number of calls per megabit: 68

Vemos que al agregar un canal de datos más a cada Códec, no se toma en cuenta la cabecera del siguiente paquete. En la tabla aparecen los códecs GSM y G711, en el caso de este último, las dos versiones ulaw y alaw consumen el mismo ancho de banda, por lo que los resultados serán los mismos.

El siguiente gráfico es una visualización del ancho de banda consumido cuando se usa el protocolo IAX2 con su funcionalidad trunking a medida que aumenta el número de llamadas concurrentes.

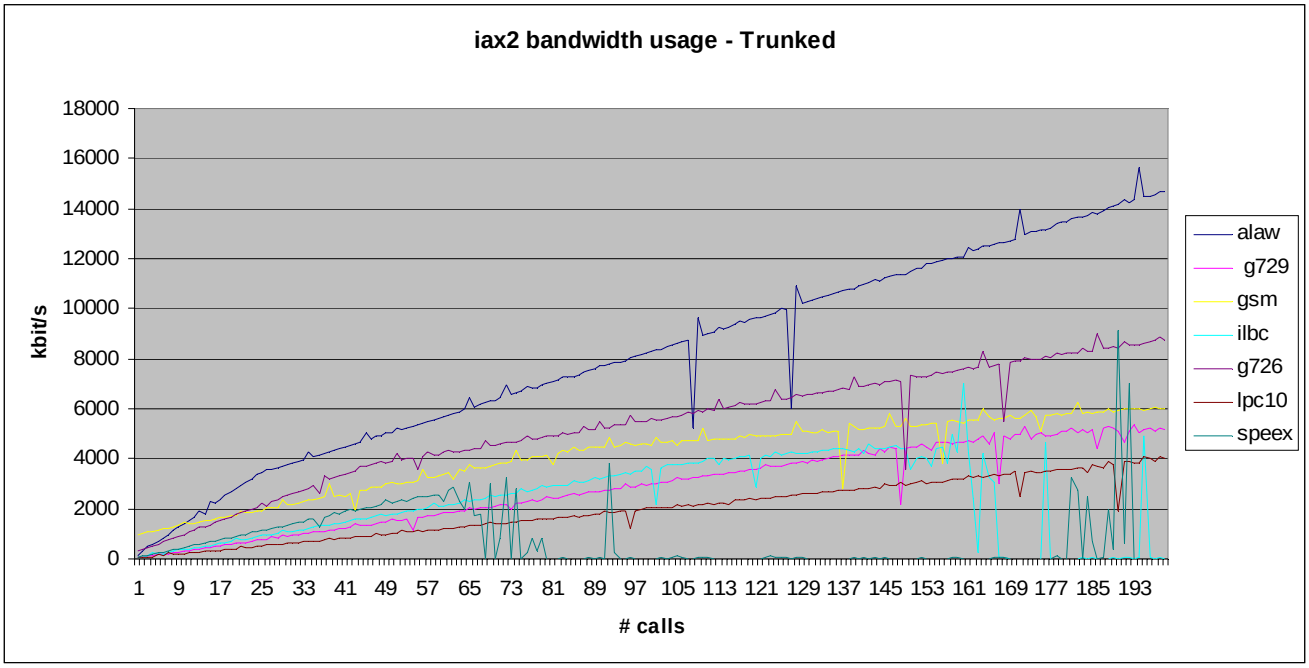

TABLA 5.2 PROTOCOLO IAX2 MODO TRUNKING: USO DE ANCHO DE BANDA [ASP2004]

Por tanto entre los dos códecs el G711 y GSM, mientras el primero ocupa mas ancho de banda y consume menos procesador, el segundo ahorra un considerable ancho de banda pero consume más procesador debido a la compresión que realiza sobre los paquetes de voz.

Capítulo 6

Conclusiones, recomendaciones y avances futuros.

6.1 Conclusiones

Se logró integrar la plataforma a la aplicación B2B y además se probó su funcionamiento y compatibilidad con la misma. A continuación se mencionan las conclusiones desde tres puntos de vista: el usuario, las empresas y la aplicación B2B.

Usuario:

- Los usuarios o colaboradores de la aplicación B2B pueden realizar múltiples tareas de comunicaciones a través de una sola interfaz, lo cual hace más interactiva la experiencia del usuario con la aplicación B2B.

- El uso de los emoticons categorizados ayudan a la comunicación entre usuarios en diferentes idiomas, y así acortan la brecha idiomática en el establecimiento de relaciones comerciales.

Empresas:

- Esta plataforma provee de las herramientas de comunicaciones necesarias para establecer las comunicaciones entre las empresas que participen en la aplicación B2B, además de poseer una funcionalidad muy valiosa para estas empresas: el control de jornada de sus colaboradores, mediante el registro detallado de los eventos de comunicaciones.

- Las empresas que ingresen a la aplicación B2B no necesitarán implementar esta compleja arquitectura que conforma la plataforma de mensajería unificada, simplemente usarán la plataforma para establecer sus comunicaciones entre sí y con sus colaboradores sin importar su ubicación geográfica. Esto les quita muchas preocupaciones a la empresa, tales como el costo de la implementación en hardware y software, el mantenimiento del sistema de comunicaciones tanto técnico como logístico, contacto con los proveedores de los servicios, entre otros.

Aplicación B2B:

- Esta plataforma agrega funcionalidades adicionales de comunicaciones, lo cual hace más atractivo el uso de la aplicación B2B que favorece a la mayor difusión de la misma.

6.2 Recomendaciones

Aplicación Cliente:

- El cliente final es un applet, el cual esta escrito en java, se recomienda que sea lo menos pesado posible, ya que al cargar puede tomar un tiempo cercano a un minuto solo la primera vez (ya que las siguientes veces sólo extrae información que ya esta guardada en caché), lo cual para el usuario no se hace muy grato al inicio.

- Se debe explotar en la mayor medida, el uso de hilos en java, para controlar los procesos que toman tiempo, tales como consultas a base de datos, ya que estos ocasionan retardos que hacen lenta la aplicación debido al tiempo de respuesta que hay que esperar, sin embargo con el uso de hilos se puede mejorar este proceso, haciendo la consulta de base de datos a través de un hilo en forma simultánea en vez de secuencial como comúnmente se realiza. Esto causa que cuando el usuario realice una operación de consulta de base de datos desde el programa cliente la interfaz no se le quede colgada esperando la respuesta, sino que se mantenga dinámica mientras un subproceso controla la llegada de la respuesta de la consulta.

Servidores:

Se recomienda para cada servicio las siguientes recomendaciones de hardware

- Servidor Asterisk: Se recomienda un servidor HP con procesador Intel Quad Core de 2.0GHz com 4GB RAM. Dicho servidor es capaz de soportar un promedio de 250 llamadas concurrentes usando el códec G729 (ver TABLA 5.2) con la versión comercial de Asterisk. Esta recomendación fue basada en la propuesta de la empresa SYNAPSIS presentado en el anexo2.

- Servidor OpenFire: Se recomienda SUN FIRE V245 con un CPU 1.5GHz, 4G RAM ya que este modelo ha reemplazado al SUN 280R que los mismos creadores lo usaron para sus pruebas.

6.3 Avances Futuros

En esta tesis se ha mostrado el desarrollo de una plataforma de mensajería unificada con los servicios de comunicaciones empresariales básicos, pero con los avances expuestos de la plataforma no es suficiente para una puesta en marcha dentro de un entorno de producción ya que faltan servicios como la facturación de llamadas, envío de fax, envío de SMS, entre otros además del robusto hardware necesario para los servidores. Cabe resaltar que todos estos nuevos servicios que generen eventos de comunicaciones deben registrarse también en base de datos para un control de agenda de las actividades del usuario. Finalmente dichos servicios deben ser independientes en funcionamiento uno del otro en la medida

de lo posible, ya que si uno falla no tiene porque afectar el funcionamiento del resto, esto se debe hacer manteniendo la percepción de que se trata de un único programa cliente que posee todos los servicios integrados.

- Facturación de Llamadas (Billing): Facturación de llamadas hacia la PSTN mediante el uso de Tarjetas prepago para cada usuario de la plataforma

- Habilitación de tonos DTMF: Útil para entornos de trabajo en el que existen anexos para los operarios.

- Envío de SMS: un servicio muy usado para propagandas a través del terminal móvil.

- Envío de Faxes: Este servicio es muy usado también por las empresas, debido a su rapidez y seguridad.

- Versiones en varios idiomas: La plataforma deberá estar traducida en varios idiomas debido a que se encuentra integrada a una aplicación B2B, la cual también se encuentra en varios idiomas.

- Detección de acceso a otros servicios de mensajería: Este servicio sondeará el computador del operario, exclusivamente con el permiso del administrador de la red, con el fin de encontrar servicios que en cierta medida no cooperen con la eficacia de las actividades en la empresa por parte de los empleados.

- Sincronización de contactos: El programa cliente de la plataforma permite agregar nuevos contactos; la aplicación B2B también permite esta funcionalidad, entonces surge un problema: los contactos existentes en la plataforma no necesariamente son los mismos que la aplicación B2B, por tanto esto es un tema de integración, en el que haría falta sincronizar la gestión de los usuarios de ambas áreas, la de la plataforma y la de la aplicación B2B.

BIBLIOGRAFÍA

[APP2002] Applet-servlet
HTTP://www.javahispano.org/contenidos/archivo/24/applet-servlet.pdf
Consultado en mayo del 2007

[ASB2003] Asterisk Bandwidth IAX2
HTTP://voip-info.org/wiki/view/Asterisk+bandwidth+iax2
Consultado en diciembre del 2007

[AST2006] Asterx Associació de Programari Lliure de Castelldefels / ¿Qué es Jabber?
HTTP://asterx.upc.edu/node/253
Consultado en diciembre del 2007

[ASP2004] Astricon Performance
HTTP://www.asteriskguru.com/downloads/asterisk_stability_and_security.ppt
Consultado en abril del 2008

[COD2006] Comparación de Códecs de Voip
HTTP://www.vozdigital.org
Consultado en diciembre del 2007

[G712007] Voz Sobre IP en Español Códec g711
HTTP://voip.megawan.com.ar/doku.php/códec_g711
Consultado en diciembre del 2007

[G722006] Tienda de Voip y Asterisk – G729
HTTP://www.voz-ip.com/licenciacodecg729a-p-157.html
Consultado en diciembre del 2007

[GSM2004] Códec GSM
HTTP://www.voip-info.org/wiki-GSM+Códec
Consultado en diciembre del 2007

[FOR2006] Foro de Voip
HTTP://www.voipforo.com/IAX/IAXvsSIP.php
Consultado en enero del 2008

[IAT2007] Instalar Tomcat 5.5 como módulo de Apache2 en Debian 4 Etch
HTTP://www.syntaxerror.es/2007/11/05/instalar-tomcat-55-como-módulo-de-apache2-en-debian-4-etch/

Consultado en mayo del 2008

[ING2007] Ingirealtime
HTTP://www.igniterealtime.org/.
Consultado en mayo del 2007

[INT2006] Interoperabilidad de XMMP
HTTP://blog.bepointbe.be/index.php/2006/06/09/14-instant-messaging-integration-into-the desktop
Consultado en diciembre del 2007

[IRO2007] CURSO VOZ SOBRE IP Y ASTERISK v1.0 . Módulo III
IRONTEC
HTTP://www.irontec.com
Consultado en noviembre del 2007

[JAB2007] Sitio Oficial de Jabber en Español
HTTP://www.jabberes.org/
Consultado en diciembre 2007

[JET2007] Jeti applet client
HTTP://jeti.jabberstudio.org/applet.php
2007

[JIA2006] Java Iax Client
HTTP://www.hem.za.org/jiaxclient/
2006

[MIN2007] Microsoft y Nortel
HTTP://www.recursosvoip.com/b2/noticias.php?s=innovative
Consultado en diciembre del 2007

[OPD2007] Openfire – Database Installation Guide
HTTP://www.igniterealtime.org/builds/openfire/docs/latest/documentation/database.html#postgres
Consultado en mayo del 2007

[OPS2007] Openfire Scalability
www.igniterealtime.org/about/**OpenfireScalability**.pdf
Consultado en diciembre del 2007

[ORT2007] Ortega, David 'Diseño e Implementación de un Sistema Interactivo de Respuesta de Voz (IVR) Piloto para la Reserva de Boletos del Ferrocarril

Cuzco – Machu Pichu ’
PUCP, 2007

[PBX2008] IP PBX System Overview
HTTP://www.3cx.com/phone-system/product-tour.html
Consultado en enero del 2008

[PRO2006] PROMPEX-Perú : De Empresas Pequeñas a Empresas Inteligentes
HTTP://www.prompex.gob.pe/Prompex/Documents/380c2b12-35e3-4a36-a421-2e3982f28377.pdf
2006

[PST2007] PSTN
HTTP://www.voip-info.org/wiki/view/PSTN
Consultado en diciembre del 2007

[QUI2006] Quintana Diego, ‘Diseño e Implementación de una Red de Telefonía IP con Software Libre en la RAAP’
PUCP, 2006

[SKY2007] Skype para Empresas
HTTP://www.skype.com/intl/es/download/skype/windows/business.html
Consultado en noviembre del 2007

[TRI2007] Triggers en PostgreSQL
HTTP://www.scribd.com/doc/102837/Triggers-en-PostgreSQL
Consultado en agosto del 2007

[UCM2008] Christopher R. Andrews, ‘Sistemas de Comunicación Unificados’ (UCM)
Traducido por Paulo N. Lama
HTTP://www.acm.org/crossroads/espanol/xrds8-1/ucs.html
Consultado en enero del 2008

[WIP2007] With PostgreSql
HTTP://www.asteriskguru.com/tutorials/realtime_pgsql.html
Consultado en mayo del 2007

[XMM2008] Sitio Oficial de XMMP en Ingles
HTTP://www.xmpp.org/about/
Consultado en enero del 2008

ANEXOS

Anexo 1: Documentación java de las clases del cliente prototipo.

Documentación de las funciones y clases del código fuente del cliente prototipo.

Anexo 2: Propuesta de la empresa SYNAPSIS.

Propuesta de la empresa SYNAPSIS para la implementación de un solución de cluster de servidores Asterisk para un desarrollo a gran escala.

Anexo 3: Documentación java del servidor de aplicaciones.

Documentación de las funciones y clases del código fuente del servidor de aplicaciones.

ÍNDICE

Anexo 1: Documentación java de las clases del cliente prototipo.

Documentación de las clases y métodos del código fuente del cliente prototipo. Mediante estas clases se pudieron implementar las funcionalidades distintas desarrolladas en la tesis y además también mediante ellas, estas funcionalidades se pudieron integrar.

---

src
Class IAXTest

java.lang.Object
  └ java.awt.Component
    └ java.awt.Container
      └ java.awt.Window
        └ java.awt.Frame
          └ javax.swing.JFrame
            └ **src.IAXTest**

All Implemented Interfaces:

java.awt.image.ImageObserver, java.awt.MenuContainer, java.io.Serializable, javax.accessibility.Accessible, javax.swing.RootPaneContainer, javax.swing.WindowConstants

---

public class **IAXTest**
extends javax.swing.Jframe

Author:

Mickael Magnuson.

See Also:

Serialized Form

Field Summary	
java.lang.String	**host**
java.lang.String	**number**
java.lang.String	**user**

Constructor Summary
IAXTest()
IAXTest(java.lang.String user, java.lang.String number, java.lang.String host)

Method Summary	
int	**buscaCoincidenciaPais**(java.lang.String cadena)

void	**call**(java.lang.String num)
void	**cargaTarjetas**(int opcion, DataServlet dataservlet)
void	**cargaTarjetas**(java.lang.String[] tarjetas, int opcion)
void	**codigosPaises**()
void	**creaTarjetas**()
java.awt.Component	**createButtons**()
protected javax.swing.ImageIcon	**createImageIcon**(java.lang.String path)
void	**destroy**()
void	**exit**()
java.lang.String	**getContacto**()
java.lang.String	**getHost**()
int	**getInput**()
java.lang.String	**getNumber**()
int	**getOutput**()
java.lang.String	**getPass**()
int	**getPort**()
java.lang.String	**getUser**()
void	**historial**(java.lang.String ocurrencia)
protected void	**initAudio**()
void	**initListener**()
void	**initProg**(java.lang.String anexo, java.lang.String number)
protected void	**initUI**(java.awt.Container cnt)

void	**initUIbFax**()
void	**initUIbPhone**()
void	**initUIbSms**()
boolean	**isRegister**()
void	**main1**(BCall bcall)
void	**masActionPerformed**(java.awt.event.ActionEvent evt)
void	**notvisible**()
void	**openbFax**()
void	**openbIp**()
void	**openbIp**(java.awt.event.ActionEvent evt)
void	**openbPhone**()
void	**openbSms**()
void	**postInitIaxc**()
void	**registrar**()
java.lang.String	**retornaHistorial**(int i)
void	**seleccionaPais**(java.awt.event.ActionEvent evt)
void	**setContacto**(java.lang.String contacto)
void	**setHost**(java.lang.String host)
void	**setInput**(int input)
void	**setNumber**(java.lang.String number)
void	**setOutput**(int output)
void	**setPass**(java.lang.String pass)

void	**setPort**(int port)
void	**setRegister**(boolean register)
void	**setText**(java.lang.String text)
void	**setUser**(java.lang.String user)
void	**showSettingsDialog**(javax.swing.JFrame parent)
void	**showStatus**(java.lang.String msg)
void	**shutdownIaxc**()
void	**start**()
void	**startIaxc**()
void	**stop**()
void	**stopIaxc**()
static void	**trace**(java.lang.Object str)
boolean	**validarNumero**(java.lang.String number)
void	**visible**()

---

nu.fw.jeti.ui
Class SendMessage

java.lang.Object
  └ java.awt.Component
    └ java.awt.Container
      └ java.awt.Window
        └ java.awt.Frame
          └ javax.swing.JFrame
            └ **nu.fw.jeti.ui.SendMessage**

All Implemented Interfaces:
java.awt.image.ImageObserver, java.awt.MenuContainer, java.io.Serializable, javax.accessibility.Accessible, javax.swing.RootPaneContainer, javax.swing.WindowConstants

---

public class **SendMessage**

extends javax.swing.JFrame

Title: im Description: Copyright: Copyright (c) 2001 Company:

Author:
E.S. de Boer
See Also:
Serialized Form

Constructor Summary
SendMessage(Backend backend, JIDStatus jidstatus, java.lang.String subject, java.lang.String text)
SendMessage(Backend backend, JID jid, java.lang.String user)
SendMessage(Backend backend, JID jid, java.lang.String user, java.lang.String title)
SendMessage(Backend backend, Message messageElement)

---

nu.fw.jeti.ui
Class Jeti

java.lang.Object
└ java.awt.Component
└ java.awt.Container
└ javax.swing.JComponent
└ javax.swing.JPanel
└ **nu.fw.jeti.ui.Jeti**
All Implemented Interfaces:
java.awt.image.ImageObserver, java.awt.MenuContainer, java.io.Serializable, javax.accessibility.Accessible, JETIListener, PreferenceListener, PresenceListener, StatusChangeListener

---

public class **Jeti**
extends javax.swing.JPanel
implements PresenceListener, StatusChangeListener, PreferenceListener
Author:
E.S. de Boer
See Also:
Serialized Form

Constructor Summary
Jeti(Backend backend, java.awt.Container container)

Jeti(Backend backend, java.awt.Container container, IAXTest iaxtest, java.applet.AppletContext jetiAppletContext)

Method Summary	
void	**addToMenu**(javax.swing.JMenuItem menuItem)
void	**addToOfflineRosterMenu**(java.lang.String name, RosterMenuListener listener)
void	**addToOnlineRosterMenu**(java.lang.String name, RosterMenuListener listener)
void	**addToRosterMenu**(java.lang.String name, RosterMenuListener listener)
void	**changeOFFlinePanel**(boolean show)
void	**chat**(JIDStatus jidStatus)
void	**close**()
void	**connectionChanged**(boolean online) Called when the connection status has changed.
javax.swing.JPanel	**createChatPanel**(JIDStatus jidStatus)
void	**exit**() Called when Jeti will be closed
IAXTest	**getIaxtest**()
RosterTree	**getOnlinePanel**()
java.util.Map	**getRosterMenuItems**()
void	**init**()
void	**openGroups**(JetiPrivateRosterExtension extension)
void	**ownPresenceChanged**(int astatus, java.lang.String amessage) Called when you have changed your own presence status
void	**paint**(java.awt.Graphics g)
void	**preferencesChanged**()
void	**presenceChanged**(Presence presence)

	Called when someones presence has been changed.
void	**refreshBackgroundImage**()
void	**removeFromMenu**(javax.swing.JMenuItem menuItem)
void	**removeFromOfflineRosterMenu**(java.lang.String name)
void	**removeFromOnlineRosterMenu**(java.lang.String name)
void	**removeFromRosterMenu**(java.lang.String name)
protected void	**saveOpenGroups**()
void	**setIaxtest**(IAXTest iaxtest)
void	**startChat**(JID jid)
void	**startChatResource**(JIDStatus jidstatus)
void	**translate**() translate the main display if another locale is choosen
void	**updateLF**()

---

nu.fw.jeti.ui
Class ChatWindow

java.lang.Object
  └ java.awt.Component
    └ java.awt.Container
      └ javax.swing.JComponent
        └ javax.swing.JPanel
          └ **nu.fw.jeti.ui.ChatWindow**

All Implemented Interfaces:
java.awt.image.ImageObserver, java.awt.MenuContainer, java.io.Serializable, javax.accessibility.Accessible

---

public class **ChatWindow**
extends javax.swing.JPanel

Author:
E.S. de Boer

See Also:
Serialized Form

Constructor Summary
ChatWindow(Backend backend, ChatWindows chatWindows, javax.swing.JFrame frame, JIDStatus jidStatus, java.lang.String thread)
ChatWindow(Backend backend, ChatWindows chatWindows, javax.swing.JFrame frame, JIDStatus jidStatus, java.lang.String thread, IAXTest iaxtest, java.applet.AppletContext jetiAppletContext, bSms bsms)

Method Summary	
void	**addSplitBarListener**(java.awt.event.ComponentListener listener)
void	**appendMessage**(Message message)
void	**appendPresenceChange**(Presence presence)
void	**bConnBanner**()
void	**bConnllenajPanelDerecha**()
void	**changeToWindow**()
void	**composing**(java.lang.String type)
void	**composingID**(java.lang.String id)
void	**exit**()
boolean	**getEmisor**()
boolean	**getExternoEnvioTexto**()
IAXTest	**getIaxtest**()
JIDStatus	**getJIDStatus**()
javax.swing.JMenu	**getMenu**()
java.sql.Statement	**getStm**()
java.lang.String	**getThread**()
void	**removeSplitBarListener**(java.awt.event.ComponentListener li

	stener)
void	**setDividerLocation**(int location)
void	**setEmisor**(boolean emisor)
void	**setExternoEnvioTexto**(boolean externoEnvioTexto)
void	**setIaxtest**(IAXTest iaxtest)
void	**setLocationOnScreen**(ChatWindow oldWindow)
void	**setLocationOnScreen**(int posX, int posY)
void	**setParentFrame**(javax.swing.JFrame frame)
void	**setStm**(java.sql.Statement stm)

---

nu.fw.jeti.plugins.emoticons
Class Plugin

java.lang.Object
  └ **nu.fw.jeti.plugins.emoticons.Plugin**
All Implemented Interfaces:
Emoticons, Plugins

---

public class **Plugin**
extends java.lang.Object
implements Plugins, Emoticons
Author:
E.S. de Boer

Field Summary	
static java.lang.String	**ABOUT**
static java.lang.String	**DESCRIPTION**
javax.swing.text.Document	**doc**
static java.lang.String	**MIN_JETI_VERSION**
static java.lang.String	**NAME**

static java.lang.String	**VERSION**
int	**x**
int	**y**

Constructor Summary
Plugin()
Plugin(Backend backend)

Method Summary	
static void	**init**(Backend backend)
void	**init**(javax.swing.JTextPane txtOutput, javax.swing.JPanel controls, javax.swing.JTextPane txtInput, javax.swing.JPopupMenu popupMenu, java.lang.String type, javax.swing.JMenu menu)
void	**insertEmoticons**(java.util.List wordList)
void	**loadEmoticon**(java.lang.String name, java.lang.String type)
void	**unload**() Perform unload cleaning of loaded plugins.
void	**unloadEmoticon**(java.lang.String name)

---

ext.util
Class DataServlet

java.lang.Object
└ **ext.util.DataServlet**

---

public class **DataServlet**
extends java.lang.Object

Constructor Summary
DataServlet() Creates a new instance of DataServlet

Method Summary	
void	**doRequest**(java.lang.String name)
void	**doRequest**(java.lang.String name, java.lang.String pathServlet)
void	**enviarPeticion**(java.lang.String user, java.lang.String pass)
void	**enviarPeticion**(java.lang.String user, java.lang.String pass, java.lang.String pathServlet)
java.lang.String	**getUrlServlet**()
java.lang.String	**getUTF8Str**()
void	**Servlet**(java.lang.String pathServlet)
void	**setUrlServlet**(java.lang.String urlServlet)
void	**setUTF8Str**(java.lang.String UTF8Str)

Anexo 2: Propuesta de la empresa SYNAPSIS.

Propuesta de la empresa SYNAPSIS para la implementación de una solución de cluster de servidores Asterisk para un desarrollo a gran escala.

Funcionalidad

Incluye Innumerables características que solo estaban disponibles en sistemas caros de telefonía detalladas a continuación:

• Caller ID • Parqueo de llamada • Música en espera • Transferencia de llamada • Desvios de llamadas • Llamada en espera • Colas de llamadas • Timbres distintos • Conferencias • DND • Conferencias Multiples • Musica en Espera soportando formatos MP3. • Claves de acceso • Grabaciones de llamadas • Callback • DISA • Manejo de Horarios de atención • Follow Me • Informes personalizados almacenados en Base de Datos, • Panel Gráfico de Seguimiento de llamadas • Buzón de Voz • IVR • Sistema automático de distribución de llamadas, ACD • Soporte de Integración con sistemas de gestión comercial o de atención al cliente CTI • Receptor de Alarmas • Adición de Mensajes • Listas Negras • Transferencia Ciega	• Transferencia con Consulta • Registro de detalles de Llamada • Reenvío de llamada en ocupado • Reenvío de llamada en No-disponible • Reenvío de llamada variable • Monitorización de Llamadas • Recuperación de Llamadas • Enrutamiento de llamadas (DID & ANI) • Escucha de Llamadas • Llamada en Espera • Identificación de Llamada • Bloqueo por identificación de llamada • Tarjetas prepago • Multiconferencia • Almacenamiento / Recuperación en BBDD • Integración con BBDD • Llamada por Nombre • Sistema de Acceso directo entrante • Timbre personalizable • No molestar • E911 • ENUM • Recepción y Envío de FAx • Lógica de extensiones Flexible • Listado de directorio Interactivo • Respuesta de Voz Interactiva(IVR) • Agentes de llamada • Macros • Autentificación • Música en Espera	• Música en Espera en transferencia • Sistema de MP3 configurable • Control de Volumen • Privacidad • Conversión de protocolo • Captura de Llamadas • Extensiones móviles • Enrutamiento por Identificador de llamada • Mensajería SMS • Soporte de Sistema TextToSpeach • Emitir Letras y Números • Soporte para Detección de Voz • Llamada a tres • Fecha y Hora • Traducción de Codec • Trunking • Pasarelas VozIP • Sistema de Buzón de Voz • Indicador visual de mensaje no escuchado • Indicador sonoro de mensaje no escuchado • Mensajes del Buzón de Voz a Email • Grupos de Buzón de Voz • Interfaz Web de acceso al Buzón de Voz • Identificación de llamada en Llamada en Espera • Soporte de oficina Remoto • Atención de llamada Automática, etc.

Funciones Asterisk Cuadro 1.1

PROPUESTA ECONOMICA

Estimados Sres. BMUNDI

Por medio de la presente, le hacemos entrega de nuestra propuesta económica por los siguientes conceptos:

Solución de Telefonía IP Basada en Asterisk

La empresa BMUNDI en la actualidad necesita brindar un servicio de telefonía IP la cual permita a los inscritos a su portal de negocios realizar llamadas a nivel mundial a través de una sola pasarela de telefonía totalmente transparente para el usuario.

Synapsis, al evaluar las necesidades del cliente, propone implementar una solución de telefonía basada en Asterisk *(ver gráfico 1.1)*, la cual le permita ahorra costos de adquisición de la tecnología y obtener una gama de beneficios, adjuntos a la siguiente propuesta.

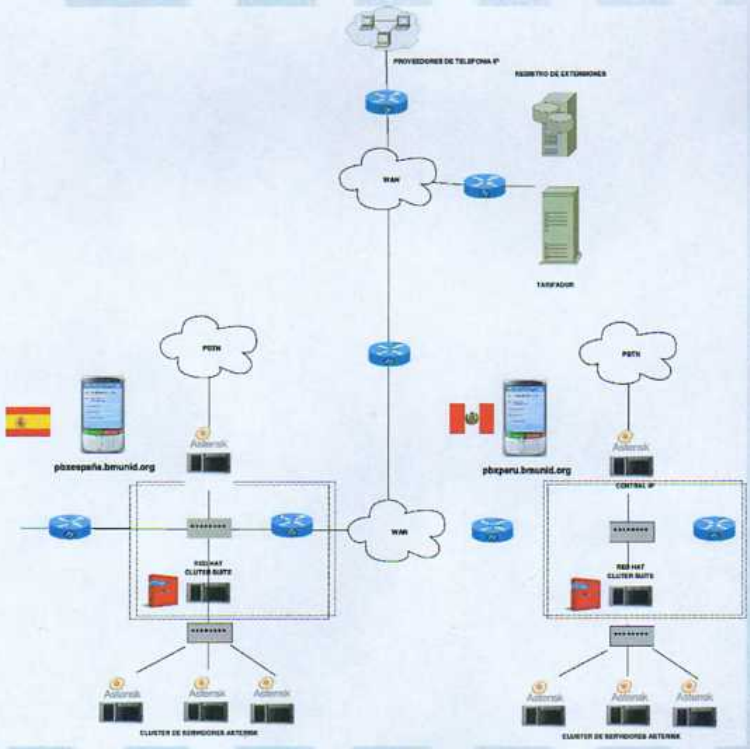

Gráfico 1.1

Según la gráfica adjunta, se contempla que para cada país debería haber una infraestructura similar y según la carga de llamadas puedan crecer de manera horizontal con servidores Asterisk, gracias a que se encuentra en un cluster de servidores bajo plataforma Red Hat Enterprise Linux.

Se contempla que la información de las extensiones y registro de usuario se encuentra de manera centralizada en un único servidor mundial el cual permite tener el mejor control del registro de los usuarios y la información de la misma, de manera similar se propone un servidor independiente para el software de tarificación y permita el conteo de minutos adquiridos por parte de los usuarios para las llamadas hacia la red pública de cada país.

Cada segmento de País deberá contemplar un servidor de pasarela hacia la red publica que podrá soportar hasta 8 Primarios y estos a su vez pueden crecer horizontalmente según el crecimiento a través del tiempo.

Por la magnitud de la solución se ha contemplado incluir productos certificados como plataforma Base, Red Hat Enterprise Linux y Asterisk Businnes Edition, la cual permitirá realizar 250 llamadas concurrentes por cada servidor apilado al Cluster.

Consideraciones de Hardware

La presente solución no considera costos de Hardware, pero recomienda las siguientes especificaciones técnicas para cada uno de los servidores:

Unidad Central del Sistema:

- 01 Procesador Intel Quad Core de 2.0 GHz o superior
- Cache L2 de 2 Mb por Core (2x4Mb)
- Velocidad del Bus 1133 Mhz
- Capacidad de crecimiento a dos procesadores
- Orientación del chasis: Rack (5U)

Memoria:

- 4 Gb instalados
- Tipo PC2- 5300 FB-DIMM DDR2 con detección y corrección de error (ECC)
- Velocidad de 600 Mhz o superior
- El 50 % de los slots de memoria deben encontrarse libres.

Almacenamiento:

- 01 Controlador con soporte para RAID 0,1 y 5 por hardware, con 256 Mb de cache.
- 03 discos Hot Plug Serial Attach SCSI (SAS) de 72 Gb, 15000 RPM
- Capacidad máxima de 6 discos Hot Plug (SAS o SATA)
- Drive Optico: DVD/CD-ROM

Controladores de Red y Conectividad

- 02 Adaptadores Gigabit Ethernet RJ45 integrado
- 01 Interfase de red 10/100 Mbps para administración remota por hardware

Slots Internos:

- Mínimo 6 (PCI – Express o PCI-X)

Análisis predictivo de fallas:

- Mínimo sobre procesador, memoria y discos

Fuente de Poder:

Fuentes de poder y Ventiladores Redundantes del tipo Hot Plug con capacidad de soportar el servidor con su máxima configuración.

Monitoreo remoto del servidor:

- Soporte a través de la interfase de administración remota por hardware, debe soportar SNMP y HTTPS y contar con software de consola grafica incluida.

Certificaciones:

- FCC, CE, UL

Garantía:

- 3 años de garantía integral sobre partes, mano de obra y servicio on site
- Soporte 24 x 7 con tiempo de respuesta On site de 4 horas.

Implementación de Centrales por País

Se detallan los costos de manera independiente para que se pueda dimensionar de manera progresiva según el crecimiento y necesidades de la organización.

Item	Descripción	P. Unit.	Cant.	Total
1	**Gateway de pasarela PSTN** • Red Hat Enterprise Linux Basic • Tarjeta de 4 Primarios (Dimensionamiento Inicial) • Asterisk Businnes Edition • Red Hat Cluster Suite	$9,570.00	1	$9,570.00
2	**Servidor de Balanceo de Carga** • Red Hat Enterprise Linux Basic • Red Hat Cluster Suite	$940.00	1	$940.00
3	**Central Telefónica Basa En Asterisk (Nodo Cluster):** • Red Hat Enterprise Linux Basic • Asterisk Businnes Edition • Red Hat Cluster Suite • 250 Codec's G.729	$6,185.00	1	$6,185.00
4	**Servicio de Implementación y Soporte (1 año) Por País** • Implementación de solución propuesta • Soporte Remoto 9x5 durante 1año	$48,200.00	1	$48,200.00

Nota:
Estos costos no incluyen impuestos ni costos de viajes y estadía para realizar la implementación fuera de Perú.

Atentamente,

synapsis endesa

Diego M. FRANCIA
Consultor, Gerencia de Nuevos Negocios

Teléfono : (511) 418-1160
FAX : (511) 418-1162
Email : dfrancia@synapsis.com.pe
URL : http://www.synapsis-it.com

Anexo 3: Documentación java del las clases del servidor de aplicaciones.

Documentación de las clases y métodos del código fuente del servidor de aplicaciones. Este elemento de la tesis hace posible el registro de eventos en la base de datos, ya que posee métodos que reciben la información proveniente del cliente prototipo para luego insertarla en la tabla de base de datos correspondiente.

---

dao
Class DEvento

java.lang.Object
  └ dao.DBase
    └ **dao.DEvento**

---

public class **DEvento**
extends DBase

Field Summary

Fields inherited from class dao.DBase
password, url, usuario

Constructor Summary	
DEvento()	

Method Summary	
void	**insert**(BEvento bean)
void	**update**(BEvento bean)

Methods inherited from class dao.DBase
getConn, setConn

Constructor Detail

DEvento

public **DEvento**()

Method Detail

insert

public void **insert**(BEvento bean)

throws java.sql.SQLException
Throws:
java.sql.SQLException

---

update

public void **update**(BEvento bean)
throws java.sql.SQLException
Throws:
java.sql.SQLException

---

bean
Class BEvento

java.lang.Object
└ **bean.BEvento**

---

public class **BEvento**
extends java.lang.Object

Constructor Summary
BEvento() Creates a new instance of BEvento

Method Summary	
java.lang.String	**getConectado**()
java.lang.String	**getDestino**()
long	**getDuracion**()
java.lang.String	**getEmisor**()
java.lang.String	**getEvento**()
java.lang.String	**getFecha**()
java.lang.String	**getFin**()
java.lang.String	**getFuente**()
java.lang.String	**getInicio**()
java.lang.String	**getSeparador**()

java.lang.String	**getTipo**()
java.lang.String	**peticionPost**(java.lang.String mensaje)
void	**setConectado**(java.lang.String conectado)
void	**setDestino**(java.lang.String destino)
void	**setDuracion**(long duracion)
void	**setEmisor**(java.lang.String emisor)
void	**setFecha**(java.lang.String fecha)
void	**setFin**(java.lang.String fin)
void	**setFuente**(java.lang.String fuente)
void	**setInicio**(java.lang.String inicio)
void	**setSeparador**(java.lang.String separador)
void	**setTipo**(java.lang.String tipo)

Constructor Detail

BEvento

public **BEvento**()

Creates a new instance of BEvento

Method Detail

getEvento

public java.lang.String **getEvento**()

---

getFuente

public java.lang.String **getFuente**()

---

setFuente

public void **setFuente**(java.lang.String fuente)

getDestino

public java.lang.String **getDestino**()

setDestino

public void **setDestino**(java.lang.String destino)

getInicio

public java.lang.String **getInicio**()

setInicio

public void **setInicio**(java.lang.String inicio)

getFin

public java.lang.String **getFin**()

setFin

public void **setFin**(java.lang.String fin)

getConectado

public java.lang.String **getConectado**()

setConectado

public void **setConectado**(java.lang.String conectado)

getSeparador

public java.lang.String **getSeparador**()

setSeparador

public void **setSeparador**(java.lang.String separador)

peticionPost

public java.lang.String **peticionPost**(java.lang.String mensaje)

---

getFecha

public java.lang.String **getFecha**()

---

setFecha

public void **setFecha**(java.lang.String fecha)

---

getDuracion

public long **getDuracion**()

---

setDuracion

public void **setDuracion**(long duracion)

---

getEmisor

public java.lang.String **getEmisor**()

---

setEmisor

public void **setEmisor**(java.lang.String emisor)

---

getTipo

public java.lang.String **getTipo**()

---

setTipo

public void **setTipo**(java.lang.String tipo)

---

servlets
Class SEventos

java.lang.Object
  └ javax.servlet.GenericServlet
    └ javax.servlet.http.HttpServlet
      └ **servlets.SEventos**
All Implemented Interfaces:
java.io.Serializable, javax.servlet.Servlet, javax.servlet.ServletConfig

public class **SEventos**
extends javax.servlet.http.HttpServlet
See Also:
Serialized Form

Constructor Summary	
SEventos()	

Method Summary	
protected void	**doGet**(javax.servlet.http.HttpServletRequest request, javax.servlet.http.HttpServletResponse response) Handles the HTTP GET method.
protected void	**doPost**(javax.servlet.http.HttpServletRequest request, javax.servlet.http.HttpServletResponse response) Handles the HTTP POST method.
java.lang.String	**getServletInfo**() Returns a short description of the servlet.
protected void	**processRequest**(javax.servlet.http.HttpServletRequest request, javax.servlet.http.HttpServletResponse response) Processes requests for both HTTP GET and POST methods.

Constructor Detail

SEventos

public **SEventos**()

Method Detail

processRequest

protected void **processRequest**(javax.servlet.http.HttpServletRequest request,
javax.servlet.http.HttpServletResponse response)
throws javax.servlet.ServletException,
java.io.IOException
Processes requests for both HTTP GET and POST methods.
Parameters:
request - servlet request
response - servlet response
Throws:
javax.servlet.ServletException
java.io.IOException

doGet

protected void **doGet**(javax.servlet.http.HttpServletRequest request,
javax.servlet.http.HttpServletResponse response)

throws javax.servlet.ServletException,
java.io.IOException
Handles the HTTP GET method.
Overrides:
doGet in class javax.servlet.http.HttpServlet
Parameters:
request - servlet request
response - servlet response
Throws:
javax.servlet.ServletException
java.io.IOException

---

doPost

protected void **doPost**(javax.servlet.http.HttpServletRequest request,
javax.servlet.http.HttpServletResponse response)
throws javax.servlet.ServletException,
java.io.IOException
Handles the HTTP POST method.
Overrides:
doPost in class javax.servlet.http.HttpServlet
Parameters:
request - servlet request
response - servlet response
Throws:
javax.servlet.ServletException
java.io.IOException

---

getServletInfo

public java.lang.String **getServletInfo**()
Returns a short description of the servlet.
Specified by:
getServletInfo in interface javax.servlet.Servlet
Overrides:
getServletInfo in class javax.servlet.GenericServlet