JAvatar Móvil: un desarrollo experimental JME para redes IMS

Isabel Kimura (isabel_kimura@hotmail.com)
Matías Brown (matibrown@gmail.com)
Roberto Guisandez (rguisandez@gmail.com)

Trabajo de Cátedra de la Asignatura Laboratorio de Software
Facultad de Informática - Universidad Nacional de La Plata
Profesora: Claudia Alejandra Queiruga (claudiaq@info.unlp.edu.ar)
Jefe de Trabajos Prácticos: Jorge Horacio Rosso (jrosso@info.unlp.edu.ar)

Abstract. JAvatar Móvil es una aplicación JME diseñada y desarrollada para ejecutarse sobre una red en la que convergen múltiples servicios, ofrecidos sobre una infraestructura común de transmisión de datos IP, IMS (IP Multimedia Subsystem). **JAvatar Móvil** permite crear y compartir ***avatares*** entre usuarios de la red IMS, pudiéndose integrar con aplicaciones popularmente usadas como redes sociales y de mensajería que usan una representación visual de sus usuarios. **JAvatar Móvil** es un trabajo de la cátedra "Laboratorio de Software" de la Licenciatura en Informática de la Facultad de Informática de la Universidad Nacional de La Plata.

Palabras Clave: IMS, SIP, JME, P2P, JAVA, MIDP

1 Introducción

JAvatar Móvil es una aplicación móvil que permite crear y compartir ***avatares*** entre usuarios de una red IMS (IP Multimedia Subsystem).

JAvatar Móvil surge como un trabajo de la cátedra "Laboratorio de Software", asignatura de 4º año de la carrera Licenciatura en Informática de la Universidad Nacional de La Plata, y es una actividad enmarcada en el convenio de la Facultad de Informática [1] con la empresa Ericsson Argentina [2] que fomenta, entre otras cosas, el desarrollo de aplicaciones basadas en IMS. Con el objetivo de desarrollar clientes JAVA basados en redes IMS, optamos por realizar una aplicación móvil, teniendo en cuenta que actualmente los dispositivos móviles son el medio de comunicación y de entretenimiento que más ha crecido y que JME es la tecnología estándar adoptada en la industria de dispositivos móviles para desarrollo de aplicaciones.

La ventaja de contar con una red convergente como IMS de mayor ancho de banda móvil y con soporte de calidad de servicios, nos permite construir nuevas aplicaciones y servicios multimedia, que incrementan la experiencia de los usuarios [3].

1.1 Fundamentos de IMS

Estamos en un mundo en el que las comunicaciones digitales han modificado la forma en que las personas se comunican. La telefonía móvil tiene un rol central en este cambio, ya que no sólo se utiliza para "hablar", sino también para enviar datos, compartir videos, fotos, voz y mucho más [3].

Históricamente, los servicios de voz, datos y video se han prestado sobre infraestructuras de red dedicadas, incluso por distintos proveedores de telecomunicaciones. A su vez se necesitan distintos tipos de dispositivos para acceder a cada servicio (teléfono, computadora, televisión, etc.) usando diferentes identidades y sin poder "movernos" (roaming) de forma transparente entre los dispositivos [3].

La telefonía clásica usa un esquema basado en un dispositivo final conectado a una red de conmutación de circuitos con servicios suministrados únicamente por el proveedor de telefonía o de servicios específicos. Este esquema en la evolución de la "redes" tiende a desaparecer. Es un hecho que en las redes IP cualquier estación de trabajo (host) puede actuar como consumidor o como proveedor de servicios, dando lugar en Internet al surgimiento de redes P2P [3].

1.1.1 Definición

IMS (IP Multimedia Subsystem) es una arquitectura de red en la que convergen múltiples servicios, ofrecidos sobre una infraestructura común de transmisión de datos IP. IMS es un conjunto de especificaciones que describen la arquitectura de las redes de la nueva generación (Next Generation Network) para implementar telefonía basada en IP y servicios multimedia.

IMS define una arquitectura y un *framework* completo que permite la convergencia de voz, video, datos y tecnologías de red móvil a través de una infraestructura basada en IP. Une la brecha entre los dos paradigmas de comunicación de mayor éxito, los celulares y la tecnología de Internet. Esta es la visión inicial de IMS: ofrecer acceso móvil a todos los servicios que ofrece Internet [3].

La arquitectura IMS sobre las redes de telecomunicación ofrece una convergencia total de servicios estándares (telefonía fija y móvil, TV por cable y satélite, Internet). Con IMS, los operadores de telecomunicaciones podrán integrar las llamadas fijas con las móviles y la voz con el acceso a Internet de banda ancha, la televisión y el video bajo demanda, abriendo las puertas a nuevas aplicaciones multimedia residenciales y empresariales [3].

1.1.2 Arquitectura

El diseño de la arquitectura IMS propuesto por el 3GPP (3rd Generation Partnership Project) está basado en capas funcionales. Esta solución facilita disponer de los servicios ofrecidos a través de la red IMS en forma independiente de la red de acceso. De esta manera, cualquier subscriptor de la red IMS que se comunica mediante su teléfono móvil o PC, utiliza el mismo servicio de presencia y de manejo

de grupos, por nombrar algunos, sin importar la tecnología de acceso usada. Sin embargo, las posibilidades de ancho de banda y de latencia de las conexiones podrían se diferentes, así como la velocidad de procesamiento en el dispositivo IMS [3].

El corazón de la arquitectura IMS es el CSCF (Call Session Control Function) y sus funciones principales son la registración, el establecimiento de sesión y estrategias de ruteo SIP. A su vez la arquitectura IMS especifica un servidor de base de datos principal llamado HSS (Home Subscriber Server) que almacena la información de todos los suscriptores y datos relacionados a los servicios. El diseño basado en capas minimiza la dependencia entre ellas, facilitando el agregado de nuevas redes de acceso a la red IMS [3].

La **Figura 1** muestra la arquitectura de una red IMS.

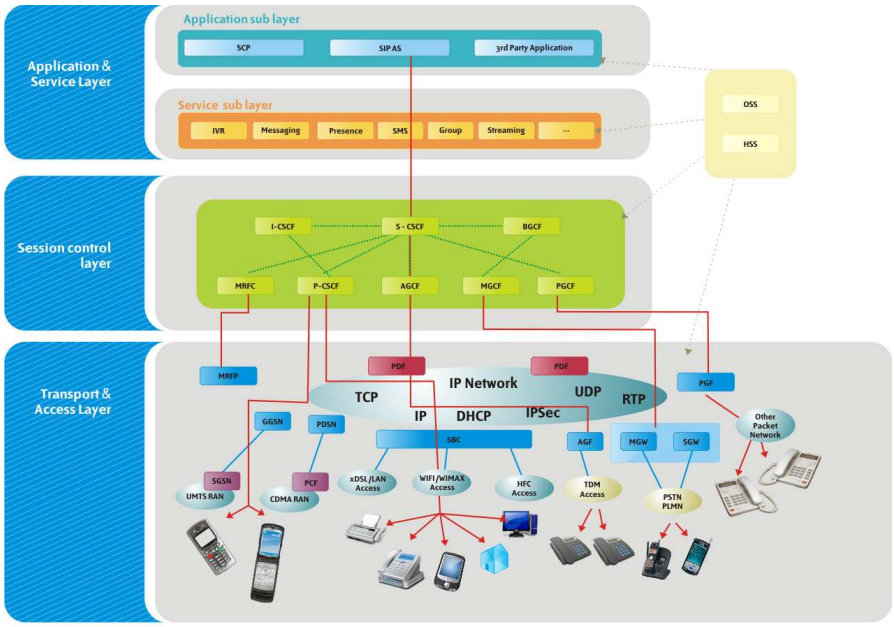

Figura 1: Arquitectura IMS

1.1.3 El protocolo SIP

SIP (Session Initiation Protocol) es un protocolo desarrollado por el IETF (Internet Engineering Task Force) perteneciente a la capa de aplicación del modelo TCP/IP y es usado para establecer, modificar y finalizar sesiones multimedia en una red IP. La especificación completa está disponible en la RFC3261 del IETF. Las aplicaciones típicas están relacionadas con video, voz, juegos, mensajería, control de llamadas y presencia. Una sesión SIP podría ser una llamada telefónica entre dos o más personas o una videoconferencia. El rol de SIP en estas aplicaciones es, durante el

establecimiento de la sesión, la negociación de los parámetros de la comunicación entre sus participantes [3].

Una de las características de diseño de SIP es su capacidad para funcionar en colaboración con otros protocolos usados para tareas específicas [3].

En resumen, la función de SIP en relación a estas aplicaciones es la administración de sesiones y el consenso de sus atributos mediante el intercambio de mensajes textuales, delegando en otros protocolos las tareas específicas [3].

La identificación de los usuarios en IMS está basada en el esquema de direcciones propuesto por SIP: similares a las de correo electrónico anteponiéndole el prefijo *sip:* o *sips:* (SIP seguro) y a su vez pueden contener parámetros adicionales que permiten indicar preferencias [3].

SIP es un protocolo del tipo requerimiento-respuesta y sus mensajes tienen un formato similar a los de HTTP (Hypertext Transfer Protocol) y SMTP (Simple Mail Transfer Protocol), utilizados en los servicios de páginas Web y de distribución de e-mails respectivamente. Uno de los objetivos del diseño de SIP fue transformar a la telefonía en un servicio más de la Internet, es por este motivo que se basó en dos de los protocolos más ampliamente difundidos [3].

1.2 ¿Por qué JME?

La neutralidad de la plataforma JAVA, la capacidad de estar disponible "en todos lados" desde dispositivos móviles con escasos recursos de hardware hasta servidores incluyendo PCs de escritorio, laptops, etc. y la estandarización de las APIs para acceder a las funcionalidades incorporadas y construir nuevas sin depender de implementaciones particulares de los fabricantes, posiciona a JAVA como una plataforma de desarrollo ideal y conveniente para la construcción de las nuevas aplicaciones convergentes a través de IMS [3]. Otra característica destacable de JAVA es su disponibilidad en diferentes IDEs (Integrated Development Enviroment) de la comunidad de software libre como las iniciativas Eclipse [8] y NetBeans [9].

Los teléfonos celulares cada vez tienen más prestaciones, sus pantallas son de mayor precisión y tamaño, tienen cámaras fotográficas y de video incorporadas, reproducen música y tienen navegadores GPS. Son dispositivos móviles que tienen la capacidad de estar siempre conectados (always on), representando un desafío para las nuevas aplicaciones que dejan de ser entidades aisladas que intercambian información a través de la interfaz de usuario. La nueva generación de aplicaciones será *peer-to-peer* facilitando compartir música, juegos, pizarras, video en vivo, etc. [3].

El JCP (Java Community Process) tiene varias actividades enfocadas en la definición de múltiples JSRs (Java Specification Request) que proveen soporte para el desarrollo de aplicaciones basadas en SIP e IMS en los distintos *sabores* de las tecnologías JAVA [3].

Es de interés de este trabajo analizar las iniciativas de JAVA para dispositivos móviles.

La plataforma JME ha evolucionado en respuesta al crecimiento de las capacidades de los dispositivos que se han introducido en el mundo móvil, basándose en la especificación MIDP (Mobile Information Device Profile). Actualmente la tendencia en la industria del software es estandarizar el acceso a estas nuevas capacidades de

hardware de los dispositivos móviles y a los nuevos servicios provistos a través de Internet. El JCP adhiere a esta tendencia habiendo elaborado una especificación construida por encima de MIDP llamada MSA (Mobile Service Architecture).

El JCP continuó trabajando en una nueva versión de MSA llamada MSA Advanced o MSA 2 (JSR 249) que aumenta la funcionalidad estándar incorporando tecnologías actuales con buena proyección al futuro, especialmente en el área de multimedia como es el caso de la JSR 281 llamada IMS Services. Esto pone de manifiesto que la tecnología IMS estará disponible en la plataforma JME, promoviendo su adopción por parte de las empresas manufactureras de dispositivos móviles y por otro lado impactando favorablemente en el desarrollo de aplicaciones para estos dispositivos [3].

2 Descripción de JAvatar Móvil

JAvatar Móvil es una aplicación basada en la especificación MIDP para el entorno JAVA ME, es decir, es un **MIDlet** que permite crear y compartir ***avatares*** entre usuarios de una red IMS utilizando el modelo de comunicación *peer-to-peer*. La **Figura 2** muestra la pantalla inicial de **JAvatar Móvil** en la que es posible visualizar las diferentes opciones que provee la aplicación.

Figura 2: Menú Principal

Cada usuario puede crear ***avatares*** eligiendo entre diferentes componentes, tales como cabellos, cabeza, accesorios, ojos, boca, etc. Un usuario siempre tiene asignado un ***avatar*** personal que lo representa, pero a su vez puede también agregar nuevos

avatares a una galería para luego seleccionar uno entre ellos como su representante. Todos los ***avatares*** en la galería pueden ser editados y / o enviados a otros contactos de la red IMS que estén conectados. Sólo es posible realizar el envío de ***avatares*** a quienes estén en la lista de contactos. La **Figura 3** muestra la pantalla del editor y la galería de ***avatares***.

Figura 3: Editor y Galería de Avatares

La lista de contactos es una colección de usuarios a los cuales se acepta como amigos para compartir los ***avatares***. En esta lista es posible visualizar el ***avatar*** que representa a cada uno de los amigos, verificar el estado de la conexión de cada contacto (conectado / desconectado) y ver o editar información personal (email, teléfono, apodo). El estado y el ***avatar*** de cada contacto se actualizan automáticamente cuando se establece la sesión entre los usuarios. La **Figura 4** muestra la lista de contactos.

Figura 4: Lista de Contactos

La aplicación puede ser personalizada a gusto de cada usuario pudiendo elegir entre diferentes temas visuales. La **Figura 5** muestra diferentes personalizaciones de **JAvatar Móvil**.

Figura 5: Uso de diferentes temas en la aplicación

2. 1 ¿Cómo usamos IMS?

A continuación describiremos las etapas de interacción de **JAvatar Móvil** con las diferentes componentes IMS.

Registración

Cuando un usuario inicia por primera vez la aplicación **JAvatar Móvil** debe registrarse con su servidor CSCF (Control Session Control Function), indicando su identificación de usuario IMS, contraseña y demás datos de registro. Si los datos ingresados son válidos, se establece una conexión IMS con el servidor que permitirá luego iniciar sesiones con otros usuarios de **JAvatar Móvil**. Los datos ingresados son persistidos en la base de datos del dispositivo móvil, para ser utilizados en usos posteriores de la aplicación. La **Figura 6** muestra la pantalla de registro en la red IMS.

Figura 6: Pantalla de Registro de Usuario

Comunicación / Envío de datos

Las comunicaciones se realizan directamente entre los usuarios, cliente a cliente. Cada vez que un usuario inicia la aplicación, automáticamente se establece la conexión con el servidor y se avisa a todos sus contactos, comenzando una sesión con todos aquellos pares que estén conectados.

Cada vez que se actualiza el ***avatar*** personal del usuario se informa a cada contacto conectado utilizando las sesiones ya establecidas. La **Figura 7** muestra visualmente la actualización automática realizada en respuesta al cambio del ***avatar*** de un usuario:

1. A la izquierda vemos dos teléfonos ejecutando **JAvatar Móvil**, arriba el usuario Alice está editando su ***avatar*** personal, y abajo el usuario Bob visualiza su lista de contactos.
2. En la imagen central Alice selecciona la opción de Guardar los cambios realizados en su ***avatar*** personal. En este punto se envía el nuevo ***avatar*** a todos los contactos de Alice.
3. A la derecha puede observarse que la edición del ***avatar*** de Alice se refleja automáticamente en ambos dispositivos.

Figura 7: Secuencia de actualizaciones

2.2 Tecnologías utilizadas

Para el desarrollo de **JAvatar Móvil** hemos utilizado diferentes tecnologías de software libre: librería de persistencia Floggy [4], librería grafica LWUIT [5] y el Wireless Toolkit de SUN [6].

A su vez cabe destacar que **JAvatar Móvil** ha sido probado en múltiples emuladores de teléfonos celulares de forma satisfactoria.

2.2.1 Sun Java Wireless Toolkit

El Java Wireless Toolkit para CLDC (Connected Limited Device Configuration) es un conjunto de herramientas para desarrollar aplicaciones móviles basadas en CLDC y MIDP (Mobile Information Device Profile), para ser ejecutadas en teléfonos celulares, asistentes digitales de uso masivo, *palmtops* y otros dispositivos móviles pequeños. Incluye los entornos de emulación, facilidades de optimización de performance y configuración, documentación y ejemplos. Es posible usar el toolkit de manera separada o incorporarlo en diversos IDEs (Integrated Development Environments).

2.2.2 Persistencia: Floggy Persistence Framework

Floggy es un *framework* de persistencia de objetos para aplicaciones J2ME/MIDP. Su objetivo principal es abstraer al desarrollador de los detalles de persistencia de datos, reduciendo el esfuerzo de desarrollo y mantenimiento. Está basado en RMS (Record Management System), que la API de JAVA tradicionalmente utilizada para almacenar datos en aplicaciones MIDP, y si bien no es una base de datos, permite a los desarrolladores trabajar con comandos de persistencia de alto nivel. Floggy tiene la Licencia Apache versión 2.0, facilitando su uso en proyectos de software libre.

2.2.3 Gráficos: Lightweight User Interface Toolkit (LWUIT)

Lightweight User Interface Toolkit (LWUIT) es un conjunto de herramientas de componentes gráficos desarrollado por Sun Microsystems para facilitar el desarrollo de interfaces de usuario gráficas para JAVA ME. Está inspirado en Swing y, aunque es mucho más simple, soporta varias de sus facilidades, incluyendo *layout managers* y *look and feel plugabble*.

LWUIT tiene una clara separación entre el modelo, la vista y el controlador, y se basa en una arquitectura de jerarquía Componente/Contenedor. Los contenedores son componentes y pueden ser anidados para construir *layouts* elaborados. A su vez los componentes pueden ser representados tanto por temas y estilos externos como programáticamente por los desarrolladores. Además los componentes LWUIT tienen soporte para el manejo de eventos generados por pantallas táctiles.

La librería LWUIT se integra fácilmente al proyecto durante la etapa de desarrollo, no requiriendo de configuraciones especiales. Es una tecnología abierta, con su código fuente libremente accesible para uso individual o comercial.

2.2.4 SDS (Service Development Studio) de Ericsson

El IDE (Integrated Development Environment) que hemos utilizado es el SDS de Ericsson, versión 4.1 FD1 [7].

SDS es una herramienta integral para el desarrollo y pruebas *end-to-end*, tanto del lado cliente como del lado servidor, de aplicaciones convergentes sobre IP (IMS). SDS contiene un simulador de red IMS, y emuladores de servicios de comunicación (COSE). Utiliza las prácticas comunes y los estándares de la comunidad JAVA. Proporciona una API de alto nivel para ocultar al diseñador la complejidad de la red IMS y los dispositivos terminales.

SDS se basa en el entorno de desarrollo integrado Eclipse (IDE) y en el Wireless Toolkit de Sun (WTK). Además incluye un núcleo emulador de red IMS y también se puede configurar como un entorno de ejecución de servidor JavaEE/SIP para pruebas con usuarios directos o de servicios de mercado.

2.3 Nuestra Experiencia

JAvatar Móvil es el resultado de muchos debates sobre las aplicaciones posibles de realizar utilizando la nueva tecnología IMS.

Consideramos que la posibilidad de contar con una infraestructura de red convergente y de alta calidad, como es IMS, en la que es posible integrar servicios comúnmente brindados por redes heterogéneas, constituye un desafío para la construcción de aplicaciones novedosas sobre dispositivos móviles, en las que la comunicación y el compartir juegan un rol central [3]. Con esa idea en mente, surge la propuesta de realizar una aplicación de entretenimiento, en la que creamos nuestro propio ***avatar*** y lo podemos compartir con nuestros amigos.

Una vez que decidimos cuál sería la aplicación a desarrollarse, empezamos a investigar sobre las diferentes posibilidades de implementación.

En un comienzo pensamos que podíamos hacer nuestra aplicación para el cliente móvil que nos ofrece SDS, que es un celular con Symbian OS. Esta plataforma no fue diseñada para ejecutar **MIDlets**, sino aplicaciones JAVA utilizando las APIs del **Abstract Window Toolkit** (AWT). Con ello en mente, y como sabíamos que la aplicación requeriría una interfaz de usuario muy rica, buscamos diferentes librerías JAVA para la implementación de interfaces gráficas, entre las que consideramos JavaFX y la tecnología de gráficos vectoriales bidimensionales SVG para construir la interfaz. Siguiendo con la investigación, tuvimos que descartar la opción de utilizar JavaFX, ya que Symbian OS no tiene soporte para esta tecnología. De esta forma, decidimos hacer nuestra aplicación para Symbian OS, con SVG. Sin embargo, cuando empezamos a realizar las pruebas utilizando los ejemplos que trae el SDS, nos topamos con diversos tipos de problemas, entre ellos la dificultad para configurar IMS dentro del celular Symbian. Aún sin poder llegar a una solución para los problemas que estábamos teniendo, mediante una teleconferencia, los desarrolladores de IMS en Canadá nos advirtieron que la API para IMS de Symbian OS, llamada **IMS Client Platform** (ICP) estaba en camino de ser obsoleta, por lo cual abandonamos esta línea de desarrollo. En esa misma teleconferencia, nos recomendaron utilizar la tecnología

JME con la API IMS conocida como **IMS Java Client Utility** (IJCU). A diferencia de ICP, IJCU cumple con la especificación de la API de Servicios IMS definida en el **JSR 281**.

Dentro de los ejemplos que trae el SDS, existe IJCUchat, el cual probamos y tomamos como referencia para nuestro desarrollo.

Con esta nueva línea de desarrollo, nuevamente tuvimos que ponernos a investigar sobre librerías gráficas para JME. Y así descubrimos LWUIT, que nos permitió realizar una interfaz gráfica de una forma bastante sencilla y agradable. Utilizando LWUIT solucionamos la parte visual de nuestra aplicación. Lo siguiente que necesitábamos era tener persistencia en el celular, ya que teníamos que almacenar la información del usuario. Al comienzo teníamos la idea de guardar esa información en archivos XML, y por ello comenzamos a investigar las posibilidades para llevarlo a cabo, sin llegar a ningún resultado. Buscando librerías de persistencia para tecnologías móviles nos encontramos con Floggy, con el cual hicimos algunas pruebas y obtuvimos los resultados que deseábamos. Lo siguiente a resolver era poder compartir un ***avatar*** entre los usuarios de la red IMS. Buscando en la API de IJCU, vimos que era posible enviar por la conexión un arreglo de bytes. Y así, lo que hicimos es una prueba, modificando el IJCUChat para que en vez de enviar texto enviase un ***avatar***. Para poder lograrlo, fue necesario *serializar* y *des-serializar* el objeto ***avatar***, haciendo para ello un desarrollo propio. Dado que la API de IJCU no contempla el envío de objetos, no disponíamos de una manera estándar de especificar el envío de este tipo de contenido. Convirtiendo los objetos ***avatar*** en arreglos de bytes logramos que **JAvatar Móvil** finalmente comparta ***avatares*** entre contactos.
En **JAvatar Móvil** necesitábamos establecer múltiples sesiones P2P, una por cada contacto disponible (en-línea) y en los modelos analizados de los ejemplos disponibles en el SDS y de otros repositorios cada cliente utiliza una única sesión. La solución propuesta consistió en el uso de la librería de *threads* (Hilos) JAVA para disponer de múltiples sesiones concurrentes.

Considerando la interacción del usuario con **JAvatar Móvil** evaluamos diferentes emuladores de celulares. Teniendo en cuenta aspectos como la manipulación directa altamente usada en la interfaz de usuario, comprobamos que el uso de pantallas táctiles aumenta la usabilidad y amigabilidad de la aplicación. Incorporar la interacción mediante pantallas táctiles fue inmediato ya que la librería LWUIT provee soporte para celulares con esta característica.

3 Alcance de JAvatar Móvil

La visión de **JAvatar Móvil** es ser la aplicación para crear ***avatares*** que representen a los usuarios de la red IMS en una variedad de aplicaciones popularmente aceptadas. Por ejemplo, podría integrarse con distintas redes sociales como Facebook, Twitter o MySpace, aplicaciones de mensajería instantánea y llamadas de voz. Los cambios realizados en los ***avatares*** a través de **JAvatar Móvil** se propagarán en todas estas aplicaciones. De esta forma se lograría que un ***avatar*** identifique uniformemente a un mismo usuario en diferentes aplicaciones.

Pensamos que **JAvatar Móvil** podría utilizarse como un producto de marketing, incluyendo accesorios de diferentes marcas y empresas. Por ejemplo, si compramos un par de anteojos de una determinada marca, éstos podrían traer de regalo la versión **JAvatar** de dicho producto de forma tal que el ***avatar*** del usuario refleje la nueva adquisición. El ***avatar*** al compartirse estaría formando parte de un "boca a boca digital". A su vez, como los ***avatares*** tienen accesorios, éstos podrían ser adquiridos en un e-shop de forma similar a la compra de medios digitales, como un tema musical.

4 Conclusión

JAvatar Móvil es un trabajo experimental, actualmente en funcionamiento, que nos permitió ganar experiencia en el desarrollo de aplicaciones móviles en JAVA y en el uso de tecnologías de comunicación con buenas perspectivas de adopción en los próximos años tales como son las redes IMS.

Asimismo el modelo de comunicación utilizado en **JAvatar Móvil** sirve como prototipo inicial del cual es posible abstraer el mecanismo de comunicación entre múltiples sesiones *peer-to-peer* para redes IMS.

Por otro lado, la tarea de evaluar e integrar diferentes tecnologías para hacer un mejor aprovechamiento de los dispositivos móviles, como son Floggy y LWUIT, nos permitió tener una visión general de las tecnologías de código libre disponibles para llevar a cabo el desarrollo concreto de una aplicación móvil.

5 Referencias

[1] Ericsson IMS training in Argentina.
http://www.ericsson.com/developer/sub/articles/other_articles/0801002_argentina

[2] Secretaría de Extensión: Actividades Laboratorio de Software.
http://extension.info.unlp.edu.ar/convenios/index.html

[3] Capítulo "Developing IMS JAVA applications for Mobile Phones", Javier Díaz, Claudia Queiruga, Jorge Rosso, del libro "Mobile Web 2.0: Developing and Delivering Services to Mobile Phones". Editorial CRC Press, ISBN: 1439800820.

[4] Proyecto Floggy. http://floggy.sourceforge.net/

[5] Proyecto LWUIT. https://lwuit.dev.java.net/

[6] SUN Java Wireless Toolkit for CLDC. http://java.sun.com/products/sjwtoolkit/

[7] SDS 4.1 FD1.
http://www.ericsson.com/developer/sub/articles/other_articles/090220_SDS_FD1

[8] Eclipse IDE. http://www.eclipse.org/

[9] NetBeans IDE. http://netbeans.org/