ENVÍO DE SMS Y E-MAIL

1. ENVÍO DE SMS

El envío de SMS es realmente simple. Consiste en importar una clase de Android, utilizar dos métodos y dar un único permiso a la aplicación. Veamos el código.

La clase necesaria es la siguiente:

```
import android.telephony.SmsManager;
```

Esta clase contiene todo lo necesario para manejar y trabajar con mensajes de texto. En este ejemplo trabajaremos con un par de métodos, uno para enviar un SMS simple y otro para enviar un SMS que contenga más de 160 caracteres, que es el número a partir del cual Android divide el mensaje en más de una parte.

Para poder usar estos métodos, necesitamos instanciar un objeto de la clase SMSManager. Lo haremos mediante la sentencia:

```
SmsManager manager = SmsManager.getDefault();
```

Veamos los métodos. El primer método es el siguiente:

sendTextMessage(String destinationAddress, String scAddress, String text, PendingIntent sentIntent, PendingIntent deliveryIntent).

Los parámetros que necesita son:

- destinationAddress: Es el número de teléfono al que se enviará el SMS
- scAddress: Es la dirección de la central de servicio de SMS (centralita). Null para utilizar la dirección por defecto.
- text: Es el cuerpo del mensaje.
- sentIntent: Intent que se envía en broadcast cuando el SMS se ha enviado o ha fallado, obteniendo el resultado de la operación (éxito o error). Si no queremos utilizarlo, lo ponemos a null.
- deliveryIntent: Si no es null, el Intent se envía cuando el mensaje se entrega al destinatario.

Para el tema de los PendingIntents necesitamos un BroadcastReceiver, de modo que para el ejemplo los dejaremos en null.

Ahora bien, cuando el mensaje que queremos enviar contiene más de 160 caracteres y lo hemos de dividir, utilizamos el siguiente método:

sendMultipartTextMessage(String destinationAddress, String scAddress, ArrayList<String> parts, ArrayList<PendingIntent> sentIntents, ArrayList<PendingIntent> deliveryIntents).

Los parámetros son igual que en el método anterior, pero con un ArrayList que contendrá las partes en las que haya sido necesario dividir el texto y otros dos ArrayList con tantos PendingIntents como partes del mensaje se vayan a enviar.

Para ejecutar los métodos, basta con escribir:

manager.sendTextMessage(Params),

O bien:

manager.sendMultipartTextMessage(Params)

Estos métodos lanzan la excepción IllegalArgumentException cuando el campo destinationAddress o el campo text/parts están vacíos, y la debemos de capturar.

Por último, para que la aplicación funcione, hemos de añadir el siguiente permiso al archivo AndroidManifest.xml:

```
<uses-permission android:name="android.permission.SEND_SMS"/>
```

Existe también la clase android.telephony.SmsMessage, para utilizar objetos SmsMessage aunque, curiosamente, parece que no existe un solo ejemplo de su uso en toda la red pero tampoco parece que esté obsoleta.

2. ENVÍO DE E-MAIL

Esta aplicación no envía los e-mails en sentido estricto. Lo que hace es aprovechar las aplicaciones que tengamos instaladas en nuestro teléfono: recopila los datos de envío, crea un intent adecuado y se lo pasa a la aplicación que seleccionemos, en caso de tener más de una. El mecanismo sería, más o menos, recopilar los datos, "preguntar" quién sabe mandar e-mails y enviarle los datos a la aplicación que conteste.

Para empezar, necesitamos importar la siguiente clase:

```
import android.net.Uri;
```

Utilizaremos esta clase en caso de querer enviar archivos adjuntos. Pero empecemos por crear el intent.

Este intent será un poco distinto de los que se han visto en clase. La forma de declararlo es la siguiente:

```
Intent intent = new Intent(Intent.ACTION_SEND);
```

No le indicamos a qué actividad debe llamar, ya que nuestra aplicación ignora el nombre o nombres de las aplicaciones que tenga el usuario instaladas en su teléfono. Lo que indicamos es que es un intent que se enviará "de alguna manera".

Ahora procedemos a añadir los extras necesarios con putExtra(key, valor). Para la parte de la clave, utilizaremos constantes de la clase Intent en lugar de los tipos de datos que hemos utilizado en los intents que llaman a una actividad.

Para añadir el destinatario, que será un String[] con las direcciones (un String simple no funciona):

```
intent.putExtra(Intent.EXTRA_EMAIL, to);
```

Para añadir el asunto (String):

```
intent.putExtra(Intent.EXTRA_SUBJECT, issue);
```

Para añadir el texto (String):

```
intent.putExtra(Intent.EXTRA_TEXT, body);
```

Ahora que tenemos todos los datos en el intent, establecemos el tipo de datos que se enviará en el email, en este caso será texto mixto:

```
intent.setType("multipart/mixed");
```

¿Y si quisiéramos enviar archivos adjuntos, por ejemplo, una imagen o un archivo .zip? Pues muy sencillo. Sólo necesitamos añadir otro extra y, en el caso del archivo .zip, cambiar el tipo del intent:

```
intent.putExtra(Intent.EXTRA_STREAM, Uri.parse("android.resource://" +
getPackageName() + "/"+ R.drawable.zorro));
```

```
intent.putExtra(Intent.EXTRA_STREAM, Uri.parse("file://" + “ruta del
archivo a adjuntar));
intent.setType(“application/zip”);
```

Una vez que tenemos los archivos adjuntos en el intent, sólo resta “preguntar” qué aplicación puede enviar nuestro email. Para ello utilizamos el método:

```
startActivity(Intent.createChooser(intent, "E-Mail"));
```

En este caso, el método startActivity() no lanza una aplicación en concreto, sino que crea un chooser con las aplicaciones que “respondan”, de las cuales escogeremos la que nos interese. En caso de que solo tengamos instalada una única aplicación, se lanzará automáticamente (por alguna razón, la aplicación de correo por defecto no la reconoce. En mi caso solo sale la de Gmail).

3. BIBLIOGRAFÍA

http://androideity.com/2012/04/03/sms-en-android-usando-eclipse/: Ejemplo de envío y recepción de SMS sin PendingIntents.

http://stackoverflow.com/questions/5944345/sending-sms-in-android: Envío de SMS con ejemplos de PendingIntents.

http://developer.android.com/reference/android/telephony/SmsManager.html: Documentación oficial de Android Developer.

http://www.maestrosdelweb.com/editorial/curso-android-enviar-emails/: Ejemplo de envío de emails.