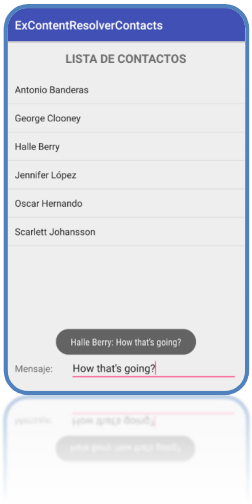
UNIDAD 5: ContentResolver Contactos

Vamos a ver como funcionan los *ContentProvider* suministrados por Android utilizando como ejemplo el acceso a los contactos de nuestro dispositivo.

Para ello vamos a crear una aplicación con una sóla actividad en la que se muestre una lista con nuestros contactos (los que tenemos almacenados en el móvil), pero para hacerlo un poco más interesante se mostrarán sólo aquellos en que hayamos guardado al menos un número de teléfono.

Además, vamos a añadir una caja de texto para recoger un mensaje que será enviado al contanto que seleccionemos de la lista.



CREACIÓN DE LA INTERFAZ

Como se ha señalado, la aplicación sólo va a contar con una actividad y por tanto tendremos que crear un único layout, en el que añadiremos la lista (ListView) y una caja de texto para recoger el mensaje a enviar.

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    tools:context="com.acme.excontentresolvercontacts.MainActivity"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical">

    <TextView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="LISTA DE CONTACTOS"
        android:textSize="20sp"
        android:textStyle="bold"
        android:gravity="center"
        android:layout_marginVertical="16dp"/>

    <ListView
        android:id="@+id/lvContactos"
        android:layout_width="match_parent"
        android:layout_height="480dp"/>

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginHorizontal="16dp">
        <TextView
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_weight="3"
            android:text="Mensaje: "
            android:textSize="16sp"/>
        <EditText
            android:id="@+id/txtSMS"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:hint="Intoduzca mensaje"/>
    </LinearLayout>

</LinearLayout>
```

CLASE: MAINACTIVITY

En la clase *MainActivity* declararemos los elementos necesarios para mantener la lista: un *ArrayList* de *String* para almacenar el nombre de los contactos, el *ListView* donde se van a mostrar los contactos y el adaptador, así como las constantes que utilizaremos para identificar las solicitudes de permisos (*READ_CONTACTS, SEND_SMS*) si fuera necesario.

```
public class MainActivity extends AppCompatActivity {
  protected ListView lvContactos;
  protected ArrayAdapter<String> adaptador;
  protected ArrayList<String> listacontactos;
  protected String contacto = "";
  protected final int REQUEST_READ_CONTACTS = 123;
  protected final int REQUEST_SEND_SMS = 124;
  ...
}
```

- ***onCreate***

En el método *onCreate* generaremos la vista de la actividad y prepararemos la lista.

A continuación, comprobaremos si disponemos del permiso para leer los contactos, en cuyo caso llamaremos al método "obtenerContactos" que se va a encargar de obtener los contactos del dispositivo y de mostrarlos en la lista.

Por último, añadimos un *listener* a la lista, de manera que cuando pulsemos sobre un elemento de la misma, recojamos el nombre del contacto y, tras comprobar que disponemos del permiso adecuado, le enviaremos el mensaje.

```
protected void onCreate(Bundle savedInstanceState) {
  super.onCreate(savedInstanceState);
  setContentView(R.layout.activity_main);

  listacontactos=new ArrayList<String>();
  adaptador = new ArrayAdapter<String>(this,
      android.R.layout.simple_list_item_1, listacontactos);
  lvContactos=(ListView)findViewById(R.id.lvContactos);
  lvContactos.setAdapter(adaptador);

  // Comprobamos el permiso READ_CONTACTS y obtenemos los contactos
  if ((android.os.Build.VERSION.SDK_INT>=android.os.Build.VERSION_CODES.M) &&
    ContextCompat.checkSelfPermission(this,Manifest.permission.READ_CONTACTS)
    != PackageManager.PERMISSION_GRANTED) {
    ActivityCompat.requestPermissions(this,
    new String[]{Manifest.permission.READ_CONTACTS}, REQUEST_READ_CONTACTS);
  } else {
    obtenerContactos();
  }

  // Enviar mensaje a contacto seleccionado
  lvContactos.setOnItemClickListener(new AdapterView.OnItemClickListener(){
    @Override
    public void onItemClick(AdapterView<?> adapterView,View v,int i,long l){
      TextView txtContacto = (TextView)v;
      contacto = txtContacto.getText().toString();
      if ((android.os.Build.VERSION.SDK_INT>=android.os.Build.VERSION_CODES.M)
        && ContextCompat.checkSelfPermission(MainActivity.this,
        Manifest.permission.SEND_SMS)!=PackageManager.PERMISSION_GRANTED) {
        ActivityCompat.requestPermissions(MainActivity.this,
        new String[]{Manifest.permission.SEND_SMS}, REQUEST_SEND_SMS);
      } else {
        enviarSMS(contacto);
      }
    }
  });
}
```

- ***onRequestPermissionsResult***

Este método se encarga de gestionar el resultado de las solicitudes de permisos:

- *READ_CONTACTS*: si tras la solicitud obtenemos el permiso llamamos al método "obtenerContactos", y en caso contrario mostramos un mensaje.
- *SEND_SMS*: si tras la solicitud obtenemos el permiso llamamos al método "enviarSMS", y en caso contrario mostramos un mensaje.

```
@Override
public void onRequestPermissionsResult(int requestCode,
  @NonNull String[] permissions, @NonNull int[] grantResults) {
  super.onRequestPermissionsResult(requestCode, permissions, grantResults);

  switch (requestCode) {

    case REQUEST_READ_CONTACTS:
      if (grantResults.length>0 &&
          grantResults[0] == PackageManager.PERMISSION_GRANTED) {
        obtenerContactos();
      } else {
        int dura = Toast.LENGTH_SHORT;
        String mensa = "No dispone de permisos para leer los contactos";
        Toast.makeText(getApplicationContext(),mensa,dura).show();
      }
      return;

    case REQUEST_SEND_SMS:
      if (grantResults.length>0 &&
          grantResults[0]==PackageManager.PERMISSION_GRANTED) {
        enviarSMS(contacto);
      } else {
        int dura = Toast.LENGTH_SHORT;
        String mensa = "No dispone de permisos para enviar SMS";
        Toast.makeText(getApplicationContext(),mensa,dura).show();
      }
      return;
  }
}
```

- **obtenerContactos**

Se encarga de obtener todos los contactos del dispositivo, y filtrar aquellos que tengan al menos un número de teléfono.

En primer lugar, obtenemos un objeto *ContentResolver* que nos va a permitir acceder a los datos del *ContentProvider* mediante *getContentResolver()*.

```
ContentResolver cr = getContentResolver();
```

Vamos a ejecutar una consulta (*query*) en la que queremos obtener todos los contactos, de manera que sólo necesitamos establecer la URI y la lista de campos a obtener de cada contacto (*proyection*). Para obtener esos valores utilizamos la clase *ContactsContract.Contacts* que nos los facilita en modo de constantes de tipo *String*.

En cuanto a esta lista de campos sólo necesitamos el nombre del contacto (*DISPLAY_NAME*) y si dispone de algún teléfono (*HAS_PHONE_NUMBER*), pero se han añadido los campos *ID* y *PHOTO_URI* a modo de ejemplo.

Se ha añadido también como criterio de ordenación la segunda columna ("2") para obtener los contactos ordenados de forma alfabética por el "*DISPLAY_NAME*".

El método "*query*" nos devuelve un cursor, para que podamos recorrer los registros obtenidos en nuestra consulta.

```
String proyeccion[] = {
  ContactsContract.Contacts._ID,
  ContactsContract.Contacts.DISPLAY_NAME,
  ContactsContract.Contacts.HAS_PHONE_NUMBER,
  ContactsContract.Contacts.PHOTO_URI
};

Cursor cur = cr.query(ContactsContract.Contacts.CONTENT_URI, proyeccion,
  null, null, "2");
```

A continuación, comprobaremos si la consulta ha obtenido algún resultado, en cuyo caso vaciamos el *ArrayList* de contactos para no duplicar los que contenga de una lectura anterior.

Mediante "*getColumnIndex*" obtenemos el número de columna de cada uno de los campos requeridos, ya que el acceso a los datos se realiza por medio de ese índice.

Recorremos los registros mediante el cursor, obteniendo el nombre y si dispone de teléfono de cada uno de esos registros, y en caso de que tengan teléfono, añadimos su nombre al *ArrayList* de contactos.

Una vez recorridos todos los registros notificamos los cambios al adaptador para que el *ListView* muestre los datos del *ArrayList* actuales, y finalizamos cerrando el cursor.

```
if (cur.getCount() > 0) {
  listacontactos.clear();
  int posName = cur.getColumnIndex(ContactsContract.Contacts.DISPLAY_NAME);
  int posTlf = cur.getColumnIndex(ContactsContract.Contacts.HAS_PHONE_NUMBER);
  while (cur.moveToNext()) {
    String name = cur.getString(posName);
    if (Integer.parseInt(cur.getString(posTlf)) > 0) {
      listacontactos.add(name);
    }
  }
  adaptador.notifyDataSetChanged();
}
cur.close();
```

El método completo quedaría así:

```
public void obtenerContactos (){
  ContentResolver cr = getContentResolver();
  String cols[] = {
    ContactsContract.Contacts._ID,
    ContactsContract.Contacts.DISPLAY_NAME,
    ContactsContract.Contacts.HAS_PHONE_NUMBER,
    ContactsContract.Contacts.PHOTO_URI
  };

  Cursor cur =
    cr.query(ContactsContract.Contacts.CONTENT_URI,cols,null,null,"2");
  if (cur.getCount() > 0) {
    listacontactos.clear();
    int posName = cur.getColumnIndex(ContactsContract.Contacts.DISPLAY_NAME);
    int posTlf = cur.getColumnIndex(ContactsContract.Contacts.HAS_PHONE_NUMBER);
    while (cur.moveToNext()) {
      String name = cur.getString(posName);
      if (Integer.parseInt(cur.getString(posTlf)) > 0) {
        listacontactos.add(name);
      }
    }
    adaptador.notifyDataSetChanged();
  }
  cur.close();
}
```

- **enviarSMS**

Esté método recibe el nombre de un contacto (*DISPLAY_NAME*) y envía un mensaje a todos los números de teléfono de ese contacto.

Lo primero que hacemos es recoger el texto del mensaje y en caso de que no contenga nada mostramos una alerta indicando que no se puede enviar el mensaje.

```
int duracion = Toast.LENGTH_SHORT;
String mensaje =
  ((EditText)findViewById(R.id.txtSMS)).getText().toString();
String alerta;
if (mensaje.equals("")) {
  alerta = "No ha escrito el mensaje";
  Toast.makeText(getApplicationContext(), alerta, duracion).show();
}
```

Tras realizar esta comprobación obtenemos el *ContentResolver* y ejecutamos el método *query* para obtener el ID del contacto cuyo nombre (*DISPLAY_NAME*) hemos recibido como parámetro.

```
ContentResolver cr = getContentResolver();
  String colId[] = {ContactsContract.Contacts._ID};
  String selName = ContactsContract.Contacts.DISPLAY_NAME + " = ?";
  String argsName[] = {nombre};
  Cursor cur = cr.query(ContactsContract.Contacts.CONTENT_URI, colId,
    selName, argsName, null);
```

A continuación, comprobamos si la consulta ha ofrecido algún resultado, y si es así obtenemos el gestor de SMS para poder enviar el mensaje, recorremos los contactos cuyo nombre coincide con el recibido, y para cada uno de ellos obtenemos su ID, y a partir de ella realizamos una nueva consulta para recoger todos los números de teléfono de ese contacto.

```
if (cur.getCount() > 0) {
  SmsManager smsManager = SmsManager.getDefault();
  Uri phoneUri = ContactsContract.CommonDataKinds.Phone.CONTENT_URI;
  int posId = cur.getColumnIndex(ContactsContract.Contacts._ID);

  while (cur.moveToNext()) {
    String id = cur.getString(posId);
    String selId = ContactsContract.CommonDataKinds.Phone.CONTACT_ID + " = ?";
    String argsId [] = {id};
    Cursor curTlf = cr.query(phoneUri, null, selId, argsId, null);
    int posTlf=curTlf.getColumnIndex(ContactsContract.CommonDataKinds.Phone.DATA);
```

Para obtener la información de detalle de cada una de las cuentas de un contacto, se utiliza la clase: *ContactsContract.CommonDataKinds.Phone*.

Puedes obtener más información de los campos de esta clase en los siguientes enlaces:

https://developer.android.com/reference/android/provider/ContactsContract.CommonDataKinds.Phone.html

https://developer.android.com/reference/android/provider/ContactsContract.Data.html

Por último, intentamos enviar el mensaje a cada uno de los teléfonos del contacto obtenidos generando una alerta que nos indica si se ha enviado el mensaje o no. Y finalizamos el procedimiento cerrando los cursores abiertos.

```
      while (curTlf.moveToNext()) {
        String telefono = curTlf.getString(posTlf);
        try {
          smsManager.sendTextMessage(telefono, null, mensaje, null, null);
          alerta = nombre + ": " + mensaje;
          Toast.makeText(getApplicationContext(), alerta, duracion).show();
        } catch (Exception e) {
          alerta = nombre + ": mensaje no enviado";
          Toast.makeText(getApplicationContext(), alerta, duracion).show();
        }
      }
      curTlf.close();
    }
    cur.close();
  }
}
```

El método completo quedaría así:

```
private void enviarSMS(String nombre){
  int duracion = Toast.LENGTH_SHORT;
  String mensaje =
    ((EditText)findViewById(R.id.txtSMS)).getText().toString();
  String alerta;

  if (mensaje.equals("")) {
    alerta = "No ha escrito el mensaje";
    Toast.makeText(getApplicationContext(), alerta, duracion).show();
  } else {

    ContentResolver cr = getContentResolver();
    String colId[] = {ContactsContract.Contacts._ID};
    String selName = ContactsContract.Contacts.DISPLAY_NAME + " = ?";
    String argsName[] = {nombre};
    Cursor cur = cr.query(ContactsContract.Contacts.CONTENT_URI, colId,
      selName, argsName, null);

    if (cur.getCount() > 0) {
      SmsManager smsManager = SmsManager.getDefault();
      Uri phoneUri = ContactsContract.CommonDataKinds.Phone.CONTENT_URI;
      int posId = cur.getColumnIndex(ContactsContract.Contacts._ID);

      while (cur.moveToNext()) {
        String id = cur.getString(posId);
        String selId = ContactsContract.CommonDataKinds.Phone.CONTACT_ID+" = ?";
        String argsId [] = {id};
        Cursor curTlf = cr.query(phoneUri, null, selId, argsId, null);
        int posTlf =
          curTlf.getColumnIndex(ContactsContract.CommonDataKinds.Phone.DATA);

        while (curTlf.moveToNext()) {
          String telefono = curTlf.getString(posTlf);
          try {
            smsManager.sendTextMessage(telefono, null, mensaje, null, null);
            alerta = nombre + ": " + mensaje;
            Toast.makeText(getApplicationContext(), alerta, duracion).show();
          } catch (Exception e) {
            alerta = nombre + ": mensaje no enviado";
            Toast.makeText(getApplicationContext(), alerta, duracion).show();
          }
        }
        curTlf.close();
      }
      cur.close();
    }
  }
}
```

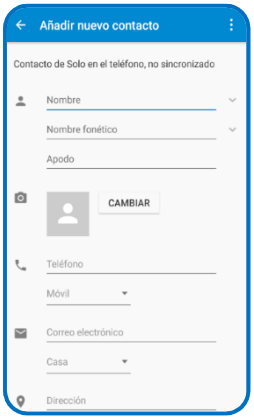
MANIFIESTO

Para poder acceder a la información de los contactos y enviar SMS no hemos de olvidar solicitar los permisos correspondientes en nuestro archivo de manifiesto.

```
<?xml version="1.0" encoding="utf-8"?>
<manifest
    xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.acme.excontentresolvercontacts">

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

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/AppTheme">
        <activity android:name=".MainActivity">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>

</manifest>
```

PRUEBAS

Para poder probar la aplicación desde el dispositivo virtual deberás agregar contactos a través de la aplicación de contactos del mismo.



Tras introducir uno o varios contactos en la agenda del teléfono comprueba que la aplicación funciona correctamente y prueba a cambiar se sitio el bloque de código que se encarga de obtener los contactos entre *onCreate*, *onResume* y *onRestart*, cerrando la app y volviéndola a abrir en cada uno de los casos, ¿Cuál sería el lugar idóneo para obtener los contactos?