API DE SERVICIOS PARA DISPOSITIVOS MÓVILES

J.A Plá [1], D.J Álvarez [2], L.K Zayas [1], D. Ilizástegui [1], Y. López [3], D. Buedo [2], Y. Viera [2]

[1] Departamento de Técnicas de Programación, Universidad de las Ciencias Informáticas, {jpla, zayas, dilizastegui}@uci.cu

[2]Departamento de Ingeniería y Gestión de Software, Universidad de las Ciencias Informáticas, {dalvarez, dbuedo, yviera}@uci.cu

[3] Policlínico Ernesto Che Guevara, Universidad de las Ciencias Informáticas, yanicabrera@uci.cu

RESUMEN / ***ABSTRACT***

Durante el transcurso de los últimos años los servicios de mensajería inalámbrica (SMS, MMS y WAP Push) han tenido un gran impacto en el mercado internacional. Haciendo uso de estos, se pueden generar una amplia variedad de servicios, permitiendo a las diferentes empresas de este sector obtener ganancias millonarias. En la actualidad existen varios protocolos de comunicación que permiten la transferencia de estos mensajes a los SMS Centers (SMSC) y MMS Centers (MMSC), en lo adelante Centros de Mensajería Inalámbrica.

Debido a la gran variedad de protocolos, así como las soluciones existentes en el mercado, han surgido ideas de cómo estandarizar dicha comunicación independientemente de cual sea el protocolo que se utilice para la transferencia de la información. En tal sentido surge en el año 2004 una especificación liderada principalmente por Nokia y Sun Microsystems conocida como SAMS.

El presente trabajo trata sobre una implementación de dicha especificación, así como el desarrollo de los drivers para los protocolos PDU (Protocol Data Unit) y SMPP (Short Message Peer to Peer) v3.3 que permitan la transferencia de mensajes SMS, MMS y WAP Push entre las aplicaciones desarrolladas por las Empresas Proveedoras de Servicios y los Centros de Mensajería Inalámbrica.

Se presenta además el desarrollo de elementos que no forman parte de la especificación pero que a su vez enriquecen la solución, dándole a la misma un mayor nivel de usabilidad.

Palabras claves: JSR-212, MMS, SAMS, SMS, WAP Push

SERVICES API FOR MOBILE DEVICES

During recent years wireless messaging services like SMS, MMS and WAP Push, have experimented a great impact on the international market. The use of these generates a wide variety of services, allowing different dedicated companies of the sector, become rich. There are currently several communications protocols that enable the transfer of these messages to SMS Centers (SMSC) and MMS Centers (MMSC) as forward Wireless Messaging Centers.

Due to the wide range of protocols, as well as existing solutions on the market, have emerged ideas on how to standardize such communication independent of the protocol being used for the transfer of the information. In this sense emerges in 2004 a specification led mainly by Nokia and Sun Microsystems known as SAMS.

This article deals with an implementation of this specification and development of drivers for the PDU and SMPP v3.3 protocols that allow the transfer of SMS, MMS and WAP Push between applications developed by Service Providers and Wireless Messaging Centers.

It also presents the developments of elements which are not part of the specification but instead turn to enriches the solution, giving it a higher level of usability.

Key words: JSR-212, MMS, SAMS, SMS, WAP Push

INTRODUCCIÓN

Durante el transcurso de los últimos lustros, las tecnologías de la informática y las comunicaciones han alcanzado un alto nivel de desarrollo en todo el mundo. Muchos han sido los avances experimentados en este sentido, así como los resultados obtenidos por investigadores de todo el planeta.

Como parte de este proceso evolutivo surge una novedosa vía de comunicación; las redes inalámbricas. Estas consisten en la no existencia de un medio físico para establecer la conexión (cables), lo cual brinda una mayor movilidad, permitiendo la conexión desde un área mucho más extensa a como era anteriormente. A partir de este momento se comienzan a

producir nuevos dispositivos, que facilitan en gran medida las comunicaciones y el trabajo diario de todos sus usuarios.

Años más tarde, surge un nuevo modelo de telefonía basado en esta misma tecnología, conocido como "telefonía celular". Este ha tenido una gran aceptación entre toda la población mundial, debido a las grandes facilidades que brinda con respecto al modelo tradicional. Las empresas de telecomunicaciones también han apostado fuertemente por su masificación, puesto que permite la prestación de servicios realizando pequeñas inversiones y disminuye considerablemente los costos de mantenimiento.

Producto de este auge y desarrollo alcanzado en tan corto plazo se comienzan a crear compañías que aprovechando las infraestructuras erigidas por los operadores de telefonía celular, se enfocan hacia la prestación de servicios. Estas instituciones han reportado en los últimos años ganancias millonarias, generando un nuevo modelo de negocio; la prestación de servicios de valor agregado.

Parte importante de estos servicios que brindan los VASP (del inglés: Value Added Service Provider) son servicios de mensajería, para lo cual resulta necesaria la comunicación con un Centro de Mensajería Inalámbrica, ente encargado de realizar las tareas de envío y recepción de mensajes desde y hacia los terminales móviles.

De ahí deriva que el objetivo general de este trabajo sea desarrollar un componente de software que permita el envío y recepción de mensajes independientemente de los protocolos de comunicación utilizados por los Centros de Mensajería.

Como objetivos específicos se plantea:

- Definir las interfaces que permitan la comunicación entre las aplicaciones desarrolladas por terceros y el componente de software a desarrollar.
- Permitir el envío y recepción de mensajes SMS (del inglés: Short Message Service), MMS (del inglés: Multimedia Message Service) y notificaciones WAP Push (del inglés: Wireless Application Protocol Push).
- Implementar protocolos de comunicación entre el componente y los Centros de Mensajería Inalámbrica.

En lo adelante, se describe cual fue el estándar seleccionado y sus características, así como las modificaciones que se implementaron para cumplir con los objetivos trazados en la investigación. Por otra parte se especifica, que protocolos de comunicación se seleccionaron y el por qué. El documento concluye con una descripción de los resultados alcanzados.

MATERIALES Y MÉTODOS

Como consecuencia del incremento a nivel mundial del número de usuarios de la telefonía móvil, los operadores se han enfocado hacia la creación de nuevos servicios que brinden un valor agregado a sus clientes.

Muchos de estos servicios son basados en la mensajería inalámbrica, por lo que resulta necesario un componente que permita la comunicación desde las aplicaciones que conforman el servicio y los centros de mensajería inalámbrica. A partir de ese momento comienzan a surgir alternativas que brindan soluciones parciales al problema, sin llegar a un punto de convergencia, donde las tareas de envío y recepción de mensajes fueran independientes de las características tecnológicas de los operadores.

En el año 2004 surge SAMS-M (del inglés: Server APIs for Mobile Services - Messaging), JSR 212 (del inglés: Java Specification Request) [1], especificación desarrollada principalmente por Nokia y Sun Microsystems que plantea una solución independiente de las particularidades de cada operador para el envío y recepción de SMS[2][3], MMS [4][5] y notificaciones WAP Push [6].

SAMS-M es independiente de los protocolos de red, el formato de datos y las plataformas para la comunicación con los centros de mensajería inalámbrica. Su arquitectura está basada en la utilización de drivers específicos para cada protocolo, de forma tal que estos puedan ser añadidos o removidos en tiempo de ejecución. Dichos drivers abstraen a los desarrolladores de todos los detalles de envió y recepción de información, ocultado las particularidades tecnológicas de cada operador.

Como parte del JSR 212 se proponen dos implementaciones básicas: una para la plataforma Java SE (Java Standard Edition) y otra para la plataforma Java EE (Java Enterprise Edition).

Como se muestra en la figura 1, la implementación de SAMS-M utilizando Java SE está compuesta por dos capas fundamentales: la capa de servicios, la cual provee todas las interfaces necesarias para la transferencia de mensajes desde y hacia las aplicaciones clientes y la capa de recursos que proporciona las interfaces para el desarrollo de los drivers específicos para cada protocolo de comunicación.

La segunda implementación de SAMS-M está basada en la plataforma Java EE. La figura 2 muestra una vista arquitectónica de los componentes del API y la interacción entre ellos en dicho entorno.

Utilizando Java EE, SAMS-M sigue estando constituido por las mismas dos capas planteadas en el caso anterior. La principal diferencia consiste entonces en que esta vez la capa de servicios se implementa utilizando la tecnología EJB (del inglés: Enterprise Java Beans) y la capa de recursos mediante JCA (del inglés: Java Connector Architecture).

La arquitectura bajo la cual fue concebida esta especificación hace que sea una solución robusta y adaptable a las disímiles características tecnológicas de cada uno de los operadores, facilitando la interacción entre los centros de mensajería de los mismos y las aplicaciones desarrolladas por terceros.

SAMS-M necesita de drivers para cada uno de los protocolos de comunicación, incrementando su valor a medida que se van incluyendo mayor número de estos. Como parte del desarrollo realizado se decide implementar un driver para el protocolo SMPP v3.3 [7][8][9] debido a que es uno de los protocolos más difundidos a nivel mundial y basándose también en el hecho de que el operador de telefonía celular en Cuba posee un centro de mensajería que utiliza dicho protocolo, además de otro para PDU [10] considerando la utilización de módems en

lugar de una conexión directa con centros de mensajería inalámbrica.

Para la implementación de dichos drivers se deben utilizar las interfaces que se proveen como parte de la capa de recursos. No obstante no existe un modelo o estilo arquitectónico previamente establecido para ello, sino que es de libre elección por parte del desarrollador.

RESULTADOS Y DISCUSIÓN

El JSR 212 define un API con dos implementaciones básicas: una utilizando Java SE y otra Java EE. Dichas plataformas utilizan el lenguaje de programación Java, lo cual constituye una importante restricción si se piensa en la cantidad de aplicaciones no desarrolladas bajo dicho lenguaje que necesitarían realizar tareas de envío y recepción de mensajes.

En tal contexto se decide agregar una tercera implementación al componente utilizando servicios Web, lo cual permitiría una mayor interoperabilidad entre tecnologías diferentes. Dicha característica brinda sin lugar a dudas un mayor grado de usabilidad al API desarrollada debido que se reduciría considerablemente la barrera impuesta por la tecnología utilizada para la implementación de las aplicaciones.

Otro aporte realizado durante del desarrollo de dicho componente es la inclusión de nuevas clases para el manejo de forma más sencilla de las notificaciones WAP Push. Dichos mensajes no son incluidos en la especificación debido a que estos pueden ser manipulados utilizando mensajes SMS, lo cual requiere de un conocimiento más profundo sobre el tema por parte del usuario final del componente. Al adicionar dichas clases se permite el manejo de forma sencilla de las notificaciones WAP Push, ocultando la lógica de cómo son creadas y manipuladas las mismas, lo cual facilita a usuarios no expertos la utilización de SAMS-M en sus desarrollos.

Durante la etapa de desarrollo se realizó además una modificación al diseño del API donde se permite el uso de varios archivos de configuración para establecer diferentes canales de recepción de mensajes en el caso de ser necesario.

El JSR-212 plantea una solución asincrónica para el envío y recepción de mensajes. Utilizando este mecanismo resulta un tanto complejo el manejo de los errores que pueden surgir durante los procesos antes mencionados. En tal sentido los autores del presente trabajo consideraron pertinente realizar otra implementación que fuera totalmente sincrónica para casos en los cuales sea necesario un tratamiento más detallado de los errores.

Estas funcionalidades constituyen un valor añadido a la implementación del estándar, incrementando el número de escenarios en que puede ser utilizado el componente.

CONCLUSIONES

A modo de conclusión se pueden establecer las siguientes afirmaciones:

- Se tomó como referencia el JSR 212 para la solución del problema planteado.
- El componente de software desarrollado brinda mecanismos para el envío y recepción de mensajes SMS, MMS y WAP Push.
- Como parte de la solución propuesta se desarrollaron servicios Web que permiten la utilización de este componente desde otras plataformas.
- Se adicionaron clases para simplificar el manejo de los WAP Push.
- Se realizó una implementación asincrónica para facilitar el tratamiento de los errores.
- Se realizó la implementación de dos drivers para la comunicación.

REFERENCIAS

1. **JAVA COMMUNITY PROCESS**: "SAMS-M. JSR 212. Server API for Mobile Services: Messaging - SAMS", disponible en: http://jcp.org/en/jsr/detail?id=212.
2. **TECHTARGET**: "Short Message", disponible en: http://searchmobilecomputing.techtarget.com/.
3. **PHONESCOOPE**: "MMS", disponible en: http://www.phonescoop.com/glossary/term.php?fid=43.
4. **TECHTARGET**: "Multimedia Messaging Service", disponible en: http://searchmobilecomputing.techtarget.com.
5. **PHONESCOOPE**: "SMS", disponible en: http://www.phonescoop.com/glossary/term.php?gid=86.
6. **AME INFO**: "Wap Push", disponible en: http://www.ameinfo.com/31971.html.
7. **GOOGLE CODE**: "JSMPP", disponible en: http://code.google.com/p/jsmpp.
8. **SMS FORUM**: "SMPP Specification", disponible en: http://www.smsforum.net.
9. **LOGICA**: "SMPP Open Source", disponible en: http://opensmpp.logica.com.
10. **LE BODIC, GWENAE**: "Mobile Messaging Technologies and Services SMS EMS and MMS", Alcatel, France, 2003. s.n., 2003.

AUTORES

José Antonio Plá Rodríguez, Ing. Ciencias Informáticas, profesor instructor, Universidad de las Ciencias Informáticas, Carretera a San Antonio de los Baños, km 2 ½, Boyeros, Ciudad de La Habana, Cuba, jpla@uci.cu.

Darien Jesús Álvarez de la Cruz, Ing. Ciencias Informáticas, profesor instructor, Universidad de las Ciencias Informáticas,

Carretera a San Antonio de los Baños, km 2 ½, Boyeros, Ciudad de La Habana, Cuba, dalvarez@uci.cu.

Lex Karel Zayas Hernández, Ing. Ciencias Informáticas, profesor instructor, Universidad de las Ciencias Informáticas, Carretera a San Antonio de los Baños, km 2 ½, Boyeros, Ciudad de La Habana, Cuba, lzayas@uci.cu.

Damián Ilizástegui Arriba, Ing. Ciencias Informáticas, profesor instructor, Universidad de las Ciencias Informáticas, Carretera a San Antonio de los Baños, km 2 ½, Boyeros, Ciudad de La Habana, Cuba, dilizastegui@uci.cu.

Yanisleydi Lopez Cabrera, Ing. Ciencias Informáticas, Policlínico Ernesto Che Guevara, Universidad de las Ciencias Informáticas, Carretera a San Antonio de los Baños, km 2 ½, Boyeros, Ciudad de La Habana, Cuba, yanicabrera@uci.cu.

Denys Buedo Hidalgo, Ing. Ciencias Informáticas, profesor instructor, Universidad de las Ciencias Informáticas, Carretera a San Antonio de los Baños, km 2 ½, Boyeros, Ciudad de La Habana, Cuba, dbuedo@uci.cu.

Yunisel Viera Vargas, Ing. Ciencias Informáticas, profesor instructor, Universidad de las Ciencias Informáticas, Carretera a San Antonio de los Baños, km 2 ½, Boyeros, Ciudad de La Habana, Cuba, yunisel@uci.cu.

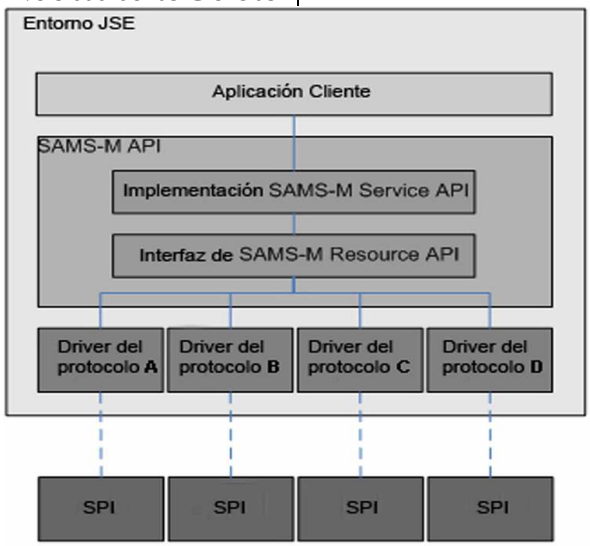

Figura 1. SAMS-M, entorno Java SE.

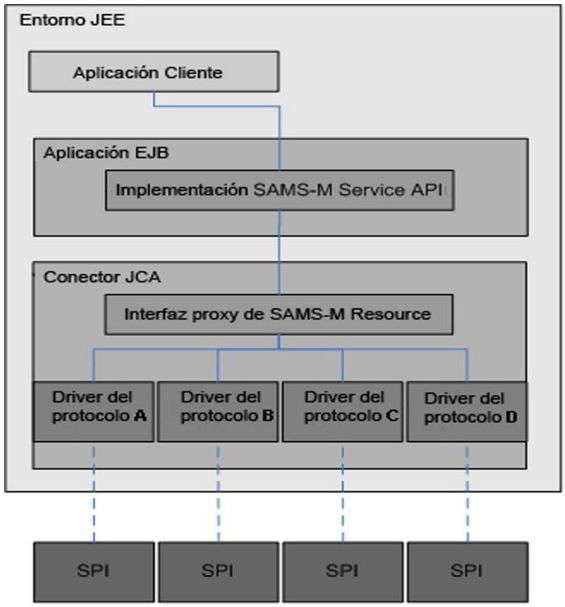

Figura 2. SAMS-M, entorno Java EE.