Recomendaciones Para el Desarrollo de Aplicaciones P2P Utilizando el Protocolo SIP

Wilson Yecit Ortiz Sánchez

Director: Ing. Javier Alexander Hurtado

Universidad del Cauca
Facultad de Ingeniería Electrónica y Telecomunicaciones
Departamento de Telemática
Línea de Investigación en Ingeniería de Sistemas Telemáticos
Popayán, Enero de 2007

TABLA DE CONTENIDO

1. INTRODUCCIÓN 1

2. EL PROTOCOLO SIP Y EL MODELO PEER TO PEER 4

2.1 Protocolo de Inicio de Sesión (SIP) 4

2.1.1 Características de SIP 4

2.2 EL MODELO PEER TO PEER 8

2.2.1 Funcionamiento Común de las Redes P2P 8
2.2.2 Clasificación de las Redes P2P 9

3. PANORAMA DE LAS TECNOLOGÍAS PARA EL DESARROLLO DE APLICACIONES MÓVILES BASADAS EN SIP 11

3.1 CARACTERISTICAS GENERALES DE LAS PLATAFORMAS PARA EL DESARROLLLO DE APLICACIONES MOVILES. 11

3.1.1 Symbian 11
3.1.1.1 Plataformas de Referencia 12
3.1.1.2 Dispositivos Móviles con Symbian 13
3.1.1.3 Plataformas de Desarrollo Soportadas por Symbian 14
3.1.1.4 Desarrollos en C++ 14
3.1.1.5 Desarrollos J2ME 15
3.1.1.6 Desarrollos en Python 15
3.1.1.7 Desarrollos en Visual Basic y C# 16
3.1.1.7 Desarrollos en Flash Lite 16
3.1.1.8 Soporte Proporcionado por Symbian 16
3.1.2 J2ME 17
3.1.2.1 Teléfonos con J2ME 17
3.1.2.2 Soporte Proporcionado por J2ME 18
3.1.3 BREW 18
3.1.3.1 Teléfonos con BREW 19
3.1.3.2 Plataformas de Desarrollo Soportadas por BREW 19
3.1.3.3 Soporte Proporcionado por BREW 20
3.1.4 Linux 21
3.1.4.1 Plataformas de Desarrollo para Linux 22
3.1.4.2 Teléfonos con Linux 23
3.1.4.3 Soporte Proporcionado por Linux 24
3.1.5 Windows Mobile 24
3.1.5.1 Teléfonos con Windows Mobile 25
3.1.5.2 Soporte Proporcionado por Windows Mobile 25
3.1.5.3 Plataformas de Desarrollo 26
3.2 SOPORTE SIP DE LAS PLATAFORMAS ESTUDIADAS 26

4. RECOMENDACIONES PARA EL DESARROLLO DE APLICACIONES SIP P2P… 29

4.1 RECOMENDACIONES INICIALES 30

4.1.1 Puntos importantes 30
4.1.2 Elección de la Plataforma 34
4.1.3 Elección del IDE 36

4.1.3 Elección del plugin o SDK SIP 37
4.1.3 Conexión de red 38
4.1.3 Consideración de las aplicaciones 38
4.2 RECOMENDACIONES REFERENTES A LAS HERRAMIENTAS 40
4.2.1 Instalación del Plugin para Symbian 40
4.2.2 Instalación del SDK para J2ME 40
4.2.3 Uso de Multi-Emulación en un Mismo PC 40
4.2.4 Configuración de uso de Recursos 41
4.3 RECOMENDACIONES DE IMPLEMENTACIÓN 41
4.3.1 Pautas para Mejorar el Rendimiento de las Aplicaciones en C++ 41
4.3.2 Estado de la Sesión 41
4.3.3 Envío y Recepción de Mensajes 42
4.3.4 Orden de las Cabeceras 43
4.3.5 Contenido de las Cabeceras de una Respuesta 43
4.3.6 Almacenamiento del Contenido de las Cabeceras 44
4.3.7 Creación del Valor del Campo Call-ID 45
4.3.8 Procesamiento de Respuestas 45
4.3.9 Envío de Respuesta Provisional 45
4.3.10 Construcción de Peticiones 46
4.3.11 Orden de Procesamiento de Peticiones 46
4.3.11 Uso del Diálogo 46
4.3.12 Uso de OPTIONS 47
4.3.13 Construcción de una Solicitud OPTIONS 48
4.3.14 Construcción de una Solicitud REGISTER 49
4.3.15 Construcción una Solicitud CANCEL 49
4.3.16 Manejo de Presencia 50
4.3.17 Orden Recomendado para Anular los Objetos del API de Cliente SIP 50
4.3.18 Creación de una Clase para la Gestión de SIP 51
4.4 EJEMPLOS DE APLICACIONES MÓVILES P2P CON SIP 52
5. PROTOTIPO DE APLICACIÓN P2P CON SIP 55
5.1 INTRODUCCION 55
5.2 DESCRIPCIÓN GENERAL 55
5.3 PROCESO DE DESARROLLO 57
5.3.1 Análisis 57
5.3.1.1 Funciones de la Aplicación móvil P2P con SIP ChatSIP 57
5.3.1.2 Ubicación en la Arquitectura de Red 58
5.3.1.3 Diagrama de Casos de Uso 60
5.3.1.4 Descripción de algunos Casos de Uso 61
5.3.2 Diseño 66
5.4 ARQUITECTURA DEL SISTEMA 67
5.5 DIAGRAMA DE CLASES 69
5.6 DIAGRAMAS DE SECUENCIA 71
5.7 IMPLEMENTACIÓN EN J2ME 76
5.8 CÓDIGO IMPLEMENTADO 79

5.8 PRUEBAS DE RENDIMIENTO 91
6. CONCLUSIONES Y TRABAJOS FUTUROS 93
GLOSARIO 96
BIBLIOGRAFÍA 99

LISTA DE FIGURAS

Figura 1. Arquitectura SIP 7
Figura 2. Arreglo bidimensional con la información de un INVITE recibido. 44
Figura 3. Secuencia de mensajes para una solicitud OPTIONS. 48
Figura 4. Aplicación ChatSip en el emulador. 56
Figura 5. Sesión P2P pura entre dos UAs. 59
Figura 6. Sesión P2P a través de un servidor Proxy. 60
Figura 7. Casos de uso del sistema. 61
Figura 8. Opción “Registrarme” de la sección Registro. 62
Figura 9. Interfaz principal del Chat SIP. 63
Figura 10. Interfaces de dos usuarios en una sesión de juego Trique. 64
Figura 11. Interfaz para el envío de imágenes con SIP. 65
Figura 12. Interfaz para el envío de mensajes de texto con SIP. 66
Figura 13. Arquitectura del sistema. 68
Figura 14. Diagrama de clases de la aplicación ChatSIP. 69
Figura 15. Diagrama de secuencia para el caso de uso Realizar registro. 72
Figura 16. Diagrama de secuencia para el caso de uso Iniciar riq. 73
Figura 17. Diagrama de secuencia para el caso de uso Jugar Trique. 74
Figura 18. Diagrama de secuencia para el caso de uso Enviar imagen. 75
Figura 19. Diagrama de secuencia para el caso de uso Enviar mensaje. 76
Figura 20. Clase conexiónSip implementada en J2ME. 77

LISTA DE TABLAS

Tabla 1. Comparación entre las diferentes plataformas estudiadas. 27
Tabla 2. Soporte de J2ME y Symbian 35
Tabla 3. IDEs que soportan los SDKs y/o el Plugin SIP. 37
Tabla 4. tecnologías y medios transmitidos. 39
Tabla 5. Resultados obtenidos a diferentes velocidades de transferencia. 91
Tabla 6. Resultados obtenidos con diferentes cantidades de memoria. 92
Tabla 7. Tamaños de los paquetes que envía la aplicación. 92

1. INTRODUCCIÓN

En la actualidad las aplicaciones de voz, video e intercambio de contenidos (entendiéndose como archivos de cualquier tipo, sea documentos, imágenes y aplicaciones entre otras), se están convirtiendo en herramientas claves para la comunicación entre personas. El desarrollo de estas aplicaciones se ha soportado en el protocolo IP desarrollado inicialmente para Internet, el cual facilita el intercambio digital de diversos contenidos a través de las redes públicas y privadas. Debido a la naturaleza del protocolo las aplicaciones han tomado algunas características importantes, destacándose el concepto de sesión en la cual después de iniciada, permite intercambiar no solo información como la voz sino una variedad de contenidos. En la llamada clásica, una comunicación generalmente solo sirve para transmitir un solo tipo de medio y una sola vez, si se desea enviar diversos medios se debe establecer una comunicación por cada medio involucrado. En este momento se habla de un camino lógico. En cuanto a la identificación de los usuarios, en esta nueva tendencia dominada por las redes IP se maneja una dirección de usuario en lugar de un número de terminal, lo cual elimina la dependencia a un terminal específico para realizar la comunicación. Esto le permite a un usuario comunicarse desde cualquier terminal que tenga a la mano y que contenga el soporte necesario para establecer una sesión. Entre estos equipos se encuentran los teléfonos celulares, computadores personales, laptops y pdas. [1] [2]

A este tipo de comunicaciones se les denomina comunicaciones *Peer-to-peer* (P2P) o persona a persona, donde el objetivo es llevar la comunicación hasta donde se encuentre el usuario y no donde esté el equipo terminal. Un ejemplo de este tipo de comunicaciones persona a persona es el Messenger [3], el cual permite el intercambio de mensajes de texto, archivos, voz y video, desde la ubicación donde se encuentre el usuario, pudiendo usar cualquier terminal disponible y una identificación personal (login

y password) con los cuales se establece una sesión de comunicaciones con otros usuarios.

Las ventajas de las comunicaciones *Peer-to-peer* son mayores frente a la llamada clásica, pero su implementación ha traído una serie de complicaciones básicamente relacionadas con el manejo de la sesión, debido principalmente a la movilidad de los usuarios entre los diferentes equipos o sistemas terminales, que por ser propietarios manejan cada uno diferentes esquemas de direccionamiento (por ejemplo: un numero de teléfono o una dirección electrónica), administración o señalización, los diferentes medios que pueden ser utilizados para la comunicación (voz, video o datos) y los diferentes protocolos encargados de llevar la comunicación hasta los usuarios. [4]

El Protocolo de Inicio de Sesión (SIP) desarrollado por la IETF (*Internet Engineering Task Force*) para el desarrollo de conexiones VoIP (Voz sobre IP) es una herramienta ágil para la gestión de las sesiones de comunicación en redes IP. Puede crear, modificar, y terminar sesiones trabajando independientemente de los protocolos de transporte subyacentes y sin la dependencia en el tipo de medio que se está compartiendo. [5]

Para el desarrollo de aplicaciones que implementen el protocolo SIP se encuentran disponibles una variedad de tecnologías y herramientas con las cuales es posible crear los diferentes componentes de la arquitectura SIP para diversos equipos terminales. Esta variedad de recursos puede generar problemas a los desarrolladores nuevos que pretendan utilizar SIP en sus aplicaciones, ya que para culminar un desarrollo además del conocimiento del funcionamiento del protocolo, el diseño e implementación se debe comenzar por una serie de elecciones como: la tecnología de desarrollo más conveniente, las herramientas disponibles y apropiadas y ejemplos de aplicación entre otras. Los posibles fracasos o dificultades generados por la falta de conocimiento de una tecnología nueva para el desarrollador, puede generar una aplicación no satisfactoria, inapropiada o lo que es peor el abandono del proyecto.

En el presente trabajo se realiza un estudio de las tecnologías más importantes que permitan el desarrollo de aplicaciones móviles P2P con SIP, con el fin de generar una serie de recomendaciones y ejemplos de aplicación sobre herramientas comunes en la comunidad de la FIET que puedan servir de guía a la hora de realizar implementaciones de este tipo.

2. EL PROTOCOLO SIP Y EL MODELO PEER TO PEER

2.1 Protocolo de Inicio de Sesión (SIP)

Es un protocolo de control de la capa de aplicación desarrollado por la IETF (Internet Engineering Task Force) para el desarrollo de conexiones VoIP (Voz sobre IP). Se utiliza para iniciar, manejar y terminar sesiones interactivas con uno o más usuarios sobre redes IP. El contenido de una sesión puede ser de cualquier tipo (voz, vídeo, datos, etc) y se puede modificar en cualquier momento, ya sea añadiendo capacidades de los usuarios o variando el número de los mismos que toman parte en la comunicación. Está inspirado en los protocolos HTTP (para Web) y SMTP (para correo electrónico), lo que lo hace ideal para integrar la voz como un servicio más de Internet. Adicionalmente puede utilizar MIME (Multipurpose Internet Mail Extensión, Extensión de Correo de Internet Multipropósito) para la descripción de contenidos, lo que permite transmitir datos y aplicaciones junto a la llamada de voz, haciendo fácil el envío de tarjetas, fotos, archivos MP3, información codificada, etc. durante una llamada [6].

SIP es un protocolo '*peer-to-peer*' que permite que en el control de la sesión intervengan terceros agentes o aplicaciones, capaces de modificar los mensajes SIP que se intercambian entre los extremos de una comunicación, permitiendo con esto realizar funciones como el desvió de llamadas a otro equipo terminal o la transferencia de sesiones de video conferencia a un ordenador en particular entre otras. La facilidad con que el protocolo SIP puede incorporar nuevas extensiones (nuevas cabeceras, métodos, parámetros, etc.) hace prácticamente infinitas las posibilidades de desarrollo de servicios en SIP.

2.1.1 Características de SIP

El protocolo SIP presenta un conjunto único de características que lo hacen idóneo para el desarrollo de aplicaciones que incorporen funcionalidades de comunicación multimedia y en particular de Telefonía IP. Éstas pueden ser resumidas en:

- Localización de Usuario: SIP encuentra al usuario llamado en el dispositivo correspondiente de la red, y establece la conexión incluso cuando la identificación del dispositivo (dirección IP) es dinámica o compartida.
- Características de la llamada: No sólo se encarga de establecer la llamada en sí con el destinatario sino que además negocia las funcionalidades (aplicaciones) que estarán disponibles durante la sesión entre los terminales.
- Disponibilidad de la llamada: Determina si el destinatario de la llamada está disponible y en caso afirmativo, si acepta o no la llamada. En caso de no disponibilidad o aceptar, SIP admite la toma de acciones definidas por las aplicaciones de control o de usuario.
- Gestión de participantes: Durante una llamada, las partes pueden añadir nuevos participantes a la llamada o comunicación, así como cancelar participantes de la misma en cualquier momento.
- Cambio de parámetros durante la sesión: Se admite que los participantes de una comunicación cambien los parámetros y características de ésta establecidos al inicio de la misma, por ejemplo, el paso de una sesión de voz a una de audio y texto o vídeo, durante la primera.
- Diferentes formatos de respuesta: SIP permite responder una invitación a una sesión con un formato diferente al solicitado, por ejemplo, un usuario puede responder una llamada de voz con una página Web con los números alternativos de contacto.
- Direccionamiento de Internet: SIP utiliza el mismo formato de direcciones de Internet, para los nombres y para las direcciones IP, por ejemplo, sip:nombre_usuario@nombredominio.com
- Protocolo encapsulado en texto: La utilización de texto plano para la implementación de los mensajes SIP, permite una integración en aplicaciones Web más simples, facilidad de diagnóstico y control de errores.
- Terminales inteligentes multi-funcionales: SIP implementa en cada uno de los dispositivos participantes una comunicación con un elevado grado de inteligencia. Dicha implementación puede estar tanto en terminales telefónicos, ordenadores

personales (PCs), asistentes personales inalámbricos (wi-fi PDAs) u otros dispositivos de comunicación, como por ejemplo los teléfonos 3G (UMTS).

2.1.2 Funciones SIP

SIP soporta una variedad de funciones de las cuales se listan algunas a continuación:

Facilidades básicas (llamada en espera, reenvío de llamadas, bloqueo de llamadas etc.), Videoconferencia, Mensajería Unificada, Presencia, Conferencias ad-hoc y programadas Mensajes Instantáneos, Encuéntrame/Sígueme, y Colaboración. De las cuales, la de presencia es una de las aplicaciones que produce más expectativas de SIP, ya que permite localizar a un usuario y determinar su disponibilidad para comunicarse vía teléfono, email, texto o vídeo. Tanto las personas como las aplicaciones pueden usar información de "Presencia", para integrar las comunicaciones.

2.1.3 Arquitectura SIP

La figura 1 muestra la arquitectura de SIP, la cual es similar a la de HTTP. Las solicitudes son generadas por el cliente y enviadas al servidor. El servidor procesa las solicitudes y envía una respuesta al cliente. Una solicitud y su respuesta conforman una transacción. SIP tiene los mensajes INVITE y ACK que definen el proceso de abrir un canal confiable sobre el cual los mensajes de control de la llamada pueden viajar. Esta arquitectura es escalable, flexible y distribuida. Las funcionalidades tales como proxy, redirección, locación y registro pueden residir en un único servidor o en varios servidores distribuidos permitiendo incorporar nuevas funciones o procesos sin afectar los demás componentes. El protocolo conserva información de estado en los extremos, permitiendo recuperarse de fallas de alguno de los componentes. [7] [8] [9].

Los componentes de la arquitectura SIP son: los agentes de usuario (UA) que pueden funcionar tanto como cliente como servidor y los servidores que pueden realizar tres tipos de operaciones: Proxy, Registro y Redirección, estos servidores pueden estar empotrados en un único servidor físico. En un sistema real aparecen de forma frecuente los denominados Marshal Server quienes hacen las labores de autenticación

y de Gatekeepers. En el despliegue de SIP se pueden encontrar agentes de usuario de tipo hardware (p.e. terminales telefónicos) o software. De la misma forma se pueden encontrar integrados los proxy's, redirectores y registros en un único sistema que se gestiona de forma remota a través de un interfaz Web [10] [11][12].

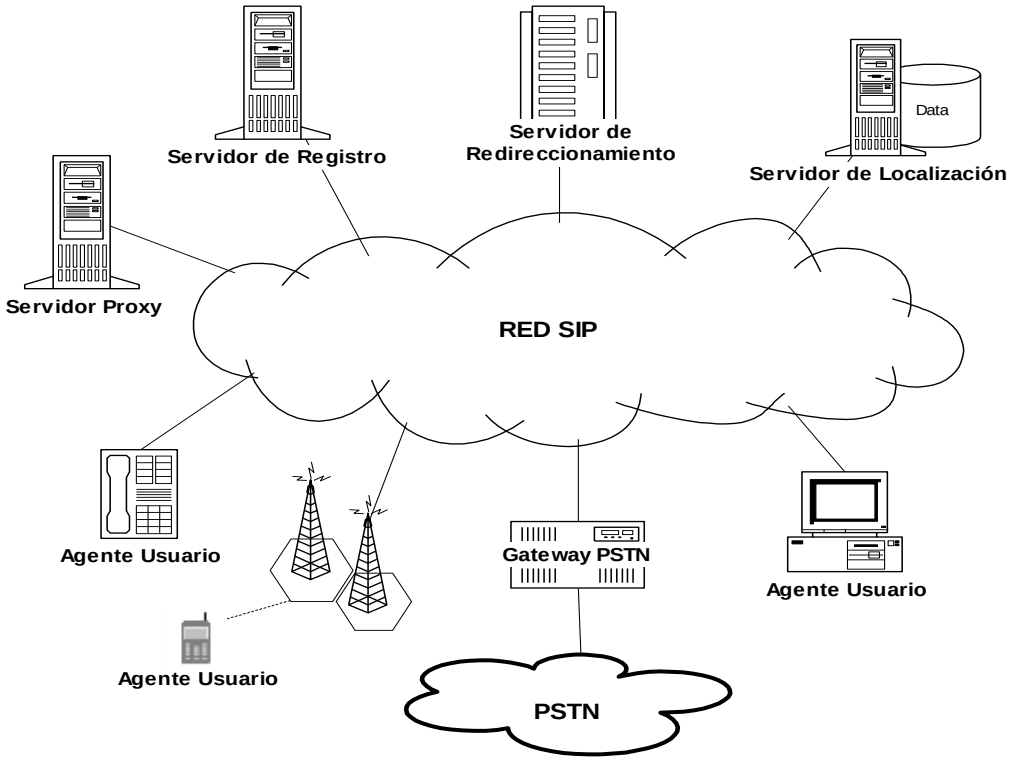

Figura 1. Arquitectura SIP

El establecimiento de una llamada típicamente sigue los siguientes pasos: un agente de usuario se registra ante un proxy, e inicia la localización de un usuario. El proxy haciendo uso de la información disponible (por ejemplo DNS) localiza el proxy del usuario final. Al realizar la consulta por el usuario al proxy final es posible que sea necesario realizar una búsqueda en un servidor de localización o una redirección hasta llegar al agente de usuario destinatario. Una vez que se confirma el establecimiento de la llamada por el mismo camino de los servidores proxy, los flujos de información se realizan de forma P2P [9] [13].

2.2 EL MODELO PEER TO PEER

Peer to peer (se traduce de par a par- o de punto a punto, es más conocida como P2P) significa de igual a igual, es decir, una conexión entre equipos terminales del mismo nivel. Teóricamente las redes P2P son aquellas que no tienen coma base grandes servidores físicos para manejar el transito a través de ellas, sino que utilizan recursos de todas las maquinas que se encuentran conectadas para hacerlo. Una red entre iguales se refiere a una red que no tiene clientes y servidores fijos, sino una serie de nodos que se comportan simultáneamente como clientes y como servidores de los demás nodos de la red. Este modelo de red difiere del modelo cliente-servidor, el cual se rige por una arquitectura donde no hay distribución de tareas entre sí, solo una comunicación en donde el cliente y el servidor no pueden cambiar de roles.

En una red P2P cualquier nodo puede iniciar, detener o completar una transacción. La eficacia de los nodos en el enlace y transmisión de datos puede variar según su configuración local (cortafuegos, NAT, ruteadores, etc.), velocidad de proceso, disponibilidad de ancho de banda de su conexión a la red y capacidad de almacenamiento en disco.

2.2.1 Funcionamiento Común de las Redes P2P

Debido a que la mayoría de los ordenadores domésticos no tienen una IP fija, sino que le es asignada por su proveedor (ISP) en el momento de conectarse a Internet, no pueden conectarse entre sí porque no conocen con anterioridad las direcciones que usaran. La solución habitual es realizar una conexión a un servidor (o servidores) con dirección conocida (normalmente una IP fija), que se encarga de mantener la relación de direcciones IP de los clientes de la red, de los demás servidores y habitualmente información adicional, como un índice de la información de que disponen los clientes. Con esto, los clientes ya tienen información sobre el resto de la red, y pueden intercambiar información sin intervención de los servidores.

2.2.2 Clasificación de las Redes P2P

Una tipo de clasificación de las redes P2P es debido a su tipo centralización, la cual las cataloga en redes P2P centralizadas, puras e híbridas.

2.2.2.1 Redes P2P Centralizadas:

Este tipo de red P2P se basa en una arquitectura donde todas las transacciones se hacen a través de un único servidor que sirve de punto de enlace entre los nodos, y que a la vez gestiona los nodos donde se almacenan los contenidos. Poseen una administración muy dinámica y una disposición permanente de contenido, sin embargo, está muy limitada en la privacidad de los usuarios y en la falta de escalabilidad de un sólo servidor. Otro inconveniente es que puede ofrecer problemas en puntos únicos de fallo, situaciones legales y enormes costos en el mantenimiento así como un gran consumo de ancho de banda.

Una red de este tipo posee las siguientes características:

- Se rige bajo un único servidor que sirve como punto de enlace entre nodos y como servidor de acceso al contenido, el cual distribuye a petición de los nodos.
- Todas las comunicaciones (como las peticiones y encaminamientos entre nodos) dependen exclusivamente de la existencia del servidor.

2.2.2.2 Redes P2P "Puras" o Totalmente Descentralizadas

Estas redes P2P son las más versátiles al no requerir de un gestionamiento central; los nodos actúan como clientes y como servidores, y el trafico interno es distribuido entre todos los nodos conectados a la red. La necesidad de un servidor centralizado con grandes cantidades de recursos es inexistente, lo que genera mayores ventajas como: menor costo en infraestructura y menor ancho de banda. Las redes de este tipo tienen las siguientes características:

- Los nodos actúan como cliente y servidor.
- No existe un servidor central que maneje las conexiones de red.
- No existe un enrutador central que sirva como nodo y administre direcciones.

La implementación de este tipo de redes es muy complicado por lo que generalmente las redes que comienzan de esta manera al poco tiempo mutan a redes Hibridas; que se describe a continuación.

2.2.2.3 Redes P2P Híbridas, Semi-centralizadas o Mixtas

En este tipo de red, se puede observar la interacción entre un servidor central que sirve como hub y administra recursos como el ancho de banda, enrutamiento y comunicación entre nodos pero sin conocer la identidad de cada nodo y sin almacenar información alguna, por lo que el servidor no comparte archivos de ningún tipo a ningún nodo. Tiene la característica de funcionar de ambas maneras, es decir, puede incorporar más de un servidor que gestione los recursos compartidos, pero también en caso de que él o los servidores que gestionan caigan, el grupo de nodos que sigue en contacto a través de una conexión directa entre ellos, continúan su comunicación sin problemas; con este tipo de red, es posible seguir compartiendo información en ausencia de los servidores. Las características que presentan las redes hibridas son:

- Tiene un servidor central que guarda información en espera y responde a peticiones para esa información.
- Los nodos son responsables de almacenar la información (el servidor central no almacena contenidos), que permite al servidor central reconocer los recursos que se desean compartir, y para poder descargar esos recursos compartidos a los peers que lo solicitan.
- Las terminales de enrutamiento son direcciones usadas por el servidor, que son administradas por un sistema de índices para obtener una dirección absoluta.

Como resultado de los problemas técnicos que involucran las redes de tipo puro y centralizadas, casi todos los sistema P2P actuales pueden clasificarse como Híbridos, los cuales no solo están integradas por peers, sino también por servidores que actúan como puentes que permiten agilizar la conexión entre clientes y optimizar el flujo de la información, dejando a los peers el grueso de la información. [14] [15] [16]

3. PANORAMA DE LAS TECNOLOGÍAS PARA EL DESARROLLO DE APLICACIONES MÓVILES BASADAS EN SIP

Actualmente existen cinco plataformas importantes entre las cuales los desarrolladores pueden escoger a la hora de crear una aplicación para dispositivos móviles en general; Symbian [17] es un sistema operativo y una plataforma de desarrollo que domina el mercado europeo. El entorno de ejecución binaria para servicios inalámbricos "BREW" (Binary Runtime Environment for Wireless) [18] es muy preferido en los EE.UU debido al proceso de distribución y pago controlado por QUALCOMM, la empresa creadora de BREW. La plataforma Java J2ME de Sun Microsystems [19], que es popular porque teóricamente puede ejecutarse en cualquier sistema operativo aunque en la práctica no lo es tanto. Linux [20] que es una plataforma que empieza a ganar terreno y Windows Mobile de Microsoft [21], que se espera que algún día llegue a ser un importante sistema operativo y plataforma de desarrollo. [22]

3.1 CARACTERISTICAS GENERALES DE LAS PLATAFORMAS PARA EL DESARROLLLO DE APLICACIONES MOVILES.

Cada una de las plataformas para el desarrollo de aplicaciones móviles posee características importantes que se describen a continuación.

3.1.1 Symbian

Symbian es un sistema operativo producto de la alianza de varias empresas de telefonía celular, dentro de las que se encuentran Nokia, Sony Ericsson, Samsung y Siemens. El objetivo de Symbian fue crear un sistema operativo abierto para las diversas plataformas de teléfonos móviles.

Técnicamente, el sistema operativo Symbian es una colección compacta de código ejecutable y varios archivos, la mayoría de ellos son bibliotecas vinculadas dinámicamente (DLL en sus siglas en inglés) y otros datos requeridos, incluyendo archivos de configuración, de imágenes y de tipografía, entre otros recursos residentes. Symbian se almacena, generalmente, en un circuito flash dentro del dispositivo móvil. Gracias a este tipo de tecnología, se puede conservar información aun si el sistema no posee carga eléctrica en la batería, además de que le es factible reprogramarse, sin necesidad de ser separada de los demás circuitos.

Symbian es actualizable, tarea que puede ser realizada por el usuario, dependiendo del modelo de su equipo a través de los sitios en Internet de los fabricantes de teléfonos o también, obteniendo el disco compacto o tarjeta de memoria flash de los distribuidores autorizados.

Symbian OS es la industria global de sistemas operativos estándar para telefonos celulares patrocinada por las principales empresas de las comunicaciones móviles, las cuales representan el 85% de las ventas mundiales de teléfonos móviles. Además, es el sistema operativo telefónico móvil abierto más popular y avanzado del mundo que ha llevado a los más importantes fabricantes de teléfonos móviles a incorporarlo en sus dispositivos celulares. Symbian tiene gran penetración en el mercado, su difusión en Europa y su entrada a América latina, además de las posibilidades de llegar a clientes comerciales y personales, hacen a esta plataforma muy atractiva para la comunidad de desarrollo, que cuenta sin costo alguno con todos los recursos necesarios para generar nuevas y atractivas aplicaciones que ayudadas de SIP hacen de los teléfonos celulares, unas herramientas altamente útiles para cubrir diversas necesidades y entornos de aplicación.

3.1.1.1 Plataformas de Referencia

Para acondicionar las diferentes arquitecturas para teléfonos avanzados, Symbian estableció varias plataformas de referencia (interfaces de usuario) basadas en Symbian OS, llamadas Series. Hasta el momento se contemplan cuatro Series o tipos de dispositivos para su sistema operativo; los denominados Serie 60, para teléfonos que el usuario maneja con una sola mano; Serie 80, para teléfonos con pantalla horizontal grande y teclado; Serie 90 para dispositivos en desarrollo y UIQ1, para

teléfonos que utilizan una pluma electrónica similar a los asistentes digitales. La mayoría de los dispositivos Nokia son Serie 60, todos los de Sony Ericsson trabajan bajo UIQ, al igual que los Motorola.[23]

La Serie 60 (también denominada S60) es la más extendida y cuenta con varias ediciones y “paquetes de características” por cada edición. Actualmente Symbian Serie 60 cuenta con el primer paquete de características de la tercera edición, la cual dicen sus fabricantes incorpora nuevos elementos adicionales en el API tanto para desarrolladores Java como de C++ que permitirán obtener mejor rendimiento y mejorar la integridad de las aplicaciones.[24]

La Series 60 está diseñada para teléfonos de alta gama que cuentan con calendario, gestión de contactos, mensajería multimedia, correo electrónico, navegación y cámara; además ofrece formatos para grandes pantallas a color, con una resolución de 170x208.

3.1.1.2 Dispositivos Móviles con Symbian

Symbian OS se ejecuta actualmente en aproximadamente 85 modelos de teléfonos de fabricantes como Nokia, Sony Ericsson, Motorola y Samsung y periódicamente se adicionan nuevos modelos al mercado; durante el último trimestre del año 2006 los teléfonos Nokia P990 y M600 de Sony Ericsson entran en circulación.

Según Symbian, hasta la fecha más de 60 millones de teléfonos basados en Symbian OS se han vendido alrededor del mundo, haciendo de Symbian la elección más usual de la industria de los Smartphones. [20] [25]

La mayoría de los móviles con Symbian son de Nokia, entre los que se pueden encontrar: el 6260, 6600, 6620,6630, 6680, 7650, 7610, 6670 y el N-Gage; son muy pocos los modelos de otros fabricantes como Sony Ericsson con el P800, P802, P900, P910, P990, W950 y M600i; los Siemens SX1; Motorola A1000, A1010, A925; los FOMA y Samsung con el modelo SGH-D730 y Panasonic con los modelos X700 y X800. Una lista completa de teléfonos con este sistema operativo se encuentra actualizada en el sitio oficial de Symbian (http://www.symbian.com/phones/index.html).

3.1.1.3 Plataformas de Desarrollo Soportadas por Symbian

Las aplicaciones compatibles con Symbian se desarrollan a partir de lenguajes de programación orientados a objetos como C++, Java (con sus variantes como PJava, J2ME, etc.), Python y Visual Basic para dispositivos móviles, entre otros, incluyendo algunos lenguajes disponibles en versión libre. A partir de la tercera edición de Symbian serie 60 paquete de características 1, se encuentra disponible en algunos de los teléfonos Macromedia Flash Lite de Adobe [26], el cual presenta un proceso desarrollo mucho más sencillo y rápido que muchas otras plataformas.

La variedad de aplicaciones que se pueden desarrollar sobre Symbian van desde administradores de archivos hasta visualizadores de películas, guías de ciudades, mapas, diccionarios, emuladores de juegos, por mencionar algunas. Cada aplicación se puede instalar en el teléfono con la ayuda de una computadora, una interfaz USB [27], Bluetooth [28] o Firewire [29], dependiendo del modelo de teléfono y el soporte correspondiente. Las aplicaciones se graban en la memoria flash del teléfono dentro del proceso de sincronización.

Nokia, una de las principales compañías que respaldan el desarrollo de Series 60, cuenta con herramientas que ayudan a los desarrolladores para convertir aplicaciones escritas en UIQ a Series 60 con el objetivo de tener un mercado mayor. Por otro lado, compañías como la estadounidense Peroon tienen herramientas que hace lo contrario, convertir aplicaciones escritas para Series 60 en UIQ. Algunos fabricantes de entornos de desarrollo integrado (IDE), como Metroworks, soportan el desarrollo tanto para la serie 60 como la UIQ.

3.1.1.4 Desarrollos en C++

La plataforma S60 suministra a los desarrolladores el acceso al API Symbian de C++, la UI de la S60 y los motores de aplicación. Más de 30 conjuntos de APIs adicionales están disponibles en esta edición además del soporte para la gestión digital de derechos (DRM). La emulación de la electrónica integrada de los dispositivos de esta serie se realiza a través del paquete Carbide, recomendado por Symbian para ser usado con los SDKs S60 para C++. El SDK S60 más reciente para C++ soporta la tercera edición de Symbian paquete de características uno.

La S60 tercera edición proporciona seguridad a través de Symbian Signed y su nuevo formato binario más eficiente. Los desarrolladores en C++ tienen acceso al servicio de ubicación, el protocolo de inicio de sesión (SIP), DRM y el API de mensajería instantánea. Adicionalmente esta edición incorpora reconocimiento óptico de caracteres (OCR), OpenGL V1.1, y mejora la interacción con las características de la plataforma, como la galería y los perfiles.

3.1.1.5 Desarrollos J2ME

La tecnología Java proporciona a los desarrolladores la opción de que la misma implementación opere sobre múltiples plataformas, a diferencia de C++.

La S60 2nd edición adiciona ME / MIDP 2.0 de Java con un rendimiento mejorado del compilador para la configuración CLDC. Adicionalmente incorpora nuevas especificaciones de Java (JSRs), incluyendo el API de gráficos de 3D (JSR - 184), el API Java para Bluetooth (JSR - 82), y el API FileConnection (JSR - 75).

Con la introducción de S60 3ra edición, los desarrolladores de Java adquieren acceso a CLDC 1.1 y mayores prestaciones del API con la adición del API de seguridad y servicios seguros (JSR - 177), el API de ubicación de Java (JSR - 179), el API SIP del protocolo de inicio de sesión (JSR - 180), el API (WMA) de intercambio de mensajes inalámbrico 2.0 (JSR - 205), y el API de gráficos de vectores 2D (JSR - 226). La S60 3ra edición, también adiciona un API (AMMS) de suplementos multimedia avanzada (JSR - 234).

El funcionamiento de Java sobre la plataforma S60 comparte muchas características comunes con el funcionamiento sobre la plataforma de la Serie 40 y la plataforma de la Serie 80; esto permite que los desarrolladores extiendan el alcance de sus aplicaciones a casi toda la plataforma de forma sencilla.

3.1.1.6 Desarrollos en Python

Los desarrolladores de python tienen acceso a un intérprete de python hasta el momento para la primera y segunda edición de la serie 60. Python suministra un ambiente de desarrollo rápido para esta plataforma que puede ser utilizado para

implementar aplicaciones temporales que posteriormente se implementaran sobre C++ o J2ME.

El soporte de Python para la S60 3ra edición está disponible a partir del segundo periodo del 2006.

3.1.1.7 Desarrollos en Visual Basic y C#

Los desarrolladores de C# y de Visual Basic pueden crear aplicaciones usando tecnología de un tercero, como AppForge, la cual se integra directamente a los IDEs de Microsoft, Visual Basic 6.0 y Visual Studio .NET de Microsoft. Usando un conjunto de controles *Crossfire* y bibliotecas al mismo tiempo de Visual Basic 6, de Visual Basic .NET, o de visual C#, los desarrolladores pueden crear aplicaciones para las Series 60 y 80. Debido a que la tecnología *Crossfire* permite que la lógica de aplicación sea rehusada, las mismas aplicaciones con algunas modificaciones en la UI funcionan en dispositivos de ambas plataformas. En cuanto las aplicaciones están completas, pueden ser lanzadas usando el *Crossfire*. Las aplicaciones funcionan en cualquier dispositivo con un cliente *Crossfire*, que puede ser “lanzado” desde una aplicación o instalado por separado.

El soporte para estos lenguajes en la S60 3rd edición esta disponible a partir del segundo periodo de 2006.

3.1.1.7 Desarrollos en Flash Lite

Flash Lite permite desarrollar aplicaciones sencillas en tiempos muy cortos, y con buenos resultados gráficos. Además, no hay que preocuparse por los perfiles o configuraciones, ya que lo que se desarrolla una vez sirve para todos los dispositivos que soporten Flash Lite, con la precaución de prever que no todas las pantallas no tienen el mismo tamaño [26].

3.1.1.8 Soporte Proporcionado por Symbian

Symbian contiene una colección muy variada y extensa de bibliotecas para implementar muchos de los estándares de la industria de telefonía móvil de 2, 2.5 y 3

generación. En la capa de software de sistema, en su versión 6, incluye soporte para distintos servicios como: redes (TCP/IP, PPP, TSL, SSL, IPSec, FTP), comunicaciones (Bluetooth, IrDA, Obex), Seguridad (DES, RSA, DSA, DH), mensajes (POP3, IMAP4, SMTP, SMS, BIO), navegación (HTML, HTTPS, WAP, WML), telefonía (GSM, GPRS, fax), gráficos, multimedia (WAV, AU, WVE, JPEG, BMP, MBM, GIF) y mucho más.

Symbian en un sistema operativo muy completo, diseñado para variados dispositivos móviles, cuenta con una comunidad de desarrolladores muy extendida que trabajan continuamente en este sistema, permite aprovechar los recursos con los que cuenta cada dispositivo móvil y las capacidades de los lenguajes de desarrollo soportados. [30] [31]

3.1.2 J2ME

Java 2 Micro Edition (J2ME) es una versión reducida de Java Estándar (J2SE) diseñada exclusivamente para dispositivos móviles con recursos limitados. Consiste de dos elementos: configuraciones y perfiles. Las configuraciones proveen las librerías y una máquina virtual para una categoría de dispositivos inalámbricos. Hay dos configuraciones para J2ME, una para dispositivos inalámbricos fijos y otra para los móviles. Los perfiles son APIs integradas por encima de las configuraciones para proveer un ambiente de funcionamiento para dispositivos específicos, como PDAs, telefonía celular, o *set-top boxes*. El perfil administra la aplicación, la interfaz de usuario, las redes y la transmisión de datos. Para soportar aplicaciones Java, los fabricantes tienen que implementar un perfil para cada dispositivo específico.[32]

Java ME se ha convertido en una buena opción para crear aplicaciones en teléfonos móviles, debido a que pueden ser emuladas en un PC durante la fase de desarrollo y luego se suben fácilmente a diferentes teléfonos que cuenten con el soporte Java; ya que al ser multiplataforma facilita la portabilidad.

3.1.2.1 Teléfonos con J2ME

En la actualidad en el mercado existen más de 600 tipos de telefonos móviles que cuentan con el soporte para J2ME. En la página de Sun Microsystems http://developers.sun.com/techtopics/mobility/device se puede encontrar la lista

actualizada de equipos que pueden correr aplicaciones J2ME. Lo importante de esta lista es que se encuentran dispositivos de todos los tipos, es decir; desde los dispositivos más sencillos hasta los dispositivos de última tecnología. Adicionalmente, de esta lista, los teléfonos Nokia 3250, 5500, E50, E60, E61, E70, E71, E73, N91 y N93 cuentan con el soporte necesario para ejecutar aplicaciones que hagan uso del protocolo SIP (JSR 180). [33] [34]

3.1.2.2 Soporte Proporcionado por J2ME

El soporte de J2ME para las diferentes tecnologías es el mismo que el presentado en la sección “Desarrollos J2ME” de plataforma Symbian, por lo cual no se mencionara nuevamente. Pero se recuerda que J2ME cuenta con el soporte pare el desarrollo de aplicaciones que utilicen el protocolo SIP, brindando la posibilidad de utilizar este protocolo desde una plataforma independiente a Symbian.

3.1.3 BREW

BREW ("Binary Runtime Environment for Wireless") es un producto de QUALCOMM Internet Services (QIS), división del grupo QUALCOMM Wireless & Internet (QWI) de QUALCOMM Incorporated. La división QIS se concentra en las aplicaciones y servicios inalámbricos de próxima generación que combinen las capacidades de voz y datos para satisfacer de mejor forma las demandas de los consumidores en un mundo donde convergen las comunicaciones inalámbricas [35].

BREW es una plataforma de ejecución de aplicaciones abierta, estándar y extensible que reside en el dispositivo inalámbrico, e intenta ofrecer la misma solución al mercado inalámbrico que J2ME, permitiendo generar aplicaciones dinámicas en el cliente con interfaces altamente gráficas y ofreciendo una alternativa de programación en los lenguajes C y C++, lo que representa una alternativa para aquellos que no están especializados en el lenguaje Java. [36]

3.1.3.1 Teléfonos con BREW

Actualmente existen alrededor de 224 teléfonos de diversos fabricantes como: Hyunday, Kyocera, Motorota, Samsung, Sony Ericsson y Nokia entre otros, que cuentan con BREW. La mayoría de los teléfonos son producidos por Kyocera, LGE, Samsung y Motorota entre los que se encuentran el Kyocera KX1, Kyocera KX2, Kyocera KX5, Kyocera KX424, Kyocera SE47, Motorola C215, Motorola T-732, Motorola V810, Samsung SCH-A815, Samsung SCH-N485, Nokia 2355, Nokia 3125, Nokia 6235i, Nokia 3155, Nokia 6012. En Colombia es posible conseguir los equipos Motorola V810, Nokia 3125 y el Samsung SCH-N415 distribuidos por el operador Movistar. Una lista completa de teléfonos con este sistema operativo se encuentra actualizada en el sitio oficial de BREW (http://brew.qualcomm.com/brew/es/developer/resources/ad/devices_op.html).

Qualcomm asegura que casi 150 millones de dispositivos Brew han sido vendidos en todo el mundo; y se espera que para finales del 2008 la cifra crezca a 200 millones de dispositivos. [37]

3.1.3.2 Plataformas de Desarrollo Soportadas por BREW

La plataforma BREW es abierta. Es compatible con otros lenguajes además del C/C++ nativo, incluyendo entornos de ejecución alternativos tales como Java, Extensible Markup Language (XML, lenguaje de marcación ampliable) y Flash. Debido a que BREW funciona sobre un terminal que tenga cualquier sistema operativo (SO) móvil, como por ejemplo Palm, es posible descargar “desde el aire” (OTA, Over-the-air programming) [38] las aplicaciones escritas para dicho SO utilizando el sistema de distribución BREW (BDS) y cobrarlas como cualquier aplicación BREW.

BREW permite a los editores y desarrolladores crear aplicaciones en el lenguaje que deseen y ponerlas a disposición de los operadores BREW, quienes a su vez emplean el mercado virtual BREW para adquirir aplicaciones y ofrecerlas a sus subscriptores; los fabricantes de equipos utilizan las herramientas BREW para responder rápidamente a los requisitos de los operadores y los subscriptores pueden evaluar (mediante una muestra previa) y compran las aplicaciones desde sus teléfonos. Para completar la solución, se simplifica el proceso de facturación para los subscriptores, así como los

pagos a editores, desarrolladores y sus socios a través del sistema de distribución BREW.

Microsoft e IBM proporcionan herramientas esenciales para respaldar el desarrollo de aplicaciones BREW. Los desarrolladores pueden utilizar entornos comerciales de desarrollo integrado (IDE), tales como Microsoft Visual Studio® e IBM WebSphere Studio Device Developer, el compilador ARM o un compilador GNU gratuito. QUALCOMM ha desarrollado herramientas adicionales para firmar aplicaciones y someterlas a prueba en los dispositivos correspondientes.[35]

3.1.3.3 Soporte Proporcionado por BREW

BREW proporciona funcionalidad a las aplicaciones a través de extensiones BREW, estas extensiones posibilitan tecnologías desarrolladas por distribuidores independientes de software para agilizar el desarrollo de aplicaciones BREW. Una extensión BREW no se utiliza como una aplicación autónoma y no se compra, descarga ni elimina de manera específica por un usuario del dispositivo. Más bien, tales funciones se efectúan dentro del contexto de la aplicación que utiliza la extensión.

Algunos ejemplos de extensiones BREW son los sistemas virtuales JAVA, reproductores de video, motores de juegos, etc.

Debido a que las interfaces de usuario del dispositivo residen sobre BREW, los fabricantes pueden emplear extensiones BREW para incorporar nuevas funciones en los teléfonos celulares y corregir errores por transmisión aérea, evitando así los enormes costos de retirar productos del mercado. Los desarrolladores externos también pueden escribir extensiones en la plataforma BREW a través del programa de extensiones, proporcionando una funcionalidad adicional a las aplicaciones ya disponibles. QUALCOMM publica las extensiones y las pone a disposición de la comunidad de desarrolladores.

Para acceder a la documentación de desarrollador, a las herramientas de desarrollo, a las extensiones BREW o al programa de extensiones es necesario ser un desarrollador autenticado, es decir poseer un suscripción de pago con QUALCOMM. Lo que ha hecho muy difícil conocer detalles importantes de esta plataforma para el proyecto, como el soporte SIP, documentación e implementación de aplicaciones.

3.1.4 Linux

Esta plataforma aparece inicialmente como un Linux empotrado, que se refiere al uso del sistema operativo Linux en un sistema embebido, como por ejemplo en celulares, asistentes personales y otros dispositivos electrónicos de consumo. Un ejemplo son las PDAs (Personal Digital Assistant) Zaurus de la empresa Sharp [39]. A pesar de sus capacidades normales y su gran costo, la Zaurus fue la PDA más popular durante la década de 1990 en el Japón, debido a que utilizaba un sistema operativo Linux, lo que genero una gran atención por parte de la comunidad Linux que extendió sus capacidades software. Actualmente se encuentran muchos proyectos en torno a estas pequeñas máquinas. Desde aplicaciones de todo tipo a ROMs (En terminología de la Zaurus es el kernel y el sistema de ficheros que se inserta en la memoria de la PDA) completas como las de openzaurus [40], pdaxrom [41], o cacko [42]. Parece ser que las capacidades normales de su hardware y su mercado restringido no la hacen muy atractiva para los usuarios comunes [43].

Hasta el momento en algunos teléfonos celulares la utilización de Linux fue realizado en su mayoría utilizando código propietario escrito en assembler. Los desarrolladores debían escribir los controladores para los dispositivos hardware y las interfaces desde cero [44]. Estos teléfonos que utilizaban Linux, eran prácticamente dispositivos “basados” en Linux, por que no existía una plataforma definida de uso común para los terminales. En este año la empresa A la Mobile [45], desarrolla una nueva plataforma Linux abierta para Smartphones. Su solución comprende el kernel, el middleware y la capa de aplicación que estará disponible para los fabricantes de terminales en septiembre del año en curso.

Según la compañía, utilizar la plataforma permitirá a los fabricantes de teléfonos reducir los costes y el tiempo de desarrollo, pruebas y despliegue. Dado su carácter abierto, los fabricantes podrán personalizarla en aspectos como los componentes del interfaz de usuario o el propio navegador lo que la diferenciará de otros sistemas operativos propietarios. Competirá con Symbian, así como con RIM, Microsoft y PalmSource.

Para acelerar la aprobación de Linux como la alternativa de sistema operativo móvil a Windows y a Symbian, A la Mobile, empresa fundada en Junio de 2005 con la firme

creencia de que la industria de la telefonía móvil está lista para recibir un sistema operativo móvil abierto y no propietario, desarrolló esta plataforma de sistema Linux llamada Convergent Linux. Diseñada para ser fácilmente adaptable a los teléfonos móviles.

Sin embargo, hay quienes afirman que un sistema operativo requiere del apoyo de los grandes operadores y de los fabricantes de terminales para que tenga éxito y sea utilizado, cosa que aun no ha logrado Convergent Linux. En cierta forma, se dice que esto puede representar una ventaja al mantener su independencia y permanecer como una plataforma libre, pero también, sin el respaldo, puede solo quedarse en una propuesta. Sin embargo, las últimas noticias indican que posiblemente un grupo de fabricantes y operadores donde se incluyen a Vodafone, NTT DoCoMo, Samsung y Motorola, podrían ponerse de acuerdo con el fin de fijar las directrices del mercado de Linux móvil, tal es el caso de la Linux Phone Standards Forum [46] y la Mobile Linux Initiative [47]. A la Mobile está relacionada con ambos grupos y con el Consumer Electronics Linux Forum [48].

3.1.4.1 Plataformas de Desarrollo para Linux

A la Mobile considera que Java ME, es una plataforma de desarrollo desplegada tan extensamente, que es un componente necesario en cualquier plataforma móvil, por lo tanto la plataforma de Linux Convergent es capaz de integrar la plataforma de Java apropiada, ya sea la plataforma de JTWI o la plataforma de arquitectura de servicios próxima que está definida actualmente por la comunidad de Java.

También Integra una colección de herramientas utilizadas por la comunidad de diseño en Flash, que se utilizan en la actualidad para el despliegue de contenido móvil. Para soportar este contenido, la plataforma de Linux Convergent integra un ejecutor Flash en su pila [49].

En cuanto a los teléfonos basados en Linux, como anteriormente se menciono, son equipos que en su mayoría utilizan código propietario, lo cual dificulta la implementación de aplicaciones a los desarrolladores particulares, ya que es casi imposible acceder al soporte necesario.

3.1.4.2 Teléfonos con Linux

Hasta el momento no existe un teléfono con sistema operativo Linux como tal, pero en el mercado se encuentra gran cantidad de teléfonos basados en Linux, como el teléfono celular Motorola A1200 , Nokia 7380 y Samsung D510. Recientemente NEC desarrollo el teléfono N902iX para la nueva red 3.5G de NTT Docomo, entre sus características se destacan que puede operar con HSDPA (*high-speed downlink packet access*) lo que permite a los usuarios realizar descargas de 3.6Mbit/s y que posee un sistema operativo basado en Linux (en la dirección electrónica http://linuxdevices.com/news/NS3110082615.html se puede encontrar mas detalles de este dispositivo).

La empresa Trolltech [43] conocida por las herramientas de desarrollo y pilas de aplicación de Linux para teléfonos y otros dispositivos movibles proporciona versiones de Linux para una gran cantidad de teléfonos móviles. Su versión principal de Linux QT, se encuentra incorporada en una variedad de teléfonos, entre los que se puede destacar el Accton VM1188T, el GW1, el pioneer E28 Limited y el PWG-500 que poseen soporte para SIP y 802.11b/g lo que los posibilita realizar comunicaciones VoWiFi (*VoIP-over-WiFi*) a trabes de redes locales inalámbricas. En el sitio de Trolltech http://www.linuxdevices.com/articles/AT9423084269.html se puede encontrar una lista completa y actualizada de los teléfonos a los que esta empresa les proporciona sus productos basados en Linux. Desafortunadamente la mayoría de estos equipos basados solo se encuentran disponibles para el mercado oriental, donde las redes celulares se encuentran muy evolucionadas.

El fabricante de telefonía celular Motorola anunció que utilizará Linux en sus unidades más avanzadas, debido a que posee mayores ventajas que los sistemas operativos Windows Mobile y Symbian OS. Según lo informa la misma empresa en el mes de noviembre de 2006 sacará un nuevo celular basado en Linux. El móvil tendrá una interfaz simple e intuitiva, cámara digital de 1.3 mp con zoom digital de 8x, flash y slot para tarjetas de memoria (entre otras características). La plataforma se llama Juix, que viene de combinar Java y Linux [50].

3.1.4.3 Soporte Proporcionado por Linux

Los teléfonos móviles basados en Linux que existen hasta el momento incorporan variedad de capacidades muy importantes, que al parecer solo pueden ser explotadas por los mismos fabricantes, ya que no se proporcionan los recursos necesarios para el desarrollo por parte de terceros. Esta situación ha reducido la evolución de Linux en los teléfonos móviles ya que la comunidad de desarrollo en Linux, a pesar de estar muy extendida no puede fijar sus ojos en estos dispositivos como plataformas sobre las cuales ejecutar sus aplicaciones.

Se espera que esta situación pueda cambiar con la introducción del sistema operativo Linux para móviles proporcionado por A la Mobile o con el reciente teléfono Qtopia Greenphone [51] basado en Linux, el cual ofrece un SDK gratis para el desarrollo rápido y sencillo de aplicaciones, se dice que su firmware podría incluir SIP, lo que permitiría usarlo como teléfono Wifi además de GSM [52]. Este teléfono estará Inicialmente disponible en Asia, Europa y Estados Unidos.

3.1.5 Windows Mobile

Es un sistema operativo compacto, combinado con una serie de aplicaciones básicas para dispositivos móviles basados en la API Win32 de Microsoft. Los dispositivos que llevan Windows Mobile son Pocket PC's, Smartphones y los Media Center portátil. Ha sido diseñado para ser similar a las versiones de escritorio de Windows y cuenta con dos versiones; Windows Mobile PPC para Pocket PC y Windows Mobile Sp para Smartphones.

Inicialmente este sistema operativo se llamaba PocketPC y utilizaba Windows CE. A partir de la tercera versión lanzada en junio de 2003, tomo el nombre de Windows Mobile y vino en tres ediciones diferentes: Windows Mobile 2003 Pocket PC Edition, Windows Mobile 2003 Pocket PC Phone Edition, diseñada para los Pocket PC que tienen características de teléfonos móviles (como HTC's Himalaya, distribuido en muchos países como Qtek, XDA, MDA o VPA [53]) y la edición Windows Mobile 2003 Smartphone Edition [54], que a pesar de sus semejanzas con la de Pocket PC, es una plataforma substancialmente diferente ya que está limitada por las características especiales de este tipo de dispositivos. Algunas de estas limitaciones son:

funcionamiento por teclas al no disponer de pantalla táctil, resolución de pantalla más baja, modelo de seguridad que impide instalar aplicaciones no firmadas y modelo de memoria diferente (diferente tipo de memoria y menor cantidad). Windows Mobile 2003 es conocido también como Windows CE 4.20.

Windows Mobile 2003 Second Edition, también conocida como Windows Mobile 2003SE, salió en marzo de 2004 e incluye un número de mejoras sobre su precursor Windows 2003SE Mobile y utiliza Windows CE 4.21

Windows Mobile 5.0, anteriormente con el nombre en clave "Magneto", salió al mercado el 9 de mayo del 2005. Utiliza Windows CE 5.0 y utiliza .NET Compact Framework 1.0 SP2, plataforma de desarrollo .NET para los programas basados en .NET que utiliza.

Microsoft está trabajando actualizaciones de Windows Mobile 5.0. La primera actualización tiene el nombre en clave de Crossbow' y está programado para ser lanzado a mediados del 2007. El segundo, con el nombre en clave de Photon y está programado para ser lanzado en la primera mitad del 2008 [55].

3.1.5.1 Teléfonos con Windows Mobile

Actualmente existen alrededor de 13 terminales con Windows Mobile de los fabricantes, HP, Hand Held Producrs, Motorota, Palm y Symbol; de los cuales los Smartphones: Cingular 2125, Cingular 3125, Motorola Q, T-Mobile Dash, T-Mobile SDA, Audiovox SMT 5600 (Cingular), Audiovox SMT 5600 (Rogers), Motorola i930 (Sprint), Motorola MPx220 (Cingular), SP-i600 Smartphone (Sprint) y Voq Professional Phone se encuentran disponibles para el continente Americano. En la página http://www.microsoft.com/windowsmobile/smartphone/default.mspx de Windows Mobile se encuentra la lista actualizada de equipos que incorporan este sistema operativo.

3.1.5.2 Soporte Proporcionado por Windows Mobile

El modelo de programación de Windows Mobile es dividido en juegos o conjuntos de interfaces, las propiedades, los métodos, las funciones, los tipos de datos y las estructuras de datos. Cada juego se centra en un área específica de la funcionalidad.

El paquete de herramientas para el desarrollo en Windows Mobile para Smartphones incluye; el soporte para manejo de connexion Bluetooth, ActiveSync, Connection Manager API, Device Configuration API, File and Application Management API, Game API (GAPI), Home Screen API, HTML Control API, Messaging API (MAPI), MIDI API, Object Exchange (OBEX) API, Pocket Internet Explorer Browser API, Pocket Outlook Object Model (POOM) API, Remote API (RAPI), Speech Recognizer API, Telephony API, User Interface API, Vibrate API y Voice Recorder Control API [56]. En cuanto al soporte para SIP, hasta el momento en esta plataforma no se ha presentado oficialmente algún tipo de recurso que permita su utilización o implementación.

3.1.5.3 Plataformas de Desarrollo

Microsoft proporciona los SDKs basado en Windows que permiten el desarrollo de aplicaciones para Smartphones con Windows Mobile. La implementación se realiza sobre Visual Studio .NET 2003 y Compact Framework de .NET con C# o Visual Basic .NET.

Para versiones anteriores al 2003 se puede realizar implementaciones en C++ sobre eMbedded Visual Tools 3.0 - 2002 Edition [57].

3.2 SOPORTE SIP DE LAS PLATAFORMAS ESTUDIADAS

Para el desarrollo de aplicaciones móviles P2P con SIP se requiere que las plataformas cuenten con el soporte necesario para el protocolo y la disponibilidad de los SDKs correspondientes. La tabla 1 muestra una comparación entre las diferentes plataformas estudiadas, donde colocaron algunos aspectos importantes de cada plataforma, para el proyecto.

Plataforma		Programación	Soporte SIP	Disponibilidad de Recursos
BREW		J2ME, C/C++	- -	Gratuitos
Linux	Convergent Linux	J2ME, Flash , C++	- -	- -
	Basados en Linux	- -	Algunos	- -
J2ME		J2ME	SI	Gratuitos
Symbian		C++, J2ME y Pitón	SI	Gratuitos
Windows Mobile		C# o Visual Basic .NET	NO	Licenciados

Tabla 1. Comparación entre las diferentes plataformas estudiadas.

En algunos campos de la tabla 1, se encuentra el caracter “- -” indicando que no fue posible conocer si la plataforma disponía del recurso. En el caso de BREW, no era posible determinar si contaba con el soporte SIP necesario, ya que para poder acceder a la información precisa de este sistema operativo era necesario adquirir mediante pago, la licencia de desarrollador BREW. En el caso de los sistemas operativos basados en Linux, tampoco era posible conocer acerca de la programación y la disponibilidad de resursos debido a que son sistemas propietarios, los cuales solo son explotados por sus propios fabricantes. Para el caso de Convergent Linux la información no se encontraba aun disponible, posiblemente por que aun le falta definir

En resumen, de las cinco plataformas estudiadas, solo J2ME es una plataforma de desarrollo. BREW, Symbian y Windows Mobile son Sistemas Operativos y Linux solo hasta hace poco paso de ser una referencia a un sistema operativo para teléfonos.

Teniendo en cuenta que de las cinco plataformas anteriormente vistas solo para J2ME y Symbian se encuentran disponibles de forma gratita los recursos necesarios para poder realizar aplicaciones sobre dispositivos móviles que hagan uso del protocolo SIP, el presente trabajo se centrará en lo referente al desarrollo de las aplicaciones P2P sobre dispositivos Móviles con SIP sobre estas dos tecnologías, resaltando además que entre las dos abarcan la mayor parte del mercado de aplicaciones para dispositivos móviles y que en la FIET los desarrollos para dispositivos móviles se realizan sobre principalmente sobre J2ME. El anexo D Tecnologías para el desarrollo

de aplicaciones móviles P2P con el protocolo SIP contiene información mas profunda sobre el desarrollo de aplicaciones móviles con SIP sobre J2ME y Symbian.

Se desea aclarar que la interacción entre aplicaciones de diferentes plataformas bajo un mismo protocolo como SIP por ejemplo, es un proceso transparente para el usuario y libre de problemas, ya que la negociación de capacidades (codecs y medios) de los equipos terminales es realizada entre las mismas aplicaciones finales, las que controlan la comunicación SIP. Esta característica de SIP genera un atractivo para el desarrollo y utilización, ya que es una tecnología “libre”, al no poseer un único o grupo cerrado de fabricantes que la puedan utilizar y que permite la comunicación desde cualquier terminal independientemente de su tecnología de desarrollo.

4. RECOMENDACIONES PARA EL DESARROLLO DE APLICACIONES SIP P2P.

Iniciarse en el desarrollo de aplicaciones móviles P2P que utilicen el protocolo SIP, puede generar algunas complicaciones y demoras, relacionadas principalmente con las herramientas a utilizar y el funcionamiento del protocolo en si. Para sobre pasar este obstáculo se hace necesario no solo tener un conocimiento del estándar del protocolo a utilizar, si no también de una de las plataformas disponibles para el desarrollo, las herramientas de desarrollo, su configuración y la implementación entre otros.

Partiendo de este hecho se describen a continuación una serie de recomendaciones orientadas a la creación de aplicaciones P2P con SIP, con las que se pretende cubrir los aspectos que se consideran relevantes en el desarrollo de aplicaciones P2P Móviles con SIP. Estas recomendaciones están dirigidas principalmente a las personas que desean iniciarse en la creación de estas aplicaciones, pero también contienen lineamientos importantes que sirven de guía para desarrollos más avanzados. Las recomendaciones cubren aspectos como la creación de los elementos mínimos necesarios que debe tener una aplicación de este tipo, para garantizar una correcta interacción entre Agentes de Usuario y la configuración de herramientas necesarias y opcionales para el desarrollo. Las recomendaciones presentan algunos ejemplos, con el fin de mostrar la forma recomendada para elaborar algún tipo de estructura.

Teniendo en cuenta que la programación en general es un aspecto subjetivo, estas recomendaciones no pretenden ser una regla imprescindible para el desarrollo de las aplicaciones móviles P2P con SIP, pero si un punto de partida que minimice las dificultades que pueden presentarse al comenzar a implementar estas aplicaciones.

4.1 RECOMENDACIONES INICIALES

4.1.1 Puntos importantes

Asumiendo que el interesado en desarrollar aplicaciones móviles SIP P2P posee conocimientos previos en algoritmos de programación y programación orientada a objetos se presentan algunos puntos iniciales que pueden indicar el camino a seguir en el desarrollo de estas aplicaciones. Los puntos son:

Documentarse respecto a SIP. Es claro que si se desea realizar aplicaciones móviles P2P con SIP es necesario que se deba conocer lo relacionado con el protocolo SIP. En la Web existe gran cantidad de documentos y páginas que hablan de SIP, esta información sirve para darse una idea general de su funcionamiento, arquitectura, servicios propuestos, futuro y relación con otros estándares; pero para llegar al desarrollo de aplicaciones es necesario conocer a fondo su estándar, por lo que es preciso revisar su especificación. La RFC 3261 es la especificación de SIP y todo lo que se debe conocer sobre su funcionamiento se encuentra en ese documento. Generalmente documentos diferentes a la especificación que hablan de SIP se basan en esta RFC y tratan el tema muy superficialmente, pero a cambio permiten entender lo relacionado con SIP de una forma más sencilla, lo que puede no suceder si se decide iniciar directamente con la especificación. Por lo cual es mejor comenzar por los documentos que hablan de SIP de una forma superficial y luego ir al documento base para lograr una mejor comprensión del protocolo.

A continuación se listan algunos documentos recomendados que se encuentran en la Web que contienen información importante para iniciar a conocer a SIP. Estos documentos también se encuentran disponibles en el CD adjunto.

- Revista Telem@tica numero 14. Telem@tica_AnoIII_No14.pdf
- Revista Telem@tica numero 15. Telem@tica_AnoIII_No15.pdf
- Estándares relacionados a la tecnología Voz sobre IP. EstandaresVoIP.pdf
- Modelo de Aplicación de Sesión Multimedia. ModeloASM.pdf

- Session Initiation Protocol (SIP) and MCIAdvantage. WP8132.a.SIP White Paper.pdf
- 01-SIP- Diego-Acosta.pdf
- VozIP.pdf

Elegir una plataforma. Algunos puntos importantes para la elección de la plataforma se comentan en el punto “Elección de la plataforma” en este mismo capitulo.

Elegir un IDE. Algunos elementos importantes para la elección de un IDE son discutidos en el punto “Elección del IDE” en este mismo capitulo.

Revisar y construir ejemplos. La revisión de ejemplos existentes es una buena forma para entender como se implementa SIP en una plataforma. Los JDKs para J2ME y Symbian incluyen algunos ejemplos guía para el desarrollo de estas aplicaciones. Por ejemplo para J2ME se encuentra el archivo MIDP_SIP_API_Example_v1_0_en.zip que contiene cuatro ejemplos básicos que muestran el manejo de las peticiones y respuestas de una forma muy clara y simplificada, este archivo también puede ser descargado de las páginas del forum de Nokia. Para Symbian existe ejemplo SIPExample que se encuentra alojado en el directorio \\Symbian\9.1\S60_3rd\S60Ex\SIPExample donde se instaló el SDK de Symbian S60, el cual contiene un ejemplo más avanzado donde se establece una sesión de juego entre dos participantes. Estos ejemplos también se encuentran disponibles en el CD adjunto al proyecto.

Complementar la documentación. Una vez entendido lo relacionado con SIP, la revisión de documentación adicional, permitirá obtener una comprensión completa y su interacción con otros protocolos y la red. Entre la documentación adicional se recomienda revisar el protocolo SDP (RFC 2327), RTP (RFC 3550) y SAP (RFC2974).

Finalmente, la página del forum de Nokia (www.forum.nokia.com) contiene información importante para la comprensión del protocolo SIP y temas que se relacionan con él, al igual que algunos ejemplos base para la construcción de aplicaciones móviles con SIP y las RFCs mencionadas.

4.2 RECOMENDACIONES INICIALES

Las características que posee una aplicación P2P con SIP influyen directamente en algunos pasos de su desarrollo, como son: la elección de la plataforma de desarrollo y el IDE mas apropiado, por lo cual a continuación se hará una revisión de estas características para determinar como realizar una mejor elección.

Las aplicaciones móviles P2P con SIP tienen dos grupos de características, el primer grupo corresponde a las características heredadas de SIP, las cuales son iguales para todas las aplicaciones P2P con SIP pero que las diferencian de las aplicaciones móviles tradicionales (entendiéndose como aplicación tradicional, una aplicación que no hace uso de SIP). Las características heredadas de SIP se listan a continuación:

- Requiere del soporte del protocolo SIP para su funcionamiento y aprovechar las capacidades de presencia y localización.
- Independencia de la red, las aplicaciones pueden ser utilizadas sobre redes celulares o redes Wifi.
- Soportan URIs SIP del tipo IPv4 y IPv6. por ejemplo tipo URL, ENUM, TEL, etc
- Puede comunicarse con diversos dispositivos diferentes a los móviles de manera más natural.
- Manejo del concepto de sesión, gestionándola mas no transportando medios.
- Realiza una descripción de sesión, mediante SDP o MIME.
- Puede negociar las capacidades disponibles para una transferencia de un medio entre dos usuarios, buscando los recursos disponibles (protocolo, codecs, etc) mas apropiados.
- Puede establecer múltiples transferencias de información durante una sesión entre dos o más usuarios al mismo tiempo.

- El control de la sesión o comunicación se realiza principalmente desde los equipos móviles, facilitando modificar los parámetros de esta en plena ejecución.

El segundo grupo corresponde a las características particulares, las cuales surgen de las capacidades y requerimientos propios de cada aplicación y que se desea que posean para llevar a cavo su funcionamiento.

- Los protocolos utilizados (para el transporte de video, audio o archivos por ejemplo).
- Despliegue en pantalla.
- Acceso a los contenidos de la memoria (datos, archivos, contactos, etc).
- Hardware requerido (cámara, vibración, control de batería, puertos etc).
- Acceso a información del sistema operativo (datos de configuración e información de operación).
- Red sobre la cual operará la aplicación (móvil celular o Wifi).

De las características anteriormente expuestas se puede encontrar que las características generales son implícitas al protocolo SIP y que sea cual sea su plataforma de desarrollo o IDE estas se mantendrán constantes. Esto se debe a que SIP se encuentra especificado para funcionar de igual manera sobre cualquier dispositivo que haga uso de él, razón por la cual los SDKs SIP para las diferentes plataformas pueden proporcionan métodos de codificación superficialmente diferentes pero que en el fondo conllevan a realizan la misma acción SIP.

En cuanto a las particulares no ocurre lo mismo, se encuentra que estas características no pueden ser pasadas por alto a la hora de la elección de la plataforma de desarrollo. Esto se debe principalmente a:

- Symbian y J2ME no proporcionan el mismo acceso a los recursos del dispositivo.
- No proporcionan APIs iguales o equivalentes de soporte.

- La riqueza de APIs y métodos disponibles para el desarrollo no es igual.
- No manejan la misma portabilidad.

Teniendo en cuenta las características particulares de las aplicaciones y las diferencias entre las plataformas presentadas se presenta a continuación la recomendación para elegir la plataforma mas apropiada.

4.1.2 Elección de la Plataforma

Para elegir la plataforma sobre la cual desarrollar aplicaciones Móviles P2P con SIP será necesario contrastar los recursos que brinda cada plataforma con las necesidades de cada aplicación. La tabla 2 presenta algunos de los requerimientos que puede tener una aplicación P2P con SIP y la existencia de soporte en cada plataforma.

Característica de la aplicación	Recurso necesario	J2ME	Symbian
Video	Capturar Fotografía	SI	SI
	Capturar Video	SI	SI
	Transferencia video	NO	SI
Audio	Grabación	SI	SI
	Lectura	SI	SI
	Transferencia	NO	SI
Mensajería	E-Mail	NO	SI
	SMS	SI	SI
	MMS	SI	SI
	Bluetooth	SI	SI
	WAP	SI	SI
	T.P	SI	SI
	FAX	NO	SI
	Seguridad	SI	SI
Manejo de hardware	Teclado	SI	SI
	Estado Batería	NO	SI
	Vibración	SI	SI
Datos	Acceso a archivos	SI	SI
	Manejo de datos	SI	SI
Pantalla	Gráficos	SI	SI

	Animaciones	SI	SI
	Juegos	SI	SI
Transporte de datos	Firma digital	SI	SI
	Encriptación	SI	SI
	Protocolos seguros	SI	SI

Tabla 2. Soporte de J2ME y Symbian

La diferencia principal en el soporte de las dos plataformas se encuentra en el transporte de audio y video en tiempo real. En J2ME no es posible transmitirlo lo que obliga a que los desarrollos que requieran de este tipo de transferencias sean necesariamente realizados en C++ para Symbian. Para las demás características no existe este inconveniente, pero esto no debe conllevar a la elección de la plataforma de una forma apresurada, se debe proseguir con la evaluación de aspectos como la portabilidad, la funcionalidad y el desempeño esperado. En cuanto a estos aspectos por condiciones propias de ejecución de cada plataforma, se considera que una aplicación puede tener un mejor desempeño en Symbian que en J2ME, esto es debido a que Symbian se ejecuta con código nativo sobre el dispositivo, logrando una mejor ejecución. J2ME por el contrario se ejecuta sobre una maquina virtual, lo cual produce una exigencia mayor de recursos, lo que conlleva a un desempeño menor que en Symbian, pero a cambio le proporciona una gran portabilidad, permitiendo que la misma aplicación terminada pueda ser utilizada en diversos dispositivo con soporte J2ME, que a diferencia de Symbian una aplicación debe ser realizada para cada edición de la S60 del sistema operativo. En cuanto a la funcionalidad Symbian provee a los desarrolladores con más componentes de interfaz, métodos de codificación y APIs que J2ME lo que facilitan el desarrollo de aplicaciones más elaboradas, intuitivas y atractivas para los usuarios.

De los aspectos anteriormente expuestos se puede recomendar que:

Si la aplicación requiere de la transferencia de audio o video en tiempo real obligatoriamente se realice en C++ para Symbian.

SI no se requiere de la transferencia de audio o video en tiempo real, pero se desea un gran desempeño y/o funcionalidad, se recomienda utilizar Symbian.

SI no se requiere de la transferencia de audio o video en tiempo real, pero se desea que la misma aplicación pueda ser utilizada en una gran variedad de dispositivos, la mejor opción es J2ME.

SI no se requiere de la transferencia de audio o video en tiempo real, pero se desea portabilidad, gran desempeño y/o funcionalidad lo recomendable es elegir Symbian y compilar la misma aplicación para cada edición de la serie.

Si la aplicación no requiere de la transferencia de audio o video en tiempo real, y la funcionalidad, desempeño y portabilidad no representan mayor problema, se podría elegir cualquiera de las plataformas o recurrir a aspectos no relacionados con la aplicación, si no a facilidades propias del desarrollador como por ejemplo cual plataforma le es mas familiar.

4.1.3 Elección del IDE

Para elegir el IDE apropiado para el desarrollo de las aplicaciones móviles P2P con SIP se debe buscar inicialmente dos cosas: que funcione sobre el sistema operativo Windows, debido a que los SDKs de la serie 60 de Nokia y el Plug-In SIP solo están disponibles para este sistema operativo y que pueda trabajar con el SDK escogido o plugin. La tabla 3 contiene algunos IDEs que soportan los SDKs de la serie 60 de Nokia y el plugin SIP, adicionalmente el IDE debe permitir los procesos de codificación, compilación, depuración y simulación de las aplicaciones.

Actualmente existen algunos IDEs extensibles (como Eclipse y Netbeans) que además de ser gratuitos permiten mediante algunos plugins adicionales la implementación de aplicaciones sobre múltiples plataformas, incluida J2ME y Symbian; el manejo y/o configuración de estos IDEs suele contener mas pasos y detalles que deben ser tenidos en cuenta, a diferencia de los IDEs no extensibles, los cuales están diseñados para soportar un solo lenguaje, su manejo y configuración suelen ser muy sencillos lo que facilita su utilización.

IDE	Soporte J2ME	Soporte C++ (Symbian)	Licencia
Eclipse	Mediante un paquete	Mediante Carbide C	Libre
NetBeans	Mediante un paquete	Mediante Carbide C	Libre
JBuilder	SI. Para versiones menores a la 7 requiere un plugin	NO	Licenciado
Visual Studio .NET	Mediante Carbide J	Mediante Carvide VS	Licenciado
Visual Studio 6.0	Mediante Carbide J	Mediante Carvide VS	Licenciado
CodeWarrior	NO	SI	Licenciado
NokiaDeveloper's Suite for the Java™ 2 Platform, Micro Edition	SI	NO	Licenciado
Sun J2ME Wireless Toolkit	SI	NO	Libre
IBM Websphere Studio Device Developer	SI	NO	Licenciado

Tabla 3. IDEs que soportan los SDKs y/o el Plugin SIP.

Para los desarrollos en J2ME se recomienda no utilizar el Wireless Toolkit como IDE de desarrollo, debido a que no se puede realizar la emulación de la aplicación sobre el. Esto dificulta y/o hace más lento el proceso de desarrollo, por lo cual es aconsejable utilizar un IDE diferente.

Para los desarrollos en C++ para Symbian se recomienda utilizar un entorno genérico como NetBeans o Eclipse debido a que el Carvide C se acopla directamente sobre el IDE creando un entorno de desarrollo independiente, es decir si tenemos instalado Eclipse e instalamos Carvide C, este genera un nuevo entorno que solo se puede utilizar para el desarrollo de aplicaciones en C++ para Symbian. La ventaja de este entorno esta en lo liviano y fácil de manejar, ya que solo contiene y presenta lo necesario para el desarrollo de aplicaciones móviles para Symbian.

4.1.3 Elección de Plugin o SDK SIP

J2ME y Symbian poseen Plugins SIP que pueden ser instalados luego de la instalación de los SDKs de la plataforma elegida, pero estos plugins muchas veces generan una serie de problemas a la hora de ser instalados, básicamente los problemas son consecuencia de que no logran localizar al SDK de la plataforma requerido, ya sea por que existe otro SDK instalado que se ha configurado con la mayor prioridad o por que en algunos casos el plugin que buscan no coincide con alguno especifico. Para evitar este tipo de inconvenientes que pueden producir una perdida de tiempo innecesaria, se recomienda instalar mejor las últimas versiones de los SDKs las cuales ya incorporan el soporte SIP, estos SDKs se encuentran disponibles para J2ME y Symbian. El SDK que incluye el soporte SIP para J2ME es el Nokia_Prototype_SDK_4_0_Beta y el SDK que incluye el soporte SIP para Symbian es el S60-SDK-200634-3.1-Cpp-f.1090b, estos SDKs pueden ser descargados del sitio www.forum.nokia.com

4.1.3 Conexión de Red

Para que las aplicaciones SIP puedan ser probadas con éxito es necesario tener una conexión red aunque se utilice los dos emuladores en el mismo equipo. Parece que cuando la tarjeta de red no posee una conexión hacia física los mensajes no pueden ser enviados al no detectarse conexión alguna, para esto se debe contar por lo menos con una conexión a un concentrador, a otro PC de forma cruzada o en ultimas (también funciona bien) tener dos tarjetas de red en el mismo PC y conectarlas cruzadas.

4.1.3 Consideraciones Adicionales

SIP por ser independiente de la red puede operar sobre redes Wifi y sobre redes celulares. En las redes Wifi los anchos de banda manejados generalmente permiten transferir cualquier tipo de medio soportado por los móviles, lo único a tenerse en cuenta es que el teléfono obligatoriamente debe poseer este tipo de conexión. En las redes celulares se debe tener claro sobre que tipo de tecnología se transmitirá la información para según esto determinar el alcance de la aplicación a implementar. La tabla 4 muestra que transferencias SIP son posibles sobre cada tecnología existente

en las redes de telefonía móvil presentes en Colombia. Las velocidades especificadas para cada tecnología son el resultado de un estudio que realizo la empresa Comcel acerca de cual es la velocidad real que se presenta.

Tecnología	Velocidad	Transferencia SIP
GSM	9 kbits/s	Nada
GPRS	30 kbits/s	Texto, pequeños archivos
EDGE	90 kbits/s	Texto, archivos, video de muy baja calidad

Tabla 4. Tecnologías y medios transmitidos.

Por las velocidades manejadas en las redes móviles de Colombia solo se pueden obtener buenos resultados en la transferencia de texto y pequeños archivos, debido a los bajos anchos de banda consumidos y a que aceptan algunos retrasos en las transferencias, esto restringe los desarrollos a aplicaciones basadas en estos tipos de transferencias, como por ejemplo: los chat, transferencia de fotografías, compartir archivos, juegos sincronizados por mensajes, colaboración, transacciones, compras, servicios de información, etc.

En cuanto al despliegue de las aplicaciones, no se considera que represente una limitante ya que en primer lugar para poder ejecutar aplicaciones móviles P2P con SIP se requiere de un teléfono con soporte SIP, y hasta el momento los equipos que cuentan con esta funcionalidad son teléfonos de gama alta, los cuales cuentan con gran capacidad operativa, de memoria, conectividad y pantalla. Como segundo se debe recordar que las aplicaciones se ejecutan sobre los teléfonos y la red solo sirve para el transporte de los datos, lo que obliga a que las aplicaciones sean cargadas a los dispositivos por los usuarios, en J2ME estos tendrán que saber o probar si es posible ejecutar la aplicación en su terminal mientras que en Symbian cada aplicación se encuentra especificada para cada serie de la edición, lo que garantiza su operación.

4.2 RECOMENDACIONES REFERENTES A LAS HERRAMIENTAS

4.2.1 Instalación del Plugin para Symbian

Para instalar el plugin para Symbian se recomienda seguir los pasos descritos en el punto Desarrollos SIP en C++ del Anexo B Preparación del entorno de desarrollo y creación de proyectos. Adicionalmente se debe tener presente ciertas situaciones:

Durante la instalación se recomienda elegir los valores que vienen por defecto para la instalación. Si por alguna razón se desea cambiarlos, se debe asegurar que

- El SDK y Perl están instalados en la misma unidad
- No existan espacios en blanco en los nombres de los directorios.

Esto se debe a que el SDK usa scripts de Perl para realizar ciertas operaciones, y si los anteriores puntos no se cumplen, puede causar problemas posteriormente.

También se debe asegurar que no se tenga ningún otro Perl en el sistema. Por ejemplo, Matlab usa Perl y tiene su propio intérprete de Perl, que puede crear conflictos con el instalado para el SDK, ya que normalmente se encuentra en el directorio de Matlab que esta en el PATH.

4.2.2 Instalación del SDK para J2ME

La instalación de este SDK es menos conflictiva pero para evitar dificultades se recomienda revisar el punto 1 Desarrollos SIP en j2me del anexo B Preparación del entorno de desarrollo y creación de proyectos.

4.2.3 Uso de Multi-Emulación en un Mismo PC

Una ayuda muy importante que presta el emulador del SDK 4.0 para J2ME es la posibilidad de emular varios dispositivos en un mismo PC. Se recomienda utilizar este

recurso para evitar conflicto de puertos que suele suceder al ejecutar una o varias aplicaciones en diferentes emuladores corriendo sobre un mismo PC. Para configurar el emulador se puede seguir los pasos descritos en el ítem 1.7 Configuración de Multi-Emulador en el ”Nokia Prototype Sdk 4.0 Beta For JME” del Anexo B.

4.2.4 Configuración de uso de Recursos

Los SDKs de la serie 60 de Nokia entre sus capacidades cuentan con la posibilidad de simular diferentes modos o niveles de operación, los cuales pretenden realizar un funcionamiento similar a como lo hace un dispositivo real mediante un comportamiento simulado, limitando y permitiendo el uso de ciertos recursos del dispositivo mediante permisos de acceso a las aplicaciones. Esta posibilidad puede también producir muchas molestias si no se configura adecuadamente, por lo cual se recomienda revisar el item 1.8 Configuración de uso de Recursos de los SDKs de la Serie 60 de Nokia del Anexo B.

4.3 RECOMENDACIONES DE IMPLEMENTACIÓN

4.3.1 Pautas para Mejorar el Rendimiento de las Aplicaciones en C++

Se recomienda revisar el documento adjunto al SDK para C++ “series_60_2_6_opengl_es_3d_sw_programmer_guide.pdf” el cual contiene un capitulo (capitulo 6, *Programming Tips*), donde se exponen varias pauntas para aumentar el rendimiento y algunas notas generales para ser consideradas al diseñar una aplicación cualquiera para Symbian.

4.3.2 Estado de la Sesión

Durante la existencia de una sesión, los UAs deben intercambiar una serie de mensajes para negociar las acciones a realizar; pero estas acciones no solo están determinadas por los mensajes recibidos sino también por el estado actual de la

sesión. Por ejemplo, no se debe tratar de la misma forma la cancelación de un INVITE enviado antes y después de recibir la respuesta temporal 100. Para determinar en que estado se encuentra la sesión se recomienda crear una variable (o varias según el caso) que guarde el estado actual de la sesión con el fin de determinar la acción mas apropiada a realizar cuando se recibe un mensaje determinado.

A continuación como ejemplo se muestra la variable EstadoSesion con tres valores posibles que toma cuando la sesión se encuentra en cada estado.

EstadoSesion = “libre”

EstadoSesion = “timbrando”

EstadoSesion = “enLinea”

4.3.3 Envío y Recepción de Mensajes

El protocolo SIP se diseño para operar de forma asíncrona, que se controla mediante solicitudes y respuestas consecutivas. La falta de un mensaje ya sea solicitud o de respuesta en una comunicación puede producir errores en los UAs. Se recomienda implementar la secuencia correcta de mensajes de comunicación ya que estos son elementos invariables del protocolo. La especificación de SIP (RF3261) presenta claramente la secuencia de mensajes que tienen lugar en una comunicación y que debe implementar un UA para proporcionar una comunicación libre de fallas. Adicionalmente se debe tener en cuenta los siguientes puntos en un intercambio de mensajes:

- Si se envía una solicitud, siempre se debe esperar por lo menos una respuesta.
- Si se recibe una solicitud, se debe enviar una contestación.
- Si se recibe una respuesta a una solicitud previa, se puede enviar una nueva solicitud.
- Si se cancela una solicitud en progreso, se espera una confirmación de la cancelación y no habrá respuesta a la solicitud.

- Si se envía una solicitud de terminación (BYE), se espera una confirmación.

4.3.4 Orden de las Cabeceras

El orden de los campos del encabezamiento no es significativo; sin embargo, se recomienda que los campos que un Proxy procesa aparezcan de primeros para lograr un rápido análisis gramatical. Entre estos campos están: Via, Route, Record-Route, Proxy-Require, Max-Forwards, y Proxy-Authorization.

4.3.5 Contenido de las Cabeceras de una Respuesta

Para llenar los valores de las cabeceras de una respuesta a una solicitud recibida se deben seguir las siguientes recomendaciones:

El campo From de respuesta debe ser igual al campo de la cabecera From de la solicitud.

El campo de la cabecera Call-ID de la respuesta debe ser igual al campo Call-ID de la cabecera de la solicitud.

El campo CSeq de cabecera de la respuesta debe ser igual al campo CSeq de la solicitud.

El contenido del campo Via de la cabecera en la respuesta deben ser igual al contenido del campo Via de la solicitud y deben mantener la misma clasificación.

Si una solicitud contiene una etiqueta To en la solicitud, el campo To de la cabecera de la respuesta debe ser igual al de la solicitud; sin embargo, si el campo To de la cabecera en la solicitud no contenía una etiqueta, la URI del campo To de la cabecera en la respuesta de la URI debe ser igual al campo To de la cabecera. Adicionalmente, el UAS debe agregar una etiqueta (tag) al campo To de la cabecera en la respuesta (excepto en las respuestas 100). La misma etiqueta debe ser usada en todas las respuestas finales y provisionales de esa solicitud con excepción de la 100.

La creación de una respuesta con sus cabeceras puede ser simplificado mediante el uso de métodos (como el initResponse en J2ME), que contienen las plataformas para

generan respuestas, los cuales llenan automáticamente los campos de las cabeceras de la misma forma como se describió anteriormente.

4.3.6 Almacenamiento del Contenido de las Cabeceras

Es recomendable que todas las cabeceras más el contenido extraído del cuerpo del mensaje sean almacenadas en variables temporales mientras dure la sesión, con el fin de poder realizar mas adelante algunos procesos como comparar el CALL-ID, incrementar el Cseq, modificar la sesión entre otros. Como el número de campos al igual que el contenido del cuerpo del mensaje puede variar en cada solicitud se puede utilizar para su almacenamiento una estructura redimensionable, como un arreglo bidimensional de cadenas, por ejemplo, en el cual se almacena la etiqueta y su contenido. La figura 2 muestra un arreglo de cadenas que contiene el contenido de las cabeceras

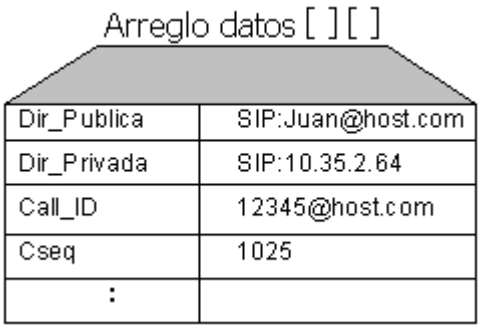

Figura 2. Arreglo bidimensional con la información de un INVITE recibido.

Aunque en una sesión la creación de respuestas puede no requerir de esta información ya que puede ser extraída de las mismas solicitudes, la modificación o ampliación de una sesión necesita de esta información para componer las solicitudes; adicionalmente debe incorporar mecanismos de actualización, ya que algún contenido puede ser cambiado o ampliado durante la negociación de capacidades o el intercambio de contenidos.

4.3.7 Creación del Valor del Campo Call-ID

El campo Call-ID requiere que el UA sea capaz de entregarle un valor que no sea igual al de otro UA. Una buena forma de crear este valor es componiéndolo de una parte aleatoria, mas la dirección del dominio del *host*; por ejemplo: <CallID:12345@gota.info.com >.

4.3.8 Procesamiento de Respuestas

Un UA debe contener lo necesario para procesar los métodos de respuesta mas convenientes de las seis clases especificadas (1xx, 2xx, 3xx, 4xx, 5xx y 6xx) y debe tratar las respuestas finales con códigos no reconocidos, como si se tratase de la primera respuesta del grupo (x00). Por ejemplo, si un UAC recibe un código de respuesta no reconocido 431, puede asumir que existe un error en la solicitud que realizó y tratarla como si hubiera recibido un código de respuesta 400 (Solicitud Mala). A continuación se muestra un ejemplo de tratamiento de los códigos de error (4xx).

SI(codRespuesta = 400) { tratamiento para solicitud incorrecta }

SI(codRespuesta = 401) { tratamiento para no autorizado }

SI(codRespuesta = 402) { tratamiento para apliación incorrecta }

:

SINO{ tratamiento para solicitud incorrecta }

4.3.9 Envío de Respuesta Provisional

El procesamiento de una invitación puede tomar algún tiempo al UA receptor por lo que todo INVITE recibido debe producir la devolución inmediata de una respuesta provisional con código 100 (intentando), para avisar al emisor de la situación. Esta recomendación no se debe aplicar a solicitudes recibidas diferentes a un INVITE, mas bien se debe generar una respuesta final lo más pronto posible.

4.3.10 Construcción de Peticiones

Con el fin de que las solicitudes realizadas por la aplicación en desarrollo sean entendidas y contengan los elementos esenciales para la comunicación con un servidor Proxy SIP, se recomienda que cualquier solicitud SIP contenga como mínimo los siguientes campos de cabecera: To, From, CSeq, Call-ID, Max-Forwards, y Via; todos éstos campos de cabecera son obligatorios en todas las solicitudes SIP.

4.3.11 Orden de Procesamiento de Peticiones

Los UASs deben seguir el siguiente orden para procesar las solicitudes recibidas:

1. Realizar la autenticación: opcional, si se incorpora mecanismos de autenticación.
2. Inspeccionar el método: el UAS debe revisar el método de la solicitud. Si el UAS reconoce pero no soporta el método de una solicitud, debe generar la respuesta 405 (Método no Permitido).
3. Revisión de los campos de cabecera: extrayendo y guardando los contenidos de los campos de las cabeceras. Si un campo de cabecera en una solicitud no es entendido, debe ser saltado y continuar con el procesamiento del mensaje.
4. Procesamiento del contenido: examinando el cuerpo del mensaje, y los campos de cabecera que lo describen.
5. Procesar la solicitud: una vez realizadas todas las verificaciones la aplicación debe efectuar lo correspondiente a la solicitud recibida.
6. Generar la respuesta: devolviendo el código de respuesta apropiado a trabes de la transacción del servidor de quien recibió la solicitud.

4.3.11 Uso del Diálogo

Ya que existe la posibilidad de enviar métodos SIP con una URI de destino o sobre diálogos establecidos, se recomienda implementar métodos sobrecargados para

manejar los diferentes métodos SIP (INVITE, MESSAGE, OPTIONS, entre otros), con estas dos posibilidades y tener en cuenta a la hora del desarrollo, que existen algunas clases que solo pueden utilizarse con diálogos establecidos, por ejemplo CSIPConnection en C++ que solo debe usarse para enviar solicitudes asociadas a un dialogo. A continuación se muestra un ejemplo que declara un método sobrecargado para el método SIP INVITE.

```
enviarINVITE(cadena URIdestino){

}

enviarINVITE(dialogo Dialog){

}
```

4.3.12 Uso de OPTIONS

Para evitar que un UA sea invitado a una sesión para la cual no posee el soporte necesario, se recomienda implementar dos cosas:

1. Para el UA que invita: antes de un INVITE enviar primero un OPTIONS para averiguar si el UA de destino tiene el soporte requerido para iniciar la sesión a la que se le va a invitar.
2. Para el UA invitado: implementar la respuesta al OPTIONS con la lista de capacidades disponibles para que solo sea invitado de acuerdo con las capacidades presentadas.

En la figura 3 se muestra un ejemplo de una invitación a una aplicación de Chat, el que invita (UA1) y el invitado (UA2), poseen el soporte para manejar mensajes OPTIONS y determinar si es viable enviar la invitación.

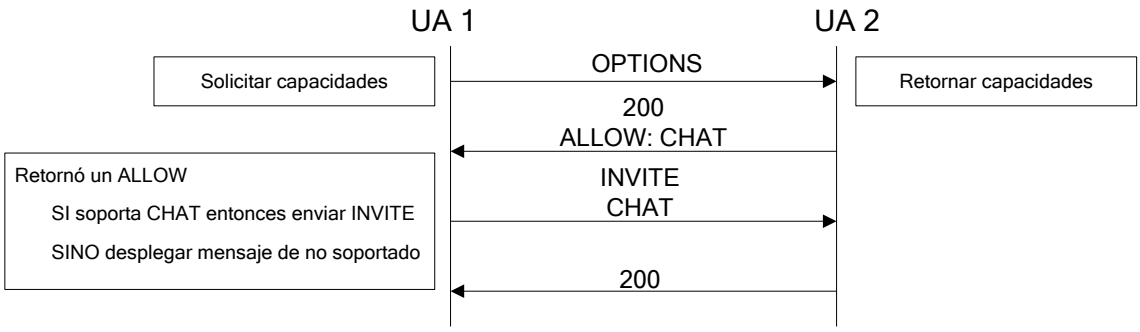

Figura 3. Secuencia de mensajes para una solicitud OPTIONS.

4.3.13 Construcción de una Solicitud OPTIONS

Una solicitud OPCIONS, es construida con los campos mínimos requeridos para una solicitud SIP como se recomienda en el punto “Construcción de peticiones” mas un campo de Contacto con la dirección privada del UA y un campo Accept que debe ser incluido para indicar el tipo de cuerpo del mensaje que el UAC espera recibir en la respuesta. Típicamente, se establece con un formato que se usa para describir las capacidades de los medios de comunicación de un UA, como SDP (application/sdp).

Ejemplo de una solicitud OPTIONS

OPTIONS sip:carol@chicago.com SIP/2.0

Via: SIP/2.0/UDP pc33.atlanta.com;branch=z9hG4bKhjhs8ass877

Max-Forwards: 70

To: <sip:carol@chicago.com>

From: Alice <sip:alice@atlanta.com>;tag=1928301774

Call-ID: a84b4c76e66710

CSeq: 63104 OPTIONS

Contact: <sip:alice@pc33.atlanta.com>

Accept: application/sdp

Content-Length: 0

4.3.14 Construcción de una Solicitud REGISTER

Para implementar la opción de registro, esta debe comenzar por activar la función de escuchar peticiones SIP, en el puerto establecido; con el fin de asegurar que el UA realmente está a la espera de peticiones en la dirección que se registró.

Para asegurar que el registro tenga éxito los campos deben tener la siguiente información:

Request-URI: Debe contener la dirección SIP del servidor Proxy o dominio al cual se desea registrar. Por ejemplo: sip:host@info.com.

To: Este campo contiene la dirección a registrar, es decir el nombre público con el que otros UAs lo intentaran buscar.

From: El campo From contiene la dirección de registro (o dirección pública), de la persona responsable del proceso de registro. El valor es igual al campo To a menos que la solicitud de registro la realice una tercera parte.

Call-ID: Debe contener un identificador globalmente único, que no coincida con el de otro UA. Ver Creación del valor del campo Call-ID.

CSeq: Este campo en las solicitudes de registro permite el ordenamiento de solicitudes de REGISTRO. Un UA debe incrementar los valores CSeq en uno para cada solicitud de REGISTRO con el mismo Call-ID.

Contact: Debe contener la dirección SIP del UA en la cual se encuentra “escuchando” peticiones SIP. Este campo puede contener uno o varios valores en los cuales estará escuchando.

4.3.15 Construcción una Solicitud CANCEL

Se debe tener en cuenta que la solicitud CANCEL solo debe ser usada para cancelar una Invitación que está en progreso, debido a que con solicitudes diferentes a un INVITE, puede crear una condición de problema para ambos UAs.

Adicionalmente para evitar otra serie de conflictos, solo se debe enviar un CANCEL después de recibir una respuesta provisional y antes de una respuesta final.

Para evitar enviar una cancelación en un momento inapropiado, la orden de cancelación generada por el usuario debe realizar primero una verificación del estado de la sesión ,para determinar si se debe esperar por la respuesta provisional, o enviar un CANCEL si el INVITE aun esta en progreso, o un BYE si el destino ya acepto la invitación. A continuación se presenta un ejemplo de implementación que revisa el estado de la sesión antes de determinar la solicitud a enviar.

HACER MIENTRAS (no se ha recibido respuesta provisional){ Esperar }

SI (destino ha aceptado){ Enviar BYE }

SINO {Enviar CANCEL}

Los campos que se deben incluir en una solicitud de cancelación son: Request-URI, Call-ID, To, CSeq y From. Estos deben contener el mismo valor que los campos del INVITE que está siendo cancelado, incluyendo las etiquetas.

4.3.16 Manejo de Presencia

En cuanto al manejo de la presencia existe la especificación JSR 164 SIMPLE Presence, que describe su manejo desde la plataforma J2ME; hasta el momento esta especificación no tiene una implementación que permita realizar desarrollos sobre esta plataforma, por lo que se recomienda manejar la presencia con el método OPTIONS. Este tema se encuentra tratado de una forma más extensa en el anexo A Presencia.

4.3.17 Orden Recomendado para Anular los Objetos del API de Cliente SIP

La eliminación de objetos que ya no se estén utilizando permite liberar al sistema de una carga innecesaria que puede afectar el rendimiento general. En J2ME esta situación no es tan crítica ya que el mismo sistema se encarga de realizar esta tarea liberando al desarrollador de esta responsabilidad. En C++ si es necesario implementar esta tarea, pero para evitar inutilizar objetos que aun se pueden estar utilizando, se recomienda seguir el siguiente orden de anulación de objetos:

- Primero la aplicación elimina el objeto CSIPTransactionBase.

- Luego elimina CSIPRefresh
- Luego elimina CSIPDialogAssocBase
- Luego elimina CSIPRegistrationBinding
- Luego elimina CSIPConnection
- Y por ultimo elimina CSIP

4.3.18 Creación de una Clase para la Gestión de SIP

La creación de una clase que lleve el control de la sesión y permita seguir los pasos para procesar solicitudes siguiendo las recomendaciones anteriormente descritas facilitaría enormemente el desarrollo de las aplicaciones Móviles P2P con SIP. Por eso a continuación se da un esquema simplificado de la estructura de una clase destinada para este fin.

```
Clase conexionSIP{
EstadoSesion = “libre”
EstadoObservación = falso
        NotificacionDeRespuestas{
                :
        }
        NotificacionDeSolicitudes{
                SI metodo(OPTIONS){}
                SI metodo(INVITE){}
                SI metodo(MESSAGE){}
                SI metodo(BYE){}
                SI metodo(CANCEL){}
        }
        //metodos
```

```
        EscucharSip(){ EstadoObservación = verdadero  }
        NoEscucharSip(){ EstadoObservación = falso  }
        EnviarINVITE(Dialogo dialog){  EstadoSesion = “invitando” }
        EnviarINVITE(Cadena URI_destino){  EstadoSesion = “invitando” }
        EnviarOPTIONS(Dialogo dialog){ EstadoSesion = “solicitando” }
        EnviarOPTIONS(Cadena URI_destino){ EstadoSesion = “solicitando” }
        EnviarCANCEL(){ EstadoSesion = “cancelando” }
        EnviarBYE(){ EstadoSesion = “terminando” }
        EnviarMESSAGE(Cadena URI_destino){}
        EnviarMESSAGE(Dialogo dialog){}
}
```

4.4 EJEMPLOS DE APLICACIONES MÓVILES P2P CON SIP

Las posibilidades de desarrollo de aplicaciones móviles P2P con SIP son muchas y están limitadas principalmente por las plataformas de desarrollo que por el mismo protocolo SIP, esto sucede mas en J2ME, el cual por motivos de seguridad no puede acceder a todas las opciones del sistema.

En una aplicación P2P, SIP actúa en la negociación de la sesión, dejando el transporte de los medios a otros protocolos. Una limitante para el desarrollo de una aplicación se presenta cuando no se posee el soporte necesario para determinados medios, por lo cual es necesario revisar el soporte de la plataforma elegida, para conocer si es posible realizar determinado desarrollo. A continuación se listan algunos tipos de aplicaciones que se consideran como desarrollos factibles de ser implementados como aplicaciones móviles P2P con SIP.

Juegos interactivos: En el desarrollo de juegos multijugador, se considera que no existe ningún problema, ya que ambas plataformas proporcionan un buen manejo de recursos

para juegos (gráficos, animación, teclado y sonidos), los cuales son la base para el desarrollo de los juegos donde compiten dos o mas jugadores.

Transferencia de video en tiempo real: las aplicaciones que requieren de la transferencia de video en tiempo real tendrán cierta limitante en J2ME, ya que en esta plataforma no se tiene aun el soporte necesario para realizarlo. Por el contrario; en Symbian si se puede crear aplicaciones que aprovechen este recurso.

Transferencia de audio en tiempo real: el soporte para realizar desarrollos con transferencia de audio en tiempo real existe en ambas plataformas.

Intercambio de archivos: Los protocolos que manejan las dos plataformas, permiten sin ningún problema realizar aplicaciones para el intercambio de archivos, entendiéndose como archivos las imágenes, documentos, Midis, mp3, videos, etc.

A continuación, se listan algunas aplicaciones móviles SIP P2P que se considera que pueden ser realizadas implementadas con el soporte actual.

Comunicaciones: mensajería instantánea, intercambio de fotos, tarjetas multimedia y otros sistemas P2P de distribución de mensajes.

Productividad: sincronización de direcciones/contactos y contenidos, e intercambio de documentos entre usuarios.

Juegos: para múltiples jugadores.

Entretenimiento: intercambio de tonos de timbre, música, video, historietas, protectores de pantalla, papeles tapices y humor.

Intercambio de aplicaciones: pudiéndose usar para comercializar o distribuir aplicaciones.

Medicina: atención médica remota y transferencia de documentación medica e imágenes al mismo tiempo.

Atención al cliente: atención por medio de audio y envío de contenidos o archivos al tiempo, así como asesoría en la diligenciación de documentos o manejo de y/o configuración de herramientas, mediante audio o video.

Industria: control y monitoreo de procesos industriales.

Herramientas colaborativas: varios usuarios trabajando en un mismo tema.

Bancarias: la asesoría se maneja vocalmente pero las claves de las cuentas se manejan mediante encriptación electrónica desde el terminal.

5. PROTOTIPO DE APLICACIÓN P2P CON SIP

5.1 INTRODUCCION

Para facilitar la comprensión del desarrollo de aplicaciones móviles P2P con SIP y aplicar las recomendaciones generadas en el cuarto capitulo “Recomendaciones para el desarrollo de aplicaciones SIP P2P”, se implementaron algunas aplicaciones a modo de ejemplo, de las cuales una será tomada para presentar su diseño en el presente capitulo. Se considera que esta implementación sirve de referencia para el desarrollo de aplicaciones móviles P2P con SIP. Esta aplicación incluye aspectos esenciales de implementación del protocolo de una manera sencilla, con el fin de que puedan ser entendidos con facilidad y que a partir de estos, se pueda construir aplicaciones más completas.

5.2 DESCRIPCIÓN GENERAL

Para cubrir aspectos como: iniciar una sesión SIP, observar solicitudes, aceptar invitaciones de sesión, cancelar una sesión en progreso, terminar una sesión, transferir archivos, enviar mensajes de texto, iniciar una conexión desde una sesión establecida, sesiones multimedia, solicitar capacidades, manejar presencia y manejar cadenas SDP, entre otros; la aplicación en desarrollo es un Chat P2P que hace uso del protocolo SIP. La aplicación se llamará ChatSIP y tendrá las siguientes características principales:

- Permitirá crear una sesión Chat con otro usuario registrado o no registrado.
- Permitirá el envío de mensajes de texto, imágenes y el inicio de un juego de Triqui.
- Las funciones de envío de imágenes y juego pueden ser utilizadas independientemente, o desde el Chat, reutilizando la sesión y manteniendo el Chat en segundo plano.

La Figura 4 muestra una visión general de la aplicación.

Figura 4. Aplicación ChatSip en el emulador.

5.3 PROCESO DE DESARROLLO

La implementación de esta aplicación, se basó en el Modelo para Construcción de Soluciones, desarrollado en el Modelo Integral para el Profesional en Ingeniería [58], tomando de este las fases o etapas necesarias para desarrollar una aplicación software.

Las etapas seleccionadas son las de análisis y diseño; la primera se centra en una investigación del problema, teniendo en cuenta este como tal y la descripción de las necesidades o requerimientos y la segunda se basa en la descripción detallada y de alto nivel de la solución lógica, estableciendo como satisfacer los requerimientos del sistema como tal. A continuación se describe el proceso de desarrollo.

5.3.1 Análisis

Esta etapa presenta un análisis detallado de los requerimientos y funcionalidades que el sistema a desarrollar debe presentar.

Para definir adecuadamente los requerimientos, se realizó un análisis de las funcionalidades generales de la aplicación ChatSIP para dispositivos móviles, un análisis de la especificación del protocolo SIP y las capacidades que debe tener el soporte SIP de la aplicación, para que pueda cumplir los requerimientos y seguir las recomendaciones del cuarto capitulo de este proyecto.

5.3.1.1 Funciones de la Aplicación móvil P2P con SIP ChatSIP

Estas funcionalidades se han establecido teniendo como base las capacidades que debe presentar la aplicación al usuario que interactúa con la ella.

- Crear registro
- Eliminar registro

- Observar Peticiones
- Iniciar sesión
- Cancelar sesión
- Aceptar sesión
- Terminar sesión
- Enviar mensaje
- Enviar imagen
- Jugar triqui
- Revisar mensajes
- Adicionar contacto
- Eliminar contactos
- Solicitar capacidades
- Revisar presencia

5.3.1.2 Ubicación en la Arquitectura de Red

En el Anexo C, titulado Señalización basada en el protocolo SIP en redes de 3G, se puede ver que una implementación P2P con el protocolo SIP, presenta dos posibles formas de conexión para una aplicación en general. La primera se realiza totalmente desde un UA a otro UA sin pasar por los elementos de la red SIP (conexión P2P pura); para llevar a cabo este tipo de conexión, solo se requieren de la red IP y las aplicaciones terminales, las cuales realizan todo el control de la sesión basándose en la dirección SIP privada del otro usuario. La figura 5 muestra este tipo de arquitectura.

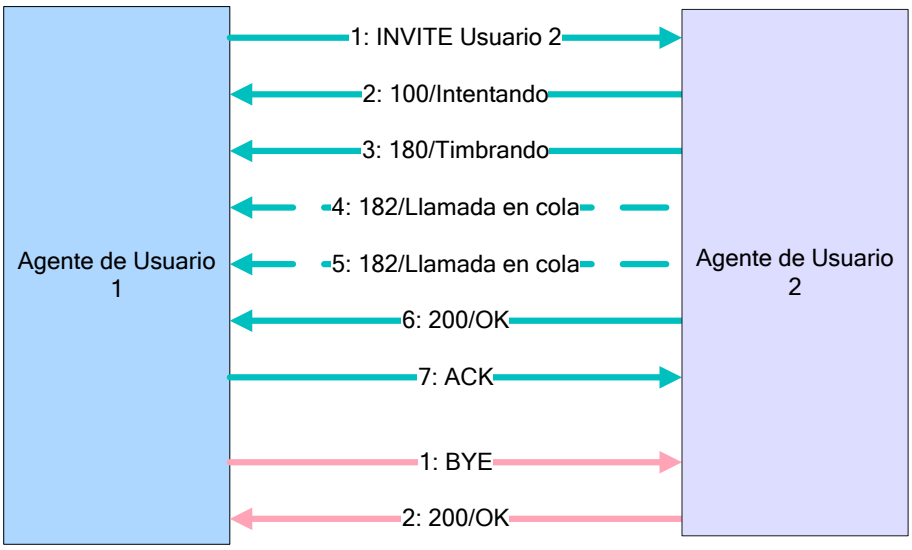

Figura 5. Sesión P2P pura entre dos UAs.

La segunda posibilidad de establecimiento de una conexión P2P que brinda SIP, se realiza a través de los servidores de la red, los que ayudan a localizar a un usuario basándose en la dirección SIP publica del usuario buscado; esta es la forma de conexión que principalmente pretende explotar SIP, ya que permite mayores posibilidades de aplicación y facilidad de manejo para el usuario (conexión P2P hibrida). Este tipo de conexión se muestra en la figura 6, en la cual se puede apreciar que el establecimiento de la sesión, se realiza a través de los servidores de la red SIP; pero el intercambio de contenidos de la sesión si es punto a punto, debido a que una vez establecida la sesión mediante las direcciones SIP publicas, los UAs toman de las cabeceras de las primeras solicitudes las direcciones SIP privadas y continúan la sesión enviando directamente los mensajes y contenidos al destino sin pasar por los servidores.

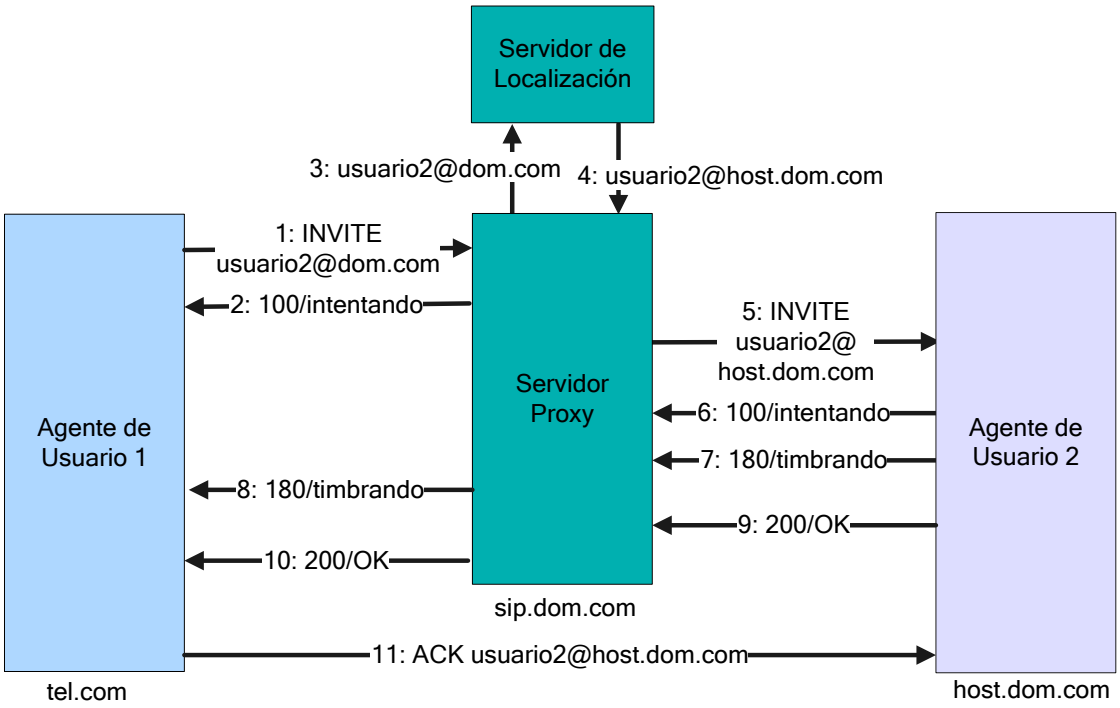

Figura 6. Sesión P2P a través de un servidor Proxy.

A nivel de operación, las dos sesiones tienen sus diferencias, pero a nivel de implementación los cambios son mínimos, ya que en ambas conexiones el protocolo proporciona el control de la sesión desde los puntos extremos. Debido a esto la aplicación a implementar contará con las dos posibilidades de conexión descritas.

5.3.1.3 Diagrama de Casos de Uso

La Figura 7 muestra el diagrama de casos de uso para las funciones generales del sistema, desde la perspectiva del desarrollador.

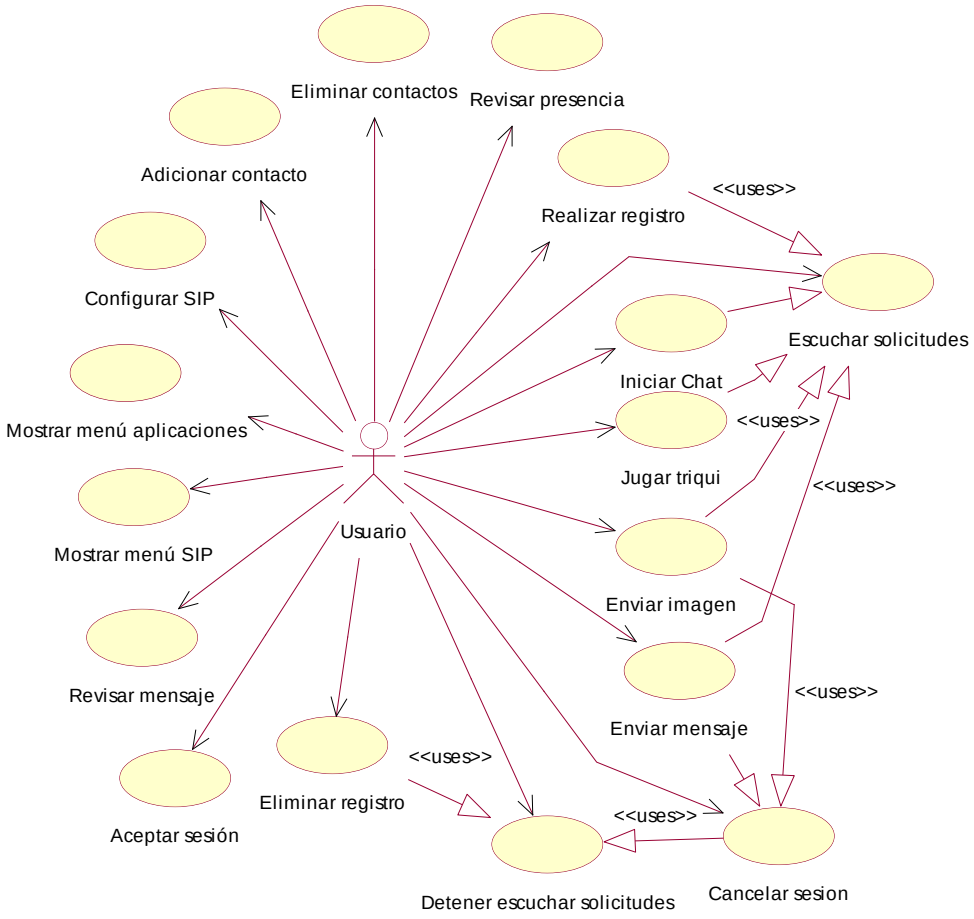

Figura 7. Casos de uso del sistema.

5.3.1.4 Descripción de algunos Casos de Uso

Caso de uso No. 1: Realizar registro

Actores: Usuario (Iniciador)

Propósito: Realizar el proceso de registro ante un servidor Proxy SIP.

Resumen: Esta opción permite al usuario crear un registro en un servidor Proxy SIP, registro que le permite ser contactado por otros usuarios de la red SIP, mediante su

dirección pública o de usuario. El registro se realiza con los parámetros configurados en la sección “Configurar SIP”. La figura 8 muestra la opción Registrarme.

Flujo principal:

- Este caso de uso inicia cuando el usuario selecciona la opción “Registrarme” de la lista de opciones en la sección de registro.
- El sistema presenta el mensaje temporal “procesando registro”.
- Cuando el registro se completa el sistema presenta un mensaje de registro exitoso.

Figura 8. Opción “Registrarme” de la sección Registro.

Caso de uso No. 2: Iniciar Chat

Actores: Usuario (Iniciador)

Propósito: Iniciar una sesión de Chat con otro usuario.

Resumen: Mediante la invitación a otro usuario se inicia una sesión de Chat, desde la cual se pueden compartir mensajes de texto. La figura 9 muestra la interfaz principal del Chat.

Flujo principal:

- Este caso de uso inicia cuando el usuario selecciona la opción “Abrir contactos” de la lista de opciones en la sección Chat SIP, que se encuentra en el submenú “Aplicaciones SIP”.
- La aplicación presenta las URIs de la lista e contactos de donde se elige una y se inicia la sesión de Chat mediante la opción, “Iniciar Chat”.
- El sistema realiza la negociación SIP con el invitado y si este acepta la invitación, se inicia el Chat, desplegando la interfaz de chat.
- Una vez en el Chat, los usuarios pueden digitar sus mensajes y enviarlos mediante la opción “Enviar mensaje” de la lista de opciones.
- El caso de uso termina cuando alguno de los dos participantes utiliza la opción “Cancelar”, durante el establecimiento de la sesión o después de establecida. Una vez cancelada la aplicación, se informa de la cancelación de la sesión mediante un mensaje en pantalla.

Figura 9. Interfaz principal del Chat SIP.

Caso de uso No. 3: Jugar triqui

Actores: Usuario (Iniciador)

Propósito: Iniciar una sesión de juego con otro usuario.

Resumen: Un usuario puede iniciar una sesión de juego Triqui con otro usuario; en el juego, cada participante es informado de su turno para colocar una ficha, hasta que alguno de los dos complete los tres en línea o se acaben los espacios disponibles. La figura 10 muestra las interfaces de dos usuarios en una sesión de Triqui.

Flujo principal:

- Este caso de uso inicia cuando el usuario selecciona la opción “Invitar contacto” de la lista de opciones en la sección “Juegos”.
- El sistema solicita el ingreso de la URI del contacto a invitar y la elección de la opción “Invitar” de la lista de opciones.
- El sistema realiza la negociación SIP con el invitado y si este acepta la invitación, se inicia el juego.
- El caso de uso termina, cuando alguno de los dos participantes utiliza la opción “Cancelar”, durante el establecimiento de la sesión, o después de establecida, esto desactiva completamente el juego en ambos terminales.

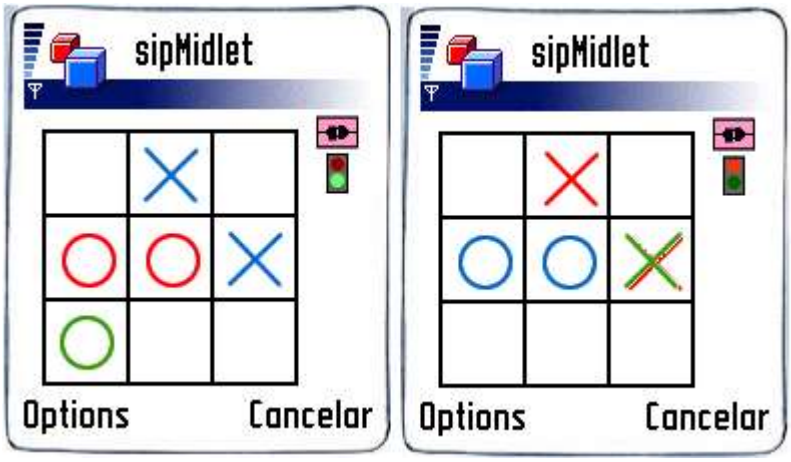

Figura 10. Interfaces de dos usuarios en una sesión de juego Trique.

Caso de uso No. 4: Enviar imagen

Actores: Usuario (Iniciador)

Propósito: Enviar una imagen a un usuario.

Resumen: Un usuario puede enviar una imagen desde archivo a un destino mediante el inicio de una sesión para transferencia de imágenes. La figura 11 muestra la interfaz para el envío de imágenes.

Flujo principal:

- Este caso de uso inicia cuando el usuario selecciona la opción “Imagen de archivo” de la lista de opciones en la sección “Transferencia”.
- El sistema solicita el ingreso de la URI del destino y la imagen a enviar.
- Luego con la opción “Enviar imagen” de la lista de opciones se efectúa la invitación.
- El sistema realiza la negociación SIP con el invitado y si este acepta la invitación, se inicia la transferencia.
- El caso de uso termina cuando alguno de los dos participantes utiliza la opción “Cancelar”, durante el establecimiento de la sesión, o cuando la imagen es transferida completamente a su destino.

Figura 11. Interfaz para el envío de imágenes con SIP.

Caso de uso No. 5: Enviar mensaje

Actores: Usuario (Iniciador)

Propósito: Enviar un mensaje de texto a un usuario.

Resumen: El usuario puede enviar un mensaje de texto que irá a un buzón de mensajes SIP del destino. La figura 12 muestra la interfaz para el envío de mensajes de texto.

Flujo principal:

- Este caso de uso inicia cuando el usuario selecciona la opción “Mensaje nuevo” de la lista de opciones en la sección “Mensajes”.
- El sistema solicita el ingreso del mensaje de texto, el asunto y URI del destino. Luego el usuario elige la opción “Enviar” de la lista de opciones.
- El sistema realiza la negociación SIP con el invitado y si este puede recibir el mensaje acepta.
- El caso de uso termina cuando el usuario que envía utiliza la opción “Cancelar”, o cuando el mensaje ha sido recibido en el destino.

Figura 12. Interfaz para el envío de mensajes de texto con SIP.

5.3.2 Diseño

Después de haber establecido las diferentes funcionalidades o requerimientos del sistema, se procede a la identificación de los casos de uso que permite tener una mejor comprensión de las funcionalidades del sistema a desarrollar, creando un diagrama de casos de uso en el cual el único actor es el usuario de la aplicación que interactúa con ella.

5.4 ARQUITECTURA DEL SISTEMA

Se pretende realizar un diseño adecuado que desacople la vista del modelo, con la finalidad de mejorar la reusabilidad. Por lo cual, para el desarrollo de las diferentes funcionalidades del sistema se ha decidido usar una arquitectura basada en el paradigma vista – control – modelo, este patrón de arquitectura de software separa los datos de una aplicación, la interfaz de usuario, y la lógica de control en tres componentes distintos. De esta forma las modificaciones en las vistas afectan en menor medida en la lógica del negocio o de datos. Este tipo de arquitectura asigna las funcionalidades de un sistema de la siguiente manera:

Modelo: Gestiona el comportamiento y los datos de la aplicación, responde a las peticiones que realizan las vistas sobre su estado y permite su actualización, normalmente desde el controlador.

Control: Interpreta las acciones del usuario, accediendo a las operaciones de negocio de la aplicación y modificando a partir de sus resultados, el estado del modelo y la navegación entre vistas.

Vista: Muestra el estado al usuario de la aplicación, redirigiendo las acciones que realiza sobre el interfaz al controlador.

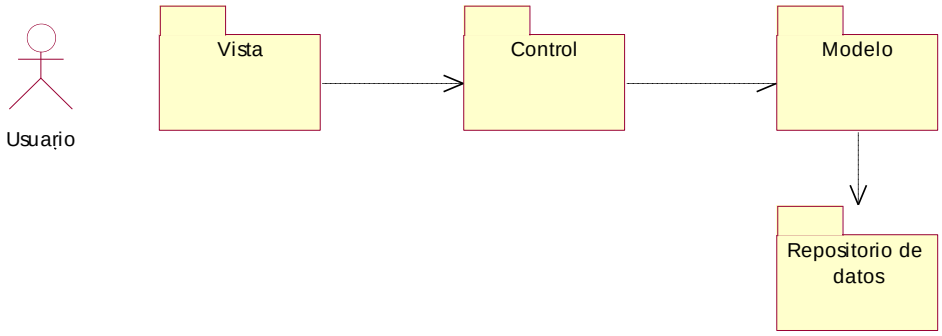

Figura 13. Arquitectura del sistema.

La aplicación ChatSIP, presenta 3 módulos principales donde se encuentran las clases que permiten realizar las diferentes funcionalidades de la aplicación; el primero de estos corresponde a la Vista, donde se presentan las interfaces graficas del sistema, con las cuales la aplicación despliega, solicita información o notifica sucesos al usuario. Adicionalmente recibe los eventos generados por el usuario y los pasa al modulo de control. Los módulos principales de la aplicación se muestran en la figura 13.

El segundo de estos componentes corresponde al Control, en este se encuentran las clases encargadas de recibir y procesar los eventos de entrada que genera el usuario de la aplicación. Algunos de estos procesamientos generan llamados al modelo.

El tercer modulo corresponde al Modelo que es donde se encuentra la funcionalidad del sistema, este contiene las clases necesarias que se encargan de recibir las peticiones del modulo de Control y entregar la información respectiva previamente almacenada.

5.5 DIAGRAMA DE CLASES

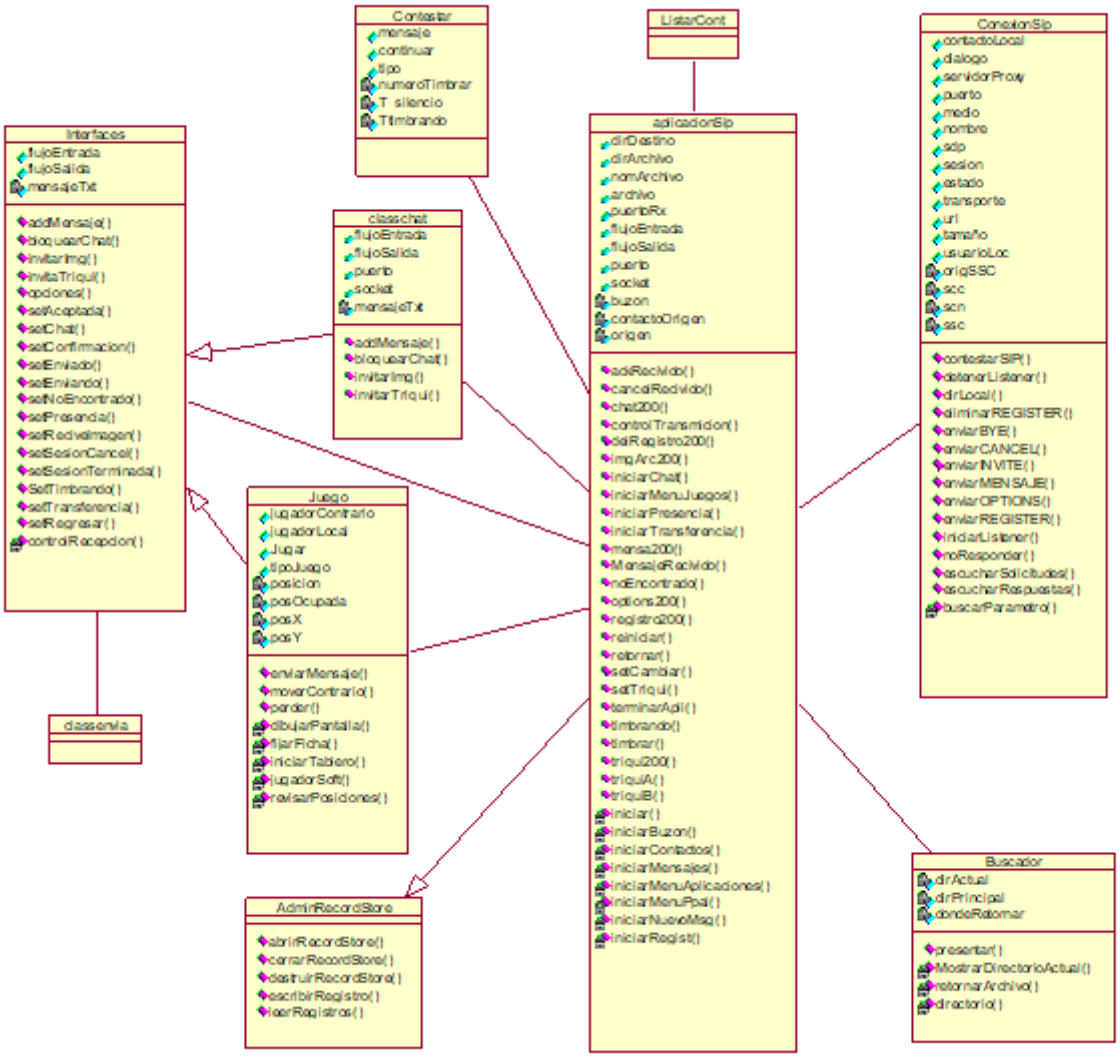

Figura 14. Diagrama de clases de la aplicación ChatSIP.

La figura 14 muestra el diagrama de clases del sistema y a continuación se presenta una breve descripción de cada una de las clases presentada.

AdminRecordStore: clase que sirve para leer y escribir datos en un RecordStore del dispositivo. La aplicación utiliza esta clase para manejar las URIs de los contactos.

AplicacionSip: esta clase realiza el control de navegación y ejecución de la aplicación. Es la clase principal y es lanzada inicialmente por la Midlet, determina las acciones a realizar, la interfaz a presentarse e instancia a las demás clases cuando se necesitan.

Buscador: clase que permite acceder a los archivos del dispositivo y realizar una búsqueda y selección de uno de estos. Se utiliza para buscar y seleccionar una imagen que posteriormente será visualizada y enviada a un destino.

ClassChat: esta clase maneja la interfaz correspondiente al Chat. Se utiliza para desplegar la interfaz principal del Chat de forma que la interfaz principal de la aplicación se mantenga en segundo plano.

ClassEnvia: esta clase maneja una interfaz desde donde se puede realizar invitaciones sobre una sesión de Chat ya establecida. Se utiliza para enviar imágenes o invitar a jugar desde una sesión de Chat dejando la interfaz de Chat en segundo plano.

ConexionSip: contiene todo lo necesario para iniciar mantener y terminar una sesión SIP, además de los métodos de observación. Esta clase se utiliza para establecer comunicaciones SIP con otros usuarios.

Contestar: esta clase genera una interfaz de “Timbrado” cuando se recibe una invitación SIP. Se utiliza para presenta la información correspondiente a la sesión y deja en segundo plano a la interfaz principal durante el tiempo que dura la solicitud de invitación.

Interfaces: contiene una colección de interfaces que presentan los diferentes mensajes informativos de la aplicación. Se utiliza para informar al usuario de la ocurrencia de algún suceso.

Juego: esta clase contiene todo lo necesario para generar y gestionar el juego Trique. Es lanzada cuando se solicita una sesión de juego.

ListarCont: esta clase presenta el listado de contactos en una interfaz, desde donde se puede seleccionar uno para su utilización desde un campo de texto por ejemplo.

El contenido de clases de una aplicación móvil SIP P2P, varía casi completamente de una aplicación a otra, manteniéndose siempre igual una clase que se encargue de las

comunicaciones mediante este protocolo. Es por eso que desde el punto de vista del desarrollo de este prototipo de aplicación móvil SIP P2P, se considera que la clase ConexionSip es la mas importante, ya que maneja toda la funcionalidad del protocolo y donde se aplican las recomendaciones realizadas en el cuarto capitulo “Recomendaciones para el desarrollo de aplicaciones Móviles SIP P2P”.

Por esta razón, a continuación el desarrollo se centrará en esta clase, que además de servir para el prototipo en desarrollo, puede ser utilizada como base para otras aplicaciones que hagan uso del protocolo SIP.

5.6 DIAGRAMAS DE SECUENCIA

Para mostrar la interacción dinámica de las clases de la aplicación, se presentan a continuación los diagramas de secuencia de los principales casos de uso que involucran la clase conexionSip.

El Anexo E Diseño de una aplicación P2P con SIP, contiene todos los diagramas de secuencia al igual que los diagramas de clases de esta aplicación.

La figura 15 muestra el diagrama de secuencia para el caso de uso Realizar registro; el cual, una vez iniciado por el usuario, provoca que *conexionSip* inicie la negociación del registro e *Interfaces* actualice la interfaz con un mensaje provisional; hasta que se obtenga una respuesta definitiva, que generara que *Interfaces* realice una actualización final de la interfaz con la información recibida.

.

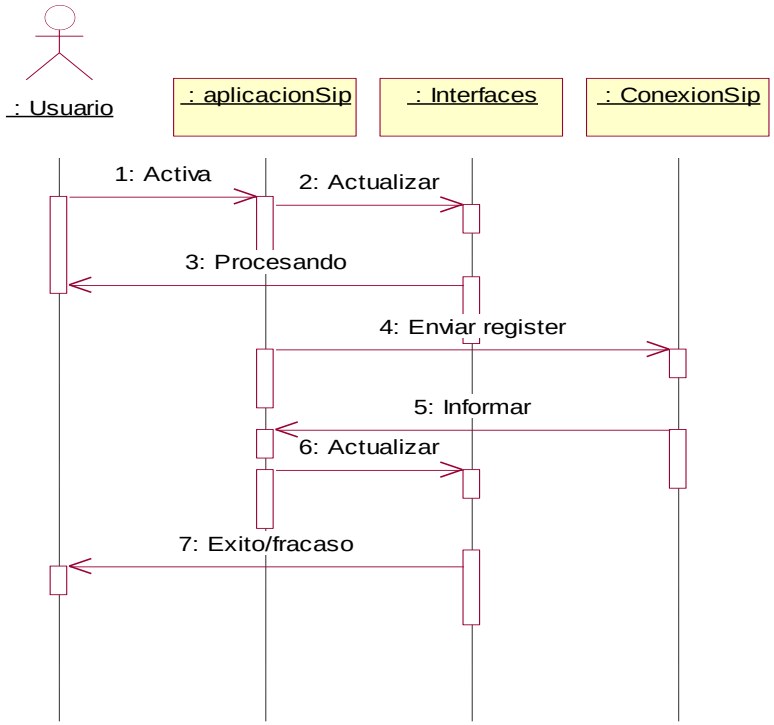

Figura 15. Diagrama de secuencia para el caso de uso Realizar registro.

La figura 16 muestra el diagrama de secuencia para el caso de uso Iniciar Chat; el cual, una vez iniciado por el usuario, *aplicacionSip* presenta una interfaz de ingreso o selección de contacto de destino; para lo cual *ListarCont* genera una lista de URIs de contactos, extrayéndolos del sistema de almacenamiento de la aplicación con la ayuda de *AdminRecordStore*; luego cuando el usuario elija un destino, *Interfaces* actualiza la interfaz con un mensaje temporal y *conexionSip* envía la solicitud de invitación; cuando el destino es encontrado *Interfaces* actualiza nuevamente la interfaz avisando de este hecho al usuario y se espera hasta que el usuario acepte la invitación, con lo que finalmente *Classchat* despliega y controla el Chat.

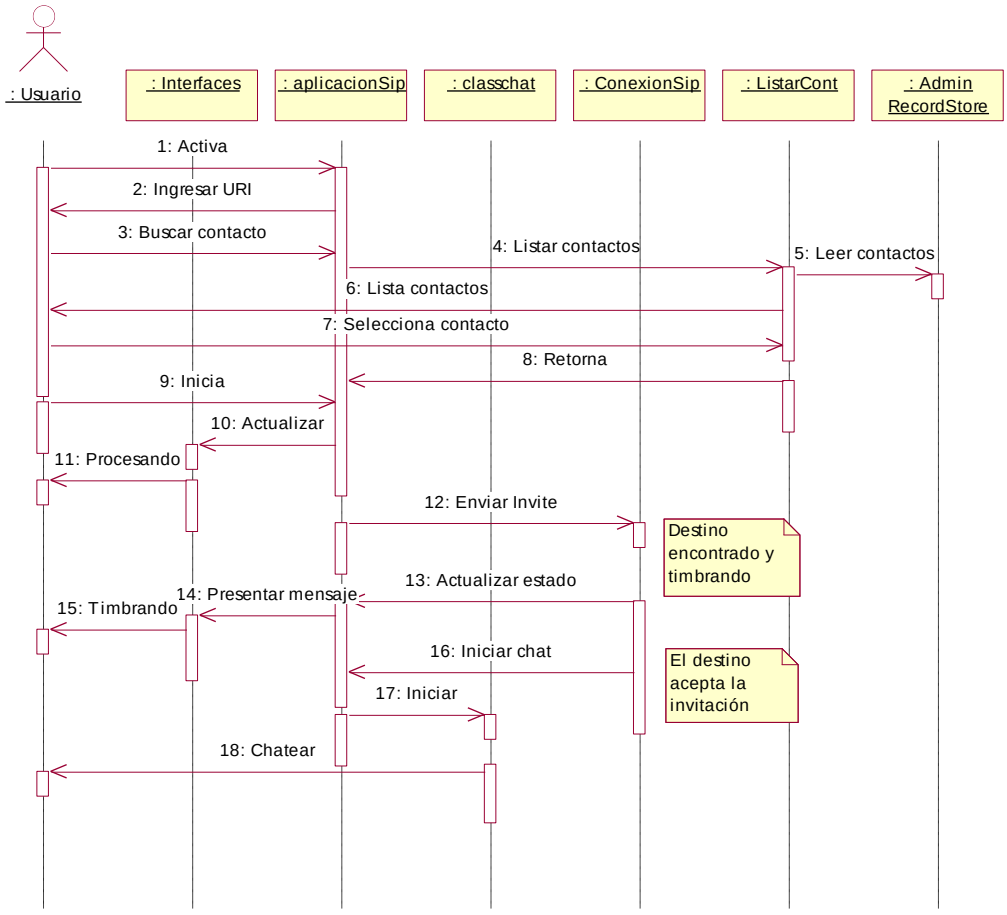

Figura 16. Diagrama de secuencia para el caso de uso Iniciar riq.

La figura 17 muestra el diagrama de secuencia para el caso de uso Jugar rique; el cual, una vez iniciado por el usuario, *aplicacionSip* presenta una interfaz de ingreso o selección de contacto de destino; para lo cual *ListarCont* genera una lista de URIs de contactos, extrayéndolos del sistema de almacenamiento aplicación con la ayuda de *AdminRecordStore*; luego cuando el usuario elija un destino y confirme la operación, *Interfaces* actualiza la interfaz con un mensaje temporal y *conexionSip* envía la solicitud de invitación; cuando el destino es encontrado *Interfaces* actualiza nuevamente la interfaz avisando de este hecho al usuario y se espera hasta que el usuario acepte la invitación, con lo que finalmente *Juego* actualiza la interfaz y controla el juego.

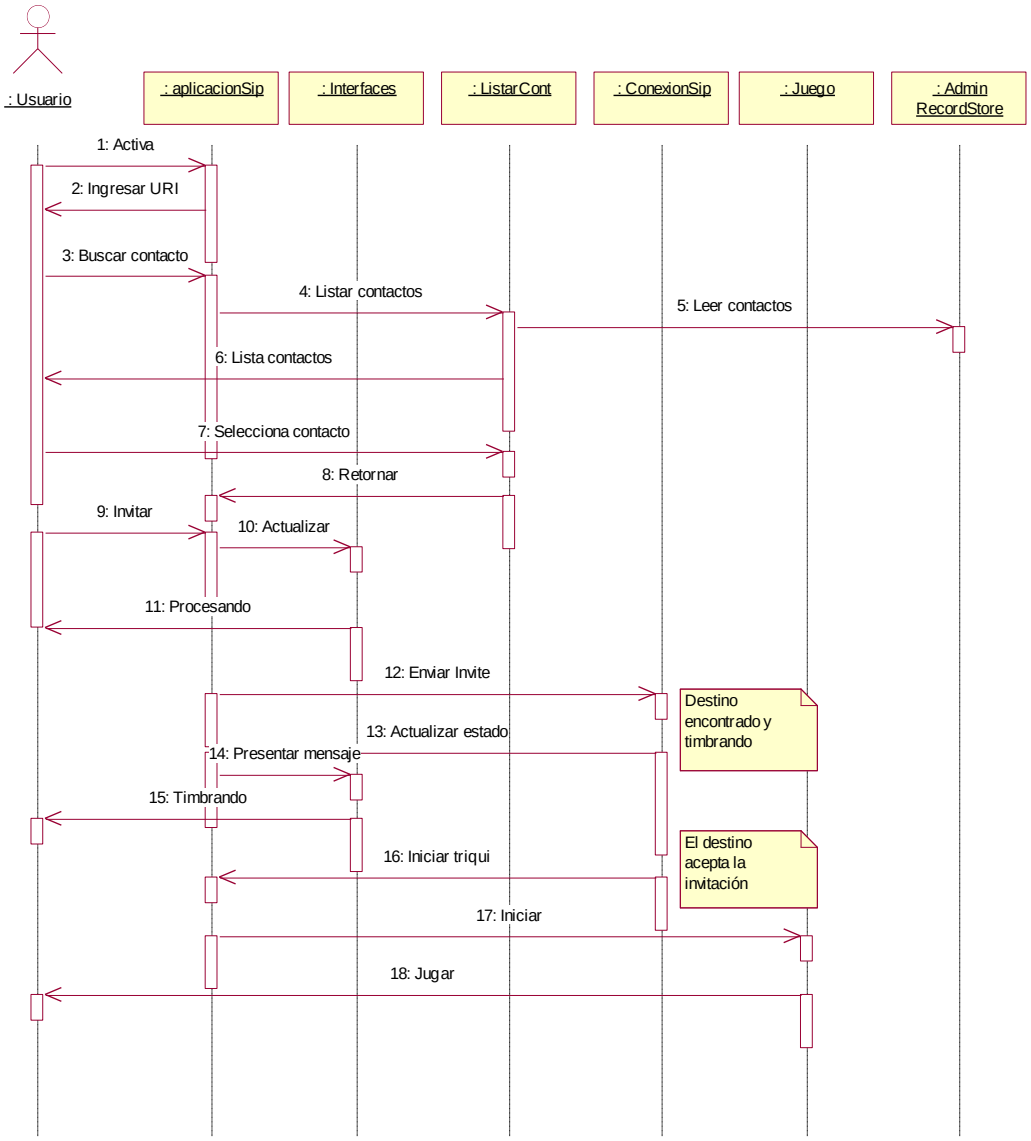

Figura 17. Diagrama de secuencia para el caso de uso Jugar Trique.

La figura 18 muestra el diagrama de secuencia para el caso de uso Enviar imagen; el cual, una vez iniciado por el usuario, *aplicacionSip* presentará una interfaz de ingreso o selección de contacto de destino y de selección de la imagen; para lo cual *ListCont* genera una lista de URIs de contactos, extrayéndolos del sistema de almacenamiento de la aplicación y *Buscador* crea un entorno de navegación entre los archivos del

dispositivo; luego cuando el usuario elija un destino y una imagen, *Interfaces* actualiza la interfaz con un mensaje temporal y *conexionSip* envía la solicitud de invitación; cuando el destino es encontrado *Interfaces* actualiza nuevamente la interfaz avisando del hecho al usuario y espera un tiempo hasta que el usuario acepte la invitación; una vez aceptada, *aplicacionSip* inicia la transferencia de la imagen e *Interfaces* actualiza el despliegue con el estado de la transferencia. Cuando la transferencia se completa *conexionSip* finaliza la sesión con un mensaje de finalización.

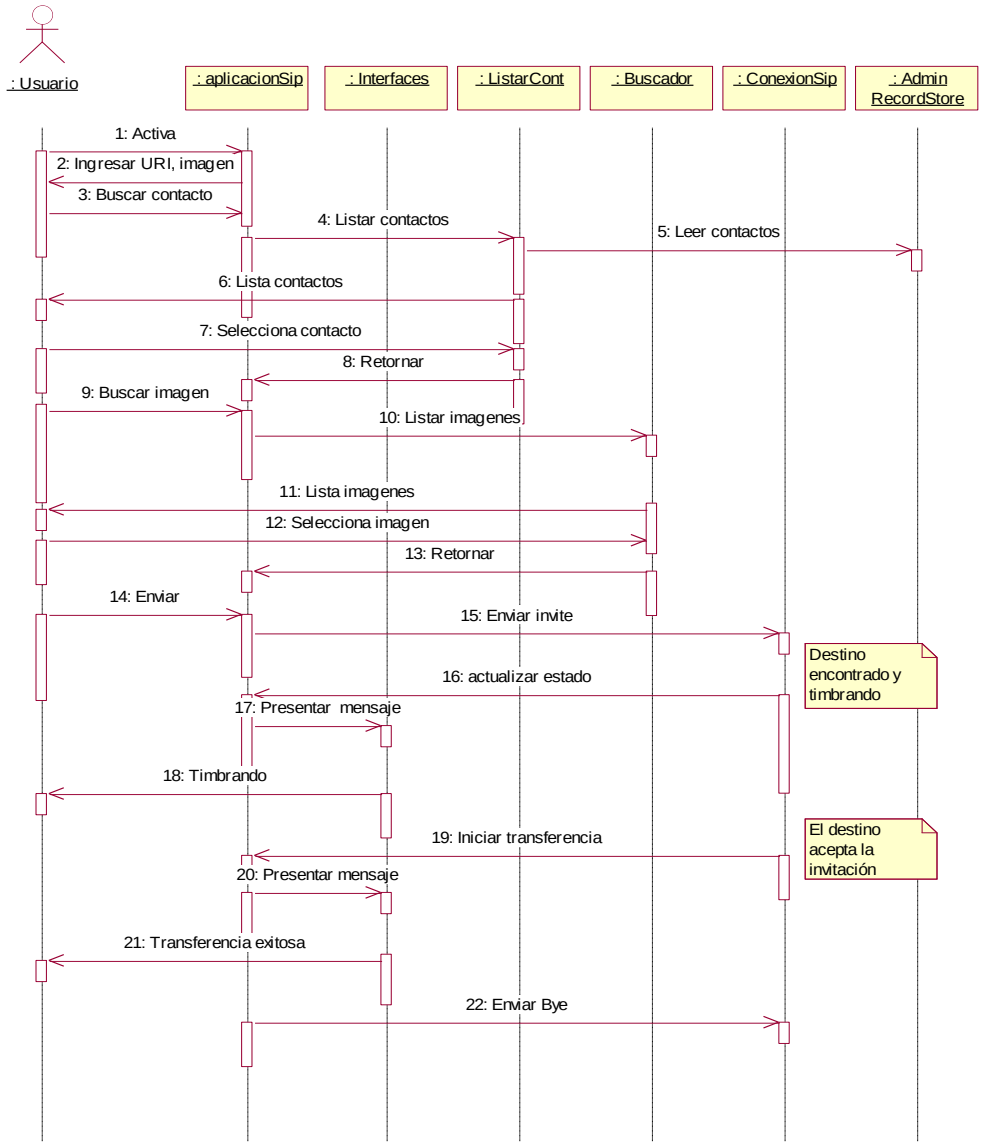

Figura 18. Diagrama de secuencia para el caso de uso Enviar imagen.

La figura 19 muestra el diagrama de secuencia para el caso de uso Enviar mensaje; el cual, una vez iniciado por el usuario, *aplicacionSip* presentará una interfaz de ingreso del asunto, mensaje y selección de contacto de destino; para lo cual *ListCont* genera una lista de URIs de contactos, extrayéndolos del sistema de almacenamiento de la aplicación; luego de confirmar la operación, *conexionSip* envía el mensaje e *Interfaces* presenta un mensaje con resultado de la operación realizada.

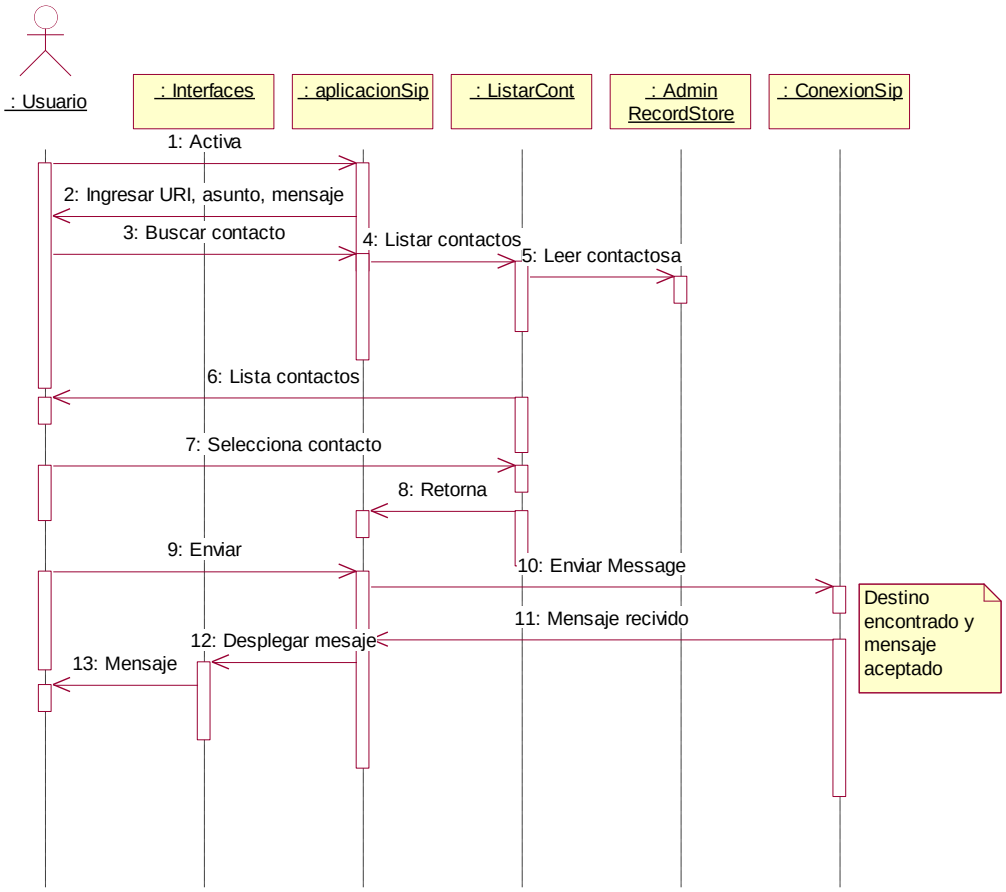

Figura 19. Diagrama de secuencia para el caso de uso Enviar mensaje.

5.7 IMPLEMENTACIÓN EN J2ME

La figura 20 presenta una descripción detallada de la clase conexionSip, implementada en J2ME. La clase se llama conexionSipJ2me y forma parte de los prototipos desarrollados en J2ME ChatSIP, Chat, rique y EnviaImagen, que se encuentran disponibles en el CD adjunto al presente trabajo.

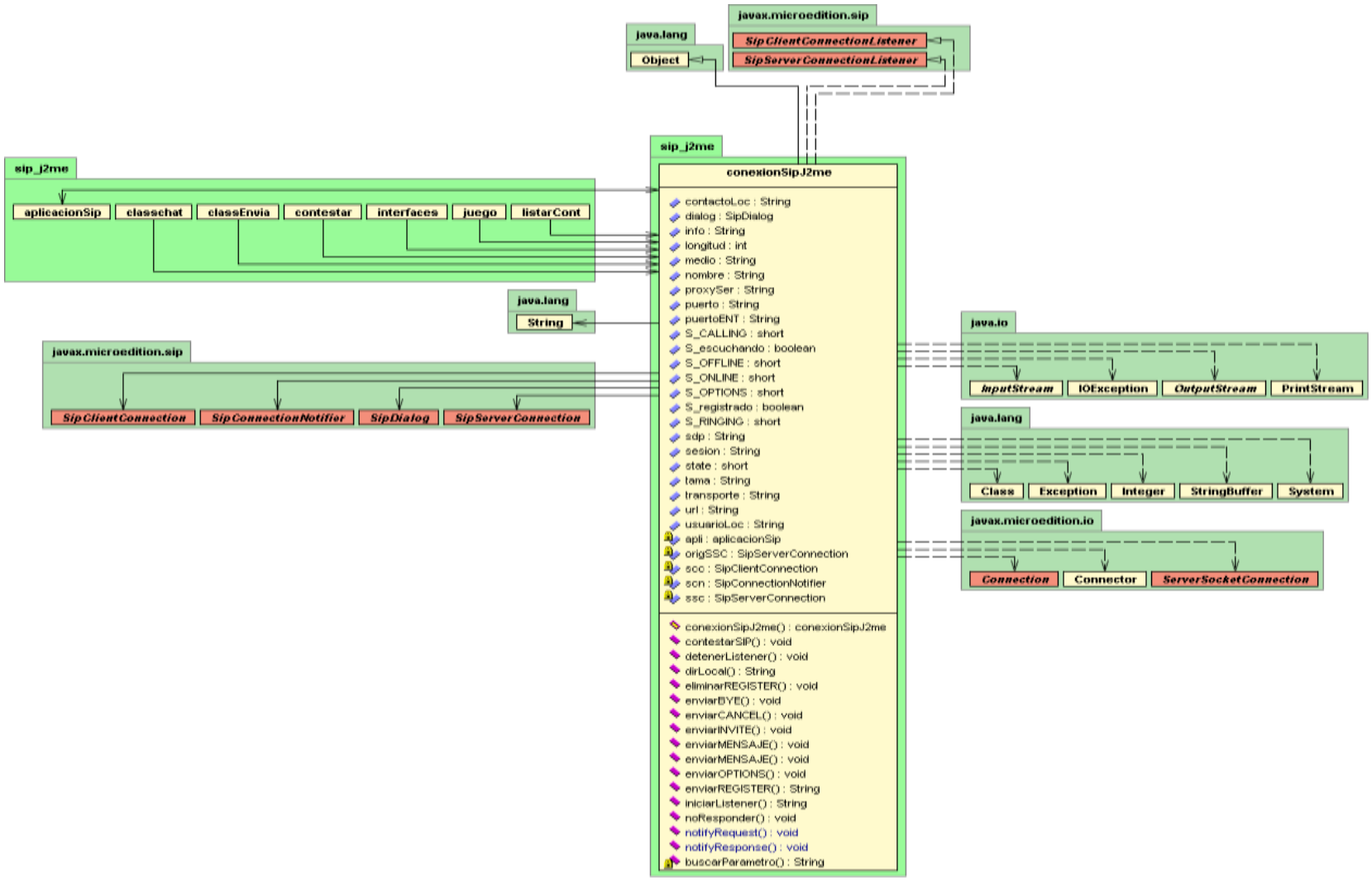

Figura 20. Clase conexiónSip implementada en J2ME.

Los métodos presentes en la clase conexionSipJ2me, son utilizados de la siguiente manera por los prototipos:

contestarSIP: método utilizado para aceptar una invitación recibida.

detenerListener: permite detener el servidor de solicitudes SIP entrantes.

dirLocal: método que retorna la dirección local.

enviarINVITE: permite el envío de una solicitud INVITE a un destino.

enviarCANCEL: envía una solicitud de cancel sobre una conexión de cliente abierta.

enviarBYE: envía una solicitud BYE sobre una sesión establecida.

enviarMENSAJE: método sobrecargado que permite el envío de una solicitud de MESSAJE a partir de una URI de destino o sobre un dialogo establecido.

enviarREGISTER: permite el envío de una solicitud REGISTER.

enviarOPTIONS: permite el envío de una solicitud OPTIONS utilizada para conocer el estado y el soporte del destino.

iniciarListener: permite iniciar el servidor de solicitudes SIP entrantes.

noResponder: método que sigue la recomendación 3.3.15 Construcción una solicitud CANCEL, del tercer capitulo "Recomendaciones para el desarrollo de aplicaciones SIP P2P" utilizado para cancelar y terminar una sesión en progreso.

notifyResponse: método de observación que debe ser obligatoriamente implementado para poder recibir respuesta a las solicitudes realizadas.

notifyRequest: método de observación que debe ser obligatoriamente implementado para poder recibir solicitudes SIP.

buscarParametro: método utilizado para extraer parámetros de las cadenas SDP

Los atributos presentes en la clase conexionSipJ2me, son utilizados de la siguiente manera por los prototipos:

contactoLoc: mantiene la dirección SIP local (dirección privada).

Dialog: mantiene el dialogo de la sesión.

Info: guarda el contenido del "titulo del medio", de la cadena SDP recibida.

Longitud: guarda el tamaño de una imagen a ser enviada.

Medio: guarda el nombre del tipo de medio especificado en la cadena SDP.

proxySer: guarda la dirección del servidor Proxy configurado.

Puerto: guarda el numero de puerto en el cual esta disponible la imagen a transmitir.

puertoENT: guarda el numero de puerto configurado para escuchar solicitudes SIP.

S_CALLING, S_OFFLINE, S_ONLINE, S_OPTIONS y S_RINGING: de acuerdo con la recomendación 3.3.2 Estado de la Sesión, del tercer capitulo “Recomendaciones para el desarrollo de aplicaciones SIP P2P”; definen los estados que puede tener la aplicación en un momento determinado de la comunicación.

S_escuchando: guarda el estado (activo/inactivo) del servidor de solicitudes entrantes.

S_registrado: guarda el estado de registro (registrado/no registrado) de la aplicación.

Sdp: guarda la cadena SDP de una invitación recibida.

Sesion: guarda el nombre de la sesión de la cadena SDP recibida en una invitación.

State: guarda el estado actual de la aplicación.

Tama: guarda el tamaño en bytes de una imagen a recibirse.

Transporte: guarda el tipo de medio especificado en la cadena SDP, que será utilizado para enviar una imagen.

url: contiene la dirección completa a la cual se debe conectar para iniciar la descarga de una imagen.

usuarioLoc: guarda la dirección publica del usuario.

5.8 CÓDIGO IMPLEMENTADO

A continuación se presenta el código en J2ME de la clase conexiónSip.

```
package sip_j2me;

import java.io.*;
import javax.microedition.io.*;
import javax.microedition.sip.*;


public class conexionSip implements SipClientConnectionListener, SipServerConnectionListener {
  public String info,tama,medio,sesion,puerto,transporte,url;
```

```
  private aplicacionSip apli;
  public SipDialog dialog;

  public final short S_OFFLINE = 0;
  public final short S_CALLING = 1;
  public final short S_RINGING = 2;
  public final short S_ONLINE = 3;
  public final short S_OPTIONS = 4;
  public short state = S_OFFLINE;
  public int longitud = 0;
  public boolean S_escuchando = false;
  public boolean S_registrado = false;

  private SipClientConnection scc = null;
  private SipConnectionNotifier scn;
  private SipServerConnection ssc = null;
  private SipServerConnection origSSC = null;
  //configuracion
  public String usuarioLoc = "sip:yo@10.35.1.20";
  public String contactoLoc ;
  public String proxySer = "sip:10.35.1.20";
  public String puertoENT = "5070";    //puerto para escuchar cuando se espera invitaciones
  //public String origen = "";
  public String sdp = "v=0\no=sippy 2890844730 2890844732 IN IP4
host.example.com\ns=example code\nc=IN IP4 host.example.com\nt=0 0\nm=message 54344
SIP/TCP\na=user:sippy";

  public conexionSipJ2me(aplicacionSip apli) {
    this.apli = apli;
    //contactoLoc = "sip:user@" + dirLocal();
    contactoLoc = "sip:" + dirLocal();   }

  /**
   * Metodo lanzado cuando se recibe una respuesta
   * @param scc SipClientConnection: cliente de conexión sobre el cual se recibió la respuesta
   */
  public void notifyResponse(SipClientConnection scc) {
       int statusCode = 0;
       try {
          scc.receive(0); // <i><b>fetch resent response</b></i>
          statusCode = scc.getStatusCode();
          if(statusCode < 200) {
            if (statusCode == 180)
               state = S_RINGING;
            dialog = scc.getDialog();
            apli.timbrando();
          }
          else if (statusCode == 200) {
            if (apli.aplicacion == apli.T_CHAT) {
              String contentType = scc.getHeader("Content-Type");
              String contentLength = scc.getHeader("Content-Length");
              String contactoDestino = scc.getHeader("Contact");
              int length = Integer.parseInt(contentLength);
              if (contentType.equals("application/sdp")) {
                //aqui se realiza el manejo SDP
              }
```

```
  dialog = scc.getDialog(); // <i><b>save dialog info</b></i>
  scc.initAck(); // <i><b>initialize and send ACK</b></i>
  scc.send();
  apli.chat200(contactoDestino);
  scc.close();
  state = S_ONLINE;
}
else if (apli.aplicacion == apli.T_ENVIMG) {
  String contentType = scc.getHeader("Content-Type");
  String contentLength = scc.getHeader("Content-Length");
  String contactoDestino = scc.getHeader("Contact");
  int length = Integer.parseInt(contentLength);
  if (contentType.equals("application/sdp")) {
    //aqui se realiza el manejo SDP
  }
  dialog = scc.getDialog(); // <i><b>save dialog info</b></i>
  scc.initAck(); // <i><b>initialize and send ACK</b></i>
  scc.send();
  apli.imgArc200(contactoDestino);
  scc.close();
  state = S_ONLINE;
}
else if (apli.aplicacion == apli.T_TRIQUI) {
  String contentType = scc.getHeader("Content-Type");
  String contentLength = scc.getHeader("Content-Length");
  String contactoDestino = scc.getHeader("Contact");
  int length = Integer.parseInt(contentLength);
  if (contentType.equals("application/sdp")) {
    //aqui se realiza el manejo SDP
  }
  dialog = scc.getDialog(); // <i><b>save dialog info</b></i>
  scc.initAck(); // <i><b>initialize and send ACK</b></i>
  scc.send();

  apli.triqui200(contactoDestino);
  scc.close();
  state = S_ONLINE;
}
else if (apli.aplicacion == apli.T_ENVMEN) {
  apli.mensa200();
  scc.close();
  state = S_OFFLINE;
}
else if (apli.aplicacion == apli.T_ENVREG) {
    String hdrs = scc.getHeader("From");
    apli.registro200(hdrs);
    state = S_ONLINE;
}
else if (apli.aplicacion == apli.T_ENVUNREG) {
    S_registrado = false;
    state = S_ONLINE;
    detenerListener();
    apli.delRegistro200();
}
else if (state == S_OPTIONS) {
    state = S_OFFLINE;
```

```
            detenerListener();
            try{
              String permitido = scc.getHeader("Allow");
              String contacto = scc.getHeader("Contact");
              apli.iniciarPresencia(contacto, permitido);
            }
            catch (Exception e) {
            }
          }


        }
        else if(statusCode >= 300) {
           if(state!=S_OFFLINE){
             state = S_OFFLINE;
             apli.cancelRecivido();
           }
           scc.close();
        }
    } catch(IOException ioe) {
        // <i><b>handle e.g. transaction timeout here</b></i>
        apli.Noencontrado();
        state = S_OFFLINE;
    }
}

/**
 * Metodo lanzado cuando se recive una petición
 * @param sn SipConnectionNotifier: notificador sobre el cual se recivio la petición
 * @throws Desde aqui se llaman a algunos metodos de la interfaz "apli" los
 * cuales realizan el procesamiento necesario ante la petición recivida. Esos metodos
 * no forman parte de esta clase ya que su procesamiento depende de la aplicación
 */
public void notifyRequest(SipConnectionNotifier sn) {
    try {
        ssc = scn.acceptAndOpen();
        if(ssc.getMethod().equals("OPTIONS")) {
          state = S_OPTIONS;
          try {
            ssc.initResponse(200);
            ssc.setHeader("Content-Length", "" + sdp.length());
            ssc.setHeader("Content-Type", "application/sdp");
            ssc.setHeader("Contact", contactoLoc);
            ssc.setHeader("Allow", "MENSAGE CHAT TRIQUI ENVIA_IMG");
            OutputStream os = ssc.openContentOutputStream();
            os.write(sdp.getBytes());
            os.close(); // close and send
          }
          catch (Exception ex) {
          }

        }

        else if(ssc.getMethod().equals("INVITE")) {
           origSSC = ssc;
           state = S_CALLING;
```

```
 String contentType = ssc.getHeader("Content-Type");
 String contentLength = ssc.getHeader("Content-Length");
 String origen = ssc.getHeader("From");
 String contactoOrigen = ssc.getHeader("Contact");

 int length = Integer.parseInt(contentLength);
 if(contentType.equals("application/sdp")) {
   InputStream is = ssc.openContentInputStream();
   byte content[] = new byte[length];
   is.read(content);
   String sc = new String(content);
   // aquí manejo de medios
   sesion = buscarParametro(sc,"s="."\n");
   if(sesion.equals("sesion chat sip")){
     apli.aplicacion = apli.T_CHAT;
     /*int m = sc.indexOf("i=");
     int n = sc.indexOf(":::",m);
     info=sc.substring(m+2,n);
     m = n+3;
     n = sc.indexOf("\n",m);
     tama=sc.substring(m,n);*/
     int n = sc.indexOf("m=");
     int m = sc.indexOf(' ', n+1);
     n = sc.indexOf(' ', m+1);
     puerto = sc.substring(m+1, n);
     transporte = sc.substring(n + 1, sc.indexOf('\n', n));
     url = transporte + puerto;
  }
   else if(sesion.equals("compartir imagen")){
     apli.aplicacion = apli.T_ENVIMG;
     int m = sc.indexOf("i=");
     int n = sc.indexOf(":::",m);
     info=sc.substring(m+2,n);
     m = n+3;
     n = sc.indexOf("\n",m);
     tama=sc.substring(m,n);
     n = sc.indexOf("m=");
     m = sc.indexOf(' ', n+1);
     n = sc.indexOf(' ', m+1);
     puerto = sc.substring(m+1, n);
     transporte = sc.substring(n + 1, sc.indexOf('\n', n));
     url = transporte + puerto;
   }
   else if(sesion.equals("triqui sip")) {
     apli.aplicacion = apli.T_TRIQUI;
   }
   if (apli.aplicacion != apli.T_LIBRE) {
     state = S_RINGING;
     //enviando respuesta 180
     ssc.initResponse(180);
     ssc.send();
     // Salvar el dialogo
     dialog = ssc.getDialog();
     apli.timbrar(origen, contactoOrigen, info);
   }
 }
```

```
}
else if(ssc.getMethod().equals("MESSAGE")) {
    String contentType = ssc.getHeader("Content-Type");
    String contentLength = ssc.getHeader("Content-Length");
    String subject = ssc.getHeader("Subject");
    String desde = ssc.getHeader("From");
    int length = Integer.parseInt(contentLength);
    if((contentType != null) && contentType.equals("text/plain")) {
        InputStream is = ssc.openContentInputStream();
        int i=0;
        byte testBuffer[] = new byte[length];
        i = is.read(testBuffer);
        String tmp = new String(testBuffer, 0, i);
        apli.mensajeRecivido(desde,subject,tmp);
    }
    ssc.initResponse(200);
    ssc.send();
}

else if(ssc.getMethod().equals("ACK")) {
  state = S_ONLINE;
  ssc.close();
  apli.ackRecivido();
}
else if(ssc.getMethod().equals("BYE")) {
   /* if (apli.aplicacion == apli.T_CHAT) {
      ssc.initResponse(200);
      ssc.send();
      ssc.close();
      apli.terminarChat();
      state = S_OFFLINE;
    }
    else if (apli.aplicacion == apli.T_ENVIMG) {
      ssc.initResponse(200);
      ssc.send();
      ssc.close();
      apli.terminarEnvMg();
      state = S_OFFLINE;
    }*/
    if (state == this.S_ONLINE){
      ssc.initResponse(200);
      ssc.send();
      ssc.close();
      apli.terminarApli();
      state = S_OFFLINE;
    }
    else if (state == this.S_RINGING) {
      ssc.initResponse(200);
      ssc.send();
      ssc.close();
      apli.retornar();
      state = S_OFFLINE;
    }

  }
  else if (ssc.getMethod().equals("CANCEL")) {
```

```
            if (state == S_RINGING) {
              ssc.initResponse(200);
              ssc.send();
              ssc.close();
              apli.retornar();
              state = S_OFFLINE;
            }
            else if(state!=S_OFFLINE){
              ssc.initResponse(200);
              ssc.send();
              origSSC.initResponse(487);
              origSSC.send();
              ssc.close();
              apli.cancelRecivido();
              state = S_OFFLINE;
            }

          }
          else if (ssc.getMethod().equals("PRACK")) {
            ssc.initResponse(200);
            ssc.send();
          }
          else {
            // 405 Metodo no permitido
            ssc.initResponse(405);
            ssc.send();
          }
        }
        catch (Exception ex) {
        ex.printStackTrace();
        // manejo de la excepcion
      }
    }
    /**
     * Metodo que inicia un SIP server connection notifier (scn) para escuchar las peticiones
entrantes
     * @return String: dirección en la cual se esta escuchando
     */
    public String iniciarListener() {
      String direccion = null;
      try {
        if (scn != null)
          scn.close();
        scn = (SipConnectionNotifier) Connector.open("sip:"+puertoENT);
        scn.setListener(this);
        S_escuchando = true;
        direccion = "sip:"+ dirLocal() +":"+ String.valueOf(scn.getLocalPort());
      }
      catch (IOException ex) {
        try {
          if (scn != null)
            scn.close();
          scn = (SipConnectionNotifier) Connector.open("sip:");
          scn.setListener(this);
          S_escuchando = true;
            direccion = "sip:"+ dirLocal() +":"+ String.valueOf(scn.getLocalPort());
```

```
          puertoENT = String.valueOf(scn.getLocalPort());
        }
        catch (IOException e) {
          S_escuchando = false;
        }
      }
       return direccion;
    }
/**
 * Metodo que detiene el SIP server connection notifier (scn)
 */
    public void detenerListener() {
  try {
    if (scn != null)
      scn.close();
    scn = null;
    S_escuchando = false;
        }
  catch (IOException ex) {
  }
}
/**
 * Metodo para solicitar las capacidades del destino
 * @param dir_destino String: dirección del usuario a invitar
 * @param dir_origen String: direccion de quien envía la invitación
 */
public void enviarOPTIONS(String dir_destino, String dir_origen) {
  try {
    state = S_OPTIONS;
    iniciarListener();
    scc = (SipClientConnection)
        Connector.open(dir_destino);
    scc.setListener(this);
    scc.initRequest("OPTIONS", null);
    scc.setHeader("From", dir_origen);
    scc.setHeader("To", dir_destino);
    scc.setHeader("Contact",
                "sip:" + scn.getLocalAddress() + ":" + scn.getLocalPort());
    scc.setHeader("Content-Length", "" + sdp.length());
    scc.setHeader("Content-Type", "application/sdp");
    OutputStream os = scc.openContentOutputStream();
    os.write(sdp.getBytes());
    os.close();
   }
  catch (Exception ex) {
    ex.printStackTrace();
  }
}

/**
 * Metodo para invitar a un usuario a una sesion
 * @param dir_destino String: dirección del usuario a invitar
 * @param dir_origen String: direccion de quien envía la invitación
 */
public void enviarINVITE(String dir_destino, String dir_origen) {
  try {
```

```
      state = S_CALLING;
      iniciarListener();
      scc = (SipClientConnection)
         Connector.open(dir_destino);
      scc.setListener(this);
      scc.initRequest("INVITE", null);
      scc.setHeader("From", dir_origen);
      scc.setHeader("To", dir_destino);
      scc.setHeader("Contact",
                  "sip:" + scn.getLocalAddress() + ":" + scn.getLocalPort());
      scc.setHeader("Content-Length", "" + sdp.length());
      scc.setHeader("Content-Type", "application/sdp");
      OutputStream os = scc.openContentOutputStream();
      os.write(sdp.getBytes());
      os.close();
      }
    catch (Exception ex) {
      ex.printStackTrace();
    }
  }
  /**
 * Metodo para aceptar una invitación recivida
 */
public void contestarSIP(){
  try {
    ssc.initResponse(200);
    ssc.setHeader("Content-Length", "" + sdp.length());
    ssc.setHeader("Content-Type", "application/sdp");
    OutputStream os = ssc.openContentOutputStream();
    os.write(sdp.getBytes());
    os.close(); // close and send
    // gusrdar Dialog
    dialog = ssc.getDialog();
    // Esperar por el ACK
    ssc.close();
  }
  catch (Exception ex) {
    ex.printStackTrace();
  }
}
  /**
   * Metodo sobrecargado para enviar mensajes sobre el dialogo abierto
   * @param mensaje String: mensaje a enviar
   * @param subject String: subject del mensaje
   */
  public void enviarMENSAJE(String mensaje, String subject) {
      try {
            SipClientConnection sc = dialog.getNewClientConnection("MESSAGE");
            sc.setHeader("From", usuarioLoc);
            sc.setHeader("Subject", subject);
            sc.setHeader("Content-Type", "text/plain");
            sc.setHeader("Content-Length", "" + mensaje.length());
            OutputStream os = sc.openContentOutputStream();
            os.write(mensaje.getBytes());
            os.close(); // cierra y envia
            //state = S_CALLING;
```

```
    } catch(Exception ex) {
        ex.printStackTrace();
    }
 }
 /**
  * Metodo sobrecargado para enviar mensajes abriendo una nueva conexión SIP
  * @param mensaje String: mensaje a enviar
  * @param subject String: subject del mensaje
  * @param dir_destino String: direccion SIP a la cual se va a enviar el mensaje
  */
 public void enviarMENSAJE(String mensaje,String subject,String dir_destino) {
   try {
       scc = (SipClientConnection) Connector.open(dir_destino);
       scc.setListener(this);
       scc.initRequest("MESSAGE", null);
       scc.setHeader("From", usuarioLoc);
       scc.setHeader("Subject", subject);
       scc.setHeader("Content-Type", "text/plain");
       scc.setHeader("Content-Length", "" + mensaje.length());
       OutputStream os = scc.openContentOutputStream();
       os.write(mensaje.getBytes());
       os.close(); // cierra y envia
       //state = S_ONSENDMES;
   } catch(Exception ex) {
       ex.printStackTrace();
   }
}
/**
 * Metodo para registrarse ante un servidor de registro SIP
 * @return String: direccion de contacto registrada
 */
public String  enviarREGISTER() {
   String dircon = "";
   iniciarListener();
    try {
        SipClientConnection scc = null;
        // Open a SipClientConnection for REGISTER targeting the
        // registrar address
        scc = (SipClientConnection) Connector.open(proxySer);
        scc.setListener(this);
        scc.initRequest("REGISTER", null);
        // Set necessary headers
        scc.setHeader("From", usuarioLoc);
        scc.setHeader("To", usuarioLoc);
        scc.setHeader("Contact", contactoLoc+":"+scn.getLocalPort());
        dircon = contactoLoc+":"+scn.getLocalPort();
        // Send it out
        scc.send();
    } catch(Exception ex) {
        ex.printStackTrace();
    }
    return dircon;
 }
 /**
  * Metodo para eliminar el registro de un servidor de registro
  * @return String
```

```
     */
    public void eliminarREGISTER() {
          try {
              SipClientConnection scc = null;
              scc = (SipClientConnection) Connector.open(proxySer);
              scc.setListener(this);
              scc.initRequest("REGISTER", null);
              // Set necessary headers
              scc.setHeader("From", usuarioLoc);
              scc.setHeader("To", usuarioLoc);
              scc.setHeader("Contact", contactoLoc+":"+scn.getLocalPort());
              scc.setHeader("Expires","0");
              scc.send();
          } catch(Exception ex) {
              ex.printStackTrace();
          }
        }


/**
 * Metodo para terminar una sesión.
 * Requiere de un dialogo (dialog)
 */
public void enviarBYE() {
  if (dialog != null) {
    try {
      SipClientConnection sc = dialog.getNewClientConnection("BYE");
      sc.send();
      boolean gotit = sc.receive(10000);
      if (gotit) {
        if (sc.getStatusCode() == 200) {
        }
        else
      }
      sc.close();
      state = S_OFFLINE;
    }
    catch (IOException iox) {
    }
  }
  else {
  }
}
/**
 * Metodo para cancelar una sesión.
 * Requiere de un SipClientConnection (scc)
 */
public void enviarCANCEL() {
  state = S_OFFLINE;
  if (scc != null) {
    try {
      SipClientConnection cancel = scc.initCancel();
      cancel.send();
      if (cancel.receive(30000)) {
        //state = S_OFFLINE;
      }
```

```
        else {
        }
      }
      catch (Exception ex) {
        ex.printStackTrace();
      }
    }
    else{
      }
  }
  public void noResponder() {
    if(state == S_RINGING) {
      try {
        ssc.initResponse(486);
        ssc.send();
      }
      catch (Exception ex) {
        ex.printStackTrace();
      }
    }
    else {
      enviarBYE();
    }
    state = S_OFFLINE;
  }
  /**
   * Metodo para adquirir la dirección local del dispositivo
   * @return String: direccion local
   */
  public String dirLocal() {
   String direccion = "";
   try {
     ServerSocketConnection ssc1 = (ServerSocketConnection) Connector.open("socket://:");
     direccion = ssc1.getLocalAddress();
     ssc1.close();
   }
    catch (Exception e) {
    }
    return direccion;
  }
  private String buscarParametro (String cadena, String parametro, String separador){
    String contenido = "";
    int m = cadena.indexOf(parametro);
    int n = cadena.indexOf(separador, m);
    contenido = cadena.substring(m+parametro.length(), n);
    return contenido;
  }
}
```

5.9 PRUEBAS DE RENDIMIENTO

A la aplicación ChatSIP se le realizaron algunas pruebas para verificar su desempeño, estas pruebas se realizaron con software de simulación que proporcionaba la capacidad de modificar algunos parámetros de operación como son el ancho de banda y la memoria disponibles para el dispositivo.

Se desea aclarar que por la falta de un dispositivo real la aplicación fue probada en simulación solo con la transferencia de los diferentes mensajes SIP y como aplicación única en ejecución, por lo que estas pruebas solo representan la carga que puede producir el manejo del protocolo y transferencia de mensajes SIP mas no del funcionamiento de una aplicación P2P con SIP que proporciona la transferencia de diversos medios.

La tabla 5 muestra el comportamiento obtenido por la aplicación para diferentes velocidades de la red. De donde se puede observar que necesariamente se requiere de una red que provea velocidades iguales o superiores a 33600 b/s para implementar sencillas aplicaciones con SIP.

Velocidad (b/s)	Resultado
2400	Produce errores en la comunicación
9600	Repetición de mensajes
14400	3.25 segundos aprox
19200	Retardo de 2.83 segundos aprox
28800	Retardo de 1.60 segundos aprox
33600	<1 segundo
56000	<1 segundo
112000	<1 segundo

Tabla 5. Resultados obtenidos a diferentes velocidades de transferencia.

La tabla 6 muestra el comportamiento obtenido por la aplicación para diferentes cantidades de memoria disponible en el dispositivo. Esta tabla muestra que la aplicación sola trabajando con SIP no requiere de cantidades especiales de memoria del dispositivo.

Memoria disponible	Resultado
512K	OK
256K	OK
128K	OK
64K	OK
32K	OK

Tabla 6. Resultados obtenidos con diferentes cantidades de memoria.

La tabla 7 muestra el tamaño de los paquetes que envía la aplicación. Esto sirve para darse una idea del consumo del canal de datos que puede producir el solo protocolo o una aplicación basada en mensajes SIP.

Mensaje	Tamaño aproximado
INVITE	455 bytes
MESSAGE	382 bytes
BYE	243 bytes
200 OK	211 bytes
180 TIMBRANDO	259 bytes

Tabla 7. Tamaños de los paquetes que envía la aplicación.

Los resultados obtenidos de estas pruebas confirman las características de SIP, las cuales especifican que este protocolo por ser diseñado a partir de tecnologías Web como HTTP y SMTP es muy liviano y produce poca carga y exigencia de recursos. Como ejemplo a continuación se calcula los Bytes requeridos para una proporcionar una sesión de Chat con 500 mensajes con alrededor de 30 caracteres por mensaje.

INVITE+180+200+500 MESSAGE +BYE = 187.66 KBytes

Para transferir una imagen de 10KBytes se requiere:

INVITE+180+200+IMAGEN+BYE (10KBytes) = 11.14KBytes

6. CONCLUSIONES Y TRABAJOS FUTUROS

Las recomendaciones generadas permiten al desarrollador que desea conocer e iniciar en la implementación de aplicaciones móviles P2P con SIP, conocer de antemano elementos importantes que facilitaran el aprendizaje y desarrollo de una aplicación de manera mas orientada y evitando posibles errores que pueden dificultar concretar su objetivo.

Hasta el momento, solo las plataformas J2ME y Symbian proporcionan libremente el soporte necesario para el desarrollo de aplicaciones móviles P2P con SIP, pero dada la tendencia de las comunicaciones hacia una convergencia, en un futuro es posible que más plataformas móviles se sumen a este grupo, proporcionando los recursos requeridos para la realización de este tipo de aplicaciones.

El funcionamiento de SIP para ambas plataformas, es igual al descrito en su especificación, pero se debe tener en cuenta que existen diferencias importantes a nivel de implementación para cada plataforma, debiéndose revisar cuidadosamente la documentación para cada una.

Entre las dos plataformas sobre las cuales se puede desarrollar aplicaciones móviles P2P con SIP, no se puede afirmar que una sea mejor que la otra. Aspectos como la funcionalidad, desempeño, portabilidad y soporte necesario deben ser evaluados en cada caso, para determinar cual de las dos es la más apropiada para un desarrollo específico.

Para el desarrollo de aplicaciones móviles P2P con SIP que requieran del servicio de presencia, se debe hacer uso de una forma alterna mediante el método OPTIONS, ya aunque existe una especificación para el manejo de la presencia; hasta el momento no se encuentra disponible un API específico que permita su implementación.

El aprendizaje del desarrollo de aplicaciones móviles P2P con SIP, tiende a ser mas sencillo en J2ME que en Symbian, debido a que Symbian cuenta con un mayor contenido de APIs, clases y métodos para el desarrollo, existe mayor disponibilidad de ejemplos sencillos en J2ME que explican claramente una implementación, mucha documentación para J2ME se encuentra disponible en español y se presenta mayor facilidad para encontrar grupos de desarrollo y foros donde se implementa aplicaciones móviles con SIP sobre J2ME.

El escenario actual de las comunicaciones evolucionará hacia una convergencia tecnológica, cuyo propósito será comunicar a los usuarios independientemente de la ubicación y el equipo terminal. Las características de SIP hacen de este protocolo un elemento importante para alcanzar este objetivo

La sencillez, flexibilidad y robustez de SIP, hacen de este protocolo una herramienta apropiada para el desarrollo de nuevas y novedosas aplicaciones y mejorar las existentes, permitiendo la incorporación de características que resultan muy atractivas y funcionales para los usuarios finales.

Las recomendaciones para el desarrollo de aplicaciones móviles P2P con SIP no son un elemento esencial para el desarrollo de este tipo de aplicaciones, pero su conocimiento brinda la posibilidad de partir desde un punto mas adelantado, pudiendo centrarse más en la esencia del problema a resolver, al reutilizar un conocimiento que ya se encuentra disponible.

Las aplicaciones móviles P2P con SIP sobre redes celulares son técnicamente realizables en Colombia, a nivel comercialmente y de despliegue tal vez requieran de un análisis mas profundo para que sean atractivas para los usuarios.

Las especificaciones y las aplicaciones generadas en este trabajo corresponden a un paso inicial en el conocimiento del protocolo SIP sobre dispositivos móviles, que puede

ser ampliado con el diseño y construcción de aplicaciones mas completas, que permitan una interacción multimedia entre diferentes dispositivos y/o aplicación a entornos donde un desarrollo de este tipo puede ser de gran utilidad o solucionar una necesidad específica.

El conocimiento que proporciona este trabajo puede ser ampliado, con el estudio del protocolo en las plataformas que se excluyeron por falta de soporte, pero que se considera que en un futuro muy cercano se podrá contar con los elementos necesarios para poder implementar a SIP sin dificultad.

GLOSARIO

Agente Usuario (UA): Entidad lógica que puede actuar como cliente de agente de usuario (UAC) y servidor de agente de usuario (UAS). Ver también, UAC y UAS.

API: (A*plication Programing Interface* - Interfase de programación de aplicaciones). Corresponde a un conjunto de especificaciones de comunicación entre componentes software. Uno de los principales propósitos de una API consiste en proporcionar un conjunto de funciones de uso general; de esta forma, los programadores se benefician del API haciendo uso de su funcionalidad y evitándo programar todo desde el principio.

CDC: (*Connected Device Configuration*) configuración para dispositivos con capacidades no tan reducidas, presentada en la tecnología j2me.

CLDC: (*Connected Limited Device Configuration*) configuración para dispositivos móviles con limitaciones, presentada en la tecnología j2me.

Cliente de Agente Usuario (UAC): Es una entidad lógica que crea una solicitud SIP, y la envía haciendo uso de una conexión de cliente. El papel del UAC, sólo se mantiene para la duración de una transacción; por ejemplo, si una aplicación inicia una solicitud, actuará como un UAC para la duración de esa transacción.

Dialogo SIP: Corresponde a una comunicación establecida entre dos UAs, en la cual se intercambian diferentes solicitudes y respuestas. Los SDKs para J2ME y Symbian permiten guardar la información referente a un diálogo SIP en una clase, facilitando el envío de mensajes y respuestas subsecuentes sin necesidad de abrir una nueva conexión.

Emulador: sistema que simula en un computador el funcionamiento de un equipo terminal móvil, permitiendo probar las aplicaciones desarrolladas sin necesidad de poseer el equipo real.

IDE: (*Integrated Development Environment* - Entorno de Desarrollo Integrado) Un IDE es un entorno de programación que ha sido empaquetado como un programa de aplicación,

es decir, consiste en un editor de código, un compilador, un depurador y un constructor de interfaz gráfica GUI.

J2ME: Java 2 Micro Edition, version de java para dispositions moviles.

Método: El método es una función primaria que significa una demanda invocada a un servidor. El método es llevado en el propio mensaje de solicitud. Algunos métodos del ejemplo son INVITE y BYE.

MIDP: (*Mobile Information Device Profile*) perfil de j2me construido sobre la configuración CLDC para desarrollar aplicaciones para dispositivos móviles con limitaciones de memoria y de capacidad de procesamiento.

Observar: se le denomina al proceso por el cual, el sistema esta pendiente de alguna solicitud entrante.

Peer-to-peer: Arquitectura en la cual el intercambio de información se hace directamente entre los terminales involucrados. Se suele abreviar como P2P.

Perfil: Biblioteca de clases específicas orientadas a implementar funcionalidades de más alto nivel para familias específicas de dispositivos.

Plugin: (o plug-in) es un programa de ordenador (adicional) que interactúa con otro programa para aportarle una función o utilidad específica .Se utilizan como una forma de expandir programas de forma modular, de manera que se puedan añadir nuevas funcionalidades sin afectar a las ya existentes ni complicar el desarrollo del programa principal.

SDK: (*Software Development Kit* - kit de desarrollo de software) conjunto de herramientas de desarrollo que le permite a un programador crear aplicaciones.

Servidor de Agente Usuario (UAS): Un servidor de agente de usuario es una entidad lógica que genera una respuesta a una solicitud SIP. La respuesta puede aceptar, rechazar, o remitir la solicitud. El papel de UAS sólo se mantiene para la duración de esa transacción.

Symbian: Sistema operativo para teléfonos móviles.

Transacción SIP: Una transacción SIP ocurre entre un cliente y un servidor y comprende todos los mensajes desde la primera solicitud enviada desde el cliente al servidor hasta la última respuesta (diferente a 1xx) enviada del servidor al cliente. Si la solicitud es un INVITE y la contestación final no es un 2xx, la transacción también incluye al ACK de la

respuesta. El ACK para una respuesta 2xx a una solicitud INVITE es una transacción separada.

Respuesta provisional: Respuesta usada por el servidor para indicar progreso, pero no termina una transacción SIP. Las respuestas 1xx son respuestas provisionales, las demás son consideradas finales.

Respuesta final: Respuesta que termina una transacción SIP. Todas las respuestas 2xx, 3xx, 4xx, 5xx y 6xx son respuestas finales.

BIBLIOGRAFÍA

[1] Castañeda Segura Rodolfo. Protocolos para voz IP. Abril de 2005. www.cudi.edu.mx/primavera_2005/presentaciones/rodolfo_castaneda.pdf

[2] Siemens Communications. SIP The Session Initiation Protocol For Enterprise SIP Solutions. Octubre de 2004. https://developer.ubiquitysoftware.com/business/whitepapers/sip_in_nextgen_mobile_nets.pdf

[3] Microsoft. Windows Live Messenger. http://imagine-msn.com/messenger/launch80/default.aspx?locale=es-es

[4] Moreno Marta, Sánchez Antonio, Fernández María, Arance Francisco, Fernández Miguel Ángel. Los servicios conversacionales de nueva generación. Marzo 2001. www.ugr.es/~rlopezc/archivosParaDescargarWebEspanol/sepln2001.pdf.

[5] Ubiquity. Understanding SIP Today's Hottest Communications Protocol Comes of Age. Julio de 2004. http://www.sipcenter.com/sip.nsf/html/WEBB5YNVK8/$FILE/Ubiquity_SIP_Overview.pdf

[6] IETF. Session Initiation Protocol (sip). [en línea] http://www.ietf.org. [consulta: 10 septiembre de 2006].

[7] Poyeaux Nelson. El Protocolo SIP: propuesta del IETF para la transmisión de VoIP (parte 1). [en línea] Revista electrónica Telem@tica. Octubre de 2004. año 3 No 14

[8] Poyeaux Nelson. El Protocolo SIP: propuesta del IETF para la transmisión de VoIP (parte 2). [en línea] Revista electrónica Telem@tica. Octubre de 2004. año 3 No 15

[9] Landaure Enrique. Estándares relacionados a la tecnología Voz sobre IP (VoIP): Su Clasificación. Diciembre de 2001.

[10] Castañeda Segura Rodolfo. Protocolos para voz IP. Abril de 2005. www.cudi.edu.mx/primavera_2005/presentaciones/rodolfo_castaneda.pdf

[11] Henning Schulzrinne, Elin Wedlund. Voz Mobility Support using SIP. Octubre de 2005. http://www.agapea.com /Mobility Support using SIP.pdf.

[12] Stokle Mariano. Session Initiation Protocol. Noviembre de 2004. www.cs.columbia.edu/~coms6181/slides/11/sip_long.pdf

[13] Manso Callejo Miguel Ángel. SIP: El protocolo para los servicios multimedia del futuro. Resumen. Marzo de 2003.

[14] Charly. Nociones basicas sobre peer to peer Capitulo 1. Noviembre de 2005. www.tectimes.com/lbr/Graphs/revistas/ldrme017/capitulogratis.pdf

[15] Wikipedia. Peer to peer. [en línea] http://en.wikipedia.org/wiki/Peer-to-peer [consulta: 12 de noviembre de 2006]

[16] Redes P2P en movilidad de intercambio legal de contenidos multimedia. Noviembre de 2006 internetng.dit.upm.es/premio%20NAI2006/P2P.pdf

[17] Symbian. Symbian http://www.symbian.com/

[18] Qualcomm. BREW. http://brew.qualcomm.com/brew/es/

[19] Sun Microsystems. Java ME – Micro App Development Made Easy. [en línea] http://java.sun.com/javame/index.jsp

[20] Nokia. Device Specifications. [en línea] http://www.forum.nokia.com/devices/matrix_s60_1.html [consulta: 20 de octubre de 2006]

[21] Microsoft. Windows Mobile. [en línea] http://www.microsoft.com/windowsmobile/default.mspx [consulta: 12 de noviembre de 2006]

[22] Briody Dan. Los relegados – The Feature. Abril de 2003. [en línea] http://brew.qualcomm.com/brew/es/press_room/bitn/2003/04_28_03.html [consulta: 16 octubre de 2006]

[23] Wikipedia. Symbian. [en línea] http://es.wikipedia.org/wiki/Symbian [consulta: 20 de octubre de 2006]

[24] Nokia. S60 Plataform. [en línea] http://www.forum.nokia.com/main/platforms/s60/index.html#overview [consulta: 23 de octubre]

[25] Nokia no oficial. Qué es Symbian. [en línea] www.nokiagratis.com [consulta: noviembre de 2005]

[26] Adobe. Adobe Flash Lite. http://www.adobe.com/products/flashlite/

[27] USB Implementers Forum. Universal Serial Bus. http://www.usb.org

[28] Special Interest Group. Bluetooth. http://www.bluetooth.com

[29] 1394 Trade Association. 1394. http://www.1394ta.org

[30] Rodríguez José Luís. Sistemas operativos en el cinturón. [en línea] http://www.enterate.unam.mx/Articulos/2005/abril/sistopera.htm [consulta: 21 de octubre de 2006]

[31] Symbian. Enterprising Symbian. [en línea] http://www.symbian.com/symbianos/enterprise/enterprising.html [consulta: 21 de octubre de 2006]

[32] Pedra Marcelo. Glosario informático y de Internet. [en línea] http://www.marcelopedra.com.ar/glosario_J.htm [consulta: 20 de octubre de 2006]

[33] Nokia, Nokia Prototype SDK 4.0 Beta for JME. Directorio de instalación del Nokia_Prototype_SDK_4_0_Beta\Nokia_Prototype_SDK_4_0_Beta\docs

[34] Sun Microsystems. Java ME APis & Docs. [en línea] http://java.sun.com/javame/reference/apis.jsp [consulta: 21 de octubre de 2006]

[35] BREW. La solución BREW. [en línea] http://brew.qualcomm.com/brew/es/developer/resources/gs/brew_solution.html [consulta: 20 de octubre de 2006]

[36] Osmosis Latina. Aplicaciones Inalámbricas un Modelo Nuevo. [en línea] http://www.osmosislatina.com/aplicaciones/j2mebrew.htm. [consulta: 20 de octubre de 2006]

[37] Qualcomm . Market Research. [en línea] http://brew.qualcomm.com/brew/en/developer/resources/ds/market_research_data.html [consulta: 20 de octubre de 2006]

[38] Wikipedia. Over the air programming. http://en.wikipedia.org/wiki/Over-the-air_programming

[39] Sharp. Handhelds Zaurus. http://www.sharpusa.com/products/TypeLanding/0,1056,112,00.html

[40] Kaushal Sheth. Open Zaurus. http://www.openzaurus.org/wordpress/

[41] PdaXrom. http://www.pdaxrom.org/

[42] Maslovsky Anton. My zaurus. http://my-zaurus.narod.ru/cacko.html

[43] Amphora. Ponga una zaurus en su vida. [en linea] http://libertonia.escomposlinux.org/story/2006/1/10/19306/4027 [consulta: 6 de noviembre de 2006]

[44] Wikipedia. Linux Empotrado. [en línea] www.wikipedia.org [consulta: 22 de octubre de 2006].

[45] A la mobile. Open Linux System Platform & Open Source Technology Provider for Mobile Handsers. http://www.a-la-mobile.com/

[46] LIPS Linux Phone Standards Forum. http://www.lipsforum.org/

[47] OSDL. http://www.osdl.org/

[48] CE Linux Forum. http://www.celinuxforum.org/

[49] A la mobile. Convergent Linux. [en linea] http://www.a-la-mobile.com/solutions/convergent.html [consulta: 25 de octubre de 2006]

[50] Xtandard. [en linea] http://xtandard.com/2005/07/15/%c2%a1linux-en-celulares-tambien/ [consulta: 24 de octubre de 2006]

[51] TROLLTECH. Qtopia Greenphone. http://www.trolltech.com/products/qtopia/greenphone/index

[52] Barrapunto.com. Greenphone, un teléfono móvil basado en Linux. [en línea] http://barrapunto.com/article.pl?sid=06/08/15/2013253&mode= thread [consulta: 9 de noviembre de 2006]

[53] Qtek. http://www.myqtek.com/

[54] Windows Mobile. http://www.microsoft.com/spain/windowsmobile/default.mspx

[55] Wikipedia. Windows Mobile. [en línea] www.wikipedia.org [consulta: 27 de octubre de2006]

[56] MobileDeveloper wiki. Documentation for the Windows Mobile [en línea] http://channel9.msdn.com/wiki/default.aspx/MobileDeveloper.APIs [consulta: 22 de octubre de 2006]

[57] MSDN Microsoft Windows Mobile Developer Center. Legacy Tools. [en línea] http://msdn.microsoft.com/windowsmobile/downloads/tools/legacy/default.aspx [consulta: 27 de octubre de 2006]

[58] Modelo Integral para el Profesional en Ingeniería. Autor: Carlos Enrrique Serrano Castaño. Año : 2002