UNIVERSIDAD POLITÉCNICA
DE MADRID
ESCUELA TÉCNICA SUPERIOR
DE INGENIEROS DE
TELECOMUNICACIÓN

11 de junio de 2009

TITULO: PLATAFORMA ABIERTA PARA EL DESARROLLO Y DESPLIEGUE DE COMPONENTES DE SERVICIOS TELCO JAVA EN VoIP

NOMBRE: JONATHAN GONZÁLEZ SÁNCHEZ

Resumen

El objetivo principal del presente proyecto es proveer una plataforma abierta para el desarrollo y creación de servicios de telecomunicación para la Red de Nueva Generación (NGN)

El proyecto investiga en el desarrollo de herramientas de software libre para entornos telco que cumplan con los fuertes requisitos de disponibilidad y fiabilidad de estas redes. Como resultado, la plataforma resultante se liberará como código abierto con licencia GPL, lo cual supone un cambio de mentalidad frente el uso exclusivo de software propietario en este tipo de entornos.

El proyecto innova en la aplicación de sistemas inteligentes a las telecomunicaciones. Frente el fenómeno emergente de la sobre-comunicación potenciado por el auge de las comunicaciones personales, el proyecto investiga en la integración de técnicas inteligentes para gestionar dichas comunicaciones según las preferencias de los usuarios.

Palabras Clave

SIP, VoIP, Java, SIP Servlets, NGN, motor de reglas, reglas de negocio, sistemas BRMS, personalización, JBoss Rules, Drools, Sailfin, Asterisk, OpenSIPS.

A mis padres,

con mucho cariño

Agradecimientos

Me gustaría escribir unas líneas para acordarme de todas aquellas personas que me han apoyado y ayudado durante estos años en la Escuela.

En primer lugar, me gustaría acordarme de todos aquellos que han compartido conmigo las innumerables clases y horas de prácticas durante estos años. No mencionaré nombres que al final siempre se me olvida alguien, muchas gracias!

Me gustaría también agradecer a todos aquellos profesores de los que he aprendido y que han sabido transmitir ese interés por ciertas materias haciéndolas más interesantes y llevaderas. Mención especial a Carlos Ángel que desde primero me ha ayudado y orientado durante todos estos años. Gracias!

También me gustaría acordarme de Íñigo, él me enseñó a jugar. Me transmitió su inquietud y siempre ha estado ahí cuando le he necesitado, muchas gracias!

Y por último, y no por ello menos importante, agradecer a mis padres, amigos, mi hermano (Raúl) y a Rosa, su ayuda y comprensión en estos años de carrera dentro y fuera de la Escuela. Con ellos, toda tarea complicada de la escuela se ha hecho más llevadera.

A todos: GRACIAS!

Índice general

Índice de figuras

Índice de cuadros

1 Introducción al Proyecto

Este proyecto se enmarca dentro del proyecto Telcoblocks[1]. El objetivo del proyecto es proveer una plataforma abierta para el desarrollo y creación de servicios telco para la Red de Nueva Generación NGN empleando tecnologías JAIN SLEE [2], SIP Servlets [3]y Parlay [4], el proyecto se enmarca en el eje de actuación del Desarrollo del Sector TIC, fomentando la creación de nuevos servicios de valor añadido para la construcción del Internet del Futuro.

1.1. Motivación

La convergencia de las redes de telefonía tradicional e Internet [5] está proporcionando una nueva arquitectura para el desarrollo de servicios telco en las así llamadas Redes de Nueva Generación (*Next Generation Networks*; [6]). Una de las principales características de NGNs[7] es el desacoplamiento de redes y servicios, permitiendo que se ofrezca de forma separada y que evolucionen independientemente.

Por tanto, en las arquitecturas NGN no hay una separación clara entre las funciones de los servicios y las funciones de transporte, proporcionando interfaces abiertas entre ambos. NGN permite el aprovisionamiento de los servicios existentes y de nuevos servicios independientemente de la red y tipo de acceso empleado. Los principales retos para esta nueva arquitectura NGN son por una parte, proporcionar un entorno de ejecución de servicios tolerante a fallos y con calidad de operadora (*carrier grade*), que sea al menos tan robusto y seguro como la actual Red de Telefonía Conmutada (RTC); por otra parte, debe ofrecer un entorno flexible que permita desarrollar nuevos servicios de forma ágil, dado que la alta competitividad del entorno telco está demandando esta flexibilidad.

Para conseguir esta flexibilidad. Moyer [5]señala que las redes de nueva generación deben ser más abiertas que la RTC a los desarrolladores de servicios, ofreciendo APIs abiertas y facilitando bloques de servicios primitivos que los desarrolladores puedan reutilizar. En esta línea, el software de telecomunicaciones NGN ha seguido tres enfoques: JAIN SLEE[2], Parlay[4] y SIP Servlets[3].

El sector telco tiene unas características muy específicas para el desarrollo software. En definición de requisitos de usuario, las operadoras suelen definir ellas mismas los requisitos con técnicos altamente cualificados, y tienen reglas muy estrictas para la implantación de nuevos sistemas, con requisitos procedentes de diferentes departamentos, e incluyen requisitos de integración con sistemas de gestión de red, pruebas en maqueta, requisitos de alta disponibilidad y tolerancia a fallos así como acuerdos a nivel de servicio. Además, el tiempo de reacción (*time to market*) es un tema clave debido a la competitividad y dinamicidad del entorno. Por tanto, es un entorno donde es necesario definir un proceso de desarrollo que combine agilidad con aseguramiento de calidad del software.

Hasta ahora, el desarrollo de servicios se hacía con plataformas propietarias de los fabricantes de red, que limitaban la agilidad y la posibilidad de creación de nuevos servicios. Las plataformas NGN basadas en estándares abren nuevas oportunidades de negocio a toda la cadena de valor. Por una parte, las operadoras serán capaces de hacer servicios de forma más sencilla, sin tener que replicar el servicio para cada tipo de red (fija, móvil, ...), por otra, los integradores pueden generalizar sus soluciones independientemente de la infraestructura de cada operadora, favoreciendo la creación de productos generales que se adapten a las necesidades de cada cliente, que redunda en plazos y costes de desarrollo menores, y mayor robustez de las soluciones. Los integradores de red pueden ofrecer conectores de sus pilas de protocolos siempre que cumplan con los estándares de las plataformas de componentes. El empleo de estándares abiertos, en definitiva, crea un nuevo ecosistema de componentes y abre un nuevo mercado de componentes.

Sin embargo, el desarrollo de aplicaciones con nuevas tecnologías y, en concreto, con nuevos modelos altamente asíncronos supone un reto importante de investigación y supone un alto impacto en las herramientas y metodología de desarrollo.

1.2. Objetivos

Los principales objetivos de este proyecto son:

1. Definir una metodología ágil asistida por herramientas para guiar el desarrollo de nuevos servicios telco, centrados en VoIP(*Voice over Internet Protocol*), y favorecer la reducción de su tiempo de desarrollo. Las metodologías tradicionales diseñadas para desarrollos largos no son aplicables y los requisitos de calidad de operadora (en términos de procedimientos definidos) deben reconciliarse con las metodologías ágiles emergentes.
2. Diseñar un entorno para el despliegue de servicios VoIP empleando herramientas Open Source.
3. Desarrollar un un Framework Extensible de Desarrollo para las Aplicaciones Telco, proporcionando una biblioteca de eventos y servicios de VoIP.
4. Desarrollar un entorno de desarrollo (IDE) que facilite el desarrollo y despliegue de servicios de red, integrado en entornos de desarrollo populares como NetBeans o Eclipse.
5. Desarrollar una aplicación que permitan validar la metodología y herramientas desarrolladas.

1.3. Metodología y Fases del Proyecto

El proyecto se ha desarrollado según las fases definidas a continuación:

1. **Estudio preliminar**

   Estudio de los diferentes proyectos opensource existentes relacionados con SIP.

2. **Profundización en las diferentes soluciones Java**

   Estudio de las diferentes soluciones Java existentes para implementar diferentes servicios SIP.

3. **Análisis y diseño del sistema**

   Obtención de la especificación del sistema software a desarrollar, así como la arquitectura y diseño de los diferentes componentes que lo formen.

4. **Implementación**

   Ciclo de vida basado en diversas iteraciones. Integración de las diferentes partes que compondrán al sistema, dando lugar a un sistema compacto.

5. **Elaboración de la memoria**

   Documentación y código generado, así como las reflexiones realizadas.

2 Estado del Arte

2.1. Introducción

La sociedad de la información está viviendo una auténtica revolución en cuanto a servicios e infraestructuras se refiere. Los proveedores de Internet y telefonía móvil están desplegando lentamente sus redes de banda ancha, redes de nueva generación, abriendo un nuevo mundo de posibilidades para los servicios de internet. Esta revolución es consecuencia de la convergencia tecnológica. La convergencia tecnológica permite el acceso, en cualquier momento y lugar, a la información digitalizada, multimedia e interactiva, con capacidad de personalización.

Es importante conocer cuáles son las características de la convergencia tecnológica para poder comprender su importancia en el despliegue de nuevos servicios telco. Algunas de estas características son las siguientes:

- **Digitalización**: Permite que la información sea tratada sin discriminación, independientemente de la fuente que la genera.
- **Interactividad:** Un mensaje se relaciona con una serie de elementos previos**.**
- **Capacidad**: necesidad de caudal impuesta por el usuario, no por el proveedor.
- "**Always-on**": La conmutación de paquetes permite a los usuarios estar conectados siempre, sin ningún coste adicional.
- **Cobertura**: soluciona el problema de la ubicuidad.
- **Personalización**: aplicaciones ajustadas al perfil de usuario.
- **Usabilidad**: aplicaciones convergentes sencillas.

Es en este contexto donde nace la tecnología VoIP como sustitución a la red telefónica conmutada (RTC). El desarrollo de esta tecnología junto con el desarrollo

de los terminales e infraestructuras abre un abanico de posibilidades a la empresas desarrolladoras de servicios de VoIP.

En los apartadas sucesivos se presenta de forma concisa una descripción de la tecnología de VoIP y de uno de sus protocolos de señalización: SIP.

Se concluye el apartado, analizando brevemente las diferentes soluciones implementadas sobre tecnología Java que permiten el desarrollo de servicios VoIP.

2.2. Tecnología VoIP (Voice Over IP - Voz Sobre IP)

Se entiende por Voz sobre IP a toda comunicación establecida gracias al intercambio de paquetes digitalizados de voz sobre el protocolo IP. Esta tecnología presenta una seria de ventajas frente a la telefonía tradicional que se pueden englobar en los siguientes puntos:

- Reducción de costes tanto en el gasto que suponen las llamadas en sí, como en el precio del equipamiento necesario así como la simplificación de las tareas de Gestión y operación necesarias.
- Integración de servicios ya existentes en Internet: correo, videoconferencia, TV...
- Desarrollo de nuevos servicios: click-to-dial, conferencias multimedia, indicación de presencia...

Pero no son todo ventajas frente a la RTC. VoIP presenta también una serie de desventajas:

- El servicio telefónico tiene requisitos estrictos de calidad de sonido especialmente de retardo extremo a extremo. Según el retardo que se presente en la llamada, se puede clasificar el servicio como sigue[1]:

[1]Imagen extraída de los apuntes de CNM1.

Figura 2.1: Calidad Percibida según retardo en VoIP

- Conmutación de Paquetes (*Best Effort*) no pensada para este tipo de tráfico que introduce una serie de efectos que deben contrarrestarse:
  - Retardo extremo a extremo grande.
  - Posibilidad de pérdida y desorden de paquetes.
  - Consumo de capacidad extra.

  Todas estas desventajas han sido tenidas en cuenta a la hora de diseñar la tecnología, no se puede asegurar la calidad del servicio pero sí se han diseñado algunas soluciones para mejorar las prestaciones: supresión de silencios, compresión de cabeceras RTP/RTCP[8], etc.

La voz ha de codificarse para poder ser transmitida por la red IP. Para ello se hace uso de códecs que garanticen la codificación y compresión del audio o del video para su posterior decodificación y descompresión antes de poder generar un sonido o imagen utilizable. Según el códec utilizado en la transmisión, se utilizará más o menos ancho de banda. La cantidad de ancho de banda suele ser directamente proporcional a la calidad de los datos transmitidos. Los códecs más empleados en VoIP encontramos son: G.711[9], G.723.1 y G.729.

Para poder establecer una llamada de VoIP es necesario el intercambio de una serie de mensajes de señalización. En VoIP, existen dos protocolos, H323[10] y SIP[11], que se está imponiendo lentamente. H323 es el estándar propuesto por la ITU-T[12] que ha sido utilizado por la mayor parte de los proveedores de VoIP, así como por gran parte de las compañías que ofrecen este tipo de soluciones. Es utilizado sobretodo en equipos destinados a videoconferencia. Por su parte SIP (*Session Initiation Protocol*), es el protocolo de señalización de VoIP más utilizado en la actualidad, la gran variedad de funcionalidades que presenta hacen de SIP el estándar de facto.

2.3. Señalización en VoIP: SIP

SIP[11] es un protocolo desarrollado por el IETF con la intención de ser el estándar para la iniciación, modificación y finalización de sesiones interactivas de usuario donde intervienen elementos multimedia. Es un protocolo muy versátil. Se encuentra en la pila OSI en el nivel de Sesión, y el nivel de transporte sobre el que se usa puede ser tanto UDP como TCP o incluso SCTP.

La sintaxis de sus operaciones se asemeja a las de HTTP[13] y SMTP[14]. Esta similitud es natural ya que SIP fue diseñado para que la telefonía se vuelva un servicio más en Internet. En Noviembre del año 2000, SIP fue aceptado como el protocolo de señalización de 3GPP[15] y elemento permanente de la arquitectura IMS (*IP Multimedia Subsystem*).

El protocolo SIP se concentra en el establecimiento, modificación y terminación de las sesiones, se complementa, entre otros, con SDP[16], que describe el contenido multimedia de la sesión, por ejemplo qué direcciones IP, puertos y códecs se usarán durante la comunicación. También se complementa con RTP (*Real-time Transport Protocol*[8]), verdadero portador para el contenido de voz y video que intercambian los participantes en una sesión establecida por SIP.[2]

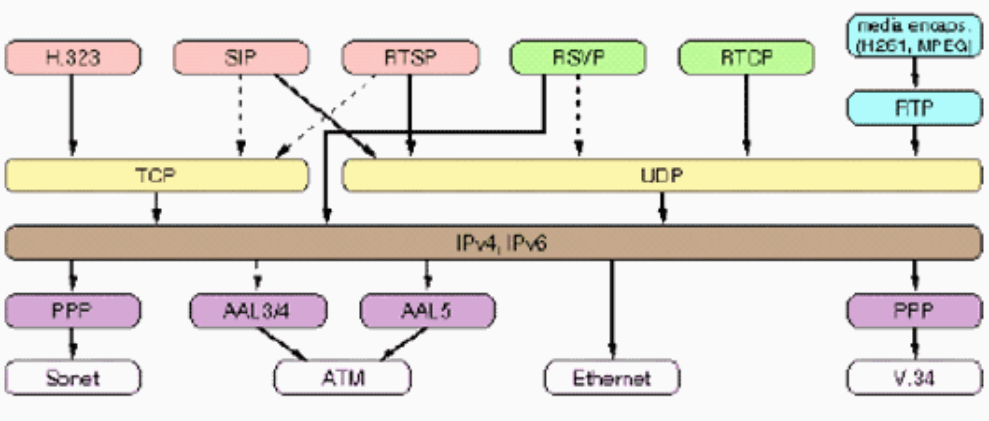

Figura 2.2: Arquitectura Protocolo SIP

[2]Imagen extraída de los apuntes de CNM1

2.3.1. Principales ventajas de SIP

SIP es un protocolo complementario a MGCP[17]. MGCP se encarga de la gestión de los elementos Hardware mientras que SIP lo hace del control de la sesión. Algunos de las ventajas que presenta SIP son las siguientes:

- **Simplicidad**: SIP es un protocolo muy simple. El desarrollo de Software para este protocolo es muy sencillo comparado con la telefonía tradiciconal. Debido a la similaridad de este protocolo con SMTP y HTTP es posible reutilizar código.
- **Extensibilidad**: SIP tiene una serie bastante amplia de extensiones así como algunas funciones de compatibilidad.
- **Modularidad**: Desde el principio SIP fue diseñado orientado a la modularidad. Una de las principales características es su independencia con respecto a otros protocolos.
- **Escalabilidad**: SIP ofrece dos ventajas en cuanto a escalabilidad.
  - Procesamiento de servidores: SIP puede ser con o sin estado.
  - Multiconferencias: La coordinación de conferencias puede ser totalmente distribuída o centralizada.
- **Integración**: SIP puede integrarse con Web, email, aplicaciones de streaming y otros protocolos.
- **Interoperabilidad**: SIP está basado en una RFC abierta, por este motivo SIP puede ofrecer interoperabilidad entre diferentes plataformas.

2.3.2. Funcionamiento

El funcionamiento del protocolo se basa en el intercambio de mensajes entre cliente y servidor, no necesita transporte fiable por lo que suele ir sobre UDP normalmente. El cliente realiza una serie de peticiones que son procesadas por el servidor.

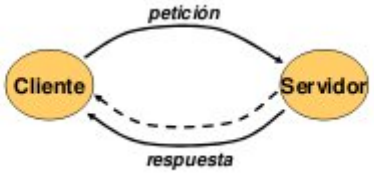

Figura 2.3: Funcionamiento SIP

Según el tipo de funciones que realice el servidor podemos distinguir los siguientes tipos de servidores SIP:

- **Agentes de usuario**(UAC): pueden actuar como cliente y como servidor, pueden comunicarse directamente o a través de servidores (por ej. teléfono físico SIP, programa en un PC, pasarela).
- **Servidores intermediarios**(UAS): cuya función principal es la localización del destinatario y el encaminamiento de la llamada, pasando por uno o varios servidores.

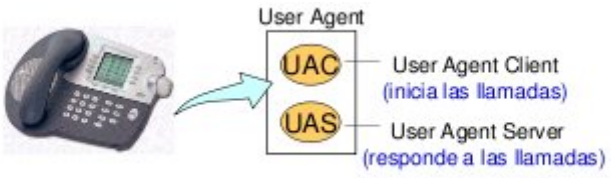

Figura 2.4: Agentes SIP

2.3.3. Mensajes SIP

SIP se basa en la premisa simple: intercambio de mensajes entre cliente/servidor. Los clientes o terminales son identificados por direcciones que son únicas. Estas direcciones tienen un formato similar al que tienen las direcciones de correo electrónico:

usuario@dominio. Cuando se habla de direcciones SIP hay que tener en cuenta las siguientes anotaciones:

- Las direcciones SIP son siempre del estilo: usuario@host.
- Usuario: nombre, teléfono, número, etc.
- Host: dominio, dirección de red en formato numérico.

Los usuarios o clientes se registran en servidores de registro SIP aportándoles a éstos información sobre su localización. Los mensajes SIP son usados para la conexión y el control de las llamadas. Hay dos tipos de mensajes SIP: las peticiones y las respuestas.

- **INVITE**: Es usado por un usuario para iniciar una llamada. La cabecera de este mensaje contiene:
  - Dirección de la persona que llama y a la que va dirigida la llamada.
  - Tema de la llamada.
  - Prioridad de la llamada / Peticiones de enrutado de la llamada / Preferencias del llamante.
  - Características deseables de la respuesta.
- **BYE**: Mensaje empleado para terminar la conexión existente entre dos usuarios.
- **REGISTER**: Aporta la información de localización a un servidor SIP, permitiendo al usuario decir al servidor cómo debe éste mapear una dirección entrante en la dirección que recibirá el usuario.
- **ACK**: Confirma que un mensaje ha llegado a su destino de forma correcta.
- **CANCEL**: Cancela peticiones pendientes.
- **OPTIONS**: Solicita información sobre lo que puede ser realizado en un extremo de la llamada, esto dependerá del tipo de terminal que esté cursando la llamada: desde un terminal analógico hasta un terminal IP Multimedia.

Además gracias a las extensiones de SIP se pueden describir algunos métodos más:

- **SUBSCRIBE**: Se usa para indicar interés en saber cuándo un usuario estará disponible.

- **NOTIFY**: Sirve para informar de los cambios de estado de un usuario.

Los mensajes de respuesta se dividen en provisionales y finales y se distinguen por un código de estado. Las respuestas provisional (códigos 1xx) se utilizan para indicar que la petición fue recibida y se está procesando. Este tipo de respuesta permite, por ejemplo, saber que un teléfono esta sonando. No hay mecanismos de la fiabilidad en SIP básico para respuestas provisionales.

Las respuestas finales (códigos 2xx-6xx) indican una conclusión a las peticiones de los UAC. Se dividen en: éxito (2xx), redirigir (3xx) y diversos tipos de las fallos (4xx-6xx). En la siguiente tabla se identifican los códigos de las respuestas SIP.[3]

CÓDIGOS DE RESPUESTA EN SIP

Código	Descripción
1xx	Provisional. Petición recibida, comienzo de procesamiento de la petición.
2xx	Éxito. La petición fue recibida, entendida y aceptada.
3xx	Redirección. Para completar la petición hay que realizar una nueva acción.
4xx	Error del cliente. La petición contiene fallos en la sintaxis o no puede ser realizada.
5xx	Error en el servidor. Fallo al procesar una petición supuestamente correcta.
6xx	Fallo global. La petición no puede atenderla ningún servidor.

Cuadro 2.1: Códigos de Respuesta en SIP

2.3.4. Ejemplos de Flujos SIP

Para concluir la descripción de SIP, se adjunta la descripción de algunas de las trazas más comunes que pueden ser capturadas en la arquitectura de despliegue

[3]PFC Juan María Javaloyes: "DESARROLLO DE UN ASISTENTE J2ME PARA ENTORNO UNIVERSITARIO CON MENSAJERÍA SOBRE EL PROTOCOLO SIP/SIMPLE "

de aplicaciones de la plataforma del proyecto. Estas capturas han sido realizadas con el analizador de protocolos Wireshark[18]. Los clientes que toman parte en las comunicaciones son tanto softphones(EKIGA,[19]) como teléfonos IP físicos, como servidor SIP se ha empleado un servidor OpenSIPS[20] cuya descripción se incluirá en capítulos posteriores.

Registro

La siguiente secuencia de mensajes SIP muestra cuál es el proceso de intercambio de mensajes en el registro de un usuario en un servidor SIP(OpenSIPS). [4]

Figura 2.5: Cronograma correspondiente al registro de un cliente

Llamada aceptada con uso de un proxy

Un cliente SIP intenta cursar una llamada con otro cliente, enviando la petición INVITE correspondiente al Servidor SIP de escenario. El servidor SIP procesa la

[4]Cronograma obtenido con Wireshark

petición, respondiendo al cliente que originó la petición(Respuesta "*Giving a try*") y reenviando la petición al cliente destino de la llamada. Una vez procesada la petición por el cliente SIP llamado, se responde al servidor SIP("Trying" y a continuación "Ringing") anunciando que está sonando, momento en el que el cliente llamante comenzará a escuchar los tonos de la llamada.

Cuando el cliente llamado contesta la llamada se transmite un mensaje que lo indica y lleva información sobre la sesión ("OK SDP"), este mensaje es enviado al proxy SIP que lo reenviará al cliente originador de la llamada. El llamante responderá con un mensaje "ACK" que llegará al llamado a través del servidor. A partir de este momento tiene lugar la conversación, transmitida mediante RTP.

Los paquetes RTP se transmiten directamente entre clientes, sin intervención del servidor SIP. Los paquetes RTP enviados antes del "ACK" no llegaron a su destino, ya que aún no se había establecido la sesión. Cuando un cliente (el llamado en este caso) decide finalizar la llamada, se envía al otro una petición "BYE" y este contesta con un mensaje "OK", ambos reenviados a su destinatario por el Servidor SIP.[5]

[5]Cronograma obtenido con Wireshark

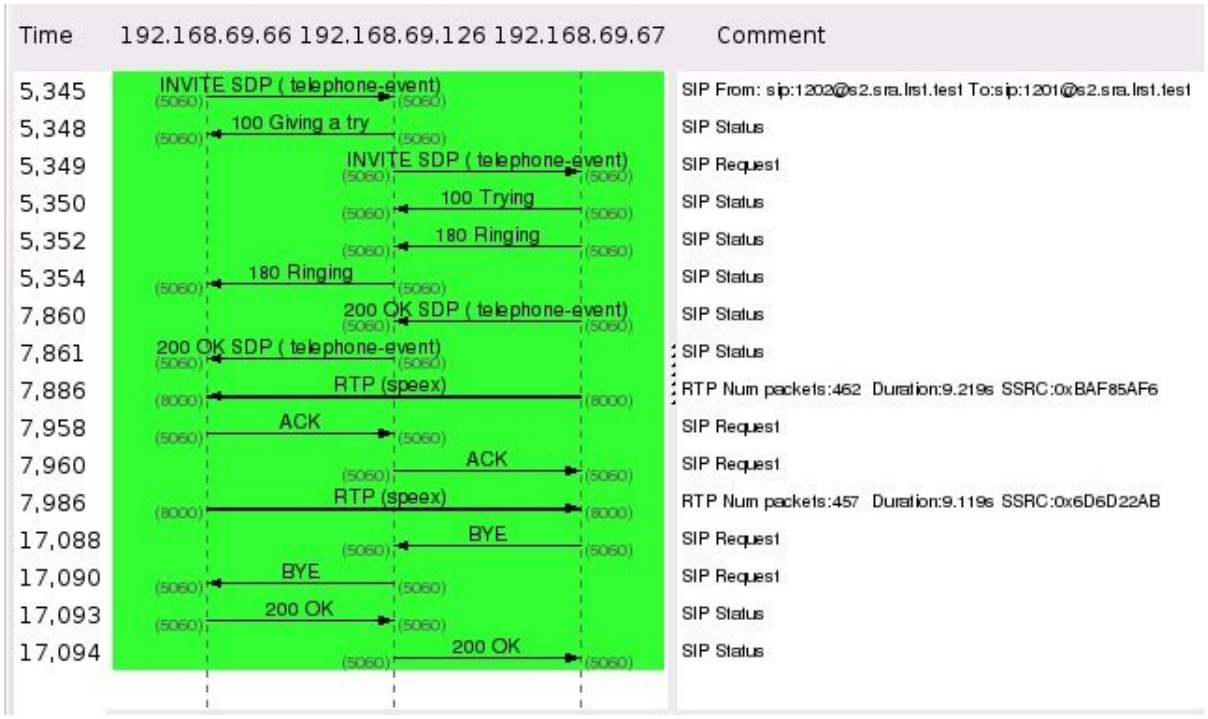

Figura 2.6: Cronograma correspondiente a una llamada local aceptada

Como se puede observar, la función del servidor SIP es meramente la de interlocutor en la conversación que tiene lugar entre los clientes, esta función va a ser la desempeñada en la mayoría de las aplicaciones expuestas en el presente proyecto por el servidor de aplicaciones SIP.

Llamada cancelada

Hasta el mensaje "Ringing" que se transmite del cliente llamado al llamante el proceso es idéntico al de la llamada aceptada descrita. Después, el llamado rechaza la llamada y se envía un mensaje "Decline" al llamante, que contesta con un "ACK". Ambos mensajes son reenviados por el servidor SIP. [6]

[6]Cronograma obtenido con Wireshark

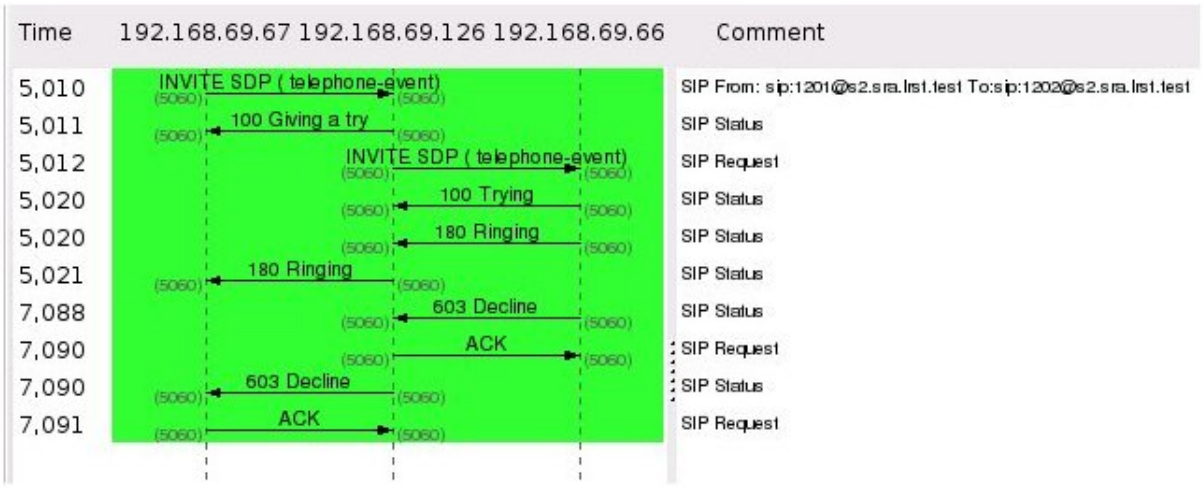

Figura 2.7: Cronograma correspondiente a una llamada local rechazada

2.4. Tecnologías JAVA para VoIP

En los últimos años, Java ha pasado de ser un lenguaje a convertirse en una plataforma. El concepto de Plataforma Java o Tecnología Java es el fruto de la política de desarrollo de Java, a través de la normalización de diferentes especificaciones con el fin de que puedan acometerse en Java desarrollos de sistemas muy diferentes, pero que todos los desarrollos sean fácilmente integrables porque emplean interfaces estandarizadas.

El mecanismo de aprobación de estos estándares es mediante JSR (*Java Specification Request*), que son propuestas y revisadas públicamente y desarrolladas por un grupo de expertos coordinadas por varias empresas líderes. De esta forma, los estándares surgen de las necesidades de las empresas y de los usuarios.

Java ha llevado este concepto de desarrollo de bibliotecas y entornos específicos. Actualmente están especificadas tres plataformas principales:

- J2SE (*Java 2 Standard Edition*). Plataforma para aplicaciones de usuario, ejecutadas en un "ordenador personal".
- J2ME (*Java 2 Micro Edition*). Plataforma para el desarrollo de aplicaciones en dispositivos pequeños (PDAs, teléfonos, vehículos, electrodomésticos, TV, ...).
- J2EE (*Java 2 Enterprise Edition*). Plataforma para el desarrollo de aplicaciones de una empresa, ejecutadas en un servidor de aplicaciones.

De forma ortogonal a las plataformas, las normas (JSR) de Java se clasifican, parcialmente, por tecnologías. Actualmente se consideran las tecnologías:

- JAIN (*Java API for Intelligent Network*). Normas relacionadas con Red Inteligente.
- OSS (*Operation Support System*). Normas relacionadas con sistemas de soporte en operaciones.
- XML (*eXtended Markup Language*). Normas relacionadas con la tecnología XML.
- JAIN SLEE (*Service Logic Execution Environment*) supone la convergencia de J2EE y JAIN. En los siguientes apartados vamos a analizar por qué ha surgido SLEE y en qué consiste.

2.4.1. JAIN SLEE

JAIN SLEE[2] es un modelo de componentes definido para un servidor de aplicaciones diseñado específicamente para aplicaciones telco. Está concebido como una plataforma de procesamiento de eventos de altas prestaciones y tolerante a fallos. Frente a las aplicaciones de empresa y web que son síncronas e intensivas en datos por naturaleza, y pueden ser modeladas e implementadas de forma adecuada con la tecnología JEE (*Java Enterprise Edition*), SLEE está dirigido a aplicaciones asíncronas, como son las aplicaciones telco, y a procesar eventos de red combinando múltiples protocolos. En la actualidad Jain SLEE es el estándar Java para entornos de ejecución de lógicas de servicio (*Service Logic Execution Environment*).

La principal ventaja de Jain SLEE es la estandarización del desarrollo y ejecución de servicios, de modo que se cubran los diferentes niveles y, muy especialmente, la capa de acceso a los elementos de telecomunicaciones. A pesar de la estandarización en el acceso a las capacidades de red ofrecida por protocolos como INAP o CAP, los cuales permiten la interconexión de redes, la complejidad de la implementación de servicios de forma directa y los requisitos de las telcos de implementación (rendimiento, distribución, confiabilidad, etc.) hacen del desarrollo de servicios una actividad muy compleja que cuenta con tan sólo un reducido grupo de profesionales con la competencia adecuada. Jain SLEE, por el contrario, ofrece un alto nivel de abstracción para el acceso a las capacidades de red, simplificando en gran medida la complejidad existente y, al mismo tiempo, ofreciendo Java como lenguaje de implementación. Paralelamente, Jain SLEE incorpora los conceptos tradicionales de la arquitectura Java (reusabilidad de componentes, facilidades de administración y concurrencia, distribución, etc.).

Sobre esta base, el desarrollo y despliegue de servicios se realizaría en un tiempo menor y por un mayor volumen de profesionales, con las consiguientes ventajas para el operador y los clientes finales: mayor número de servicios y con un ciclo de lanzamiento más corto y dinámico.

En cuanto a los servicios que se pueden construir y desplegar sobre Jain SLEE, se pueden citar como ejemplos los servicios tradicionales de inteligencia de red como VPN, traducción de números o least cost routing, servicios de localización, servicios de mensajería, como cobro en tiempo real de MMS y servicios basados en señalización SIP, así como cualquier combinación de los anteriores.

Jain SLEE, como paradigma de los entornos de nueva generación, rompe con la tradicional distinción entre servicios de inteligencia de red y servicios IP. El concepto de adaptadores de recursos (RA) permite la integración entre diferentes tecnologías en servicios innovadores y de nueva generación, así como el despliegue de servicios tradicionales basados en señalización, como la gestión de enrutamiento de llamadas. No existe un único campo de servicios adecuado para Jain SLEE; por el contrario, una de las principales ventajas es su gran alcance, el cual es además ampliable mediante la incorporación de adaptadores de nuevos protocolos. Otra de las ventajas es su definición abierta, que elimina las barreras tradicionales de integraciones

propietarias.

2.4.2. SIP Servlets

SIP Servlets[3] proporcionan un modelo de desarrollo específico del protocolo SIP, dado su papel fundamental en las nuevas arquitecturas IMS (*IP Multimedia Subsystem*). Es una tecnología alternativa a JAIN SLEE, que ofrece un modelo de desarrollo más sencillo, aunque no permite manejar otros protocolos diferentes de SIP y no es transaccional.

Tanto JAIN SLEE como SIP Servlets están ofreciendo sus propios Entornos de Creación de Servicio, y se desarrollan con conectores a la red.

Los SIP Servlets están especificados en la JSR 116[21] y pretenden implementar aplicaciones SIP siguiendo el modelo de servlets. Como los HTTP Servlets[22], extienden la clase genérica servlet. Sin embargo, dadas las diferencias entre SIP y HTTP, los SIP servlets, son capaces de:

- Iniciar peticiones SIP y recibir respuestas SIP.
- Crear respuestas SIP.
- Redirigir peticiones (proxy).

Cuando llega un mensaje SIP, el contenedor verifica si algún servlet cumple las reglas que el mensaje SIP lleva. Si es así, se lanza ese servlet y se lanzan los métodos de creación de respuestas o peticiones. El resultado de una petición no tiene por qué ser una respuesta, como en HTTP, sino que puede ser tanto una respuesta como la creación de una nueva petición SIP (cuando el servlet actúa como proxy, por ejemplo).

Los SIP servlets en su descriptor de despliegue, indican al contenedor, qué eventos y con qué parámetros (reglas) se activan. Los servidores que soporten este contenedor, es recomendable que entiendan HTTP y puedan integrarse en la plataforma J2EE.

El funcionamiento de un SIP Servlet queda reflejado en el siguiente diagrama.

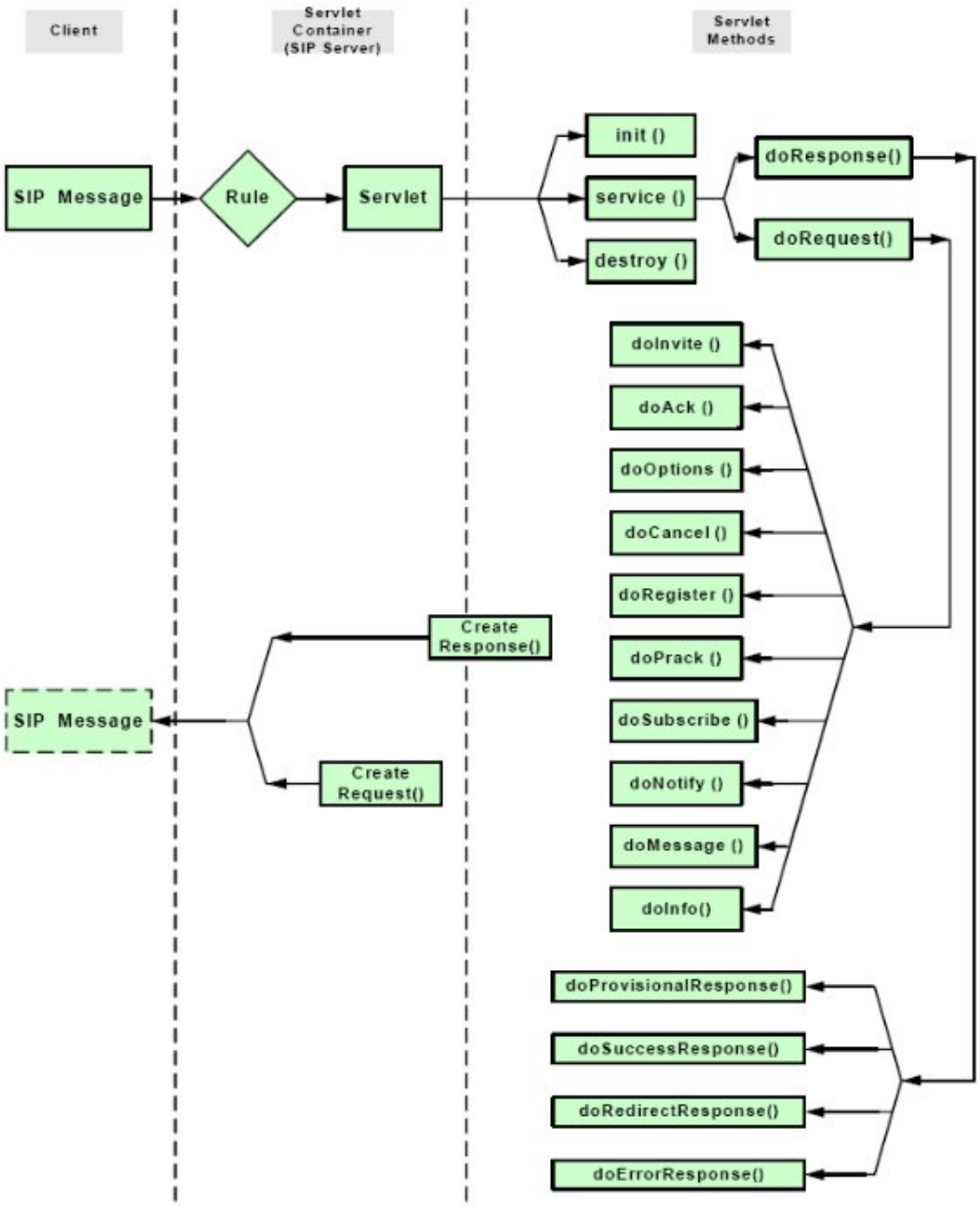

Figura 2.8: Esquema Funcionamiento SIP Servlets

La actual especificación de SIP Servlets es muy parecida a la de los HTTP Servlet y define también una API y un formato de fichero para su despliegue. Como es lógico, implementa SIP (RFC 3261), pero también algunas de sus extensiones más comunes: la dedicada a notificación de eventos (RFC 3265) y la dedicada a mensajería instantánea (RFC 3428). Los contenedores de SIP Servlet se encargan de manejar todos los recursos necesarios para establecer comunicación con clientes, pueden integrarse dentro de servidores J2EE que además pueden proporcionar otros servicios y pueden conectarse a otras APIS:

- JNDI (Java Naming and Directory Interface) para accesos a directorio.
- JDBC (Java Database Connectivity) para acceder a bases de datos por SQL.
- JMF (Java Media Framework) para manejar elementos multimedia.
- JavaMail para usar características de correo electrónico.

De esta forma, el uso de servlets en SIP es muy útil para converger con las tecnologías ya establecidas y proporcionar nuevas funcionalidades.[7]

2.5. Conclusiones

Ante todo se debe aclarar, que no sería en el actualidad tan útil VoIP sino hubiese convergencia, es decir; si no se pudiese establecer una comunicación entre un terminal tradicional que se comunica mediante RTC y otro terminal que lo hace empleando la tecnología VoIP.

La tecnología de VoIP está lo suficiente madura para ser aplicada en entornos profesionales; y es así como ha estado sucediendo en los últimos años. La aparición de diferentes proyectos opensource,estudiados en el siguiente capítulo, han favorecido a la rápida aceptación de la VoIP como solución a las comunicaciones en ámbitos variopintos.

[7]Desarrollo de un asistente J2ME para entorno universitario con mensajería sobre el protocolo SIP/SIMPLE. PFC de Juan María Javaloyes Sáez.

La tecnología VoIP nace justificada por la convergencia de las TIC, saciando la necesidad de desarrollar aplicaciones ajustadas al perfil de usuario y aplicaciones convergentes sencillas gracias en gran parte a la tecnología Java, que permite el desarrollo rápido de servicios debido a la reutilización de componentes ya existentes.

3 Plataforma de Despliegue de Aplicaciones

3.1. Introducción

La gran aceptación que ha tenido la tecnología VoIP como solución a las comunicaciones corporativas de las empresas y el mayor uso de un protocolo abierto de señalización, como es SIP, que permite implementar gran variedad de aplicaciones, ha favorecido al nacimiento de algunos proyectos de software abierto que permiten implementar la mayoría de las características que ofrecen algunas de las PBX comerciales (Avaya, Cisco...) que desde el nacimiento de esta tecnología han sido utilizadas por las grandes y medianas empresas.

En la actualidad existen diferentes soluciones de software abierto para del despliegue de una infraestructura VoIP, como no, sobre el protocolo de señalización SIP.

En este capítulo se pretende investigar en estas soluciones para crear una arquitectura integrada que permita generar un entorno de pruebas y despliegue de servicios de forma sencilla utilizando diferentes elementos.

El entorno básico de despliegue de servicios VoIP se ilustra en la figura 3.1 y consta de los siguientes elementos, que serán descritos en la siguiente sección:

- Proxy SIP - Servidor de Registro (3.2)
- Centralita VoIP (PBX) (3.3)
- Servidor Multimedia (3.4)
- Servidor Telco de Aplicaciones(3.5)

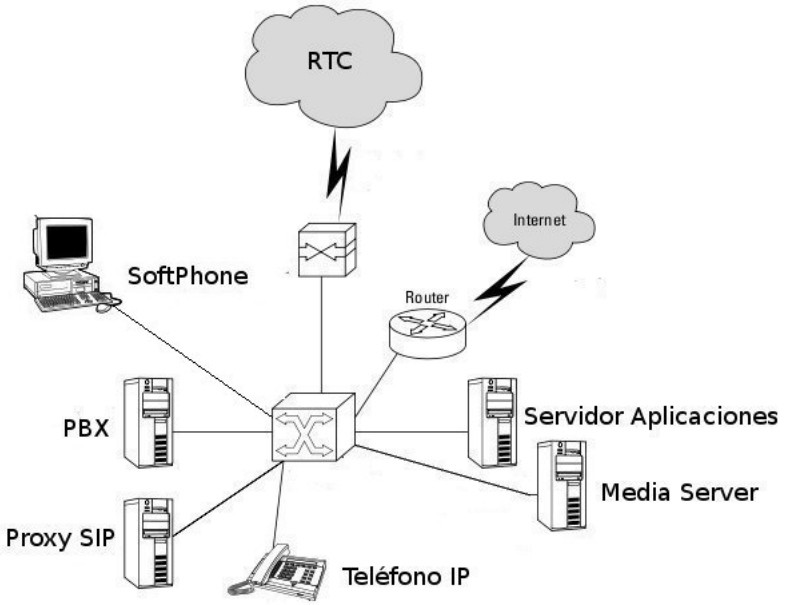

Figura 3.1: Plataforma de Despliegue de Servicios

3.2. Proxy SIP - Servidor de Registro

Un servidor de registro SIP es un servidor que permite la autenticación de usuarios en él mediante el protocolo de señalización SIP. Cada usuario tiene una dirección lógica que es invariable respecto de la ubicación física del usuario. Una dirección lógica del protocolo SIP es de la forma usuario@dominio, es decir; tiene la misma forma que una dirección de correo electrónico. La dirección física (denominada "dirección de contacto") es dependiente del lugar en donde el usuario está conectado (de su dirección IP).

Cuando un usuario inicializa su terminal (por ejemplo conectando su teléfono o softphone SIP), el agente de usuario SIP que reside en dicho terminal envía una petición con el método REGISTER a un Servidor de Registro, informando a qué dirección física debe asociarse la dirección lógica del usuario. El servidor de registro

realiza entonces dicha asociación. Esta asociación tiene un período de vigencia y si no es renovada, caduca. También puede terminarse mediante un desregistro. La forma en que dicha asociación es almacenada en la red no es determinada por el protocolo SIP, pero es vital que los elementos de la red SIP accedan a dicha información.

Por su parte, un proxy SIP es un servidor que actúa como interlocutor en un diálogo SIP, puede actuar de dos maneras:

- Como Proxy, encaminando el mensaje hacia destino.
- Como servidor de Redirección generando una respuesta que indica al originante la dirección del destino o de otro servidor que lo acerque al destino. La principal diferencia es que el servidor proxy queda formando parte del camino entre los extremos de la comunicación, mientras que el servidor de redirección una vez que indica al llamante cómo encaminar el mensaje ya no interviene más.

De entre todos los proyectos de código abierto que implementan la funcionalidad de un servidor de registro SIP, se han analizado los siguientes: SER (SIP Express Router)[23], OpenSER[20] y los dos proyectos que de él derivan llamados OpenSIPS[24] y Kamailio[25].

3.2.1. SER (SIP Express Router)

SER[23] es un servidor SIP que se distribuye como software libre bajo licencia GPL. Este servidor puede actuar como servidor de registro, proxy o servidor de redirección. Asi mismo, SER puede ser configurado para desempeñar algunas tareas más especificas como reparto de carga de la red o puede actuar también como front-end de algunos servidores de aplicaciones.

SER está diseñado para sistemas de gran escala donde probablemente cualquier otra solución no podría ser empleada. SER podría ser capaz de manejar hasta 1000 llamadas por segundo . Así mismo, SER como servidor de registro puede registrar a más de 100 000 agentes de usuario. Su modelo de datos flexible permite a SER ser empleado con otras herramientas y existentes infraestructuras.

Un escenario típico donde SER suele ser la solución empleada suele ser en empresas cuyos teléfonos se encuentran detrás de una red NAT, asi como salida a la RTC. Un

diagrama típico de este tipo de redes sería el siguiente[1].

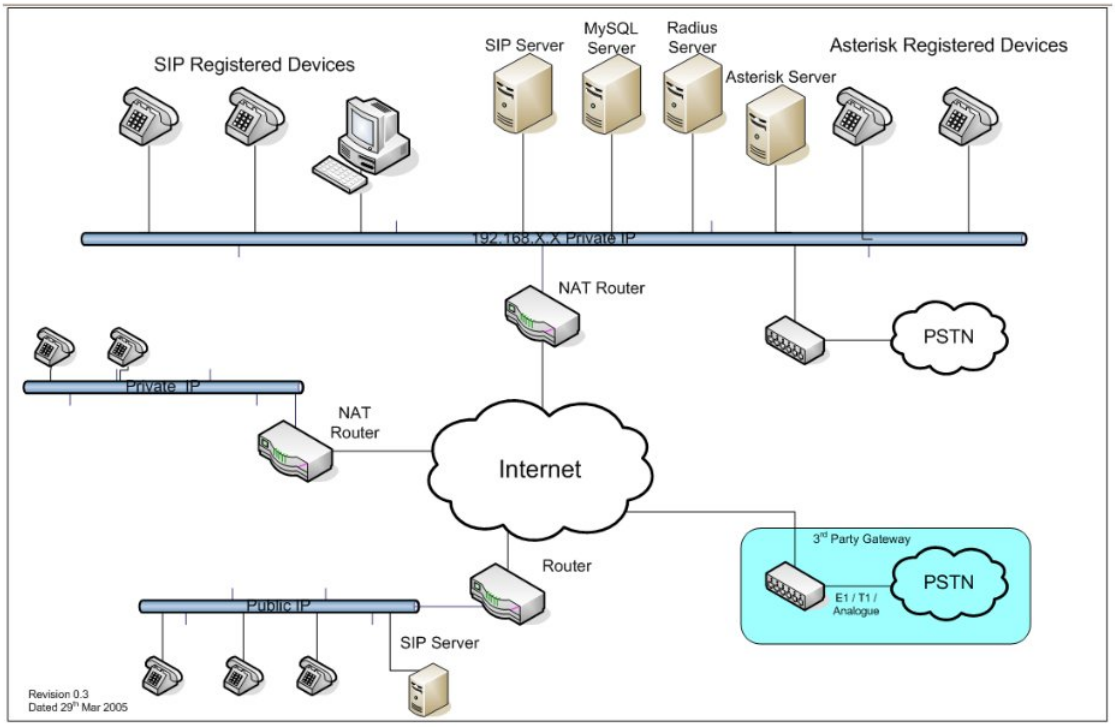

Figura 3.2: Arquitectura SER

Además de todas estas funcionalidades, SER posee otros servicios como procesamiento de CDR(*Call Detailed Records*). SER está incluido en numerosos sistemas operativos actualmente como: Debian, FreeBSD, Gentoo, NetBSD, OpenBSD, OpenSolaris o OpenSuse. Ha sido usado desde 2002 para diferentes propósitos, es común en industria ya que es empleado por grandes ISP y por algunas universidades. Algunas de las características más importantes de SER son su flexibilidad, su interoperatibilidad y que su funcionamiento no se ve afectado por la carga de la red.

Las características de SER son especificadas en detalle en la sección 3.2.5. En 2004 dos de los principales desarrolladores abandonaron el proyecto y fundaron un nuevo

[1]Imagen extraída del documento de diseño de Arquitectura de SER. URL: http://www.iptel.org

proyecto gratuito y opensource llamado OpenSER. Actualmente SER es gratuito y opensource. Pertenece a la compañía TEKELEC.

3.2.2. OpenSER

En octubre de 2005 nace oficialmente el proyecto OpenSER[20] con el lanzamiento de la primera versión 1.0.0.0. OpenSER nace como un proyecto fork de SER debido a ciertas diferencias a la hora de hacia dónde debía orientarse el proyecto. Desde el principio OpenSER recibió mucho más apoyo que SER y gracias a una activa lista de correo bastante activa, tanto por parte de los usuarios como de los desarrolladores, se fueron lanzando diferentes versiones rápidamente que supusieron muchas mejoras con respecto a lo que el proyecto SER ofrecía.

Un esquema típico de arquitectura donde OpenSER actúa como servidor de registro SIP, sería igual a la que aparece en la sección anterior. La primera versión de OpenSER ya incluía, además de lo ya mencionado en las características de SER, una mejor seguridad con TLS integrado y que favorecía el establecimiento de un entorno seguro de VoIP.

En Julio de 2008 se volvió a decidir por parte de los desarrolladores en dividir el proyecto en dos: OpenSIPS y Kamailio. Las nuevas funcionalidades implementadas por las posteriores versiones de OpenSER asi como las de estos dos proyectos son comentadas en las siguientes secciones (3.2.3 y 3.2.4).

3.2.3. Kamailio

Kamailio[25] es una de las dos ramificaciones del proyecto OpenSER. El proyecto y su comunidad habían crecido considerablemente en los últimos 3 años. Debido al tamaño de la comunidad y a la diferencia de intereses que había en ella han surgido, se decidió dividir la comunidad de en dos proyectos relacionados con OpenSER para continuar con el proyecto siguiendo dos caminos distintos; por una parte Kamailio con unos intereses más afines a los que tuvo en su momento la comunidad de SER(comunidad pequeña para mejor entendimiento entre desarrolladores) y por

otra parte OpenSIPS que mantiene el espíritu abierto de OpenSER y la mayor parte de la comunidad. El nombre Kamailio intenta representar lo que va a significar el proyecto, la solución de Servidor SIP por excelencia. Kamailio es una palabra que proviene del Hawaiano y que significa hablar, conversar.

Este proyecto empieza a partir de la versión 1.4 de OpenSER. Las características que ofrece Kamailio coinciden con las que ofrecía la última versión de Openser y que son especificadas en la sección. A parte de esas características otro factor a destacar tanto del proyecto OpenSIPS como de Kamailio es la escalabilidad:

- Kamailio (OpenSER) puede ser utilizado en máquinas con unos requisitos no demasiado exigentes de tal manera que pueda manejar cientos de llamadas por segundo.
- Reparto de carga, de tal manera que Kamailio puede establer más de 5000 llamadas por segundo.
- En sistemas con 4GB de memoria RAM, Kamailio puede tener registrados incluso más de 30000 usuarios.
- Al sistema se le pueden añadir facilmente más servidores Kamailio de tal manera que resulta ser un sistema fácilmente escalable.
- Kamailio (OpenSER) puede ser empleado para distribución geográfica de plataformas de VoIP.
- Es sencillo añadir redundancia.

3.2.4. OpenSIPS

OpenSIPS[24] es el otro proyecto que nació a partir de OpenSER al mismo tiempo que Kamailio. OpenSIPS es más que un router/proxy SIP ya que incluye ciertas funcionalidades en el nivel de aplicación. OpenSIPS, como servidor SIP, es uno de los núcleos de cualquier solución de VoIP basada en SIP.

Este proyecto tiene un mecanismo de enrutado bastante configurable, de tal manera, que permite unificar voz, video, IM y servicios de presencia de una forma bastante eficiente gracias a su arquitectura modular. OpenSIPS es uno de los servidores SIP

más rápidos del mercado, de tal manera que se confirma que puede ser una de las soluciones a nível de empresa e incluso es empleado por algún ISP. Las características que ofrece OpenSIPS son especificadas en la sección 3.2.5

3.2.5. Comparativas SIP Servers

Comparativa SER vs OpenSER

OpenSER presenta una lista de correo mucho más activa que SER además de ir lanzando nuevas versiones cada menos tiempo, lo cual hizo en un principio a este proyecto mucho más interesante para ser empleado en la infraestructura de despliegue de aplicaciones, además de esta razón existen ciertas diferencias entre ambos proyectos como puede verse en la siguiente tabla.

Característica	**SER**	**OpenSER**
Implementación de la RFC 3261 SIP	√	√
Modulos plug&play	√	√
Soporte a diferentes protocolos de transporte UDP/TCP/TLS	√	√
Soporte a SRV y NAPTR DNS		√
SRV DNS failover, IP Blacklists		√
Lenguaje de scripts en los ficheros de configuración	√	√
Interfaz de gestión del servidor	√	√
autentication, autoorización y accounting (AAA) RADIUS y DIAMETER	√	√
Autenticación por IP		√
Soporte de NAT traversal para tráfico SIP y RTP	√	√
Interfaz para aplicaciones de Java SIP Servlet		√
Load balancing	√	√
Elección de rutas con menos coste	√	√
Gateway para sms o xmpp	√	√
Diferente soporte a base de datos - MySQL, PostgreSQL	√	√
Conexión directa con pasarelas a la RTC	√	√

Cuadro 3.1: Comparativa SER vs OpenSER

Se puede apreciar que OpenSER es ligeramente superior en cuanto a características se refiere que el proyecto SER. Además, OpenSER ofrece mejores prestaciones en cuanto al rendimiento y uso de CPU que realizan ambos proyectos[26](tests realizados sobre Intel Xeon 5140 2.33 GHz CPU).

Utilización de CPU

La siguiente tabla representa el porcentaje de utilización de CPU frente al número de llamadas por segundo[2].

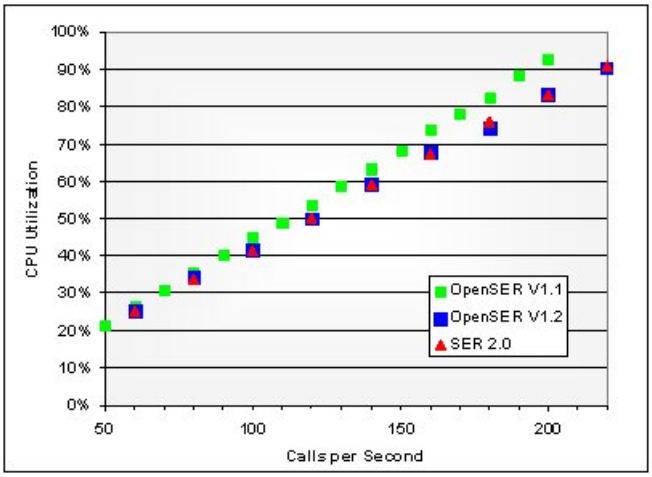

Figura 3.3: Comparativa SER y OpenSER - Uso de la CPU

Como se puede apreciar las versiones sucesivas de OpenSER han ido alcanzando las características del proyecto SER. La versión actual presenta mejores prestaciones en cuanto a utilización de CPU, lo que demuestra que OpenSER escala mejor que SER.

[2]Imágenes extraídas de "Transnexus, Performance results for openser and sip express router", disponible en http://www.transnexus.com/whiterep.

Utilización de Memoria

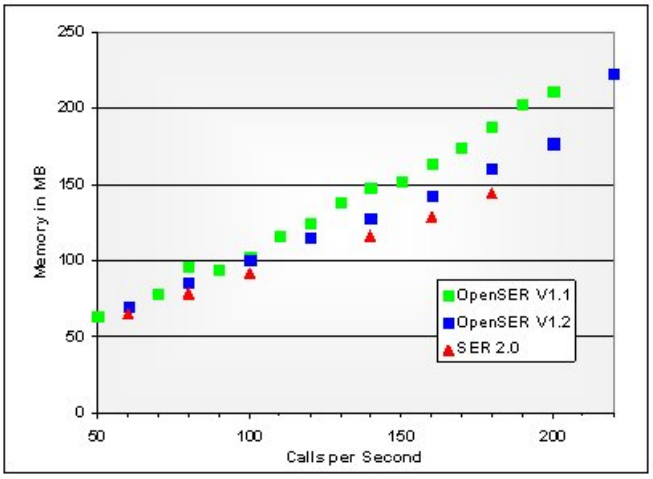

Figura 3.4: Comparativa SER y OpenSER - Utilización de Memoria

En el comportamiento de utilización de memoria, pese a no ser uno de los mayores problemas en la utilización de estas soluciones, SER 2.0 originalmente seguía requiriendo la menor cantidad de memoria para un mismo valor de llamadas por segundo.

Retardo de Llamada (PDD)

En la siguiente gráfica podemos apreciar la gran mejora que representa en este sentido OpenSER 1.2 frente a SER.

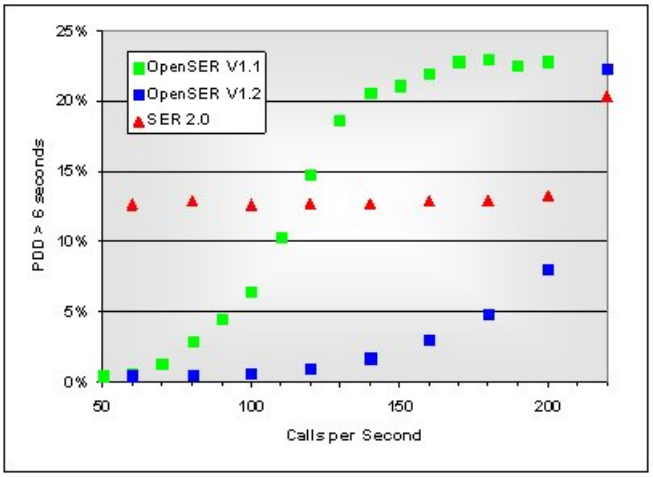

Figura 3.5: Comparativa SER - OpenSER - Retardo de Llamada

Llamadas con Éxito

La siguiente gráfica reprenta el porcentaje de llamadas que fueron realizadas con éxito cuando la ocupación de la CPU era menor del 90 %.

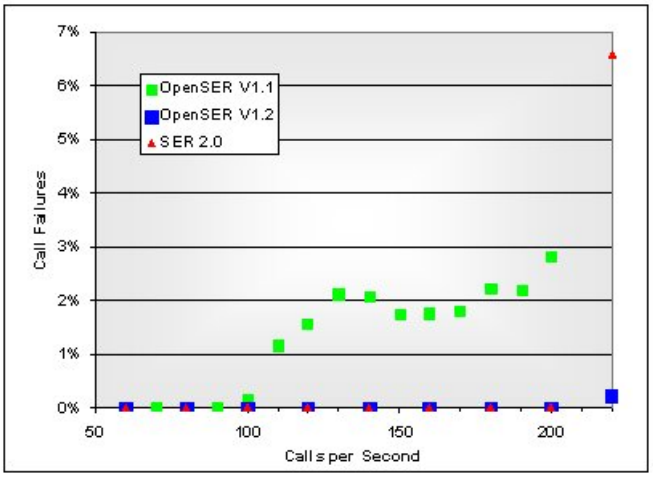

Figura 3.6: Comparativa SER-OpenSER - Llamadas realizadas con éxito

Comparativa OpenSIPS – Kamailio

Una vez decidido que OpenSER es más adecuado que SER para los intereses del presente proyecto, es necesario comparar los dos proyectos que continúan con el proyecto OpenSER de tal manera que podamos establecer cuál cumple mejor los requisitios para ser empleado en la arquitectura de VoIP del proyecto.

Característica	Kamailio	OpenSIPS
Implementación de la RFC 3261 SIP	√	√
Modulos plug&play	√	√
Soporte a diferentes protocolos de transporte UDP/TCP/TLS	√	√
Soporte a SRV y NAPTR DNS	√	√
SRV DNS failover, IP Blacklists	√	√
Lenguaje de scripts en los ficheros de configuración	√	√
Interfaz de gestión del servidor	√	√
autentication, autoorización y accounting (AAA) RADIUS y DIAMETER	√	√
Autenticación por IP	√	√
Soporte de NAT traversal para tráfico SIP y RTP	√	√
Interfaz para aplicaciones de Java SIP Servlet	√	√
Load balancing	√	√
Elección de rutas con menos coste	√	√
Gateway para sms o xmpp	√	√
Diferente soporte a base de datos - MySQL, PostgreSQL	√	√
Conexión directa con pasarelas a la RTC	√	√
Servidor IM SIP		√
Agente de presencia SIP		√
Servidor de aplicaciones SIP		√

Cuadro 3.2: Comparativa OpenSIPS - Kamailio

Tras un análisis de cada uno de los proyectos se optó por el uso de OpenSIPS, ya que parte de la última versión de OpenSER y es de las soluciones más utilizadas en el ámbito profesional.

3.3. Centralita VoIP (PBX)

Una PBX (*Private Branch Exchange*) es una central telefónica conectada directamente a la red pública de teléfono por medio de líneas troncales para gestionar, además de las llamadas internas, las entrantes y/o salientes con autonomía sobre cualquier otra central telefónica. Los usuarios de una PBX no tienen asociada ninguna central de teléfono pública, ya que es el mismo PBX que actúa como tal, análogo a una central pública que da cobertura a todo un sector mientras que un PBX lo ofrece a las instalaciones de una compañía generalmente.

Una centralita VoIP (PBX) ofrece:

- Salida a la Red Telefónica Conmutada (RTC): Los usuarios registrados en el sistema podrán realizar llamadas a la RTC.
- Llamadas a bajo coste: Una PBX es capaz de conectarse a diferentes proveedores de servicios de VoIP, permitiendo así la realización de llamadas que serán tarificadas a la plataforma por el proveedor con el que se realice la llamada.

La solución de código abierto más empleada en el ámbito profesional es Asterisk[27].

3.3.1. Asterisk

Asterisk[27] es un proyecto opensource que emula el comportamiento de una PBX. Una PBX es simplemente una central telefónica. Asterisk originalmente fue diseñado para sistemas con un número reducido de usuarios, aunque en la actualidad es posible emplearlo en sistemas de varios miles de usuarios. Todas estas características y el crecimiento espectacular del proyecto en su relativa corta existencia, hacen de Asterisk una de las soluciones a tener en cuenta para implantar VoIP en la pequeña, mediana y gran empresa.

El paquete básico de Asterisk incluye muchas características que antes sólo estaban disponibles en caros sistemas propietarios como creación de extensiones, envío de mensajes de voz a e-mail, llamadas en conferencia, menus de voz interactivos y distribución automática de llamadas. Además, se pueden crear nuevas funcionalidades mediante el propio lenguaje de Asterisk o módulos escritos en C o mediante scripts

AGI escritos en Perl o en otros lenguajes. Pero quizás lo más interesante es que Asterisk soporta numerosos protocolos de VoIP como SIP y H.323. Asterisk puede operar con muchos teléfonos SIP, actuando como "registrar" o como "gateway" o entre telefónos IP y la red telefónica convencional. Los desarrolladores de Asterisk han diseñado un nuevo protocolo llamado IAX[28] para un correcta optimización de las conexiones entre centralitas Asterisk.

Al actuar como salida a la telefonía tradicional y dar soporte a los servicios de VoIP, Asterisk permite a los desarrolladores construir nuevos sistemas telefónicos de forma eficiente o migrar de forma gradual los sistemas existentes a las nuevas tecnologías. Algunos sitios usan Asterisk para reemplazar las antiguas centralitas propietarias, otros para proveer funcionalidades adicionales y algunas otras para reducir costes en llamadas a larga distancia utilizando Internet. Con Asterisk es posible implementar servicios de VoIP bastante complejos sobre arquitecturas como las siguiente[3].

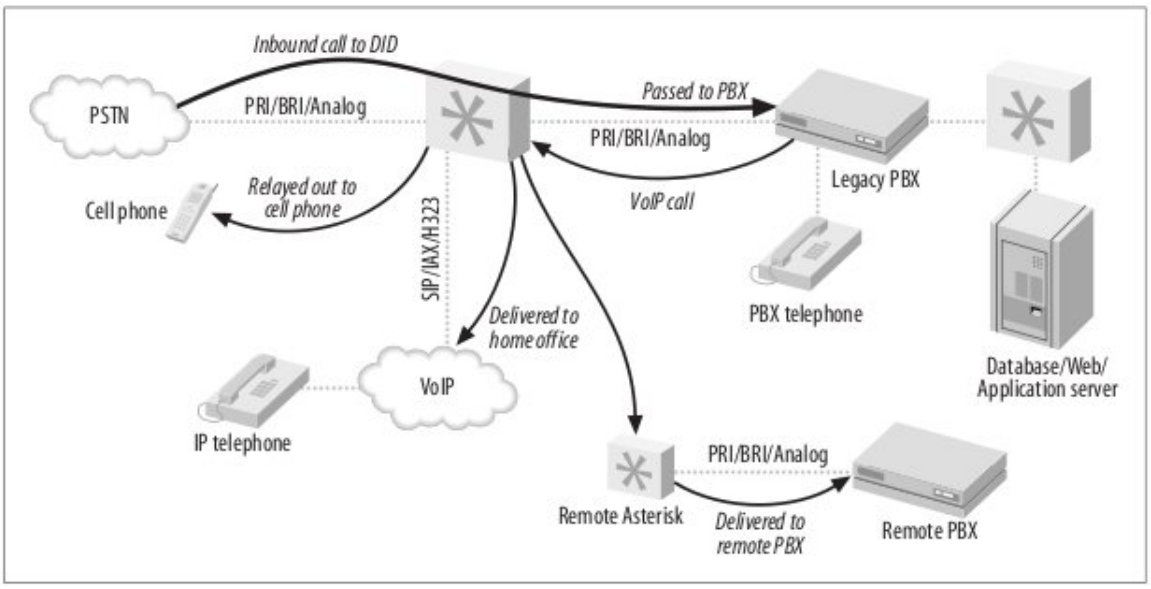

Figura 3.7: Arquitectura Asterisk

[3]Imagen extraída de “Asterisk: The Future of Technology”.

Son muchos los proyectos que han surgido en los últimos años para facilitar la configuración de una PBX Asterisk. Asterisk, por defecto, sólo permite configurarse a partir de ficheros de texto modificados en un terminal. Es por ello, que todos los proyectos surgidos para este fin están centrados en crear una interfaz web para facilitar la configuración. De entre estos proyectos, ha sido analizado por su sencillez de uso Elastix[29].

ELASTIX

Elastix es una distribución gratuita de un servidor de comunicaciones basada en Cent OS. Sobre esta distribución vienen instalados por defecto un servidor VoIP, Fax, Servicio de Mensajería instantánea y servidor de correo, de tal manera que pueden ser fácilmente configurables todos estos servicios nada más instalada la máquina. El sistema carece de servidor 'X' y es totalmente configurable desde una interfaz web que incluye todos estos servicios.

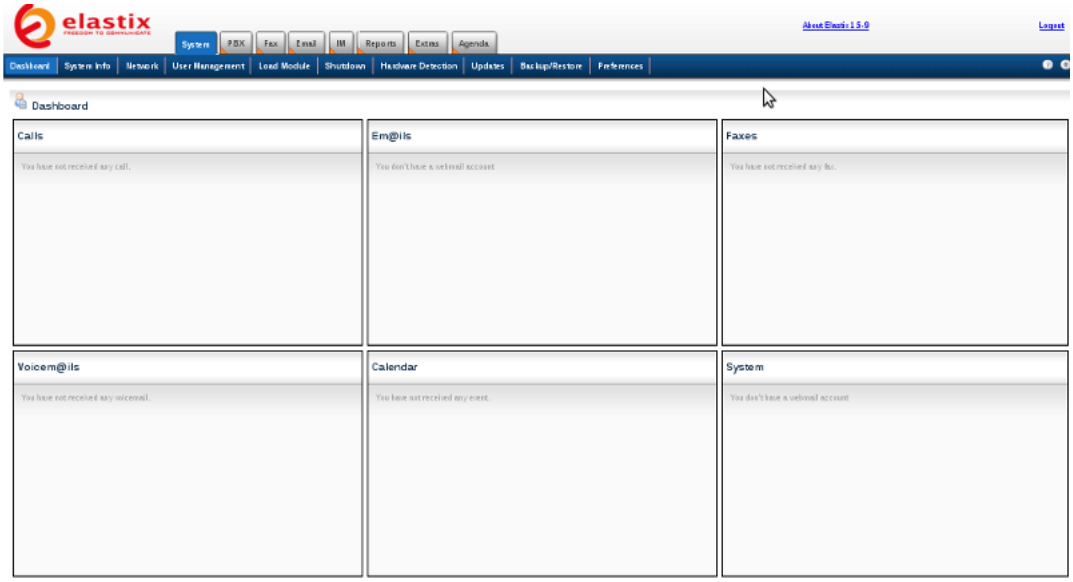

Figura 3.8: Apariencia Interfaz Web Elastix

En cuanto al servidor de VoIP, se trata de un servidor FreePBX[30] empotrado en

la plataforma, que es simplemente un servidor Asterisk configurable desde interfaz web. Su aspecto es el siguiente:

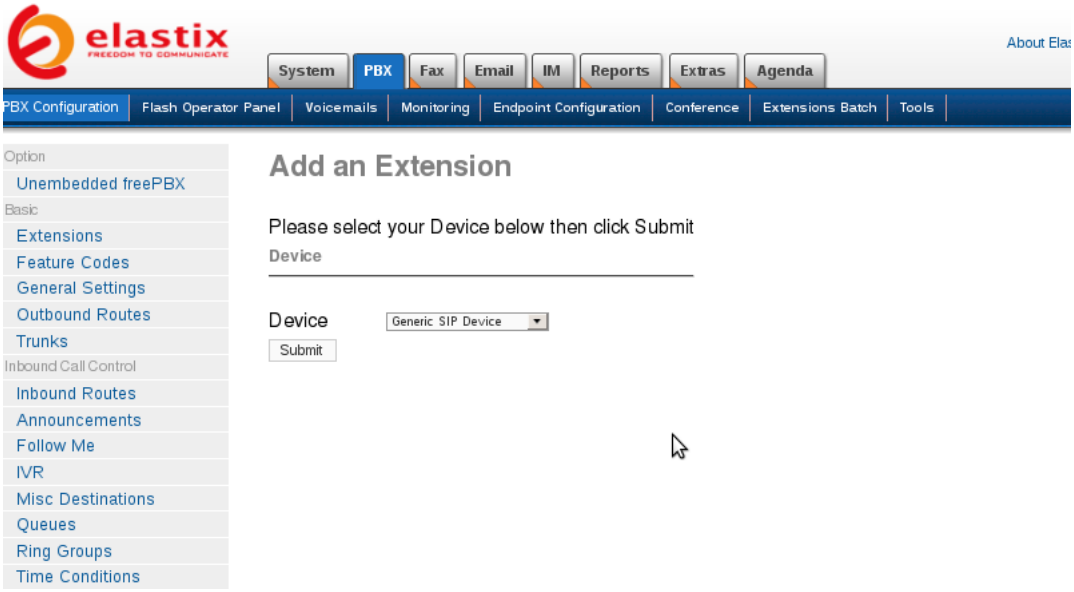

Figura 3.9: Aspecto PBX Elastix

Algunas funcionalidades destacables de este servidor, es la posible generación de reportes de llamadas asi como la configuración del cobro de llamadas (*billing*) gracias a que tiene un a2billing[31] también empotrado.

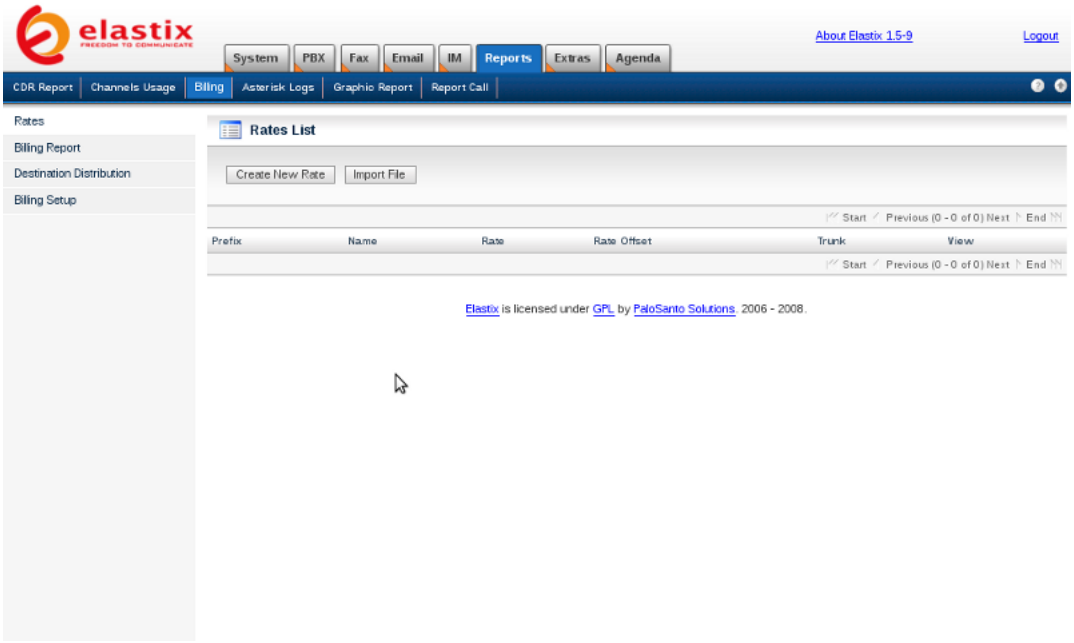

Figura 3.10: Aspecto a2billing Elastix

3.4. Servidores Multimedia

Un Servidor Multimedia (*Media Server*) permite la implementación de muchos de los servicios ya mencionados, la diferencia fundamental con los elementos de la infraestructura ya mencionados es que solo se encarga del tráfico RTP.

De entre las soluciones que implementan un servidor multimedia, han sido consideradas la solución de Mobicents (*Mobicents Media Server*[32]) y la solución del proyecto SER, llamada SEMS[33].

Uno de los motivos por los que se consideraron ambos proyectos fue que para la implementación del servicio de anuncio personalizado, era necesaria la utilización de un Media Server que implementase la RFC4240[34], que especifica la reproducción de ficheros multimedia a través del protocolo de señalización SIP.

3.4.1. RFC 4240

En la RFC 3261[11] se define el protocolo SIP sin necesidad de definir ningún servicio multimedia. Este tipo de servicios pueden ser desde la reproducción de un anuncio hasta la realización de un conversación con más de dos usuarios, conferencia.

Los anuncios son ficheros multimedia que son servidos a un usuario. Estos anuncios pueden ser ficheros estáticos, pueden ser generados dinámicamente en tiempo real, etc.

El mecanismo empleado para la reproducción de los anuncios no modifica los principios en los que está basado el protocolo SIP. Se emplea la parte de usuario de la URI SIP para así tener un indicador de servicio, pese a que los servidores multimedia no tienen soporte para diferentes usuarios.

El uso de la parte de usuario de la URI SIP tiene una serie de utilidades:

- No se modifica nada del protocolo SIP.
- Solo aquellos terminales que quieran actuar según este estándar tienen que implementarlo.
- Estas modificaciones que se van a mencionar en este fragmento solo tendrán que ser tenidas en cuenta en Media Servers.

El servidor multimedia cuando recibe un INVITE separa la parte de usuario de la petición que es el indicador de servicio solicitado. Dependiendo de este indicador, el servidor procesará la petición correctamente o bien enviará un mensaje de error con su correspondiente código.

3.4.2. Mobicents Media Server

Mobicents Media Server es un servidor bastante completo que permite la implementación de la RFC 4240[34]. Asi mismo, permite la utilización de diferentes códecs(G.711, speex y G.729).

A parte de todo esto, este servidor proporciona varias funcionalidades:

- Servicio de Anuncio.

- IVR (Grabación de conversaciones) y DTMF.
- Servicio de Conferencia.
- Loopback (usado principalmente en test).

Puede ser usado de forma local usando la API de MSC, asi como que también puede ser invocado de forma remota empleando MGCP. Presenta, asimismo, una interfaz web que permite su configuración.

3.4.3. SEMS

SEMS es un servidor de aplicaciones y recursos multimedia para servicios VoIP basados en SIP. Presenta muy buenas prestaciones realizando algunos servicios básicos como anuncios, conferencias combinándose, en algunos casos, con servidores de aplicaciones externos. Gracias a su facilidad de uso y su framework de aplicaciones flexible, así como de su soporte a Agentes de Usuario extremo a extremo, se puede unir en un mismo proceso la lógica de la aplicación con los recursos servidos por el servidor. Suele ser empleado con algunos de los servidores SIP analizados anteriormente (SER, OpenSER, OpenSIPS) de tal manera que se obtiene asi un servicio de VoIP completo.

Las principales funcionalidades de SEMS son las siguientes:

- Servicio de Voicemail: Servicio que almacena mensajes destinados a usuarios que en el momento de la recepción de los mismos no pudieron atender la llamada. Estos mensajes son enviados posteriormente al correo electrónico del usuario.
- Anuncio: Reproduce un recurso solicitado.
- Servicio de Conferencia: Permite conversaciones telefónicas por más de dos usuarios simultáneamente.
- Servicio de Videoconferencia.

Algunos de los códecs soportados por la versión estándar de SEMS son G.711, iLBC, speex y GSM.

Debido a que finalmente este servidor multimedia ha sido elegido como solución, a continuación se añade una breve descripción de su arquitectura.

SEMS proporciona una separación de media y control facilitando así el diseño y favoreciendo a la escalabilidad del sistema. En el contexto de un Servidor Multimedia SIP, esto significa:

- Señalización y la lógica de la aplicación son independientes del procesado de los recursos multimedia.
- Señalización SIP separada del streaming RTP.

3.5. Servidor Telco de Aplicaciones

El servidor telco de aplicaciones ejecuta la lógica de negocio del servicio (ClickToDial, tarificación de llamadas, personalización de servicios). La implementación de todas las aplicaciones SIP se ha realizado mediante la tecnología de SIP Servlets descrita en la sección 2.4.2. Como servidor de SIP Servlets se ha escogido Sailfin[35] gracias a sus facilidades de desarrollo integradas con el IDE Netbeans.

Sailfin es un proyecto basado en la robusta tecnología de Sip Servlets. Actualmente se encuentran implementando el JSR116 [21], aunque en un futuro no muy lejano trabajarán por hacerlo compatible con el estándar JSR289, añadiendo nuevas características a los servicios ya implementados para Glassfish.

Glassfish es un proyecto basado en la tecnología Java EE 5 cuya finalidad es la implementación de un servidor de aplicaciones. Hay una gran variedad de subproyectos entre los que se puede encontrar Sailfin. Sailfin añade la compatibilidad con JSR289, además de tratarse de un contenedor de servlets capaz de repartir la carga de tráfico tanto SIP como HTTP.

Algunas de las aplicaciones más importantes que se pueden implementar gracias a este proyecto están orientadas hacia las transacciones, seguridad, servicios web. Con el paso de los meses está comenzando a tener cada vez más usuarios y más desarrolladores, que publican diferentes aplicaciones web que demuestran las posibilidades que realmente tiene el proyecto.

3.6. Conclusiones: Arquitectura de Despliegue de Aplicaciones

Tras analizar cada una de las funcionalidades de los elementos que componen la infraestructura VoIP se ha decidido emplear los siguiente proyectos de código abierto:Proxy SIP - Servidor de Registro: **OpenSIPS,** Centralita VoIP (PBX): **Asterisk,** Servidor Multimedia: **SEMS** y como Servidor Telco de Aplicaciones: **Sailfin.**

Pese a que Asterisk y OpenSIPS aparentemente realicen las mismas funciones, no es así. Asterisk es un Agente de usuario extremo a extremo (B2BUA) mientras que OpenSIPS es un Proxy SIP[4]. Esto hace que OpenSIPS sea más rápido y ligero que Asterisk debido a que solo trata con señalización. Por otra parte, Asterisk pese a ser más lento tiene algunas características que lo hacen interesante como Traducción de Códecs (G729<->G.711), traducción de Protocolos (SIP<->H323) y otros servicios como reconocimiento de voz, encolado...

Además, OpenSIPS se encarga en muchas más ocasiones de solucionar los posibles problemas ocasionados detrás de una red que hace natting. De esta manera se puede enviar la información directamente del usuario al proveedor usando simplemente OpenSIPS, capaz de manipular las cabeceras de los mensajes SIP.

Una vez analizadas cada una de las soluciones escogidas para la arquitectura se puede concluir afirmando que las principales ventajas de esta arquitectura son:

- Escalabilidad de servicios, gracias al uso de un Servidor Telco de aplicaciones
- Soporte al protocolo AAA [36]: Integración Radius[37], Diameter [38], LDAP [39], gracias a la integración de OpenSIPS.
- Facturación y conectividad a RTC ofrecidas por Asterisk.
- Reproducción de archivos multimedia, ofrecido por SEMS.

La plataforma de despliegue se ha empaquetado como una **máquina virtual** con VMware[40], para que los desarrolladores puedan disponer de un entorno donde

[4]Comparing Asterisk and OpenSER - http://www.packtpub.com/article/comparing-asterisk-and-openser

puedan probar rápidamente los servicios desarrollados. Dicha máquina virtual, contiene todos los elementos mencionados en la arquitectura, adeacuadamente configurados, para que el desarrollador sólo deba estar pendiente del nuevo servicio que esté implementando. Más detalles acerca de la instalación de la plataforma y despliegue de la Máquina Virtual se pueden encontrar en la sección A.

4 Análisis de Requisitos de la Plataforma

4.1. Introducción

Una vez establecida la arquitectura sobre la que se va a trabajar, es necesario definir los requisitos funcionales y no funcionales lo cual permitirá definir el alcance del trabajo que hay que llevar a cabo. También se darán una serie de casos de uso que ilustrarán con mayor detalle las funcionalidades que se esperan de la aplicación.

4.2. Requisitos del Servicio

En el ámbito de este documento se van a distinguir dos tipos de requisitos:

1. **Requisitos funcionales**: Funciones que lleva a cabo el sistema.
2. **Requisitos no funcionales**: Limitaciones que condicionan el funcionamiento del sistema expuesto mediante los requisitos anteriores.

4.2.1. Requisitos Funcionales

- **Login de Usuarios:** La aplicación debe permitir el inicio de sesión de los usuarios mediante interfaz web. Estos usuarios deben ser dados de alta previamente en una base de datos, donde especifican una serie de datos que contiene su perfil, de tal manera que solo los usuarios registrados puedan utilizar el servicio web.

- **Servicio ClickToDial:** La plataforma debe ofrecer un servicio de llamada a los usuarios registrados en la plataforma. De tal manera, que una vez iniciada la sesión puedan registrar sus terminales VoIP y cursar llamadas a otros usuarios conectados al servicio. Al final de una llamada cursada con este servicio se debe mostrar al usuario la factura a la que asciende la llamada.
- **Mostrar Usuarios en la Página Web:** Una vez iniciada la sesión por el usuario en la página web del servicio, éste debe ser capaz de poder ver en la página todos aquellos usuarios a los que puede realizar una llamada, porque también hayan iniciado sesión.
- **Servicio Facturación**: La plataforma debe ofrecer un servicio mediante el cual se puedan generar facturas de las llamadas cursadas por un usuario. El detalle de estas llamadas estará almacenado en un fichero que será procesado por la base de conocimiento aplicando tarifas personalizadas a los usuarios.
- **Base de Conocimiento conforme a las necesidades del cliente:** La base de conocimiento debe ser tal, que sea capaz de seleccionar de forma astuta el anuncio más adecuado al cliente que realiza la llamada a partir de los datos que se especificaron al registrar al cliente, en el servicio de anuncio personalizado. Además, debe ser capaz de tener en cuenta también la información del usuario que va a recibir la llamada. Una vez servido el anuncio, la base de conocimiento debe facturar la llamada de acuerdo a las ofertas personalizadas definidas para cada usuario.
- **Logs de los Servidores:** Es necesaria la creación de Logs de los diferentes elementos empleados en el servicio para así conocer el estado de cada uno de ellos en un momento dado.

4.2.2. Requisitos No Funcionales

- **Sistema Multiusuario:** El sistema debe ser capaz de dar servicio a múltiples usuarios de manera simultánea, permitiendo llevar cada conversación y la información asociada a la misma de manera independiente.
- **Escalabilidad:** El sistema debe ser capaz de soportar cargas de trabajo elevadas, permitiendo el uso por parte de un gran número de usuarios sin que

ello suponga una disminución de la calidad del servicio.

- Diseño de módulos o componentes funcionales totalmente independiente del resto de elementos, de tal manera que puedan ser reutilizados en varios servicios sin realizarse grandes modificaciones.
- **Extensibilidad:** El sistema debe ofrecer una interfaz cómoda y común para el desarrollo de nuevos componentes que extiendan su funcionalidad o su interfaz gráfica y que puedan acoplarse fácilmente a él. Además su base de conocimiento debe poder ser modificada o ampliada de manera sencilla, tanto en el número de temas tratados como en la información disponible para dichos temas.
- **Compatibilidad**: El sistema debe poder ofrecer el mismo servicio independientemente del terminal que se emplee en la realización de llamadas, ya sea un teléfono IP físico o un softphone.
- **Amigable e intuitivo**: Es deseable que el sistema sea accesible para cualquier usuario y que no requiera de conocimientos técnicos excesivos para su uso. Además es recomendable que las interfaces sean agradables para los usuarios, tanto en manejo como en aspecto.

4.3. Casos de Uso

En los casos de uso se describe qué es lo que debe hacer la aplicación dependiendo de las distintas interacciones de los actores implicados con la misma.

4.3.1. Casos De Uso del Servicio de Anuncio Personalizado

Acceso de un usuario llamado a través de interfaz Web

En estos casos de uso un usuario inicia sesión en la interfaz web y a continuación recibe una llamada

Caso de Uso 1	**Acceso usuario llamado a través de interfaz web**
Descripción	Un usuario debe iniciar sesión a través de un navegador web.
Actores	Usuario
Precondiciones	El usuario se conecta a la aplicación mediante un navegador Web. Así mismo el usuario debe disponer de un cliente SIP para poder autenticarse contra el Servidor SIP y cursar la llamada.
Secuencia Normal	1.- El usuario inicia sesión en la plataforma a través de un navegador web.
	2.- El usuario inicia su cliente SIP, introduciendo los datos necesarios para autenticarse contra el Servidor SIP.
	3.- El usuario recibe una llamada de otro usuario registrado en el sistema y se extablece la conversación.
	4.- Fin de la conversación.
Postcondiciones	La interacción entre los diferentes elementos que toman parte en la acción debe ser registrada en sus correspondientes logs.

Cuadro 4.1: Caso de Uso: Acceso Usuario Llamado a través de Interfaz Web

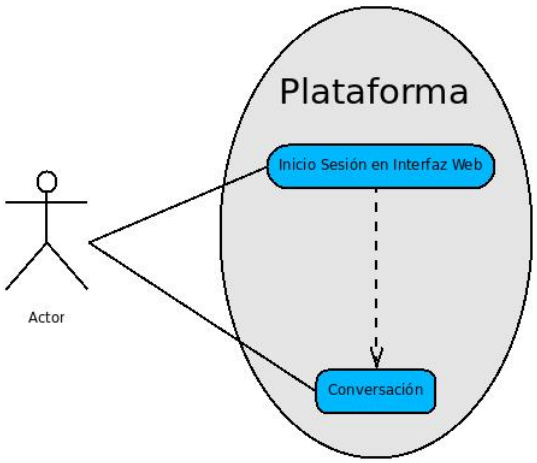

Figura 4.1: Caso de Uso: Acceso de un usuario llamado a través de interfaz web

Acceso de un usuario llamante a través de interfaz Web

En estos casos de uso un usuario inicia sesión en la interfaz web y procede a realizar una llamada a otro usuario registrado en el sistema.

Caso de Uso 2	**Acceso usuario llamante a través de interfaz web**
Descripción	Un usuario debe iniciar sesión a través de un navegador web.
Actores	Usuario
Precondiciones	El usuario se conecta a la aplicación mediante un navegador Web. Así mismo el usuario debe disponer de un cliente SIP para poder autenticarse contra el Servidor SIP y cursar la llamada.
Secuencia Normal	1.- El usuario inicia sesión en la plataforma a través de un navegador web.
	2.- El usuario inicia su cliente SIP, introduciendo los datos necesarios para autenticarse contra el Servidor SIP.
	3.- El usuario selecciona de la página web a la que ha sido redirigido tras iniciar sesión al usuario con el que desea establecer una conversación.
	4.- La petición es procesada por un proxy SIP que la redirige a la base de conocimiento. La cual puntúa todos los anuncios disponibles, seleccionando el de mayor puntuación para ser servido al usuario llamante.
	5.- El anuncio es solicitado al servidor multimedia, el cual sirve al anuncio al llamante.
	6.- Tras finalizar la reproducción del anuncio, se realiza finalmente la llamada al destinatario, estableciéndose la conversación.
	7.- Fin de la converesación.
	8.- Se muestra la factura de la llamada en pantalla.
Postcondiciones	La interacción entre los diferentes elementos que toman parte en la acción debe ser registrada en sus correspondientes logs.

Cuadro 4.2: Caso de Uso: Acceso Usuario Llamante a través de Interfaz Web

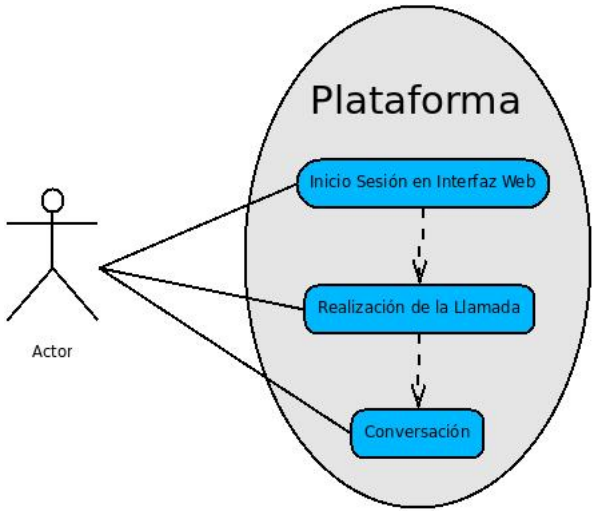

Figura 4.2: Caso de Uso: Acceso de un usuario llamante a través de interfaz web

4.3.2. Casos de Uso del Servicio de Facturación

Acceso de un usuario para generar una factura

Caso de Uso 3	**Acceso usuario a través de interfaz web para generar una factura**
Descripción	Un usuario web accede a la interfaz para obtener una factura con el detalle de sus llamadas.
Actores	Usuario
Precondiciones	El usuario se conecta a la aplicación mediante un navegador Web.
Secuencia Normal	1.- El usuario accede a la plataforma a través de un navegador web.
	2.- El usuario solicita el nombre del usuario del que desea obtener la factura mensual con el detalle de las llamadas.
	3.- La aplicación muestra por pantalla la factura al usuario
Postcondiciones	La interacción entre los diferentes elementos que toman parte en la acción debe ser registrada en sus correspondientes logs.

Cuadro 4.3: Caso de Uso: Acceso usuario a través de interfaz web para generar una factura

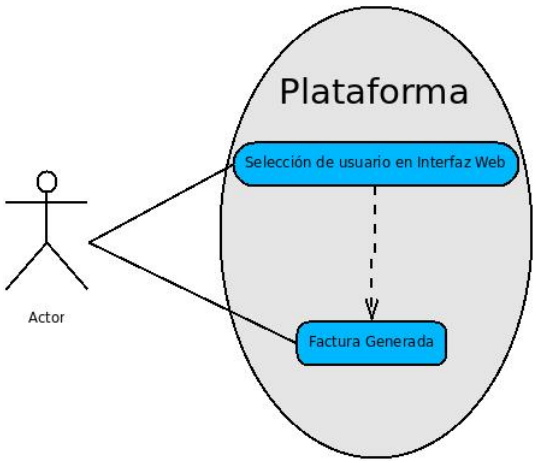

Figura 4.3: Caso de Uso: Acceso de un usuario llamado a través de interfaz web

4.3.3. Casos de Uso de Administrador

Despliegue de la aplicación

El administrador despliega la aplicación. Todas las incidencias y registros de actividad que se den durante el proceso de arranque quedan almacenados en el correspondiente log de la aplicación.

Caso de Uso 4	Despliegue de la aplicación
Descripción	El administrador del servicio despliega la aplicación sobre un servidor.
Actores	Administrador
Precondiciones	El servidor sobre el que se desplegará la aplicación debe Precondiciones encontrarse correctamente instalado.
Secuencia Normal	1.- Despliegue de la aplicación click to dial sobre servidor sailfin.
	2.- Despliegue del servidor que contiene la base de conociemiento.
	3.- Despliegue del servidor multimedia.
	4.- Toda la información del arranque de la aplicación, así como cualquier incidencia que pudiera surgir al iniciarse la misma, queda registrada en el archivo de incidencias.
Postcondiciones	La aplicación se encuentra arrancada y todos los logs de arranque Poscondiciones e incidencias quedan registrados en un archivo.

Cuadro 4.4: Caso de Uso: Despliegue de la aplicación

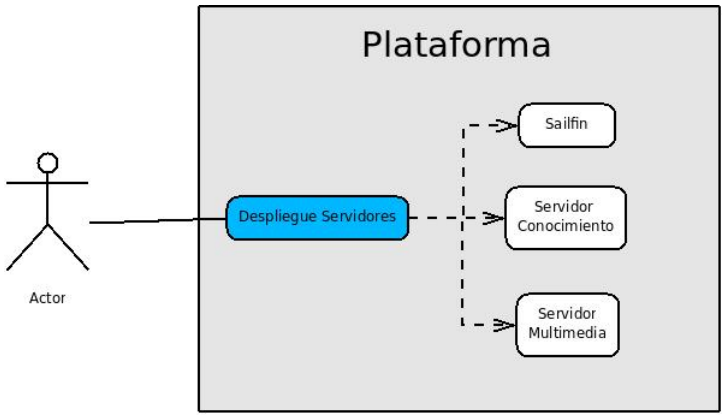

Figura 4.4: Caso de Uso: Despliegue de la aplicación

Configuración base de conocimiento

El administrador abre el archivo de configuración de la base de datos y modifica, añade o borra la entrada o las entradas correspondientes a los archivos que poseen

los temas de conocimiento que desee. Al reiniciar el sistema debe cargarse la base de conocimiento con los cambios que ha introducido el administrador.

Caso de Uso 5	**Configuración de la base de conocimiento**
Descripción	La base de conocimiento debe ser configurable.
Actores	Administrador
Precondiciones	Ninguna
Secuencia Normal	1.- El administrador abre el archivo de configuración de la base de conocimiento.
	2.- El administrador modifica, borra o añade la entrada o las entradas correspondientes a los archivos de conocimiento Secuencia normal que desee.
	3.- El administrador guarda la configuración y reinicia la aplicación.
Postcondiciones	La aplicación se encuentra arrancada y todos los logs de arranque Poscondiciones e incidencias quedan registrados en un archivo.

Cuadro 4.5: Caso de Uso: Configuración de la Base de Conocimiento

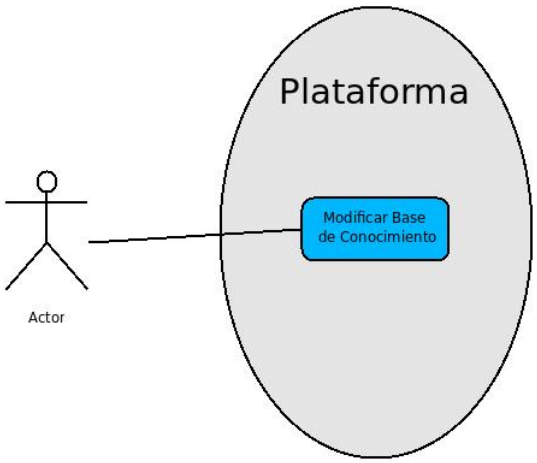

Figura 4.5: Caso de Uso: Configuración Base de Conocimiento

5 Arquitectura Plataforma de Servicios

5.1. Introducción

En el presente capítulo se definirá la arquitectura general de la plataforma de servicios, los elementos que la forman y las diferentes relaciones que existen entre ellos.

Una vez definida la arquitectura general se detallarán sus elementos. Al mismo tiempo que se define la arquitectura de las aplicaciones y de cada uno de los componentes que las forman, se explicarán las diferentes decisiones de diseño que se han tomado durante el proceso de implementación de los servicios.

5.2. Descripción de la Arquitectura

En la arquitectura de la plataforma de servicios se ha seguido el patrón MVC. Modelo Vista Controlador (MVC) es un patrón de arquitectura de software que separa los datos de una aplicación, la interfaz de usuario, y la lógica de control en tres componentes distintos:

- **Modelo**: Esta es la representación específica de la información con la cual el sistema opera. La lógica de datos asegura la integridad de estos y permite derivar nuevos datos. En la presente arquitectura el modelo coincide con el componente de personalización, se encarga de procesar las peticiones realizadas por los usuarios de la plataforma y de procesar las peticiones realizadas al servidor de conocimiento.

- **Vista**: Este presenta el modelo en un formato adecuado para interactuar, usualmente la interfaz de usuario.
- **Controlador**: Éste responde a eventos, usualmente acciones del usuario e invoca cambios en el modelo y probablemente en la vista.

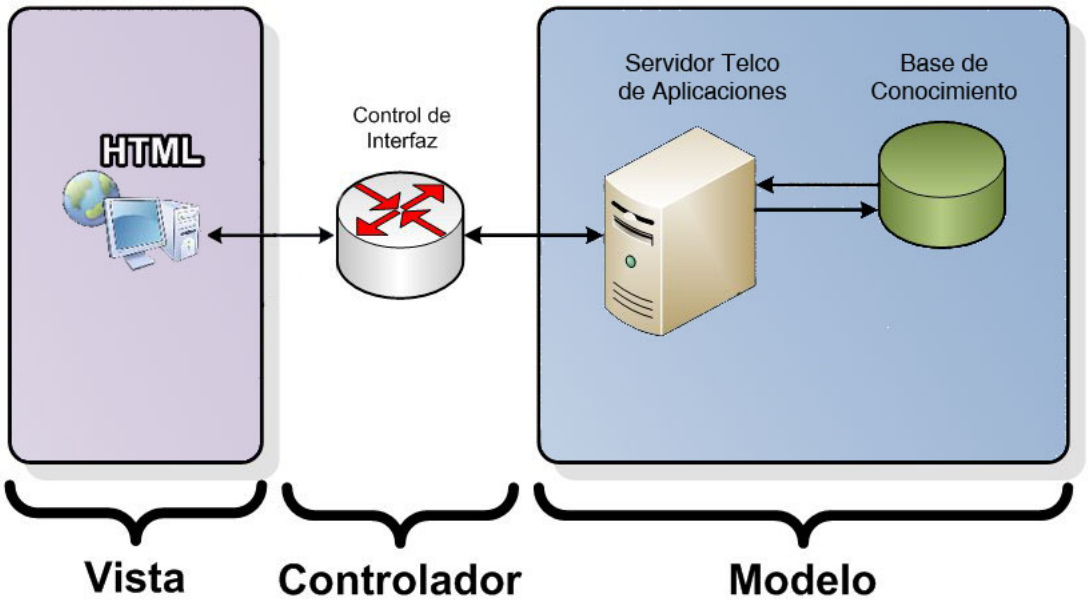

Figura 5.1: Esquema de patrón MVC de la plataforma de Servicios

La unión entre capa de presentación y capa de negocio conocido en el paradigma de la Programación por capas representaría la integración entre Vista y su correspondiente Controlador de eventos y acceso a datos, MVC no pretende discriminar entre capa de negocio de capa de presentación pero si pretende separar la capa visual gráfica de su correspondiente programación y acceso a datos algo que mejora el desarrollo y mantenimiento de la Vista y el Controlador en paralelo ya que ambos cumplen ciclos de vida muy distintos entre sí.

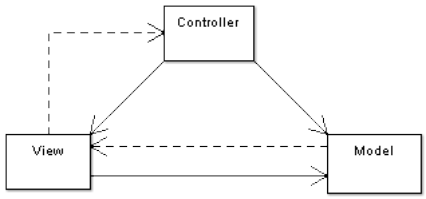

Figura 5.2: Interacción Modelo-Vista-Controlador

Así pues, siguiendo esta arquitectura la interacción con el sistema se haría de la siguiente manera:

1. El usuario accede a la interfaz web de la plataforma y selecciona el tipo de servicio al que desea acceder, realizando así una petición al controlador.
2. El controlador recibe (por parte de los objetos de la interfaz-vista) la notificación de la acción solicitada por el usuario. El controlador gestiona el evento que llega, saneando la petición y adecuándola a las necesidades del modelo.
3. El modelo recibe la petición, la analiza, procesa las bases de conocimiento y devuelve la respuesta al controlador quedando a su vez registrada la interacción en un archivo de log.
4. El controlador delega a los objetos de la vista la tarea de desplegar la interfaz de usuario. La vista obtiene sus datos del modelo para generar la interfaz apropiada para el usuario donde se refleja los cambios en el modelo. El modelo no debe tener conocimiento directo sobre la vista. Sin embargo, el patrón de observador puede ser utilizado para proveer cierta indirección entre el modelo y la vista, permitiendo al modelo notificar a los interesados de cualquier cambio.
5. La interfaz de usuario espera nuevas interacciones del usuario, comenzando el ciclo nuevamente.

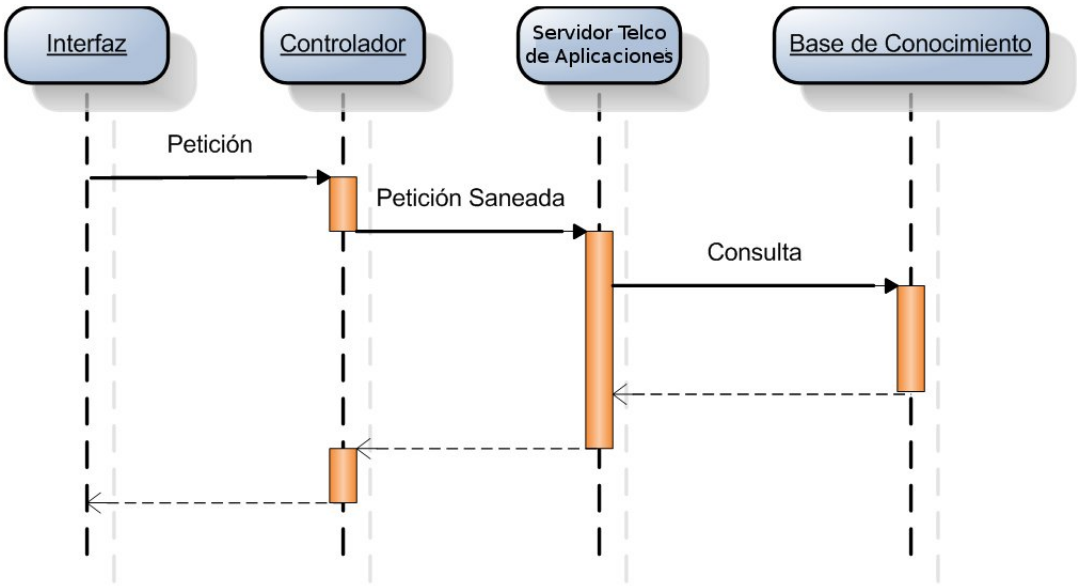

Figura 5.3: Diagrama Secuencia Aplicación

5.3. Modelo: Descripción Componente de Personalización

El modelo engloba el núcleo de la lógica de la aplicación, debe recoger la consulta, acceder a la base de conocimiento, procesar la entrada comparándola con los patrones de la base de conocimiento y devolver una respuesta.

Partimos de una infraestructura de VoIP común a todos los servicios. Para la implementación de los diferentes servicios es necesaria la integración en la plataforma de distintos bloques funcionales independientes entre sí. En el caso de los componentes necesarios para la personalización, es necesaria, ya sea implementando el servicio con SIP Servlets o JAIN SLEE; una base de conocimiento en la que se basará nuestro Sistema Experto.

El objetivo del componente de personalización es facilitar la externalización de la lógica del negocio de un servicio de telecomunicaciones. La personalización[41] es un

elemento clave para tanto el descubrimiento de nuevos servicios, como la adaptación de los servicios existentes a las características de los usuarios.

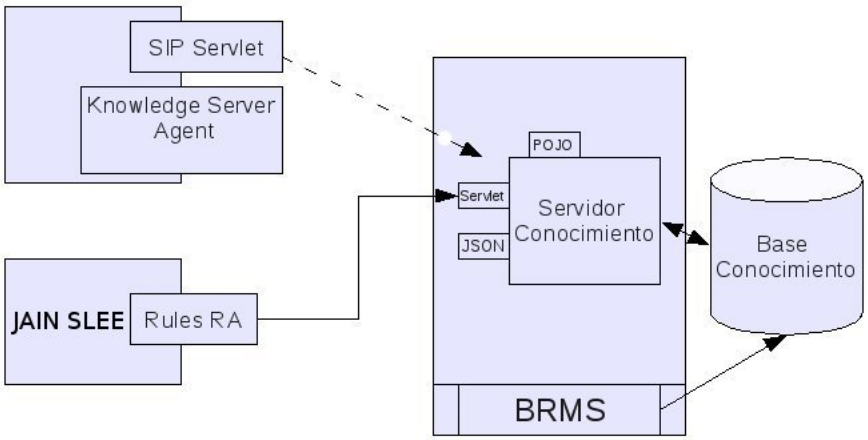

Figura 5.4: Arquitectura Plataforma de Servicios

La figura 5.4 muestra la arquitectura del componente de personalización propuesto, que consta de los siguientes elementos:

- Servidor Telco de Aplicaciones.
- Servidor de Conocimiento.

5.3.1. Servidor Telco de Aplicaciones

Como ya fue descrito en la sección 3.5, el servidor telco de aplicaciones ejecuta la lógica de negocio del servicio. Las tecnologías empleadas para la implementación de las aplicaciones de la plataforma son SIP Servlets junto con HTTP Servlets, ambos desplegados sobre un servidor de aplicaciones Java (Sailfin [35]), así como JAIN SLEE. El servidor provee de interfaces para la implementación de servicios en ambas tecnologías.

- **Interfaz para tecnología SIP Servlets**. Los sip servlets atienden las peticiones SIP, haciendo peticiones al servidor de conocimiento cuando de él precisa algo, por ejemplo; en el servicio de anuncio personalizado, realiza una petición al servidor de conocimiento, para saber el anuncio a reproducir, antes de solicitar el anuncio a reproducir al servidor multimedia. Los SIP Servlets usan los componentes *RuleAgent* directamente.
- **Interfaz para tecnología JAIN SLEE**. Para los componentes JAIN SLEE, la integración se ha realizado mediante un adaptador de recursos Rules-RA[42], que permite cargar realizar consultas a los componentes SBB. En este caso, es el adaptador de recurso el que emplea un componente *RuleAgent* para acceder al gestor de conocimiento, que es configurable mediante la interfaz de gestión con JMX.

5.3.2. Servidor de Conocimiento

El servidor de conocimiento es el encargado de gestionar la base de conocimiento, ofreciendo una interfaz para su acceso remoto La base de conocimiento define las reglas y hechos de la lógica de servicio. El servidor está integrado con otro componente, llamado BRMS (*Business Rule Management System*) que permite la edición de la base de conocimiento mediante una interfaz web. Se ha seleccionado la plataforma Drools[43] para su implementación. Ofrece diferentes interfaces para su acceso (clase Java en local, útil para desarrollo, Servlet para acceso remoto, o bien JSON para las capa de presentación). Para el acceso remoto de este servidor de conocimiento, los clientes emplean *RuleAgents*, que son clases Java que hacen de proxy del servidor de conocimiento, encapsulado toda la lógica de acceso.

5.4. Vista

El uso del patrón MVC nos permite disociar la parte de presentación de la parte lógica. Además de la funcionalidad básica de presentación y captura de entrada de usuario, las vistas tienen también una parte activa, ya que se encargan de obtener información relativa a la sesión, mostrando al usuario un mensaje de bienvenida

en caso de que acabe de iniciar sesión o mostrando una información ya solicitada en caso contrario. Es decir, tienen que hacer frente a dos situaciones y actuar en consecuencia en cada uno de los casos:

- Si es la primera interacción del usuario debe mostrar un mensaje de bienvenida que le guíe en el uso de la aplicación.
- Si no es la primera interacción se debe atender la petición del usuario y enviarla al controlador junto con la información necesaria para que éste último sepa a quién tiene que devolver la respuesta correspondiente. Una vez procesada la petición se muestra en pantalla el servicio o la información solicitada.

5.4.1. Interfaz Web

La interfaz Web constará de tres partes diferenciadas, cada una de las cuales tiene una función y unas características específicas:

- **Página de Inicio**: Es la página a través de la cual se accede a la plataforma, consta de una serie de enlaces que permite seleccionar la aplicación que se quiere emplear así como navegar por la página del proyecto. Debido a la sencilla función de la página de inicio se puede implementar mediante una página estática HTML.
- **Página Servicio Anuncio Personalizado:** Permite al usuario realizar un login en la plataforma para posteriormente realizar o recibir una llamada a otro usuario de la plataforma mediante el servicio *ClickToDial*.
- **Página Servicio Facturación:** Permite a un usuario de la plataforma generar facturas mensuales de sus llamadas.

5.5. Controlador

El control de la aplicación se divide en dos partes, en primer lugar existe un JSP que se encarga del control de las vistas y, en segundo lugar, un servlet que se encarga de la gestión de entrada de usuario, redirigiendo las consultas hacia el servicio correspondiente y recogiendo la respuesta de dicho servicio.

El control de vistas actúa como un distribuidor, se encarga de tomar la entrada del usuario y redirigirla al servlet, para, una vez que recibe la respuesta, devolvérsela a la interfaz desde la que se solicitó.

En la gestión de entrada de usuario es donde se encuentra la mayor complejidad de la capa de control de la aplicación, ya que es donde se realiza la conexión con el servicio propiamente dicho. En esta conexión el servlet de control debe identificar el servicio al que se le quiere enviar la petición y el usuario que la envía. Con estos datos realizará la consulta y obtendrá la correspondiente respuesta. Toda esa información se guardará en variables de sesión para que sea fácilmente accesible por la capa de vistas.

6 Diseño Servicio Anuncio Personalizado

6.1. Introducción

Muchas empresas están tomando conciencia que no pueden servir de forma óptima a todos los posibles clientes que existen en un mercado, esta situación se debe principalmente a los gustos, preferencias, estilo, capacidad de comprar.

Internet está haciendo que la manera de actuar en los negocios cambie. Las oportunidades de éxito son numerosas, tanto como las de fracaso. Para convertirse en un sitio de Internet con éxito es necesario ganar la confianza y lealtad de los clientes, así como conseguir que los visitantes se conviertan en clientes. Por este motivo, la publicidad personalizada es un elemento clave en la supervivencia de un sitio de Internet, una publicidad orientada al cliente y no solo al producto.

En este capítulo se describirán algunos casos de estudio de anuncios personalizados, para concluir con la descripción de una aplicación de anuncio personalizado, mediante la cual un usuario puede llamar a otro usuario de la plataforma de forma gratuita, empleando un servicio de *click to dial,* a cambio de escuchar un anuncio elegido acorde a las características que especificó en su perfil de usuario.

6.2. Casos de Estudio

A continuación se presentan algunos servicios de anuncio personalizados diseñados para portales web, que han sido empleados como ejemplo a la hora de modelar la aplicación.

6.2.1. Fundamentos para personalizar un sistema[1]

El perfil de usuario es uno de los elementos más importantes de la personalización, esto representa todos aquellos elementos que preocupan de un cliente.

La implementación de un perfil de usuario comienza con unos datos que se rellenan al principio en una serie de formularios cumplimentados por los clientes y que pueden ser completados más adelante.

Un perfil normalizado de usuario debería contener los siguientes datos:

- email
- Nombre y Apellidos
- Fecha de Nacimiento
- Sexo
- Fecha de Registro en el sistema
- Dirección, Teléfono...
- Login, Password
- Idioma

Una vez especificado el perfil inicial, es probable que se necesite completar esa información a partir de datos procedentes, por ejemplo, de la experiencia del usuario en el sistema o de otros medios. La importancia de cada una de las fuentes de las que podemos obtener información acerca de los usuarios queda reflejado en la siguiente imagen:

[1]Extracto del artículo [44]

Figura 6.1: Importancia Perfil Usuario

- Información Obtenida a partir de la experiencia del cliente en el sistema:
  - Grado de interés en diferentes categorías.
  - Palabras buscadas en el sistema.
  - Productos que contienen una serie de palabras clave.
  - Información de Localización.
  - Historial de compras.
- Información obtenida por medios externos al sistema:
  - Información de lealtad.
  - Sistemas de análisis Web.
  - Sistemas de información, soporte al cliente.

Una vez definida la información que se debe tener de un usuario, es necesario diseñar cómo se va a utilizar esa información para poder servir un anuncio personalizado a los clientes.

Componentes de un sistema de personalización basado en reglas

- **Segmento**: Un segmento es un grupo de usuarios que tienen una serie de características similares.
- **Grupos de Contenido:** Un grupo de contenido es lo mismo que un segmento pero aplicado a los productos o anuncios que deseamos servir al cliente.

Es decir, se agrupan los productos o anuncios según las características que presenten.

- ***Content Targeters*:** Son una serie de reglas que asocian a los usuarios contenido. Las reglas consisten en una serie de piezas simples que condicionan cuándo se debe mostrar qué anuncio a qué usuario.

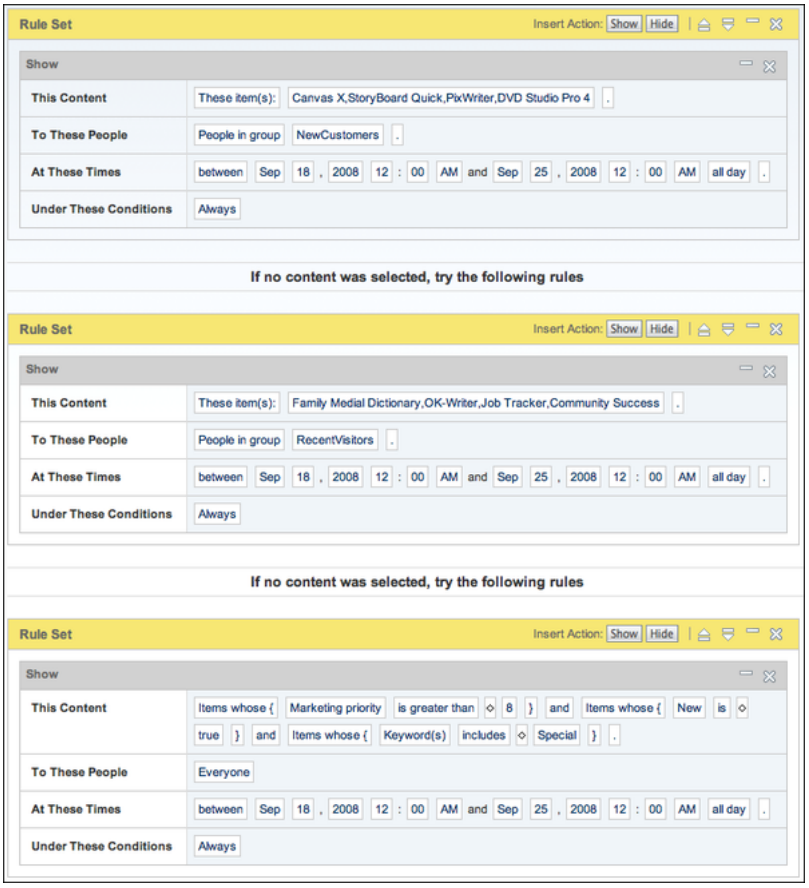

Figura 6.2: Ejemplo de reglas para personalización de anuncios

Ventajas del uso de reglas

Reutilización de reglas: Si se definen las reglas en una sección independiente de la aplicación es fácil depurar las reglas e incluso reutilizarlas para diferentes fines. Es útil reutilizar reglas por los siguientes motivos:

- Los resultados son más consistentes y están testados.
- Hay menos reglas que gestionar.
- Es más fácil depurar fallos.

6.2.2. Técnica de Scoring para personalizar anuncios[2]

Esta sección describe una nueva técnica de anuncio personalizado basado en las preferencias de un cliente hacia una categoría de productos específica. La importancia de este método reside en los resultados, bastante positivos, frente a la poca cantidad de datos necesarios para su correcto funcionamiento.

Arquitectura del Sistema

Para poder proveer anuncios personalizados, es necesaria la obtención de información del usuario, información proveniente del perfil, del historial de compras y del comportamiento en el entorno del sitio web. Para poder seleccionar estos anuncios en tiempo real, debido a problemas de tiempo, no es posible consultar todas las bases de datos para seleccionar el anuncio. Se propone, para poder superar esa dificultad, una estructura en árbol que contiene esa información de forma comprimida, de tal manera que se puede seleccionar el anuncio en tiempo real.

[2]Extracto del artículo [45]

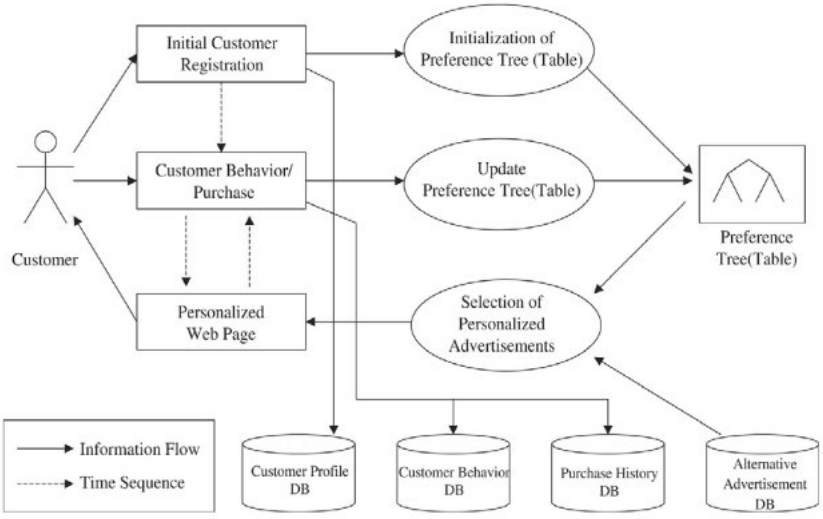

Figura 6.3: Estructura en árbol para proveer un anuncio en tiempo real

Con esta estructura, estamos suponiendo que el usuario aporta una serie de datos de lo que se puede deducir una serie de preferencias. Esta estructura se puede adaptar fácilmente a otros tipos de sitios en los que no se dispone de esa información hasta que el usuario comienza a navegar.

Cuando un usuario visita el sitio web por primera vez debe rellenar un formulario en el que se encuentran los datos que componen el perfil de usuario. Al mismo tiempo, se inicializa un árbol de preferencias que contiene información personal del usuario acerca de una serie de preferencias. Después del registro, lo normal es que el usuario navegue por el sitio web, haga click en algún anuncio y realice alguna compra. Este comportamiento es empleado para actualizar así la información del cliente, historial de compras, comportamiento ante la publicidad, intereses... Además, se establecen una serie de puntuaciones en el árbol de preferencias según el comportamiento que haya mostrado el usuario. De esta manera, cuando sea necesario escoger un anuncio personalizado, se eligen una serie de anuncios de la base de datos basados en esta puntuación. Estos anuncios son actualizados, creados y borrados por los gestores de la plataforma. Cuando se añade un nuevo anuncio a la base de datos se debe añadir información sobre la categoría a la que pertenece.

Modelos y Algoritmos de Selección de Anuncios Personalizados

Tabla de preferencias

La tabla de preferencias almacena en una tabla las puntuaciones de un cliente que muestra preferencias hacia un nível dentro de una categoría.

Una tabla de preferencias es una relación de la siguiente manera: PT = { CID, PGID, PS }

Donde CID, PGID y PS son el identificador de cliente, identificador del nivel de la categoría y la puntuación asignada. La puntuación es asignada a partir del perfil inicial, compras o muestras de interés.

Actualización de Puntuaciones

Cuando un usuario se registra, las puntuaciones según sus preferencias están basadas en su perfil inicial. Por lo tanto, la puntuación por interés de un usuario hacia una categoría puede ser distinto de cero cuando ha expresado un interés hacia esa categoría, o cero en el resto de los casos.

Cuando un cliente realice la compra de un producto de una categoría o exprese un determinado interés sobre un producto la puntuación por preferencia se actualizará.

Selección de Anuncios

Cuando el cliente visite la página principal del sitio le aparecerán una serie de anuncios personalizados basados en la puntuación que se da según sus preferencias. Aquellos anuncios que pertencen a categorías cuya puntuación sea elevada son elegidos en tiempo real. Para poder elegirlos, es necesario primero establecer cuál será el número de anuncios que se pueden mostrar al mismo tiempo, para ello es necesario establecer unas normas de selección.

Algoritmo de selección

1. Para un cliente, elegir la categoría que tenga la mayor puntuación y de ella el producto que mejor puntuación tenga.
2. Repetir el proceso hasta que se hayan elegido el número de anuncios especificado.
    a) Elegir dos anuncios que todavía no hayan sido comprados por el cliente.

b) Elegir un anuncio de la siguiente categoría con mayor puntuación, promocionando algún artículo que no haya sido adquirido por el cliente.

Árbol de Preferencias

La aproximación de la tabla de preferencias tiene un defecto y es, que no se pueden expresar las relaciones entre diferentes categorías. Por ejemplo, si un cliente compra un CD de baladas, es razonable pensar que le pueden interesar CDs de baile o R&B, antes que de música clásica. El árbol de preferencias ha sido diseñado para expresar estas afinidades.

El árbol de preferencias PT'(i) es idéntico al árbol de jerarquía de categorías. Las puntuaciones de un producto de un nivel de árbol sin descendencia se define como la media de puntuación de los nodos hijos de la misma categoría.

Con esta definición no se expresa la afinidad que puede existir entre diferentes categorías, pero ajustando la puntuación del nodo al que pertenecen se puede obtener la relación entre los diferentes níveles del árbol.

Actualización de las puntuaciones

1. El árbol de preferencias se inicializa a partir del perfil inicial. Estos valores se inicializan a un valor cuando el cliente muestra un cierto interés en un producto de una categoría o a cero en cualquier otro caso.
2. Los nodos que descienden de los anteriores se inicializan de la misma manera.
3. Los nodos de nível superior a los anteriormente descritos se inicializan haciendo la media de puntuaciones de todos los nodos hijos.

Las puntuaciones son actualizadas según las muestras de interés o compras realizadas por el cliente. Actualizando la puntuación para el nível de árbol y también para todos los progenitores, de esta manera no es necesario actualizar todas las puntuaciones del árbol.

Selección de Anuncios

El algoritmo de selección de anuncios en este caso, funciona de forma similar al anteriormente descrito. Se escoge el producto de la categoría con mayor puntuación y el producto de la siguiente categoría con mayor puntuación, de mayor puntuación.

Se van escogiendo anuncios, hasta que se completa el número de anuncios que se muestran simultáneamente, si la elección de un nodo hoja del árbol, se debe elegir otro nodo hoja que no haya sido comprado por el cliente, mientras que si la elección es un nodo que tiene descendientes se elegirá un anuncio de un nodo hoja de un producto que no haya sido adquirido por el cliente.

Una mejora de este algoritmo sería el empleo de tablas de preferencia para elegir entre anuncios que tienen la misma puntuación, de manera que en caso de empate se comparan las puntuaciones ajustadas para obtener las categorías de las productos elegidos, que será en lo que se basará el desempate.

Conclusiones

Pese a la poca información recabada por este método de publicidad personalizada, varios test realizados han demostrado que lo resultados obtenidos son bastante positivos comparados con sistemas de filtrado de información y selección aleatoria de anuncios.

Existen una serie de temas que no quedan cerrados después de este estudio, uno es cómo debe ser la estructura de datos para mejorar el funcionamiento y otro es cómo integrarlo con otros sistemas basados en reglas.

6.3. Descripción del Servicio Personalización de Anuncios para el llamante

A partir de los dos casos de estudio expuestos y utilizando la arquitectura ya presentada, se presenta la implementación práctica de un servicio de anuncio personalizado para un llamante. El servicio seleccionado consiste en la personalización de anuncios a un llamante combinado con un servicio de llamadas a través de la web *clicktodial*, figura 6.5). Los usuarios se registran en un portal web, especifican una serie de datos en su perfil, y pueden realizar llamadas pinchando en un botón (servicio *ClickToDial*). En el momento en el que se cursa la llamada, el sistema selecciona un anuncio auditivo, que es escuchado por el usuario, y una vez que termina el anuncio se cursa la llamada al destinatario. A cambio de escuchar la llamada, el usuario puede recibir alguna promoción, tales como descuentos o minutos gratis.

La figura (6.4) muestra el diagrama de servicio desarrollado con el plugin para diagramas de bloques de Telcoblocks para Eclipse, desarrollado con *Graphical Modeling Framework* (GMF)[46]

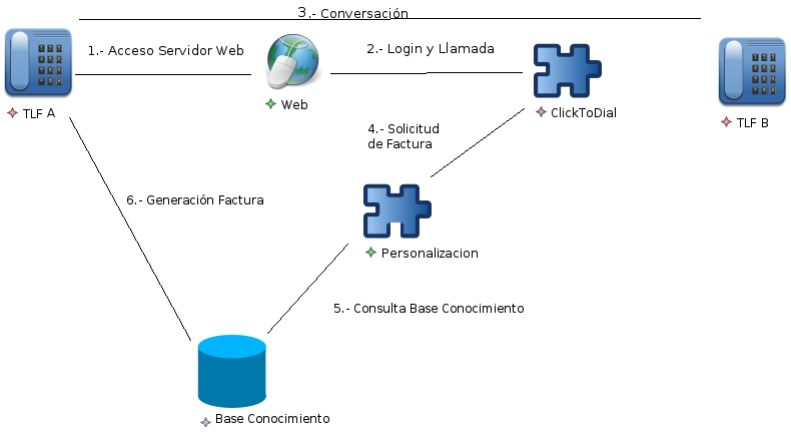

Figura 6.4: Secuencia acciones Servicio Anuncio Personalizado

6.3.1. Servicio Click To Dial

Primeramente, intentaremos abordar el problema desde el punto de vista de SIP. En el escenario de este servicio debe haber, al menos, dos clientes SIP registrados en nuestro servidor (llamante y destinatario de la llamada).

Para ello lo primero que deben realizar los usuarios es un login en la plataforma:

Figura 6.5: Interfaz Web Click To Dial

Una vez realizado este login, los usuarios deberán lanzar sus terminales SIP para poder registrarse en el Servidor de registro. El proceso de registro es llevado a cabo por un SIP Servlet, que registra las direcciones de cada uno de los usuarios en un SIP proxy usando la API de persistencia de Java. Estas direcciones serán empleadas más adelante para realizar una llamada. El seudocódigo de dicho SIP Servlet sería el siguiente:

```
/** A SIP Servlet to handle SIP REGISTER requests. */
public class RegistrarServlet extends SipServlet {
    // inject the SipFactory
@Resource(mappedName="sip/ClickToDial")
private SipFactory sf;
/** Handle a SIP Register request. */
protected void doRegister(SipServletRequest req)
```

```
        throws ServletException, IOException {
int response;
// Figure out the name the user is registering with.  This is the
// user portion of the SIP URI, e.g. "Bob" in "sip:Bob@x.y.z:port"
String username = ...     // get the Person object from the database
Person p = mf.getPerson(username);
if (p == null) {    // no person found in the database
response = SipServletResponse.SC_NOT_FOUND;
} else {  // the Expires header tells us if this is a registration or
            int expires = Integer.parseInt(req.getHeader("Expires"));
            if (expires == 0) {    // unregister
      response = handleUnregister(req, p);
            } else {            // register
       response = handleRegister(req, p);
 }
        }
    // send the response
   SipServletResponse resp = req.createResponse(response);
   resp.send();
    }
    /** Handle a registration request */
  private int handleRegister(SipServletRequest req, Person p)
     throws ServletException     {
       // Get the contact address from the request.  Prefer the
       // "Contact" address if given, otherwise use the "To" address
Address addr = sf.createAddress(req.getHeader("Contact"));
        p.setTelephone(addr.getURI().toString());
ModelFacade mf = (ModelFacade);
        getServletContext().getAttribute("Model");
        mf.updatePerson(p);
return SipServletResponse.SC_OK;
     }
}
```

Una vez registrados ambos usuarios, cada uno de ellos podrá ver a través de la interfaz web a qué usuario puede realizar una llamada porque también está registrado en el servicio tal y como muestra la siguiente figura.

Figura 6.6: Usuarios registrados en el servicio ClickToDial

Para realizar la llamada, bastará con seleccionar el botón de llamada de interfaz web invocando así a un HTTP Servlet que tendría el siguiente código:

```
/** Place a phone call when a user clicks */
public class PlaceCallServlet extends HttpServlet {

    // inject the SipFactory
    @Resource(mappedName="sip/ClickToDial")
    private SipFactory sf;
    /** Processes requests for HTTP POST method. */
    protected void processRequest(HttpServletRequest request,
```

```
        HttpServletResponse response)  throws ServletException,
        IOException      {

    // get the source and target from the request
    String sourceUserName = request.getParameter("source");
    String targetUserName = request.getParameter("target");
    // look up the relevant Person objects in the database
    ModelFacade mf = (ModelFacade)
        getServletContext().getAttribute("Model");
    Person source = mf.getPerson(sourceUserName);
    Person target = mf.getPerson(targetUserName);
    // get the phone numbers as SIP Addresses
    SipApplicationSession appSession = sf.createApplicationSession();
    Address to = sf.createAddress(source.getTelephone());
    Address from = sf.createAddress(target.getTelephone());
                            "INVITE", from, to);
    // store the second party's address in the sip session
    req.getSession().setAttribute("SecondPartyAddress", from);
    req.send();
  }
}
```

De esta manera el anterior Servlet obtiene los usuarios que participarán en la llamada, creando así una petición SIP de tipo INVITE que será procesada por otro SIP Servlets encargado de atender este tipo de peticiones. Es en el momento de recepción de esta petición cuando el SIP Servlet debe consultar al servidor de conocimiento para obtener así el nombre del anuncio que debe servir al usuario llamante.

Una vez conocido el anuncio a servir, comienza el diálogo SIP. El SIP Servlet encargado de atender las peticiones de tipo INVITE enviará una petición de este tipo al servidor multimedia (SEMS[33]) que reproducirá el anuncio solicitado al usuario que inició la llamada. La cabecera de la petición enviada por el SIP Servlet al media server tiene un formato especificado en la RFC 4240 [34]. En la cabecera de esta petición debe ir especificado el anuncio que debe ser reproducido por el servidor

multimedia, en un parámetro llamado *"play="*. Una vez finalizada la reproducción del anuncio, se finaliza la sesión para negociar la sesión con el destinatario de la llamada y establecer así la conversación solicitada.

El intercambio completo de mensajes durante la reproducción de anuncio y durante la conversación se muestra en la figura 6.7.

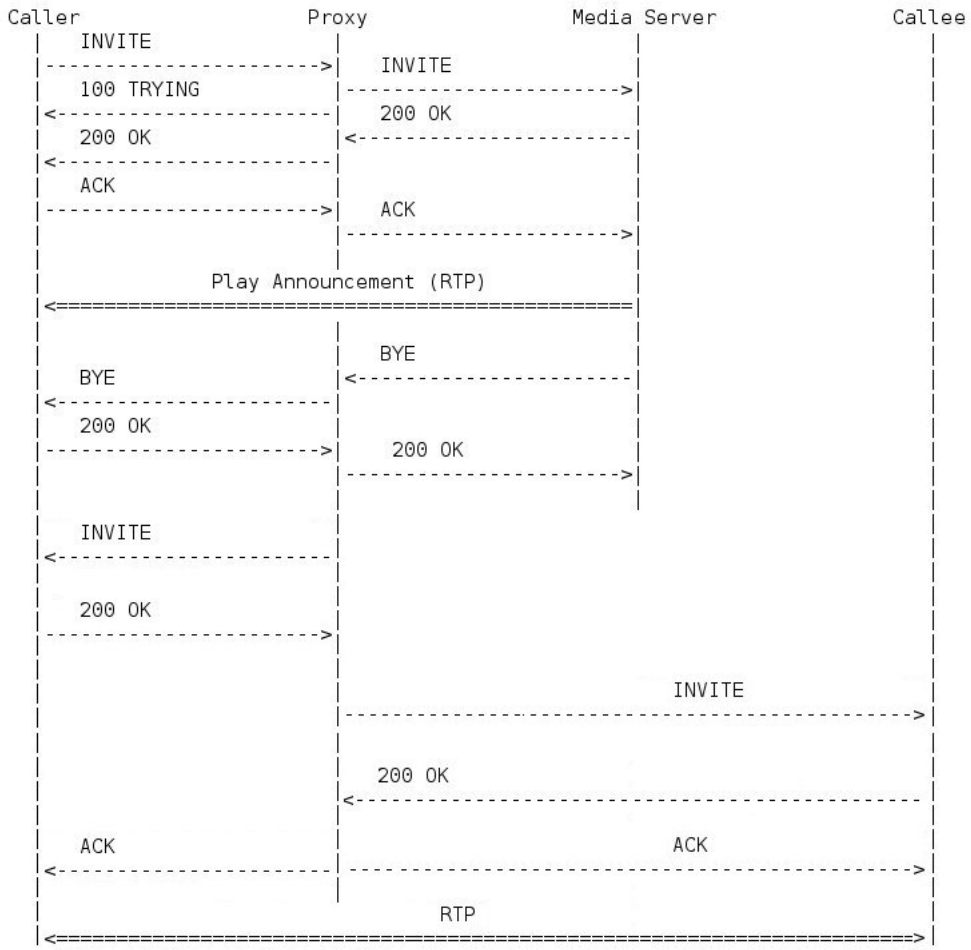

Figura 6.7: Diagrama de Trazas SIP en el servicio de anuncio personalizado

Una vez finalizada la llamada, al usuario que realizó la llamada se le mostrará por pantalla a cuánto asciende el coste total de la llamada. Para ello, el SIP Servlet que atiende los mensajes SIP de BYE deberá consultar al servidor de conocimiento ese precio, enviándole para ello los datos necesarios para la facturación: usuario llamanate, destinatario, hora del día y duración. El servidor de conocimiento generará una factura a partir de esos datos, realizando un descuento especial por haber escuchado un anuncio y el resultado de este proceso será mostrado en la interfaz del cliente.

Figura 6.8: Factura de la llamada realizada con el servicio Click To Dial

Por tanto el diagrama de acciones entre los diferentes servlets que toman parte en el transcurso de la actividad del servicio sería el siguiente:

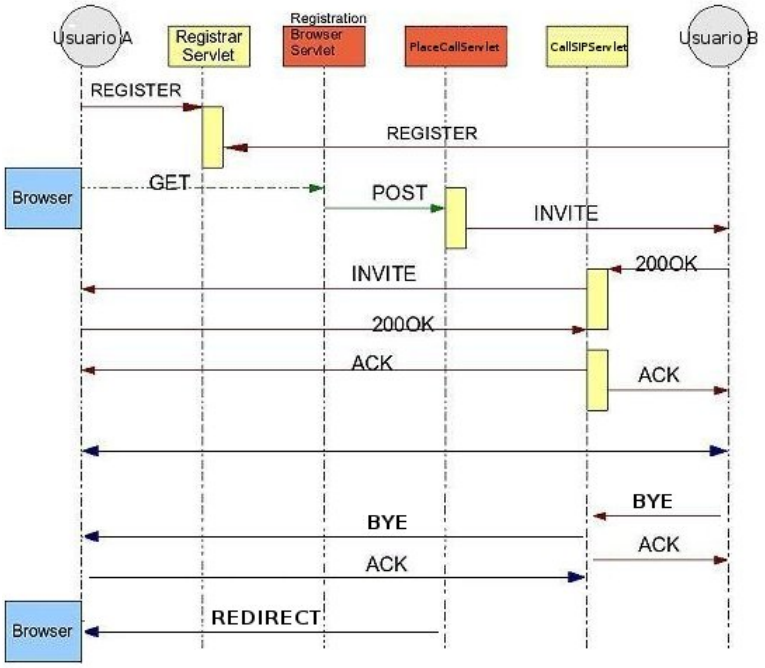

Figura 6.9: Secuencia Acciones ClickToDial

6.3.2. Selección de Anuncios

Para realizar la selección del anuncio, se emplea el componente de personalización previamente descrito. El servidor de conocimiento consta de tres bases de conocimiento a las que debe consultar para poder seleccionar un anuncio al usuario llamante.

La selección del anuncio se debe realizar en tiempo real, en el momento en el que se produce la llamada por lo que es conveniente minimizar las consultas a las bases de conocimiento que se realizan en el momento en el que se cursa una llamada. Es por ello, que se ha dividido el proceso de selección de anuncios en dos fases: una primera fase en la que se segmentará a los diferentes usuarios registrados en la plataforma y se clasificarán todos los anuncios disponibles y una segunda fase consistente en la puntuación de los anuncios según las afinidades de estos con el usuario que realiza la llamada.

La segmentación de la población, consiste en clasificar cada uno de los usuarios de la plataforma en cada uno de los segmentos de población que se han definido a partir de criterios de edad y estudio/trabajo. Esta segmentación se realizará cada vez que un usuario se registra en el sistema, o modifica uno de los campos que afecta a la segmentación. No es necesario hacer una segmentación de la población entera cada vez que se modifica un usuario, sino que basta con solo insertar a ese usuario en la base de conocimiento, partiendo del hecho de que los perfiles de usuario son estáticos en este caso.

La segmentación en este caso sería en base a criterios demográficos: edad y estudios/trabajo. De tal manera que se puedan determinar diferentes segmentos de población entre los que se pueda diferenciar de forma sencilla el poder adquisitivo y la edad del público al que se destinarán los anuncios.

La clasificación de los anuncios sigue criterios parecidos, clasificándolos en categorías de contenidos y precios cuando los anuncios son dados de alta o modificados. Asi pues la clasificación de los anuncios atiende a los siguientes criterios:

- Edad, diferenciando si van dirigidos a mayores de edad o no.
- Temática (Ocio, Cultura, Evento...)
- Precio, distinguiendo si va dirigido a un público con un poder adquisitivo elevado o no.

La última fase es la fase de elección del anuncio emplea una técnica de puntuación (*scoring*) de los anuncios según su afinidad al perfil del llamante (por edad, por aficiones, por nivel social (trabajo), región, etc) y a algunas características que tengan en común llamante, usuario llamado y anuncio. Una vez puntuados todos los anuncios siguiendo estos criterios, se elige el de mayor puntuación que será reproducido por el servidor multimedia para posteriormente cursarse la llamada solicitada.

La figura (6.10) representa el flujo de la acción: segmentación de la población, clasificación de anuncios y selección del anuncio. Esa imagen ha sido generada con el plugin de drools para eclipse que permite definir flujos de reglas.

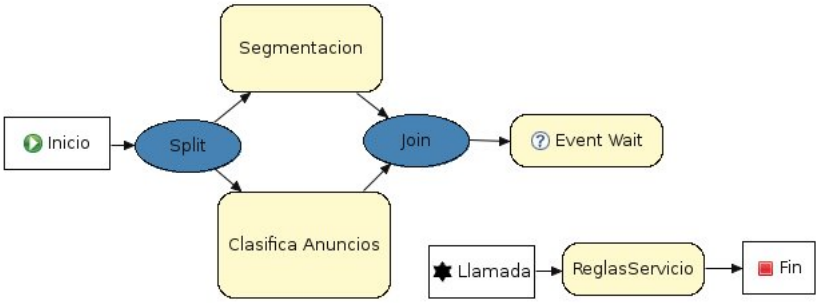

Figura 6.10: Flujo de Inferencias para la personalización de un anuncio

7 Diseño Servicio Facturación

7.1. Introducción

Existen multitud de plataformas que permiten facturar llamadas a partir de CDR(a2billing[31], jbilling[47]...) pero la mayor parte de ellas, a excepción de jbilling, no permiten definir ofertas personalizadas por lo que en esta plataforma se ha programado una herramienta que permite la definición de reglas de facturación personalizadas a cada usuario.

Son muchas las ventajas que aporta la utilización de la VoIP frente a la telefonía tradicional, entre ellas la reducción de las facturas, es por ello que la implementación de este servicio se ha enfocado en cómo conseguir un descenso radical de las facturas mediante el empleo de la tecnología VoIP.

En este capítulo, se describirá un escenario típico que justifica la utilidad de este servicio y se describirá, asimismo, el servicio de facturación implementado para la plataforma.

7.2. Escenario

En internet existen muchos proveedores de VoIP que ofrecen servicio telefónico permitiendo a sus usuarios obtener tarifas a menor precio frente a las que obtienen usando la telefonía tradicional. Esto es posible gracias a que estos proveedores de VoIP tienen contratadas un gran número de tarifas por lo que pueden cursar las llamadas seleccionando la tarifa más ventajosa y cobrando a sus usuario una mínima diferencia que les permita obtener beneficios.

En el diseño del sistema de facturación, hemos considerado que podemos ofrecer un servicio similar al que ofrecen los servidores de VoIP, limitando el escenario al ámbito de las llamadas a teléfonos móviles en España. Así pues, hemos considerado que tenemos contratadas las tarifas de los principales operadores de móviles, de tal manera que cuando un usuario use nuestro servicio de llamadas a traves de interfaz web realice las llamadas con la tarifa más ventajosa.

7.3. Descripción del Servicio

El servicio descrito en el anterior escenario estaría basado por dos módulos independientes: "selección de tarifas y operadoras" y "facturación de las llamadas". El servicio consiste en una interfaz web en la que un usuario puede seleccionar el nombre del usuario del que quiere obtener la factura mensual. El sistema procesará entonces todas las llamadas registradas del usuario solicitado mostrando por pantalla el detalle de la factura en el caso de ser facturado por la compañía que tiene contratada y en el caso de usar el servicio ofrecido por la plataforma, para así poder notar la gran diferencia existente entre ambos valores.

7.3.1. Módulo Selección de Operadoras

El sistema de selección de operadoras se basa en la idea de seleccionar la operadora con precio más económico en función de determinados parámetros relacionados con el momento en el que se realiza la llamada o las tarifas ofrecidas por las diferentes operadoras. Este proceso debe ser transparente para el usuario, de forma que el usuario realice una llamada y el sistema se encargue de la selección de la opción más ventajosa sin que el usario se vea implicado.

Cuando se realiza una llamada, en la que intervienen el usuario llamante y el usuario llamado, se conocen parámetros como la hora a la que se realiza la llamada, la operadora del llamante y la operadora del llamado. En este momento, se insertan en la base de conocimiento los siguientes hechos: la llamada que ha originado el proceso y las diferentes tarifas entre las que habrá que elegir.

Una vez insertados los hechos, se disparan las reglas para que el sistema elija la tarifa más económica según los parámetros de la llamada. En nuestro sistema se han definido tres franjas horarias: mañana (8:00-15:59), tarde (16:00-23:59) y noche (0:00-07:59). Las tarifas insertadas en la base de conocimientos están definidas por la operadora de origen, la de destino y los diferentes precios en las distintas franjas horarias. Las reglas están diseñadas de forma que:

- Primero se descartan todas aquellas tarifas cuya operadora de destino no coincida con la operadora de destino de la llamada que ha originado el proceso.
- Por último se dispara la regla correspondiente a la franja horaria en la que se realiza la llamada y se obtiene la tarifa más económica.

El código sería el siguiente:

```
package upm
rule "Tarifa Mañana"
    when

        $llamada : Llamada( $hora : hora , $tarifaLlamada : tarifa )          $
        eval( $hora >= 8 && $hora < 16 )
    then
        $llamada.setTarifa($tarifa);
        update($llamada);
    end
rule "Tarifa Tarde"
    when

        $llamada : Llamada( $hora : hora , $tarifaLlamada : tarifa )
        $tarifa : Tarifa( precioTarde < ($tarifaLlamada.getPrecioTarde()) )
        eval( $hora >= 16 && $hora < 24 )
    then
        $llamada.setTarifa($tarifa);
        update($llamada);
```

```
    end

rule "Tarifa Noche"

    when

        $llamada : Llamada( $hora : hora , $tarifaLlamada : tarifa )
        $tarifa : Tarifa( precioNoche < ($tarifaLlamada.getPrecioNoche()) )
        eval( $hora >= 0 && $hora < 8 )

    then

        $llamada.setTarifa($tarifa);
        update($llamada);

    end

rule "Descarta Tarifas"
no-loop true

    when

        $llamada : Llamada( $llamado : llamado )
        $tarifa : Tarifa( $operadorDestino : operadorDestino )
        eval( $llamada.getLlamado().getOperador() != $operadorDestino )

    then

        retract($tarifa);

    end
```

Estas reglas pueden ser escritas en lenguaje natural, a partir de un fichero de Excel o en el lenguaje de drools como será descrito en el capítulo B. Una disparadas estas reglas el servidor de conocimiento devolverá al servidor telco información sobre con qué tarifa debería realizarse la llamada. El sistema. una vez que tiene esta información, cursará la llamada empleando esa tarifa, en un proceso totalmente transparente al usuario.

7.3.2. Sistema de facturación

El sistema de facturación de llamadas de postpago es el encargado de elaborar la factura mensual de consumo de llamadas de los usuarios. El proceso ha de ser de nuevo transparente al usuario, de forma que éste realice todas sus llamadas y al finalizar el periodo de facturación, el sistema le proporcione un informe detallado de sus llamadas, indicando el número de llamadas que ha realizado a cada operadora, el horario en el que se han realizado, la duración de dichas llamadas y el precio total asociado al consumo en el periodo correspondiente. Los parámetros que definen cada llamada realizada en el periodo de facturación se obtienen a partir de los logs similares a los generados por Asterisk, es decir; un fichero que contiene los detalles de la llamadas(CDR). Además de poderse aplicar una facturación normal de acuerdo a unas tarifas preestablecidas, es posible definir ofertas personalizadas: número favorito, descuentos por llamadas a usuarios frecuentes, etc.

Cuando se solicita la facturación de las llamadas de un usuario, el sistema comienza analizando el contenido del log generado, obteniendo todos los parámetros necesarios para elaborar la factura: operadora de destino, hora y duración de cada una de las llamadas realizadas. Todas estas llamadas son insertadas en la base de conocimiento junto con el usuario llamante.

Una vez insertados los hechos, se disparan las reglas. El conjunto de reglas se ha diseñado de forma que hay una regla para cada tipo de llamada según la operadora de destino y la franja horaria en la que se realiza. Además, existen una serie de reglas para ofertas personalizadas como la de número favorito.

Finalmente, el servidor de conocimiento, una vez procesado todo el CDR devolverá al Servidor Telco de Aplicaciones la información solicitada de tal manera que éste la mostrará al usuario.

7.3.3. Interfaz Web

Los módulos mencionados están desplegados en el servidor de conocimiento por lo que falta por describir cómo interactúa la aplicación con el servidor. La interfaz de

facturación, que provee a los usuarios de una intuitiva página web representada en la siguiente figura:

Figura 7.1: Interfaz de Facturación

La secuencia de acciones que tiene lugar en el proceso de generación de la factura sería la siguiente:

1. Usuario elige el usuario del que desea obtener la factura mensual.
2. El LoginServlet se conecta al servidor de conocimiento para realizar la consulta de la factura.
3. El servidor de conocimiento recibe la información enviada por el servlet, que solamente es el nombre del usuario del que se desea obtener la factura, y extrae las llamadas de un fichero de llamadas de asterisk de dicho usuario.
4. El servidor de conocimiento, dispara las reglas correspondientes a la facturación con la tarifa asociada al usuario y devuelve la respuesta con los datos de la factura al Servlet que le envió la información.

5. El servidor de conocimiento comienza a procesar cada una de las llamadas del usuario para asociarlas a la tarifa más ventajosa posible, haciendo uso del servicio de selección de operador.
6. El servidor de conocimiento genera la factura empleando los resultados obtenidos en el paso anterior y devuelve la factura al servlet.
7. El servlet muestra al usuario las facturas solicitadas.

Figura 7.2: Interfaz de Facturación

8 Conclusiones y Trabajos Futuros

8.1. Introducción

Una vez finalizado el proyecto se puede analizar desde otra perspectiva el grado de consecución alcanzado en el mismo, así como los diferentes factores que han influido en su desarrollo.

En este capítulo se intentará no sólo valorar el trabajo que se ha llevado a cabo y las dificultades que han surgido durante todo el proceso, sino establecer las bases para posibles trabajos futuros.

8.2. Trabajo Desarrollado

La forma más objetiva de realizar una valoración del trabajo realizado a lo largo un proyecto es comprobar si se han cumplido los requisitos especificados al comienzo del mismo. Así pues en este apartado se tratará de valorar el grado de satisfacción tanto de los requisitos funcionales como de los no funcionales.

8.2.1. Requisitos Funcionales

- **Login de Usuarios:** Uno de los elementos claves a la hora de personalizar los servicios. Se ha conseguido implementar este requisito, considerando que los usuarios rellenaron un perfil de usuario al darse de alta en el sistema.
- **Servicio ClickToDial:** El Servicio ClickToDial es uno de los servicios fundamentales de la plataforma, que permite a los usuarios que hayan iniciado

sesión en ella establecer una conversación. Se ha conseguido que este servicio ofrezca una calidad de audio aceptable y que pueda ser usado por más de 2 usuarios simultáneamente.

- **Mostrar Usuarios en la Página Web:** Este requisito es clave para un funcionamiento correcto del servicio anteriormente mencionado. La plataforma muestra a los usuarios en una página web todos los usuarios que hayan iniciado sesión en ella, de tal manera que puedan seleccionar al usuario con el que quieren establecer una conversación.
- **Servicio Facturación**: Es uno de los componentes principales de la plataforma, el servicio de facturación implementado ofrece la posibilidad de facturar las llamadas realizadas por un usuario ya sea al finalizar la llamada o cuando es solicitada la generación de una factura mensual. Además permite la definición de tarifas personalizadas a los usuarios de la plataforma.
- **Base de Conocimiento conforme a las necesidades del cliente:** Las bases de conocimiento definidas permiten la personalización de los servicios ofertados. Además, otro requisito existente es la necesidad de que la consulta a las mismas se pueda realizar en tiempo real sin que el usuario perciba retardo alguno, requisito que se ha cumplido en los servicios donde era necesario.
- **Logs de los Servidores:** Todos los elementos empleados en los servicios implementados ofrecen logs durante la ejecución de los servicios, que nos ofrecen información sobre el estado de la ejecución de la lógica de negocio. Estos logs son mostrados en la consola de cada componente, por lo que sería interesante ofrecer la posibilidad de poder recoger los logs de la aplicación en ficheros de texto de forma automática.

8.2.2. Requisitos no funcionales

- **Sistema Multiusuario:** El sistema permite el login simultáneo de múltiples usuarios, asi como la ejecución de varios servicios de forma simultánea por parte de los usuarios.
- **Escalabilidad:** La plataforma permite el uso de los servicios por varios usuarios de forma simultánea sin que ello conlleve un empeoramiento de la calidad

de servicio.

- **Diseño de módulos o componentes funcionales totalmente independiente del resto de elementos**: Los diferentes módulos implementados en los servicios de la plataforma son totalmente independientes unos de otros, de tal manera que una ligera modificación del código de un módulo permite la integración de ese módulo en nuevos servicios.
- **Extensibilidad:** El actual diseño de cada uno de los servicios de la plataforma permite extender la funcionalidad de manera sencilla mediante el uso de módulos como los anteriormente descritos.
- **Compatibilidad**: El sistema permite el empleo de los servicios que ofrece desde diferente tipos de terminales, cuyos requisitos mínimos sean tener un navegador web y un cliente SIP.
- **Amigable e intuitivo**: La plataforma presenta una interfaz web muy intuitivo para los usuarios de tal manera que no requiere un profundo conocimiento de la plataforma para su correcta utilización.

8.2.3. Conclusiones

Como se puede comprobar en los apartados previos, se han cumplido todos los requisitos tanto funcionales como no funcionales, pudiendo añadirse mejoras en la recopilación de logs del sistema. Por lo tanto puede considerarse que se ha completado el desarrollo del proyecto de manera exitosa.

8.3. Dificultades y Limitaciones

El proceso de implementación de los servicios e implementación de la arquitectura ha presentado ciertas dificultades, muchas de las cuales se han podido solventar, aunque otras han tenido que ser evitadas optando por otras soluciones.

El análisis de los componentes que forman parte de la arquitectura ha sido detallado, analizando cada una de las soluciones propuestas e instalándolas para ser

probadas. En verano de 2008, el proyecto OpenSER se dividió en dos subproyectos como ya fue descrito en la sección 3.2.2. Por lo que fue necesaria la instalación del proyecto OpenSIPS que no ofrecía la posibilidad de instalación mediante repositorio ni facilidad de configuración en sus primeras versiones. Finalmente, en octubre de 2008, se consiguió una instalación exitosa y nos decantamos por su utilización en la arquitectura de la plataforma.

La mayor dificultad de estre proyecto, ha sido la gran cantidad de tiempo invertida en probar plataformas que permitiesen el despliegue de SIP Servlets, pues durante muchos meses Sailfin no se mostró como un proyecto maduro capaz de soportar adecaudamente esta tecnología. Finalmente, como en el caso anterior, el paso del tiempo permitió el lanzamiento de una versión más madura que es la que se emplea actualmente en el despliegue de los servicios de la plataforma.

Por último, una de las dificultades que han estado presentes en la última etapa del proyecto ha sido encontrar un proyecto capaz de poder desplegar las bases de conocimientos definidas con Drools. Se estudio la posibilidad de emplear Guvnor(Sección B) como repositorio de reglas, pero debido a la falta de documentación relacionada con el proyecto se optó por desplegar de forma temporal las bases de conocimiento sobre un servidor Java.

Creo conveniente destacar la gran ayuda aportada en las diferentes listas de correo de los proyectos empleados en la plataforma, sin la cual no habría sido posible solventar ciertos problemas que tuvieron lugar durante el proyecto.

8.4. Trabajos Futuros

Una vez concluido el proyecto, al menos de acuerdo a lo que fue fijado desde un principio, es necesario pararse a pensar sobre hacia dónde debe estar orientado a partir de ahora para poder añadir funcionalidades a la plataforma y poder añadir mejoras a los servicios ya implementados. Las siguientes secciones están dedicadas a describir brevemente cuáles pueden ser estas mejoras y esta orientación del proyecto.

8.4.1. Integración en la Arquitectura

En el capítulo 3, se definió una arquitectura de VoIP sobre la cuál debían ir desplegados los servicios implementados por la plataforma. En la actualidad, estos servicios no se encuentran completamente integrados en la arquitectura descrita por lo que se cree conveniente trabajar en ese sentido a partir de ahora. Es necesario autenticar a los usuarios en el Servidor de registro SIP que ha sido seleccionando, delegando la responsabilidad de añadir funcionalidad al sistema al servidor telco de aplicaciones. De esta manera, lo usuarios registrados en el servidor SIP tendrían a su alcance las mismas funcionalidades descritas hasta ahora, pero además dispondrían de una mayor seguridad gracias al soporte TLS que aporta OpenSIPS.

8.4.2. Actualización del Perfil de Usuario

Como ya fue mencionado en la sección 6.3, actualmente los usuarios de la plataforma disponen unicamente de perfiles estáticos, no pudiendo obtener información alguna sobre su experiencia en el sistema a partir de la interacción con las aplicaciones de la plataforma. Sería interesante poder implementar un mecanismo de realimentación de información que permitiese al usuario, en el caso de recibir un anuncio personalizado, evaluar si el anuncio recibido le ha resultado útil o no para asi poder mejorar la eficiencia del sistema de selección de anuncios.

8.4.3. Interfaz de Modificación de la Base de Conocimiento

Como va a ser descrito en el capítulo B, resultaría de gran utilidad la existencia de una plataforma que permita editar las reglas de personalización de las diferentes aplicaciones sin necesidad alguna de conocer en detalle la arquitectura sobre la que se encuentran desplegadas. Es por ello, que resultaría interesante trabajar en la edición de reglas que permite la plataforma Guvnor de tal manera, que tras la modificación de las reglas, se puedan desplegar en el servidor de conocimiento que en un futuro se encontrará en la dicha plataforma.

A Instalación de la Plataforma

A.1. Introducción

En el presente capítulo se explicarán los pasos necesarios para tener instalada la plataforma de aplicaciones explicada en esta memoria.

Debido a la gran cantidad de componentes necesarios para la arquitectura VoIP aquí definida se ha decidido, para facilidad del programador, generar una máquina virtual VMware[40] que ya contenga todo instalado. De esta manera el programador no tendrá que preocuparse por la configuración de los diferentes elementos y podrá centrarse en la implementación y despliegue de nuevos servicios.

A.2. Contenido Máquina Virtual VMware

Es necesario especificar cuál será el software instalado en la máquina virtual para que así el programador sepa cuáles deben ser los requisitos del sistema sobre el que desplegar la máquina virtual, de la misma manera que sepa con qué elementos cuenta para la implementación y despliegue de servicios.

La máquina virtual tiene un Sistema Operativo Ubuntu 9.04 con el siguiente software instalado y configurado:

- Servidor Asterisk versión 1.4
- Servidor OpenSIPS con soporte a TLS versión 1.4
- Servidor Multimedia SEMS versión 1.0.1.
- Servidor para despliegue de Aplicaciones J2EE y SIP Servlets Sailfin.

- Entorno de desarrollo Netbeans 6.1
- Entorno de desarrollo Eclipse 3.4 con Drools instalado.
- Entorno de edición de reglas Guvnor desplegado sobre Tomcat.

Con todas estas aplicaciones instaladas, es importante destacar que esta máquina virtual ha sido probada con diferentes cantidades de memoria asignada, concluyendo que para un correcto funcionamiento es necesario asignar a la máquina 1GB de RAM. Esto implica que el sistema sobre el que se despliegue el servidor VMware debe tener unas características ligeramente superiores.

A.3. Instalación Servidor VMware

Una vez establecido cuál es el contenido de la Máquina Virtual, es necesario explicar cuáles son los pasos a seguir para la instalación del Servidor sobre el que se desplegará la máquina. Para ello, partimos de la premisa de que el sistema sobre el que se desplegará será un sistema Debian/Ubuntu.

Para la instalación de VMware Server lo primero que se debe realizar es la descarga del instalador. Para ello debemos acceder a http://www.vmware.com/download/server/ y descargar la última versión disponible ser servidor. Debido a la licencia de VMware, será necesario aportar una serie de datos para obtener la licencia correspondiente y que el servidor funcione correctamente. Debido a que nos encontramos en un sistema Debian/Ubuntu se elegirá la descarga del fichero .tar del servidor.

Una vez descargado, será necesaria la instalación de una serie de librerías para la compilación correcta del servidor. Para ello se debe ejecutar las siguientes órdenes en una consola.

```
sudo apt-get install linux-headers-‘uname -r‘ build-essential xinetd
```

Una vez instaladas las cabeceras del kernel sobre el que se desplegará el servidor, se procede a la instalación del servidor. Para ello es necesario acceder al directorio donde fue descargado el servidor VMware y ejecutar:

```
tar xvzf VMware-server-x.y.z-wwwwww.i386.tar.gz
```

Donde x,y,z,w son caracteres que representan la versión de VMware que estemos instalando. Con esta orden extraemos el contenido del fichero que hemos descargado. A continuación, accedemos al directorio del servidor y realizamos la instalación ejecutando lo siguiente:

```
cd vmware-server-distrib/
sudo ./vmware-install.pl
```

Durante la instalación, el script que se ejecuta realizará una serie de preguntas sobre dónde se desea instalar el servidor, se pueden usar las opciones por defecto que se sugieren o modificarlas según la costumbre del programador. Una vez finalizada la instalación se debe introducir la licencia.

A diferencia de las versiones previas de VMware, la nueva versión del servidor (2.x) no ofrece ninguna aplicación de escritorio para la gestión de las máquinas virtuales. VMware decidió migrar esta aplicación al explirador web, de tal manera que se pueda acceder al servidor de manera sencilla sin necesidad de instalación de un cliente en el PC desde el que se intente acceder.

Para poder acceder a la consola se debe lanzar un cliente, por ejemplo Firefox, y poner en la barra de direcciones https://w.x.y.z:8333/ (w.x.y.z es la IP del Servidor).

Una vez accedida a la consola se deben introducir los datos de usuario y contraña de acceso al servidor para poder gestionar las máquinas virtuales.

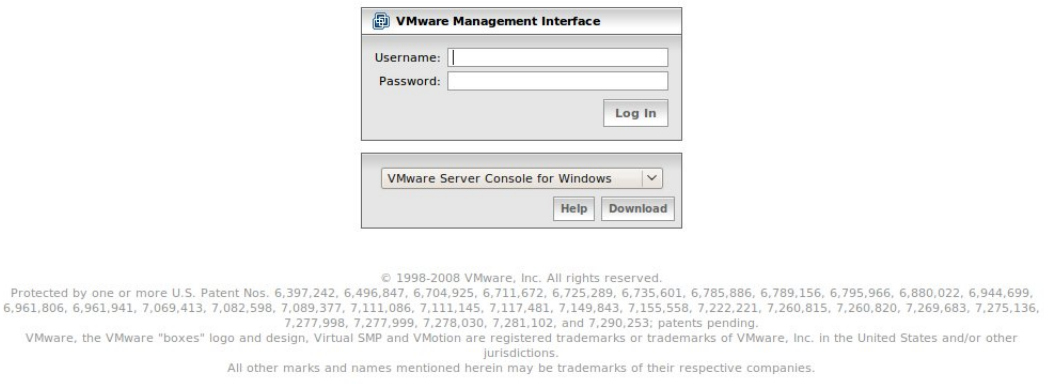

Figura A.1: Interfaz VMware: Usuario y Contraseña

Una vez introducidos los datos correctamente, accedemos a la interfaz de control de las Máquinas virtuales, donde podemos ver el rendimiento de cada una de las máquinas virtuales, ver el estado del servidor, encender máquinas, acceder a sus consolas...

Figura A.2: Interfaz VMware: Información Máquinas Virtuales

La imagen anterior muestra una serie de máquinas corriendo en el servidor. En nuestro caso estaríamos interesandos en la Máquina Telcoblocks, disponible en la página del proyecto(http://telcoblocks.germinus.com). Una vez descargada la máquina virtual, se debe trasladar al directorio donde el servidor VMware almacena las Máquinas virtuales (/var/lib/vmware/Virtual Machines/). Desde la interfaz de gestión de las máquinas aparecerá automáticamente listada y podrá ser encendida tal y como muestra la siguiente imagen.

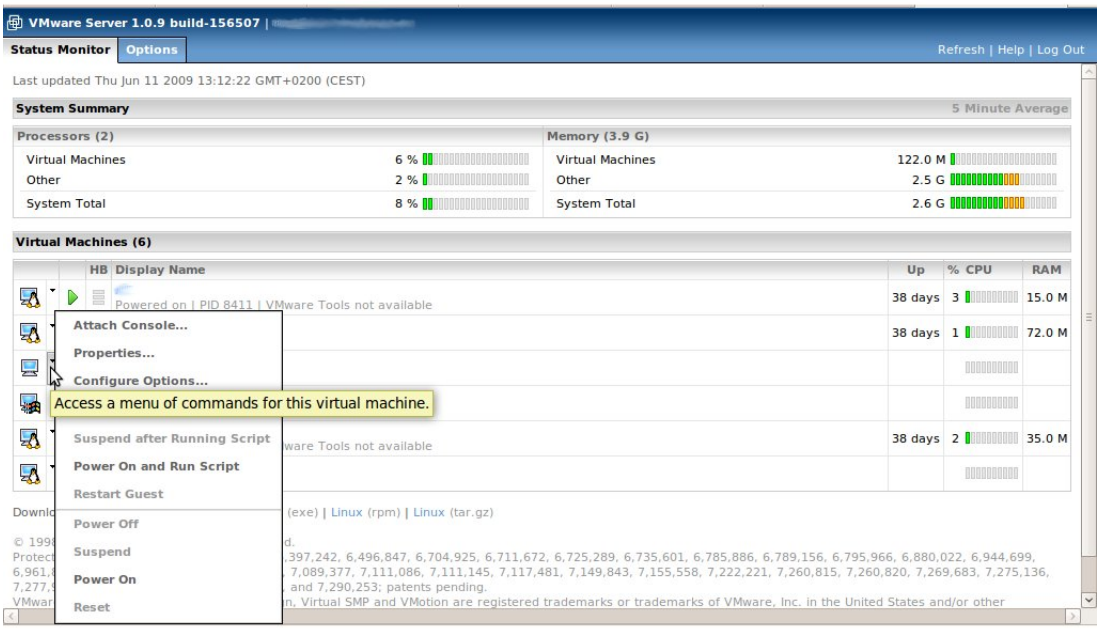

Figura A.3: Interfaz VMware: Encendido Máquina Virtual

A.4. Conclusión

Una vez arrancada la máquina virtual, el programador dispone de todos los servicios de la plataforma correctamente configurados, de tal manera que no deberá configurar desde cero la plataforma.

Se considera la existencia de esta máquina virtual clave para la misión del proyecto: despliegue rápido de servicios VoIP.

B Instalación de Guvnor y Edición de Reglas

B.1. Introducción

En el presente capítulo se hará una descripción sobre la plataforma de edición de reglas del proyecto Drools, llamada Guvnor. Además de plataforma de edición de reglas, como veremos en las secciones siguientes, Guvnor es también un repositorio de reglas en el que pueden ser desplegadas y consultadas las bases de conocimiento, la descripción de esta funcionalidad está fuera del ámbito del presente documento.

B.2. Arquitectura de la Interfaz de Gestión de Reglas de Negocio

Una interfaz de gestión de reglas de negocio(BRMS) permite a los usuarios modificar las reglas definidas en el sistema gracias al empleo de una interfaz amigable y sencilla de usar, sin necesidad de conocer un lenguaje específico para poder entenderlas.

En Drools, plataforma escogida para la implementación de este componente, la interfaz web que permite gestionar las reglas a partir de una interfaz web recibe el nombre de Guvnor. Es aconsejable el uso de esta interfaz en los siguientes supuestos:

- Necesidad de gestionar distintas versiones de reglas.
- Permitir acceso a las reglas a usuarios de diferente grado de especialización, permitiéndoles la modificación y creación de reglas.

- No existencia de otra interfaz de Gestión de Reglas.
- Necesidad de gestionar un elevado número de reglas.

La instalación de la interfaz es sencilla. La aplicación Guvnor es deplegada como un fichero .war en cualquier servidor de aplicaciones o contenedor de servlets, lo que la hace realmente atractiva para los usuarios. En el caso de este proyecto Guvnor ha sido desplegado sobre el servidor de aplicaciones de JBoss.

Como se puede observar en la figura B.1, Guvnor necesita el empleo de una base de datos capaz de almacenar series de reglas, para ello es posible el empleo de un JCR(Contenedor de Repositorios Java) estándar. La implementación más común es Jackrabbit[48], que ofrece un motor de almacenamiento que puede ser configurado como una base de datos o para poder emplear algún RDBMS(Sistema Administrador de Bases de Datos Relacionales) si fuera necesario.

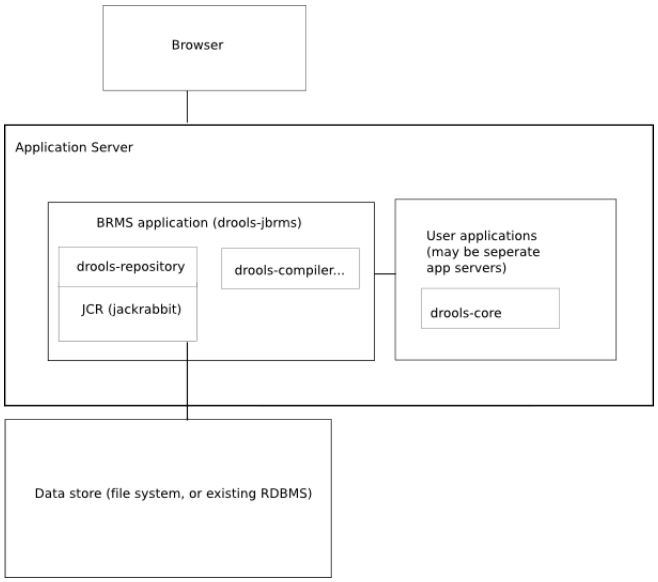

Figura B.1: Arquitectura BRMS

B.3. Descripción Guvnor

En esta sección se hará una breve descripción de Guvnor y qué se puede gestionar empleando esta interfaz.

Primeramente es necesario, como ya se ha comentado, descargar la aplicación de la página de JBoss y desplegarla en un servidor de aplicaciones, en este caso el servidor de aplicaciones de JBoss.

Una vez desplegada la aplicación, para poder acceder a ella es necesario lanzar un explorador web, ya sea Firefox o IE, poniendo en la barra de direcciones: http://<servidor>/drools-guvnor/.

En caso de ser la primera vez que lancemos Guvnor, debemos acceder a la plataforma usando el nombre de usuario: Admin y ninguna contraseña, una vez accedida a la plataforma es posible establecer una contraseña de acceso. Tras acceder a la plataforma se nos solicitará si queremos instalar ejemplos, que no serán útiles para familiarizarnos con la interfaz.

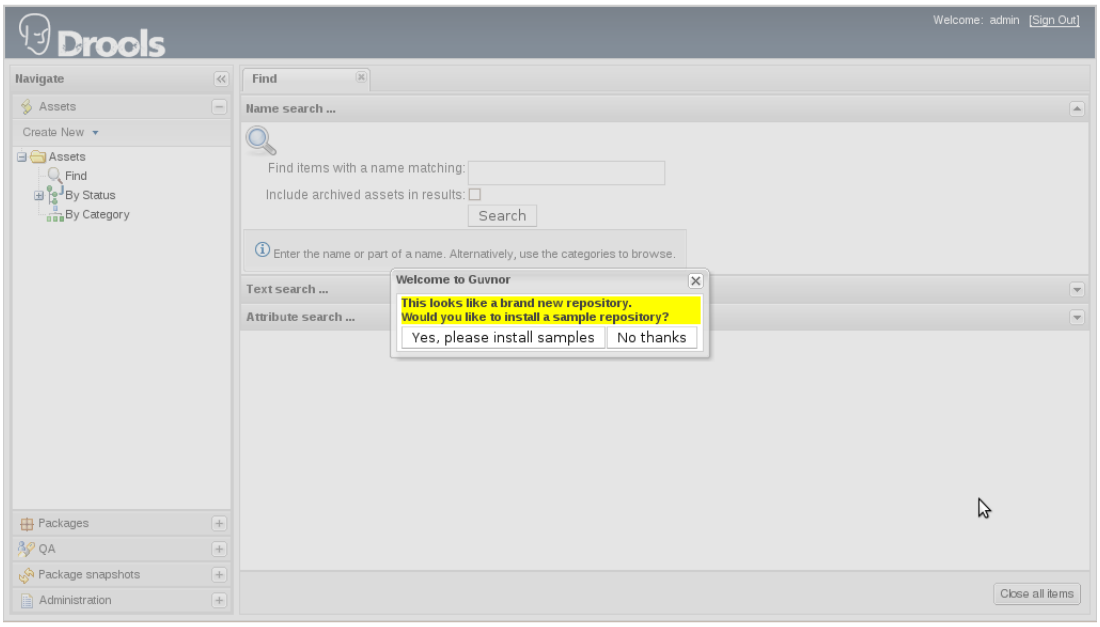

Figura B.2: Acceso Plataforma Guvnor

Tras decidir si deseamos o no instalar los ejemplos, nos encontramos ante la pantalla principal de la interfaz.

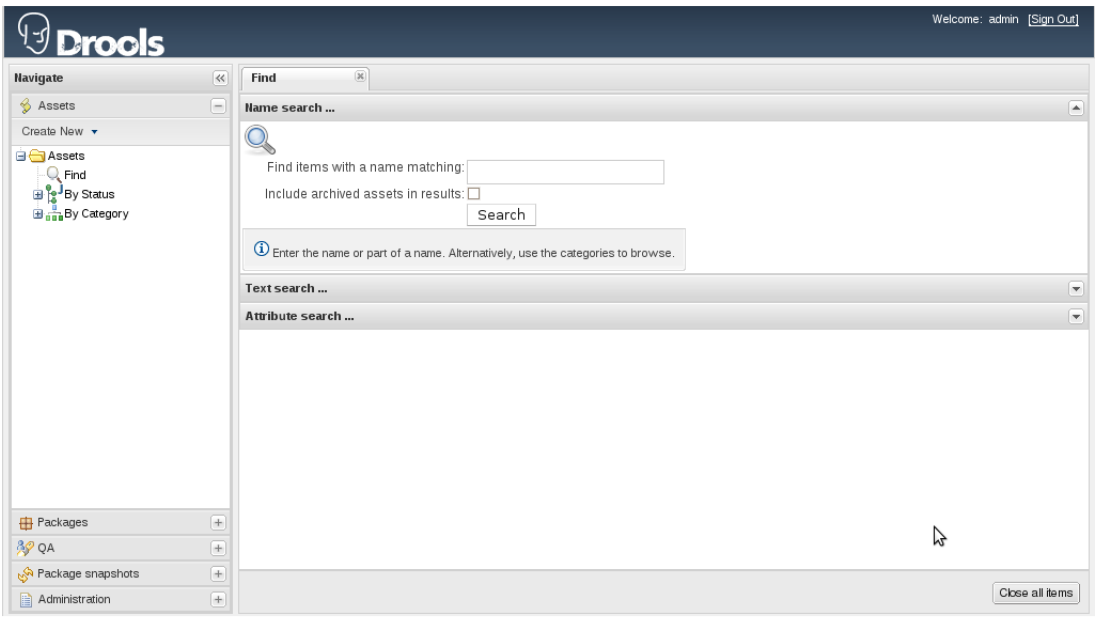

Figura B.3: Interfaz Guvnor

Las funcionalidades de este menú son las siguientes:

- Info: Pantalla inicial con links a los recursos que tiene Guvnor.
- Rules - Reglas: Categoría para poder gestionar las reglas de negocio.
- Package: Categorías para configurar y gestionar paquetes.
- Deployment: Despliegue de los diferentes repositorios.
- Admin: Funciones de Administracion(categorías, estado, importar y exportar).

Es importante describir con más detalle qué es una categoría y un paquete ya que son fundamentales para la definición de reglas y hechos en Guvnor.

Categorías

Necesitamos crear al menos una categoría, bajo la cual podremos almacenar las reglas que se vayan creando. Las categorías tienes el propósito de clasificar las reglas, por lo que es importante asignarlas nombres con significado.

Paquetes

Un paquete es el lugar donde van a ser almacenadas las reglas. También se incluye dentro de paquete todas aquellas clases importadas y variables globales.

B.3.1. Definición de Reglas con menú desplegable

La siguiente imagen muestra una de las posibilidades de definición de reglas que tiene la plataforma, que es la definición de las mismas a partir de las opciones que te da la interfaz web mediante menús desplegables.

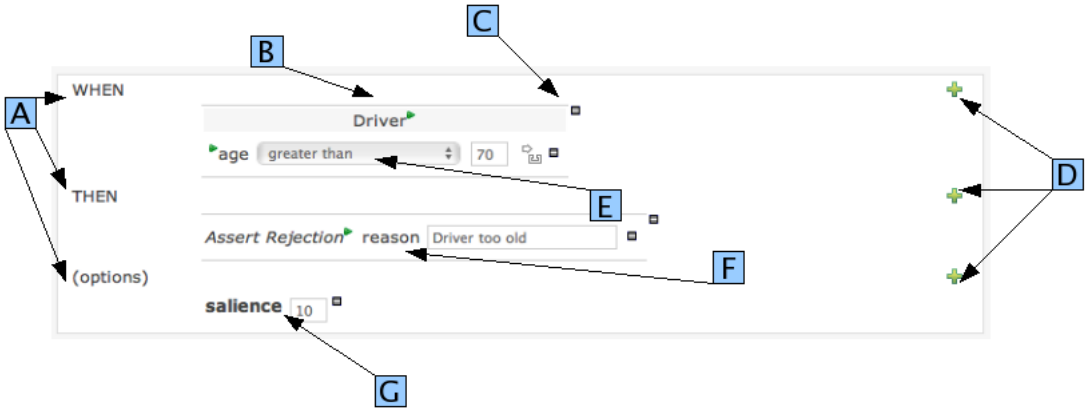

Figura B.4: Definición Reglas Guvnor

Cada una de las letras de la imagen anterior es descrita a continuacion.

A: Identifica las diferentes partes que componen una regla. Parte izquierda o condiciones de disparo(WHEN), acción en caso de cumplirse las condiciones o parte

derecha de la regla(THEN) y por último "(options)" donde se pueden añadir una serie de atributos si así se desea.

B: En este caso, la regla define que se esta buscando un hecho de tipo “Driver”. Se pueden especificar una serie de parámetros que además deba cumplir el hecho para poder ejecutar las acciones.

C: El botón "-" indica que se puede borrar la declaración “Driver”.

D: El símbolo "+" permite poder añadir más condiciones que se deben cumplir en la parte izquierda de la reglas. Cuando se pulsa, aparece una nueva ventana que lista los tipos de hechos que pueden aparecer para ser evaluados.

E: Muestra el menú desplegable en el que se puede indicar la condición que debe cumplir el parámetro “age” del hecho “Driver”.

F: Definición de la parte derecha de la regla. Acciones.

G: Opciones adicionales que se quieran añadir.

B.3.2. Definición de Reglas en Lenguaje Común

Además de la forma ya mencionada, es posible la definición de reglas en lenguaje corriente, es decir en inglés; sin necesidad de conocer la sintaxis del lenguaje de reglas de Drools.

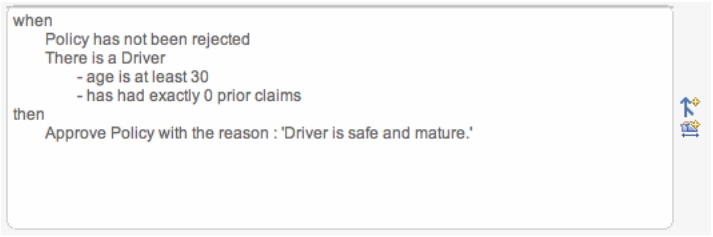

Figura B.5: Regla escrita en DSL

B.3.3. Definición de Tablas de Decisiones a través de Ficheros de Excel

Asi mismo, también es posible la definición de reglas a partir de subir un fichero Excel que las contenga a Guvnor. Guvnor tiene una opción que permite la subida de este tipo de ficheros a la plataforma.

Cuadro B.1: Tabla de Decisión Excel

También es posible definir tablas de decisiones, al igual que mediante ficheros, desde la misma interfaz web, de forma que Guvnor va orientando al usuario sobre las posibilidades de definir reglas mediante menús desplegables.

B.3.4. Definición de Reglas en Lenguaje Técnico

Por supuesto, también es posible la definición de reglas en el lenguaje que Drools ha diseñado para ello. Estas reglas son almacenadas como textos en ficheros DRL. Cada fichero puede ser o bien una serie de reglas, o una única regla.

```
salience 100 #this can short circuit any processing
when
    a : Approve()
    p : Policy()
then
    p.setApproved(true);
    System.out.println("APPROVED: " + a.getReason());
```

Figura B.6: Definición de Reglas en Ficheros DRL

B.4. Validación de Reglas

Guvnor dispone de una herramienta capaz de validar que la sintaxis de definición de las reglas es la correcta, pero además es necesario otro tipo de herramientas para confirmar que efectivamente cada una de las reglas realiza su cometido.

Al igual que en Java existe JUnit para validar el código, es necesario el empleo de una aplicación similar para validar las reglas. Las reglas cambian con el paso del tiempo y por ello es necesario probarlas cada vez que esto ocurre.

Existe una plataforma, FIT For Rules[49], que permite la validación de reglas que fue construída sobre FIT (*Framework for Integrated Testing*). Esto permite que los resultados de la validación de las reglas estén recogidos en documentos tipo Word o Excel. Fit For Rules funciona de la siguiente manera, lee una serie de datos que contienen los hechos a ser introducidos en la base de conocimiento y dispara las reglas correspondientes, escribiendo el resultado en el mismo fichero si asi se desea, comprobando si los resultados de dichas reglas son idénticos a los esperados.

Debido a que estos documentos no son código, pueden ser utilizados por usuarios que no tengan un dominio del lenguaje de reglas y que no estén muy familiarizados con las reglas.

Un ejemplo de ejecución de esta herramienta está recogido en la siguiente imagen.

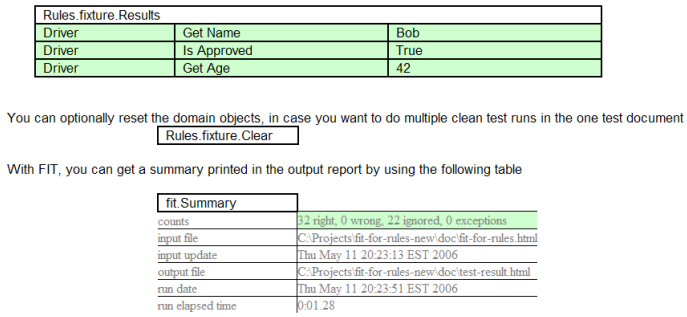

Figura B.7: Ejemplo ejecución FIT For Rules

Bibliografía

[1] "Proyecto telcoblocks," 2008. Disponible en http://telcoblocks.germinus.com.

[2] D. Ferry and S. Lim, "JSR 22. JAIN SLEE API Specification," Mar. 2004.

[3] P. O'Doherty, "Jsr 289 - sip servlet v1.1," 2008.

[4] "Osa parlay x 3.0 specifications," 2007.

[5] A. Moyer, S.Umar, "The impact of network convergence on telecommunications software," *Communications Magazine*, 2001.

[6] "Definition of next generation network," tech. rep., ITU, 2004.

[7] "Conclusions from the ngn-sg," tech. rep., ETSI 38th General Assembly meeting Nice, November 2001.

[8] R. F. H. Schulzrinne, S. Casner, "Rtp: A transport protocol for real-time applications," tech. rep., ietf - RFC 1889, 1996.

[9] ITU-T, "Itu-t recommendation g.711," tech. rep., 1989.

[10] O. Levin, "Rfc3508 - h.323 uniform resource locator (url) scheme registrat," tech. rep., ietf - RFC 3508, 2003.

[11] G. J.Rosenberg, H.Schulzrinne, "Sip: Session initiation protocol," tech. rep., ietf, 2002.

[12] ITU, "PÃ¡gina web de itu-t, disponible en http://www.itu.int/itu-t/," 2009.

[13] J. R. Fielding, J-Gettys, "Hypertext transfer protocol – http/1.1," tech. rep., ietc - RFC 2616, 1999.

[14] U. of Southern California, "Simple mail transfer protocol," tech. rep., ietc - RFC 821, 1982.

[15] "Pagina web de 3gpp disponible en http://www.3gpp.org/," 2009.

[16] V. J. M.Handley, "Sdp: Session description protocol," tech. rep., ietf - RFC 2327, 1998.

[17] I. E. M.Arango, A. Dugan, "Media gateway control protocol (mgcp)," tech. rep., ietf - RFC 2705, 1999.

[18] W. Foundation, "Analizador de protocolos, disponible en http://www.wireshark.org/," tech. rep., 2009.

[19] UCL-INGI., "Pagina del proyecto disponible en: http://www.gnomemeeting.org/," tech. rep., 2009.

[20] "Open sip express router (ser)," tech. rep., openser.org, 2007.

[21] E. Sean Olson, "Jsr 116: Sip servlet api," tech. rep., 2003.

[22] "Java Servlet 2.3 and JavaServer Pages 1.2, JSR-53, Final Release, http://jcp.org/en/jsr/detail?id=53, year = 2001," tech. rep.

[23] "Sip express router (ser," tech. rep., iptel.org, 2004.

[24] "Opensips," tech. rep., Web del proyecto disponible en http://www.opensips.org, 2008.

[25] "Kamailio," tech. rep., Web del proyecto disponible en http://www.kamailio.org, 2008.

[26] Transnexus, "Performance results for openser and sip express router, disponible en http://www.transnexus.com/whiterep., 2006.

[27] "Asterisk the future of telephony," tech. rep., asterisk.org, 2005.

[28] F. M. M. Spencer, B. Capouch, "Iax: Inter-asterisk exchange version," tech. rep., ietf - RFC 5456, 2009.

[29] B. Sharif, "Elastix without tears," tech. rep., PaloSanto Solutions, 2008.

[30] Bandwith.com., "Pagina del proyecto disponible en: http://freepbx.org/," tech. rep., 2009.

[31] A2billing, "Pagina del proyecto disponible en http://www.asterisk2billing.org/cgi-bin/trac.cgi," tech. rep., 2008.

[32] D. Silas, "Mobicents media server guide," tech. rep., JBoss, 2008.

[33] S. Sayer, "Sems-ng design overview inside the media server," tech. rep., iptego GmbH, 2006.

[34] A. S. E. Burger, J. Van Dyke, "Basic network media services with sip," tech. rep., ietc - RFC 4240, 2005.

[35] "Sailfin - contenedor de sip servlets," tech. rep., sun.

[36] P. C. S. Farrell, J. Vollbrecht, "Aaa authorization requirements," tech. rep., ietf - RFC 2906, 2000.

[37] A. R. C. Rigney, S. Willens, "Remote authentication dial in user service (radius)," tech. rep., IETF - RFC 2865, 2000.

[38] E. G. P. Calhoun, J. Loughney, "Diameter base protocol," tech. rep., IETF - RFC 3588, 2003.

[39] S. K. M. Wahl, T. Howes, "Lightweight directory access protocol," tech. rep., IETF - RFC 2251, 1997.

[40] VMware Inc, "Pagina web de VMware, disponible en http://www.vmware.com," 2009.

[41] R. Guarneri, A. M. Sollund, D. Marston, E. Fossbak, B. Berntsen, G. Nygreen, G. Gylterud, R. Bars, and A. Kerdraon, "Report on the state of the art in personalisation, common framework," tech. rep., ePerSpace - IST Integrated Project, 2004.

[42] A. Bhayani, "Mobicents rules-ra, disponible en http://groups.google.com/group/mobicents-public/web/mobicents-rules-ra," 2007.

[43] P. Browne, *JBoss Drools Business Rules*. PACKT Publishing, 2009.

[44] G. Borkowski, "Personalization fundamentals: Rules-based personalization," tech. rep., ATG, 2008.

[45] M. J. S. Jong Woo Kim, Kyung Mi Lee, "A preference scoring technique for personalized advertisements on internet storefronts," tech. rep., School of Business, Hanyang University, Seoul, Republic of Korea, 2004.

[46] Eclipse, "Graphical modeling framework tutorial, disponible en http://wiki.eclipse.org/index.php," tech. rep., 2006.

[47] S. B. S. Corp, "Pagina del proyecto disponible en http://www.jbilling.com," tech. rep., 2007.

[48] A. Org., "Pagina del proyecto disponible en: http://jackrabbit.apache.org/," tech. rep., 2008.

[49] M. Neale, "Fit for rules. keep your business rules in shape," tech. rep., Drools Project, 2006.