23. Arrancar aplicaciones integradas de Android

En la unidad anterior hemos aprendido a utilizar intenciones para ejecutar actividades desde otras del mismo paquete. Una de las características más destacadas de Android es que también permite llamar a actividades de otras aplicaciones. Incluso podemos utilizar intenciones para invocar y utilizar aplicaciones integradas en Android, como el navegador web o el teléfono.

Por ejemplo, para arrancar el navegador web haremos lo siguiente:

```
Intent intent = new Intent (android.content.Intent.ACTION_VIEW,
                            Uri.parse("http://www.eps.uam.es"));
startActivity(intent);
```

En lugar de utilizar una constante, también podemos pasar la acción al constructor como una cadena de caracteres:

```
Intent intent = new Intent ("android.intent.action.VIEW",
                            Uri.parse("http://www.eps.uam.es"));
startActivity(intent);
```

Como puedes ver, pasamos dos argumentos al constructor del objeto `Intent`:

- El primer argumento, `android.content.Intent.ACTION_VIEW`, es la acción, que, combinada con el `http` del segundo argumento, hace que los datos se muestren con el navegador web. Si la uri fuera de tipo `mailto:`, se abriría el gestor de correo; si fuera de tipo `tel:`, se abriría el marcador de teléfonos; si fuera de tipo `market:`, se abriría Google Play, y así sucesivamente.
- El segundo argumento corresponde a los datos. Utilizamos el método `parse()` de la clase `Uri` para convertir la cadena `http://www.eps.uam.es` en un objeto `Uri`. El siguiente código abre un nuevo correo con el destinatario indicado:

```
Intent intent = new Intent ("android.intent.action.VIEW",
                            Uri.parse("mailto:pepe@uam.es"));
startActivity(intent);
```

También podemos pasar al constructor la acción y posteriormente añadir los datos con el método `setData()`:

```
Intent intent = new Intent ("android.intent.action.VIEW");
intent.setData(Uri.parse("http://www.eps.uam.es"));
startActivity(intent);
```

A veces no es necesario especificar datos. Por ejemplo, para seleccionar un contacto de la aplicación de contactos, hay que hacer lo siguiente:

```
Intent intent = new Intent(android.content.Intent.ACTION_PICK);
intent.setType(ContactsContract.Contacts.CONTENT_TYPE);
startActivity(intent);
```

En general, un objeto de tipo `Intent` puede contener acción, datos, tipo y categoría. En la página de desarrollo de Android encontrarás más detalles al respecto.

A continuación, veamos cómo permitir que nuestro juego envíe información a nuestros contactos y nos permita solicitar la aplicación a través de la cual deseamos enviarla. Vamos a crear un proyecto nuevo `CCC23` que, más tarde, cuando veamos cómo se implementan los menús, añadiremos a nuestro juego. A continuación estudiamos los archivos `MainActivity.java` y `activity_main.xml`:

`/src/MainActivity.java`

```
package es.uam.eps.android.CCC23;

import android.app.Activity;
import android.content.Intent;
import android.net.Uri;
import android.os.Bundle;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;

public class MainActivity extends Activity implements OnClickListener {
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        Button button = (Button) findViewById(R.id.send);
        button.setOnClickListener(this);
        button = (Button) findViewById(R.id.call);
        button.setOnClickListener(this);
    }

    @Override
    public void onClick(View v) {

        if (v.getId() == R.id.send) {
            Intent intent = new Intent(android.content.Intent.ACTION_SEND);
            intent.setType("text/plain");
            intent.putExtra(Intent.EXTRA_SUBJECT, "Cha cha cha score");
            intent.putExtra(Intent.EXTRA_TEXT,
                "Hola ..., he llegado a ... puntos en cha cha cha ...");
            startActivity(intent);
        } else if (v.getId() == R.id.call) {
            // Incluye android.permission.CALL_PHONE en el manifiesto

            Intent intent = new Intent(android.content.Intent.ACTION_CALL);
            intent.setData(Uri.parse("tel:+620254234"));
            startActivity(intent);
        }
    }
}
```

`/res/layout/activity_main.xml`

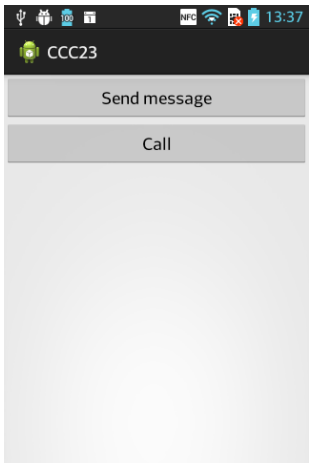
```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical" >

    <Button
        android:id="@+id/send"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="@string/sendString" />

    <Button
        android:id="@+id/call"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="@string/callString" />

</LinearLayout>
```

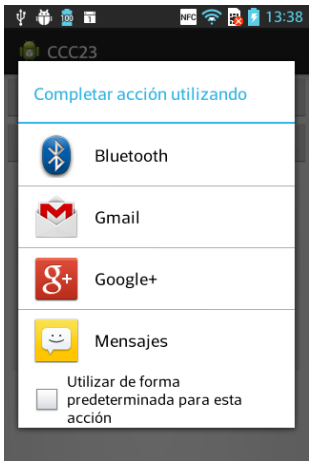
Si la aplicación tiene que marcar un número directamente, es necesario añadir el permiso `CALL_PHONE` al fichero `AndroidManifest.xml`:

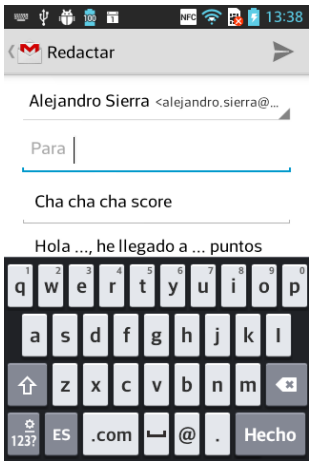
`/res/AndroidManifest.xml`

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="es.uam.eps.android.CCC23"
    android:versionCode="1"
    android:versionName="1.0" >

    <uses-sdk
        android:minSdkVersion="15"
        android:targetSdkVersion="19" />

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

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

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

</manifest>
```

Esta es la sucesión de pantallas que vemos al arrancar la aplicación, pulsar el botón Send message y seleccionar Gmail:







23.1 Ciclo de vida de las actividades

Veamos ahora cómo se ejecutan los distintos métodos del ciclo de vida de las actividades cuando se arranca una actividad desde otra. Supongamos que tenemos una actividad principal A que arranca una actividad B mediante una intención. Este sería el proceso completo de llamadas:

```
------ Se crea la app
A onCreate()
A onStart()
A onResume()

------ Se lanza la actividad B desde A
A onSaveInstanceState()
A onPause()
B onCreate()
B onStart()
B onResume()
A onStop()

------ Se aprieta Back con la actividad B visible
B onPause()
A onRestart()
A onStart()
A onResume()
B onStop()
B onDestroy()
```

Como se puede observar, antes de lanzar y de ejecutar ningún método del ciclo de vida de la actividad B, se ejecuta el método `onPause()` de la actividad A, a continuación se ejecutan los métodos `onCreate()`, `onStart()` y `onResume()` de la actividad B, que pasa a estar visible tapando completamente la actividad A.

Una vez que la actividad B está visible, se llama al método `onStop()` de la actividad A. Este proceso es igual cuando cerramos la actividad B al apretar el botón back. Esto es, primero se pausa B, a continuación se rearranca A (`onRestart()`, `onStart()` y `onResume()`) y finalmente se para y destruye B (`onStop()` y `onDestroy()`).

Viendo estas secuencias, es importante resaltar que la velocidad con que se ejecuta el método `onPause()` de una actividad influye directamente en el tiempo que tardará la siguiente actividad en mostrarse al usuario. Por lo tanto, hay que evitar tareas en el método `onPause()` que lleven demasiado tiempo y dejarlas para el método `onStop()`, una vez que la siguiente actividad ya está en pantalla.