INTEGRACIÓN DE LAS APIs WMA Y PUSH REGISTRY DE LA PLATAFORMA J2ME PARA LA RECUPERACIÓN REMOTA DE INFORMACIÓN EN TELÉFONOS MÓVILES

INTEGRATION OF WMA AND PUSH REGISTRY APIs OF THE J2ME PLATFORM FOR REMOTE INFORMATION RETRIEVAL IN MOBILE PHONES

AUTOR
ALICIA MARTÍNEZ REBOLLAR
Doctora
*Cenidet
amartinez@cenidet.edu.mx
MÉXICO

AUTOR
WILFRIDO CAMPOS FRANCISCO
Maestro
*Cenidet
mcwilfrido11c@cenidet.edu.mx
MÉXICO

INSTITUCIÓN
*CENTRO NACIONAL DE INVESTIGACIÓN Y DESARROLLO TECNOLÓGICO
CENIDET
Centro de Investigación Público
Interior Internado Palmira S/N.
MÉXICO

RECEPCIÓN: Diciembre 6 de 2011 **ACEPTACIÓN:** Febrero 2 de 2012

TEMÁTICA: Acceso y conectividad inalámbrica, Sistemas inalámbricos y móviles, Comunicaciones móviles, Teleaplicaciones.

TIPO DE ARTÍCULO: Artículo de Investigación Científica y Tecnológica.

RESUMEN ANALÍTICO

Uno de las tendencias actuales en tecnología es el uso de dispositivos móviles como herramienta auxiliar en el trabajo de los usuarios. De esta forma, los usuarios almacenan información valiosa en sus teléfonos móviles, la cual debe poder ser accedida en forma eficiente. En este artículo se presenta una nueva API (del inglés, Application Programming Interface) para teléfonos móviles denominada Arec (API de RECuperación), la cual permite recuperar de forma remota la información (listas de contactos fotos, videos y música) almacenada en un teléfono móvil. Esta API fue desarrollada como resultado de la integración de las APIs WMA y Push Registry, de la plataforma J2ME (Java 2 Micro Edition). La aplicación ayuda a los programadores a desarrollar rápidamente nuevas aplicaciones con una arquitectura cliente/servidor, ya que la misma API cuenta con interfaces y métodos para la programación del lado del cliente y del servidor. Además la API permite hacer frente a los problemas de acceso a gran distancia, debido a la utilización del Servicio de Mensajes Cortos (SMS), brindando la ventaja de acceder y recuperar información desde cualquier lugar en el que haya cobertura de red de telefonía móvil. Además, su funcionamiento es independiente del proveedor del servicio. Es importante mencionar que el teléfono móvil que desempeñe la función de servidor, no necesita de la intervención del usuario para aceptar y contestar las peticiones hechas por el equipo cliente. Con el objetivo de demostrar la utilización de la API se desarrollo un caso de estudio en el cual se muestra la recuperación remota de uno o varios contactos que coinciden con un criterio de búsqueda.

PALABRAS CLAVES: API Arec, Cómputo Móvil, J2ME

ANALYTICAL SUMMARY

One of the current trends in software technology is the use of mobile computing as an auxiliary tool in the users´ work. In this context, the mobile devices are used to store valuable information for the users, which should be accessed in an efficient way. This paper presents a new API (Application Programming Interface) for mobile phones called Arec (API of RECovery), which allows remote retrieval of information stored on a mobile phone (contact lists photos, videos and music). This API was developed as result of the integration of WMA and Push Registry APIs of the J2ME platform (Java 2 Micro Edition. The application helps developers to quickly develop new applications with client/server architecture, since the API itself provides interfaces and methods for programming the client side and the server side. Furthermore, the API allows addressing the problems with long distance access because it uses the Short Message Service (SMS), providing the advantage of accessing and retrieving information from any place in the phone network. Moreover, this API function is independent of the service provider. It is important to mention that mobile phone playing the role of server does not need the interventions of the user to accept and answer the requests made by the client phone. In order to validate the use of the proposed API a case study was developed that permits remote recovering of several phone contacts that match the search criteria.

KEYWORDS: API Arec, Mobile phone, J2ME

INTRODUCCIÓN

Hoy en día, el uso del teléfono móvil se ha incrementado considerablemente. En la última década, sólo la sociedad mexicana ha rebasado los 94 millones de usuarios registrados [1]; ésto revela la gran necesidad de la población por mantenerse comunicado en todo momento. A su vez, los costos de los equipos y los servicios han disminuido, logrando así, que sean cada día más las personas que puedan acceder a estos dispositivos.

Actualmente, los teléfonos móviles además de ser usados como medio de comunicación, también ofrecen servicios para: a) almacenamiento de una agenda telefónica, b) captura de imágenes, c) reproducción de música, etc. [2]

Debido al uso de estos nuevos servicios, los usuarios llegan a tener una gran cantidad de información en su teléfono *móvil*, convirtiéndolos en unidades de almacenamiento masivo. Derivado de lo anterior, se origina una dependencia de los usuarios hacia sus teléfonos móviles cuando tienen la necesidad de utilizar su información (lista de contactos, fotografías, música, etc.). Esto puede ser originado porque el usuario haya dejado olvidado en algún otro lugar su teléfono móvil, o porque lo haya perdido.

Con el objetivo de ofrecer una alternativa para recuperar remotamente la información contenida en un teléfono móvil, en este artículo se muestra el desarrollo de una nueva API denominada *Arec* (API de RECuperación), que al ser implementada por los programadores de teléfonos móviles, éstos pueden desarrollar aplicaciones fácilmente basadas en la arquitectura Cliente/Servidor para recuperar información de otro teléfono móvil. Por ejemplo: listas de contactos, fotos, videos, etc. Esta recuperación remota podrá hacerse a gran distancia debido al uso del sistema de mensajes cortos (SMS).

Esta nueva API desarrollada en la plataforma J2ME, puede funcionar en los teléfonos móviles que incorporan la máquina virtual de java, particularmente la denominada KVM (*Kilobyte Virtual Machine*) para dispositivos móviles.

Este artículo se encuentra organizado de la siguiente manera: En la sección 2, se explican los fundamentos teóricos; en la sección 3 se detalla la integración de las *APIs WMA y Push Registry*; la sección 4 muestra un caso de estudio: la aplicación *Arec Phonebook* implementando la nueva API *Arec*. Finalmente, en la sección 5 se detallan las conclusiones y los trabajos futuros.

1. FUNDAMENTOS TEÓRICOS

A continuación se detallan los conceptos y la plataforma de desarrollo en la que se basa la nueva API *Arec*.

1.1 CÓMPUTO MÓVIL

El cómputo móvil [3] ha crecido vertiginosamente debido a la gran cantidad de dispositivos móviles que se han desarrollado en los últimos años, haciendo uso de la parte computacional para llevar a cabo sus funciones.

Dentro de la gran cantidad de aplicaciones podemos enumerar:

- La realización de ventas directa
- Servicio de asesoría a clientes en tiempo rea
- Administración de sucursale
- Trabajo en grupo desde cualquier luga
- Oficinas móvile
- Videovigilanc
- etc.

1.1.1 Dispositivos móviles

Los dispositivos móviles, también conocidos como *handheld* (computadoras de mano, en español), son computadoras que debido a su tamaño son fáciles de ser transportadas, incluso en el bolsillo de una persona. Cuentan con limitadas capacidades de procesamiento y memoria, además de poseer pantallas y botones pequeños [4]

Algunos ejemplos de dispositivos móviles son: computadoras portátiles, teléfonos móviles, teléfonos inteligentes (*Smartphones*), PDA (del inglés, Personal Digital Assistant), etc.
De acuerdo a su nivel de funcionalidad estos dispositivos se clasifican en:

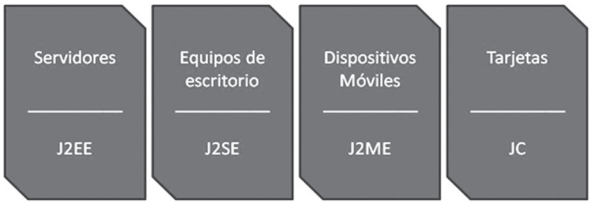
- **Dispositivo Móvil de Datos Limitados:** De pantallas pequeñas, basadas en solo texto y servicio de mensajes cortos (SMS, Short Message Service).
- **Dispositivo Móvil de Datos Básicos:** Pantalla de mediano tamaño y a color, reproducción de música, navegador web y cliente de e-mail
- **Dispositivo Móvil de Datos Mejorados:** Además de los servicios anteriores, pantallas de medianas a grandes e incluyen aplicaciones corporativas.

1.2 PLATAFORMA JAVA 2 EDICIÓN MICRO

La plataforma Java 2 Edición Micro (J2ME por sus siglas en inglés) es una edición de la plataforma Java, desarrollado por Sun Microsystem, está orientada a los dispositivos móviles [5]

En la Figura 1, se muestras las diferentes ediciones de la Plataforma Java en las cuales se detallan las arquitecturas computacionales en las que son utilizadas.

FIGURA 1. Ediciones de la plataforma Java.

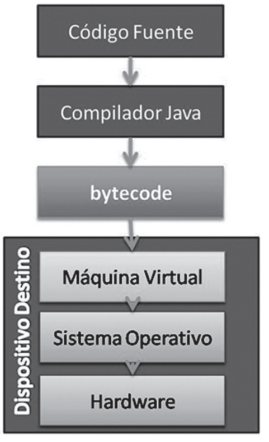


La plataforma Java ofrece un conjunto de ventajas a los programas que son desarrollados bajo este lenguaje de programación [6] A continuación, se enlistan algunas de estas ventajas:

- **Compatibilidad entre plataformas:** Se refiere a que una aplicación desarrollada con J2ME, puede ejecutarse en una amplia variedad de dispositivos móviles de diferentes modelos y fabricantes.
- **Contenido dinámico:** Con J2ME se pueden desarrollar aplicaciones dinámicas, debido a las APIs que se implementan para contenido multimedia.
- **Fuerte seguridad:** La seguridad es una de las cosas que más concierne a Java, una aplicación escrita con J2ME no puede acceder al hardware del dispositivo u otros recursos; evitando con esto la creación de virus, caballos de Troya, así como otros códigos maliciosos.

Para lograr la compatibilidad entre plataformas de una aplicación desarrollada en J2ME, es necesario compilar su código fuente a fin de crear un código llamado "java-bytecode" (archivo con extensión .jar), que es un código intermedio entre el código fuente y el código máquina entendible por el dispositivo destino [7] La Figura 2 representa a java-bytecode listo para ser ejecutado por la máquina virtual del dispositivo destino.

FIGURA 2. Java-Bytecode producto de la compilación del código fuente, el cual está listo para ser ejecutado sobre la máquina virtual de un dispositivo destino.

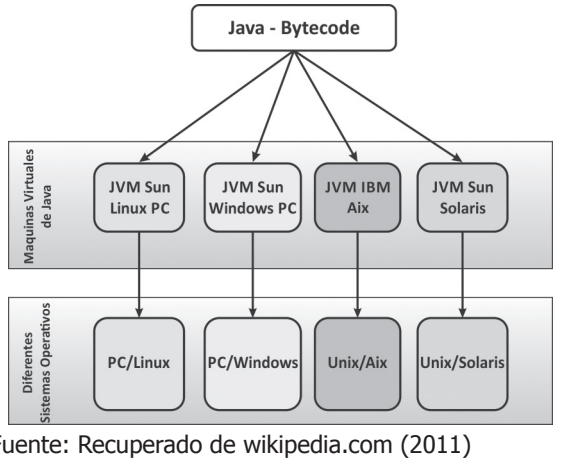


1.2.1 La Máquina Virtual KMV

La KVM (Kilo Virtual Machine) [8] es una máquina virtual Java compacta y portable específicamente diseñada para ser la base de desarrollo en dispositivos pequeños y de recursos limitados. Su objetivo es proporcionar un entorno de ejecución independiente de la plataforma de hardware y del sistema operativo, ocultando los detalles de la plataforma subyacente y permitiendo que un *java-bytecode* se ejecute siempre de la misma forma sobre cualquier plataforma.

La Figura 3 muestra como un mismo java-bytecode puede ser ejecutado en las diferentes máquinas virtuales de Java que Sun Microsystem ha desarrollado.

FIGURA 3. Máquinas Virtuales de Java.



Fuente: Recuperado de wikipedia.com (2011)

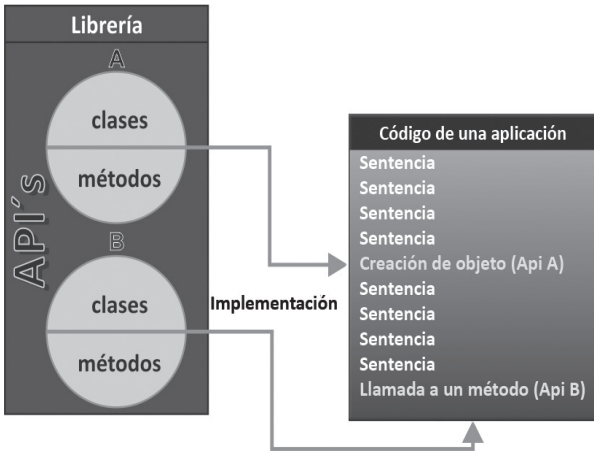
1.3 INTERFAZ DE PROGRAMACIÓN DE APLICACIONES (API)

Una API es un conjunto de clases y funciones escritas en un lenguaje de programación que residen en una biblioteca o librería [9]

Estas bibliotecas pueden ser implementadas en cualquier aplicación como una capa de abstracción. De esta manera los programadores hacen uso de la funcionalidad de las APIs, ya sea creando objetos o haciendo llamadas a las funciones contenidas dentro de las mismas, evitando con esto, reescribir todas las instrucciones de su aplicación desde el principio, favoreciendo el desarrollo más rápido de nuevas aplicaciones.

En la Figura 4 se ejemplifica una librería que contiene dos APIs, a las cuales se les ha denominado *A* y *B*, respectivamente. Cada una de ellas contiene sus propias clases y métodos. Cuando la librería se implementa en el código de una aplicación, se pueden crear objetos y hacer llamadas a los métodos de cualquiera de las APIs.

FIGURA 4. Implementación de APIs en una Aplicación



1.4 APIs DE J2ME

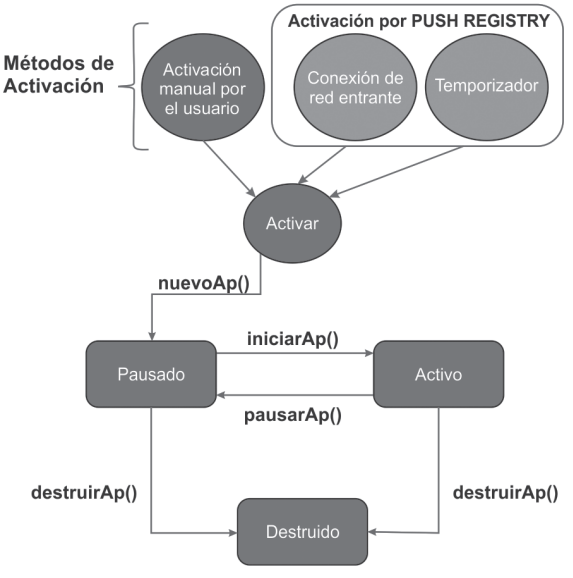
J2ME incluye un conjunto reducido de APIs para el desarrollo de aplicaciones móviles que van desde APIs de interfaz de usuario, de ciclo de vida del programa, almacenamiento persistente, juegos, trabajo en red y multimedia. A continuación se detallan las principales funciones de las APIs Push Registry y WMA que fueron integradas para el desarrollo de la nueva *API Arec* [10]

1.4.1 La API Push Registry

Antes de explicar la funcionalidad de la API Push Registry, es importante diferenciarla del PUSH REGISTRY que forma parte del Sistema Manejador de Aplicaciones (AMS por sus siglas en inglés) en un dispositivo móvil, dado que comparten el mismo nombre. En un dispositivo móvil el AMS es responsable del ciclo de vida de cada *MIDlet* (nombre que se le dá a los programas desarrollados para dispositivos móviles), desde su inicialización hasta su destrucción [11]

El PUSH REGISTRY como parte del AMS, guarda un registro de las dos vías por las cuales un MIDlet puede ser inicializado. Estas vías son: 1) activación causada por conexiones de red entrantes, 2) temporizadores.
En la Figura 5, se muestra el ciclo de vida de un *MIDlet,* además se han enmarcado las dos nuevas vías de activación.

FIGURA 5. Activación de un MIDlet vía PUSH REGISTRY



Fuente: Recuperado de oracle.com y traducido al español (2011).

Por otro lado, la API Push Registry se encapsula dentro de la clase *javax.microedition.io.PushRegistry* y proporciona los métodos para modificar los registros de las conexiones de red entrantes o alarmas de temporización almacenados en el PUSH REGISTY mediante un archivo descriptor (archivo con extensión .jad) que acompaña al MIDlet [12]

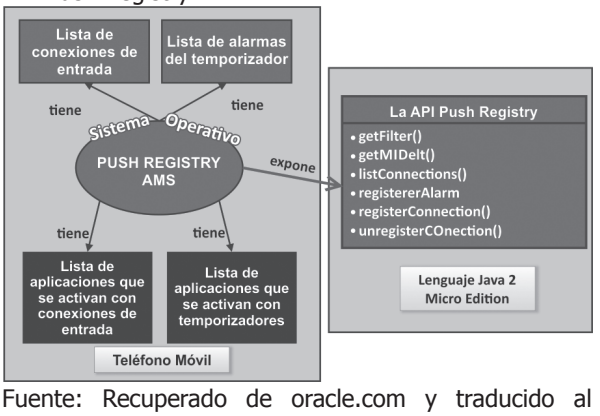
En la Figura 6, se muestra la diferencia entre el PUSH REGISTRY como parte del AMS en el teléfono móvil y la API Push Registry perteneciente a la plataforma J2ME.

El registro en el PUSH REGISTRY de las aplicaciones que se ejecutan automáticamente se lleva a cabo durante la

instalación del MIDlet. Para ello, en el archivo descriptor (archivo con extensión .jad), deberá haber una entrada con el siguiente formato:

MIDlet-Push-1: sms://:<puerto>,<Midlet-ClassName>,<>*

FIGURA 6. PUSH REGISTRY como parte del AMS y la API Push Registry.



Fuente: Recuperado de oracle.com y traducido al español (2011)

Esta sentencia indica que a la llegada de un conexión entrante, se ejecutará el constructor del MIDlet que contenga la clase <MidletClassName>, independientemente del nombre de la aplicación cliente que envía el mensaje <*>. La Tabla 1, enlista los métodos de la API Push Registry.

TABLA 1. Los Métodos de la API javax.microedition.io. PushRegistry.

La API Push Registry (javax.microedition.io.PushRegistry)
• getFilter() • getMIDelt() • listConnections() • registerAlarm • registerConnection() • unregisterConnection()

1.4.2 API WMA

La última versión de la API WMA 2.0 está orientada a la manipulación, envío y recepción de mensajes ya sea de tipo texto o binarios, a través del Servicio de Mensajes Cortos (SMS).
En J2ME las interfaces para la API de mensajería han sido definidas en el paquete javax.wireless.messaging. La interface base que se implementa para los mensajes, se llama Message y para definir el tipo de los mismos se utilizan las subinterfaces TextMessage para cadenas de texto y BinaryMessage para el arreglo de bytes. En la Tabla 2 se enlistas las interfaces y los *métodos* que conforman la API WMA [13]

TABLA 2. Las Interfaces y Métodos de la API WMA.

La API WMA (javax.wireless.messaging)	
Interfaces	**Métodos**
Message BynaryMessage TextMessage MessageConnection MessageListener	getAddress(), getTimestamp(), set Address(), getPayloadText(), setPayloadData(), getPayloadText(), setPayloadText(), newMessage(), receive(), send(), setMessagenListener(), numberofSegments()

El siguiente código de ejemplo, envía un mensaje de texto del lado del cliente utilizando el identificador 12345.

```
try {
        String telefono = "sms://7471333310:12345";
        MessageConnection conn =
        (MessageConnection) Connector.
        open(telefono);
        TextMessage msg =
        (TextMessage)conn.
        newMessage(MessageConnection.TEXT_
        MESSAGE);

        msg.setPayloadText("Mensaje de Prueba");

        conn.send(msg);

} catch (Exception e)
{
  //Funciones para manipular el error
}
```

Por el lado del servidor, el siguiente código de ejemplo recibe el mensaje que llegó por el identificador 12345.

```
try {
        String addr = "sms://:12345";
        MessageConnection conn =
        (MessageConnection) Connector.open(addr);
        Message msg = null;

        while (someExitCondition) {
        // Esperar la llegada de un sms
        msg = conn.receive();
        // Mensaje recibido
        if (msg instanceof TextMessage)
        {
                TextMessage tmsg = (TextMessage)
                msg;
                String receivedText = tmsg.
                getPayloadText();
```

```
                System.out.print("Mensaje
                recibido:" +receivedText);
            } else
            {
             //Recibe un mensaje que no es binario
            }
} catch (Exception e) //
{
  //Funciones para manipular el error
   }
```

1.5 SEGURIDAD EN LOS DISPOSITIVOS MÓVILES

Un dispositivo móvil dispone de datos y funciones privilegiadas, por ejemplo: la agenda de contactos, las funciones de envío y recepción de mensajes, así como la realización de llamadas telefónicas, entre otros. Estas funciones requiere la autorización explícita del usuario para llevarse a cabo. El acceso y la manipulación de estas funciones privilegiadas se hace a través de una aplicación móvil y dependiendo del nivel de acceso que tiene hacia el dispositivo pueden ser clasificadas en:

a) **Aplicaciones móviles de NO Confianza:** Las cuales no requieren del permiso explícito del usuario para acceder libremente a las APIS de gestión de almacenamiento, a la modificación del ciclo de vida de una aplicación y los controles de reproducción multimedia.

b) **Aplicaciones móviles de Confianza:** Las cuales requieren de la autorización del usuario para acceder, por ejemplo a la lista de contactos, enviar y recibir mensajes así como la realización de llamadas telefónicas.

En esta propuesta, la seguridad se maneja utilizando una *aplicación móvil de confianza*, debido a la necesidad de acceder a los datos y funciones privilegiadas del dispositivo móvil. Por lo tanto, los permisos de acceso se establecen en un archivo descriptor (archivo con extensión .jad) mediante la clase *MIDlet-Permissions*, en ella se permite o se deniega el acceso a las áreas restringidas mencionadas anteriormente. Por ejemplo: la llamada a la función *Javax.microedition.io.Conector de la clase MIDlet-Permissions* permite abrir una conexión en el protocolo de mensajería, así mismo *MessageConnection.send* habilita el envío de mensajes por un puerto determinado y *MessageConnectio.receive* permite a la aplicación recibir mensajes a través de un identificador (puerto) específico.

2. INTEGRACIÓN DE LAS APIS

El objetivo principal de este artículo, se centra en el desarrollo de una nueva API, que integra las funcionalidades y beneficios de las APIs Push Registry y WMA, a fin de proporcionar a los programadores de teléfonos móviles una nueva API que les permita desarrollar aplicaciones Cliente/Servidor, con el propósito de recuperar la información (lista de contactos, fotos, videos, etc.) de un teléfono móvil a otro.

2.1 LA API DE RECUPERACIÓN AREC

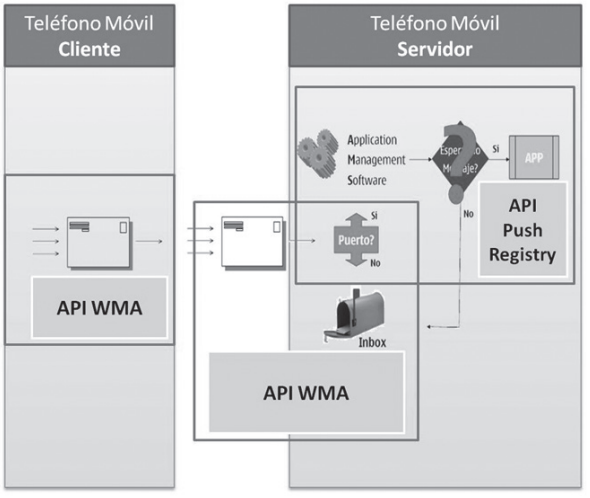
La nueva API propuesta es llamada *Arec* (API de RECuperación) y tiene como objetivo proporcionar interfaces y métodos necesarios para crear aplicaciones cliente/servidor, que recuperen de manera remota la información contenida en un teléfono móvil a través del servicio de mensajes cortos (SMS), desde cualquier lugar en el que haya cobertura de red de telefonía móvil.

La Figura 7, muestra la integración de las APIs Push Registry y WMA. En ella se observa la petición vía SMS hecha por el teléfono móvil cliente al teléfono móvil servidor, en donde se identifica si el mensaje recibido deberá ejecutar una aplicación o en caso contrario será enviando a la bandeja de entrada (*inbox*).

A continuación se detalla la lógica del funcionamiento de las aplicaciones que implementen la API Arec en los teléfonos móviles cliente y servidor.

1) *Cliente y Servidor:* Se instala la aplicación y automáticamente se reserva un número de puerto en el PUSH REGISTRY del AMS.
2) *Servidor:* Se especifica la contraseña de seguridad.
3) *Cliente:* Se realiza la petición especificado puerto, contraseña y el objeto a recuperar.
4) *Servidor:* Se recibe la petición y sólo se atiende al validar el puerto y contraseña correctos.
5) *Servidor:* Se realiza la búsqueda de la información que cumpla con los criterios solicitados.
6) *Servidor:* Se forma un nuevo mensaje con el resultado de la búsqueda o con un mensaje de *"no encontrado".*
7) *Servidor:* Se envía el mensaje resultante de la búsqueda al teléfono móvil cliente.
8) *Cliente:* Se recibe y deposita en la bandeja de entrada.

FIGURA 7. Activación de una aplicación móvil por la llegada de un SMS.



2.1.1 Implementación del lado del Servidor

La aplicación servidor que implemente la API Arec está acompañada de un archivo descriptor (archivo con extención .jad). Es importante mencionar que en un archivo descriptor, se definen los permisos que debe de tener un aplicación móvil para ejecutarse (*MIDlet-Permissions*) y también las conexiones entrantes que pueden inicializarlo (*MIDlet-Push-n*). Estos permisos quedan establecidos en el PUSH REGISTRY (que forma parte del AMS) durante la instalación de la aplicación, reservando un número de identificador (puerto) y el nombre del constructor de la aplicación que se ejecutará automáticamente.

El siguiente código muestra el método *setPushRegistry* de la API Arec, el cual crea un archivo descriptor llamado Arec.jad, e incorpora dentro de él las líneas: a) javax.microedition.io.PushRegistry; b) MIDlet-Push-1: sms://:puerto,MIDletServidor,*. En ellas se especifican el número de puerto en el parámetro *pto* y el nombre del MIDlet en el parámetro *mid*.

```
import javax.microedition.rms.*;

public void setPushRegistry(String pto, String mid){
        static final String BD = "Arec.jad";
        try
        {
          rs = RecordStore.openRecordStore(BD,
                true);
          guardaRegistro(rs,"
            javax.microedition.io.PushRegistry ");
          guardaRegistro(rs," MIDlet-Push-1:
            sms://:"+pto"+","+mid+",*");
          rs.closeRecordStore();

        }
        catch( RecordStoreException error )
        {
          //Funciones para manipular el error
        }
}// setPushRegistry
```

Una vez establecidos estos permisos, la aplicación servidor está lista para recibir peticiones de una aplicación cliente vía SMS, por lo tanto, cuando llega un mensaje con el número de puerto previamente definido, el método *startApp* será invocado e iniciará automáticamente la aplicación servidor.

En el siguiente código se detalla la recepción de un mensaje de texto que contiene en el parámetro PUERTO_SMS el número de puerto reservado para la aplicación servidor.

```
//Haciendo uso del Push Registry
protected void startApp()
{
 String direccion = "sms://:" +PUERTO_SMS;
 if( con == null )
 {

 try{
      con = (MessageConnection)
      Connector.open( direccion);
      con.setMessageListener( this );
     }catch( IOException e ) {}
 }
 conexiones =
 PushRegistry.listConnections( true );
 queryPhoneBookReception();
 notifyDestroyed();
}

//hacienda uso de la wma

public void queryPhoneBookReception()
{
try {
    mensaje = con.receive();
    if( mensaje != null && mensaje
    instanceof TextMessage )
    {
     String origen = mensaje.getAddress();
     String nombre = ((TextMessage)
     mensaje).getPayloadText();
     String clave =
     recuperarClave(nombre);
     String nombre =
     recuperarNombre(nombre);
```

```
String dato =
recuperarClaveAlmacenada();

  if(dato.equals(clave))
  {
   BuscarContactobuscarContacto =
        new BuscarContacto(nombre,this);
   String mensajeRespuesta =
   buscarContacto.getContactos();
   try{
   enviarRespuesta(mensajeRespuesta);
   }catch(Exception t)
       {//funciones para el error }
     }
 }

} catch( Exception e ) {}

} // queryPhoneBookReception
```

2.1.2 Implementación del lado del cliente

Como se pudo observar en el archivo descriptor del servidor, ya está asignado un puerto para recibir peticiones de una aplicación cliente. El siguiente código de ejemplo muestra el envío de un mensaje de texto desde la aplicación cliente a la aplicación servidor, adjuntando en el parámetro puerto el mismo número de puerto que tiene definido el servidor: Finalmente en el parámetro mensaje se tiene los detalles de la información solicitada y la contraseña de seguridad.

```
public void sendQuery(){
 try{
    String addr =
      "sms://"+txtTelefono.getString()
      +":puerto";
    String texto =txtNombre.getString();
    MessageConnection conn =
      (MessageConnection)
       Connector.open(addr);
    TextMessage mensaje =
      (TextMessage)conn.newMessage
      (MessageConnection.TEXT_MESSAGE);
   mensaje.setPayloadText
      (texto+":"+txtClave.getString());
    conn.send(mensaje);
     }catch(Exception t)
    {
     System.out.println("Ocurrio un
        error al enviar el mensaje");
    }
  } // fin de sendQuery()
```

3. CASO DE ESTUDIO: LA APLICACIÓN AREC PHONEBOOK

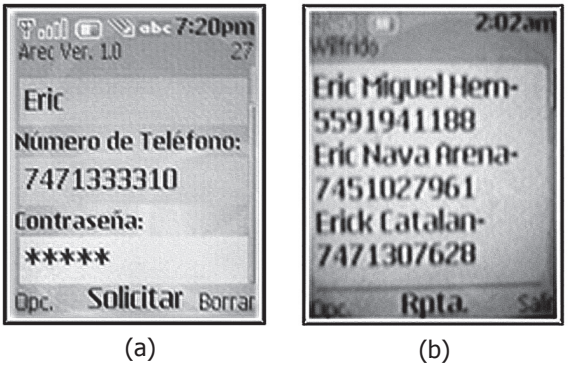
Uno de los servicios que ofrece un teléfono móvil es el de agenda telefónica [14], cuyo objetivo es almacenar y tener acceso inmediato a gran cantidad de números telefónicos, evitando así la necesidad de tener que memorizarlos; esto ha ocasionado una gran dependencia al teléfono móvil que contiene esta información.

Arec PhoneBook es un proyecto que implementa la API AREC, con la finalidad de recuperar de forma remota la lista de contactos, de otro teléfono móvil. La recuperación remota se realiza utilizando el Servicio de Mensajes Cortos (SMS). Arec PhoneBook funciona desde cualquier lugar donde haya cobertura de la red telefónica móvil. Por otro lado, la aplicación servidor no necesita la intervención del usuario para aceptar y contestar las peticiones hechas por la aplicación cliente.

Una petición se realiza introduciendo en la aplicación cliente el criterio de búsqueda, el número telefónico del teléfono móvil del cual se desea recuperar la información y la contraseña de seguridad para validar que el teléfono móvil cliente está autorizado para realizar la solicitud.

En la Figura 8(a), se muestra la interface de Arec PhoneBook Cliente realizando la petición al teléfono móvil servidor y en la Figura 8(b) se observa el resultado recibido de Arec PhoneBook Servidor.

FIGURA 8. Arec PhoneBook en ejecución.



(a) (b)

3.1 PRUEBAS REALIZADAS

Las pruebas llevadas a cabo para comprobar la funcionalidad de proyecto Arec PhoneBook presentado en este articulo se ejecutaron en un teléfono móvil de la marca Nokia modelo 2760 como servidor, el cual contiene una agenda telefónica con 31 contactos (ver Tabla 3) de los cuales se muestran sólo algunos de ellos por cuestiones de espacio. Las peticiones de información se hicieron utilizando un teléfono Nokia modelo 1650.

TABLA 3. Agenda telefónica de Arec PhoneBook Servidor.

Teléfono	Nombre	Teléfono
	Adolfo	7471345632
	Adriana Téc	5591941188
	Alberto Flor	7451027961
	Alex Cenidet	7471307628
	Almazzo EU	7444878372
	Ángel	7444965443
	Eric Miguel	5591941188
	Eric Nava	7451027961
	Erick Catalán	7471307628
	Felipe	5512482345
	Germán	7445245587
	Gilberto	7471254632
	Guadalupe	7444698723
	...	...

Se realizaron tres pruebas en las cuales el criterio de búsqueda se utilizó de la siguiente manera:

1. *Utilizando una Letra.* Para recuperar los contactos que contengan la letra especificada en cualquier parte de su nombre.

2. *Utilizando una Palabra.* Para recuperar los contactos que contengan la palabra completa en cualquier parte de su nombre.

3. *Utilizando una Cadena Vacía.* Para recuperar la lista de contactos completa.

A continuación se detallan los resultados obtenidos de las peticiones realizadas al teléfono móvil.

3.1.1 Utilizando solo una letra

En esta petición se espera recuperar aquellos contactos que contengan la letra "**N**" en cualquier parte de su nombre. Los datos introducidos en la aplicación cliente son los siguientes:

Criterio de Búsqueda: N
Número de Teléfono: 7471333310
Contraseña: 12345 (oculta por asteriscos)

La cantidad de contactos recuperados fueron 8 y se muestran en la Tabla 4.

TABLA 4. Resultados obtenidos con el criterio de búsqueda "N".

Nombre	Teléfono
Adria**n**a Téc	5591941188
Alex Ce**n**idet	7471307628
Á**n**gel	7444965443
Eric **N**ava	7451027961
Erick Catalá**n**	7471307628
Germá**n**	7445245587
Isaura Pi**n**eda	7445685247
Ramó**n**	7471658368

3.1.2 Utilizando una palabra

En esta petición se espera recuperar aquellos contactos que contengan la letra "**Eric**" en cualquier parte de su nombre. Los datos introducidos en la aplicación cliente son los siguientes:

Criterio de Búsqueda: Eric
Número de Teléfono: 7471333310
Contraseña: 12345 (oculta por asteriscos)

La cantidad de contactos recuperados fueron 3 y son mostrados en la Tabla 5.

TABLA 5. Resultados obtenidos del criterio de búsqueda "Eric".

Nombre	Teléfono
Eric Miguel Hernández	5591941188
Eric Nava	7451027961
Erick Catalán	7471307628

3.1.3 Recuperando la lista de contactos completa

En esta petición se espera recuperar la lista de contactos completa, por lo tanto el criterio de búsqueda es una cadena vacía ("'"). Los criterios establecidos en la aplicación cliente son los siguientes:

Criterio de Búsqueda:
Número de Teléfono: 7471333310
Contraseña: 12345 (oculta por asteriscos)

Para esta petición se generaron 6 segmentos de mensajes cortos de 170 caracteres cada uno, y por lo tanto se realizaron 6 envíos de mensajes cortos.

La cantidad de contactos recuperados fueron exactamente los 31 contactos mostrados en la Tabla 3.

Actualmente, la aplicación ha sido probada en teléfonos móviles de la marca Nokia, figurando entre ellos los modelos Nokia 1650, 3109, 6020, 6021, 6030, 6070, 6080 y 6101, sin embargo, se continua la búsqueda en más modelos para continuar haciendo pruebas en teléfonos móviles.

4. CONCLUSIONES Y TRABAJOS FUTUROS

Este artículo muestra el desarrollo de una nueva API llamada *AREC* para recuperar automáticamente y de forma remota, información contenida en un teléfono móvil a través del Servicio de Mensajes Cortos (SMS). Además, presenta una implementación de ésta API en una aplicación denominada Arec PhoneBook, la cual, recupera directamente de un teléfono móvil los números telefónicos asociados a un nombre de contacto.

Es importante mencionar que la implementación de la API Arec permite que las aplicaciones puedan funcionar desde cualquier lugar donde haya cobertura de red de telefonía móvil, y además es independiente de la compañía telefónica contratada. Así mismo, el teléfono móvil que desempeñe la función de servidor, no necesita la intervención del usuario para aceptar y contestar las peticiones hechas por el teléfono móvil cliente.

Con la API Arec, los programadores pueden desarrollar rápidamente nuevas aplicaciones para recuperar remotamente los objetos tales como: lista de contactos, fotos, imágenes, videos, etc., contenidos en un teléfono móvil.

A pesar de las ventajas que ofrece la mensajería SMS, esta no está exenta de problemas de seguridad en ambos extremos de la comunicación. Especialmente cuando se utiliza para intercambiar información confidencial, por esta razón surge la necesidad de dotar a la API AREC de una mayor seguridad en el envío y recepción de los mensajes. Para proporcionar esta protección se están implementando técnicas de criptografía que permitan cifrar la información que se intercambia entre el dispositivo cliente y el dispositivo servidor, lo cual evitará que la información pueda ser accedida por personas no autorizadas.

5. REFERENCIAS

[1] Comisión Federal de Telecomunicaciones, México, "Usuarios de Telefonía Móvil", Recuperado de: Dirección de Información Estadística de Mercados, cofetel, Corte a Abril de 2011.

[2] P. Brans, "Mobilize your enterprise: achieving competitive advantage through wireless technology", Editorial. Prentice Hall Professional, 2003.

[3] T. Imieliski, H. Korth, "Mobile computing", Editorial. Kluwer Academic Publisher, 2002. Junio 2002.

[4] F. Shearer, "Power management in mobile devices", Editorial. Newnes of Elsevier, Diciembre 2007.

[5] D. Flanagan, "Power Java in a nutshell", Editorial. Prentice Hall Professional, 2001.

[6] C. S. Horstmann, G. Cornell, "Core Java 2: Fundamentals", Editorial. O'Reilly Media, Inc., 2005.

[7] H. Schildt, "Java 7, A Beginner's Guide, Fifth Edition", Editorial. McGraw-Hill Prof Med/Tech, Marzo 8, 2011.

[8] S. Li, J. Knudsen, "Beginning J2ME: from novice to professional", Editorial. Apress, Abril, 2005.

[9] C. Fraizer, J. Bond, "Java API reference", Editorial. New Riders Pub, 1996.

[10] M. Jode, J. Allin, "Programming Java 2 micro edition on Symbian OS: a developer's guide to MIDP 2.0", Editorial. John Wiley & Sons, Ltd, Julio 30, 2004.

[11] V. Piroumian, "Wireless J2ME platform programming", Editorial. Prentice Hall Professional, 2002.

[12] Martin J. Wells, "J2ME game programming", Editorial. Cengage Learning, 2004.

[13] P. Smyth, "Mobile and wireless communications: key technologies and future applications", Editorial. Institution of Electrical Engineers, 2008.

[14] M. Juntao Yuan, "Enterprise J2ME: developing mobile Java applications", Prentice Hall Professional, 2004.