Envío de SMS y E-Mail

Rubén Peña Fernández

Envío de SMS

Envío de SMS

- La aplicación de ejemplo consiste en enviar un simple SMS en sus dos variantes:
  - Una, cuando el texto que queremos enviar contiene menos de 160 caracteres.
  - Otra, cuando el texto que queremos enviar contiene más de 160 caracteres.

- Al ser dos casos diferentes utilizaremos, evidentemente, dos métodos distintos, aunque para el usuario se utilizará uno u otro de forma totalmente transparente.

Envío de SMS

- Para enviar un mensaje de menos de 160 caracteres utilizamos:
  - 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 el número 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.

Envío de SMS

- Para enviar un mensaje de más de 160 caracteres utilizamos:
  - sendMultipartTextMessage(String *destinationAddress*, String *scAddress*, ArrayList<String> *parts*, ArrayList<PendingIntent> *sentIntents*, ArrayList<PendingIntent> *deliveryIntents*).

    - 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.
      - parts: ArrayList de String con las partes en las que se ha dividido el mensaje.
      - sentIntents: Un ArrayList de PendintIntents con tantos objetos como partes tenga el ArrayLIst parts.
      - deliveryIntents: : Un ArrayList de PendintIntents con tantos objetos como partes tenga el ArrayLIst parts.

Envío de SMS

- Para poder enviar un SMS necesitamos importar la clase:

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

- Esta clase es la que contiene los diferentes métodos para enviar un SMS.

- Empezaremos por instanciar un objeto de la clase:

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

Envío de SMS

- Obtenemos los datos de los distintos componentes de la interfaz de usuarios:

```
String numero = destino.getText().toString();
String sms = mensaje.getText().toString();
```

- Comprobamos la longitud del mensaje, e invocamos al método adecuado:

```
if (sms.length() > LONGITUD) {
    ArrayList<String> partes = dividirMensaje(sms);
    manager.sendMultipartTextMessage(numero, null, partes, null,
            null);
} else {
    manager.sendTextMessage(numero, null, sms, null, null);
}
```

Envío de SMS

- Si la longitud del texto es excesiva, lo dividiremos en varias parte mediante el siguiente método:

```
public ArrayList<String> dividirMensaje(String mensaje) {
    ArrayList<String> partes = new ArrayList();
    int numPartes = mensaje.length() % LONGITUD;
    int inicio = 0;

    for (int i = 0; i < numPartes; i++) {
        inicio = i * LONGITUD;
        partes.add(mensaje.substring(inicio, inicio + LONGITUD));
    }

    return partes;
}
```

Envío de SMS

- Por último, para que todo lo anterior funcione, hemos de añadir el siguiente permiso en el archivo AndroidManifest.xml:

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

Envío de E-Mails

Envío de E-Mails

- Esta aplicación no envía emails en sentido estricto. Su mecanismo sería más o menos, recopilar los datos del email, preguntar qué aplicación sabe enviar email y mandarle los datos a la que responda.
- Un ejemplo claro de cómo aprovecharse del trabajo de los demás…

Envío de E-Mails

- Vamos a crear el intent. Como no sabemos con qué aplicación vamos a enviar el email, lo crearemos de la siguiente manera:

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

- Con este argumento indicamos que va a ser un intent para enviar.

Envío de E-Mails

- Obtenemos los datos que le vamos a añdir al intent:

```
String[] to = { destino.getText().toString() };
String issue = asunto.getText().toString();
String body = mensaje.getText().toString();
```

- El destinatario debe guardarse en un String[] aunque solo sea una dirección ya que, si no, no aparecerá en la aplicación que seleccionemos para enviarlo y habrá que introducirla de nuevo.

Envío de E-Mails

- Añadimos los datos al intent. Obsérvese que utilizamos constantes de la clase Intent, para el campo ‘key’:

```
// Añadimos la dirección de destino
intent.putExtra(Intent.EXTRA_EMAIL, to);
// Añadimos el asunto
intent.putExtra(Intent.EXTRA_SUBJECT, issue);
// Añadimos el mensaje
intent.putExtra(Intent.EXTRA_TEXT, body);
```

Envío de E-Mails

- También podemos enviar archivos elementos adjuntos. En este ejemplo enviaremos una imagen y un .zip, al ser tipos de archivos muy habituales.

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

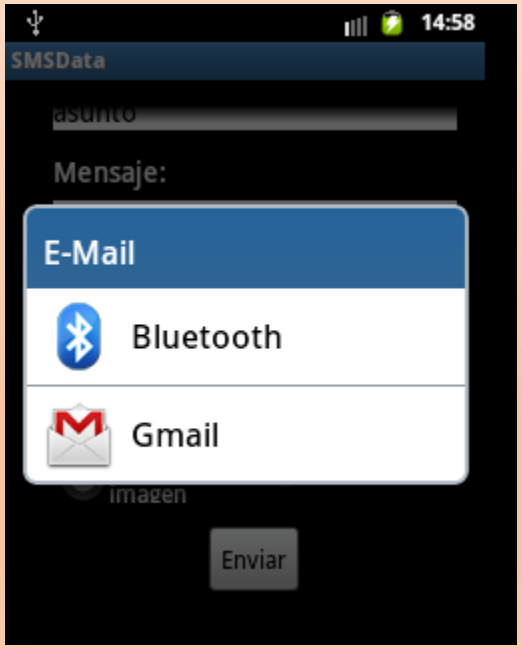
Envío de E-Mails

- Por último, creamos el chooser con las aplicaciones que respondan:

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

Envío de E-Mails

- Ahora debería parecer un chooser (si tenemos más de una aplicación que puede enviar el intent), y seleccionaremos una de las aplicaciones:



FIN

(de esta parte…)