INDICE

1. Intent en Activities

- ✓ Más información en este enlace.
- ✓ Para lanzar otra activity se hace uso de los Intent.
- ✓ **Nota Importante:** En Android cuando lanzamos una nueva activity, la anterior queda en memoria formando una cola LIFO. Al cerrar la nueva activity aparecerá la anterior. SI QUEREMOS CERRAR UNA ACTIVITY PARA ABRIR UNA NUEVA DEBEMOS DE ABRIR LA NUEVA ACTIVITY Y LLAMAR AL MÉTODO finish() de la activity que llama.

1.1. Construyendo un Intent

- ✓ http://developer.android.com/training/basics/firstapp/starting-activity.html

- ✓ http://developer.android.com/guide/components/intents-filters.html

- ✓ A efectos prácticos, una activity se viene a identificar con una pantalla. Los intent´s nos van a servir para llamar a dichas pantallas entre otras cosas.
- ✓ También van a servir para llamar a otras aplicaciones ANDROID y esperar una respuesta (o no).
- ✓ Un intent proporciona un enlace entre diferentes componentes como puede ser dos activity´s. Se puede traducir como la intención de la aplicación de hacer algo.
- ✓ Cuando llama, lo puede hacer indicando la Activity que quiere lanzar (**Intent Explícitos**) o simplemente indica la acción que quiere realizar y es el S.O. quien decide en base la acción e el tipo de información, que aplicación debe lanzar (**Intent Implícitos**).

- ✓ Para llamar a una activity desde otra tenemos el siguiente código:

```
1 Intent intent = new Intent(this, DisplayMessageActivity.class);
```

- ✓ El primer parámetro es una referencia al contexto (la clase Activity es una subclase de Context, por eso ponemos this).
- ✓ El segundo parámetro es la clase que el Sistema Operativo 'intentará' cargar (en nuestro caso la activity a cargar).
- ✓ Nota: El **context** es una interface que nos permite acceder a los recursos de la aplicación, clases y otras operaciones, como acceso a las activities…
- ✓ Ya vimos su funcionamiento en el tema de los Intents Explícitos.

1.2. Etiqueta action

- ✓ Como vimos anteriormente, cada vez que se crea una activity se añade en el AndroidManifiest.xml.
- ✓ Dita activity tiene asociado un filtro de intento (<intent-filter>) o podemos añadirlo en caso de no ser un tipo 'launcher'.
- ✓ Dentro de este tenemos la etiqueta <action....> en el que se define el nombre del filtro para la activity.
- ✓ De esta forma podemos llamar a otra activity por su nombre:
  - startActivity (new Intent (this,NombreClase.class);Esta es la opción vista hasta ahora.
  - startActivity(new Intent("nombre.definido.etiqueta.action"));Llamamos con el nombre definido en la etiqueta action.

  En este segundo caso tendremos que definir una categoría por defecto, para poder llamar a la activity por su nombre (<category android:name="android.intent.category.DEFAULT" />)

Veamos un ejemplo:

```
1      <activity
2             android:name="com.example.variasactiviy.MainActivity1"
3             android:label="@string/app_name1" >
```

```
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
        <activity
            android:name="com.example.variasactiviy.MainActivity2"
            android:label="@string/app_name2" >
            <intent-filter>
                <action android:name="com.example.variasactiviy.ActionMainActivity2" />
               <category android:name="android.intent.category.DEFAULT" />
            </intent-filter>
        </activity>
```

Para llamar a la segunda activity:

```
startActivity(new Intent("com.example.variasactiviy.ActionMainActivity2"));
```

- ✓ Nombre de los action:
  - Aunque el action puede tener cualquier nombre, por convención se usa este formato en los action definidos por Android: **.intent.action.nombre_acción**.

    Por ejemplo, 'android.intent.action.MAIN' es el action que define Android para una activity de categoría 'Launcher'.

    Podemos consultar todos los action definidos en el siguiente enlace.

    Si queremos llamar a una activity que 'atienda' a un tipo de acción tendríamos que poner:

```
Intent intent = new Intent(Intent.ACTION_MAIN);    // En este caso buscamos las que atiendan el tipo ACTION MAIN.
startActivity(intent);
```

  - En caso de las activities definidas por nosotros que no tienen un action 'predefinido' de Android, el name tendrá el formato: **nombre_paquete.nombre_accion**.

    En el ejemplo anterior: **com.example.variasactiviy**.ActionMainActivity2

Una activity puede llevar varias acciones asociadas.

Si existen varias activities con el mismo nombre el S.O. mostrará un mensaje para que escojamos cuál de ellas debe abrir.

- ✓ Los intent´s nos van a servir para llamar a otras aplicaciones que estén instaladas en el sistema operativo Android. La idea es llamar a una aplicación con un tipo de dato, y que sea el S.O. en base a los 'filter' de las aplicaciones instaladas, el que llame a la aplicación correspondiente.

Dentro del Intent irán los datos necesarios para que la aplicación que reciba la petición pueda dar el servicio. Así, si abro el navegador, puede enviarle la url que quiero que abra (es opcional mandarlo). Para mandar los datos haremos uso del método putExtra de la clase Intent (lo veremos después en otro punto).

Así, gracias a esta forma de trabajar, podemos usar Intent´s:

- Para llamar a un teléfono
- Para enviar un sms/correo
- Para abrir el navegador
- Para abrir una localización dentro de un mapa
- O para llamar a las Activities de otras aplicaciones.

Por ejemplo:

✓ Un intent para **enviar un correo electrónico**:

```
1 Intent intent = new Intent(Intent.ACTION_SEND);
2 intent.setType("application/octet-stream");
3 intent.putExtra(Intent.EXTRA_SUBJECT, "Subject");
4 intent.putExtra(Intent.EXTRA_TEXT, "Texto do Mail");
5 intent.putExtra(Intent.EXTRA_EMAIL, new String[]{"android@cursoandroid.es"});
6 startActivity(intent);
```

✓ Un intent para **mostrar el dial del teléfono**:

```
1 Intent intent = new Intent(Intent.ACTION_DIAL);
2 startActivity(intent);
```

✓ Un intent para **lanzar una busca del navegador web**:

```
1 Intent intent = new Intent(Intent.ACTION_WEB_SEARCH);
2 intent.putExtra(SearchManager.QUERY, "Android");
3 startActivity(intent);
```

✓ A un Intent podemos asociarle una acción, unos datos y una categoría.

Las activities pueden declarar el tipo de acciones que se pueden llevar a cabo y los tipos de datos que pueden gestionar.

Las acciones son cadenas de texto estándar que describen el que una activity puede hacer.

Por ejemplo android.intent.action.VIEW es una acción que indica que la actividad puede mostrar datos al usuario.

Más información en:

- http://developer.android.com/guide/components/intents-filters.html
- http://developer.android.com/reference/android/content/Intent.html

Cuando definimos una activity normalmente aparece esto en el AndroidManifiest.xml:

```
        <activity
            android:name=".MainActivity"
            android:label="@string/title_activity_main" >
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
```

Como vemos tenemos:

- <action android:name="android.intent.action.MAIN" />: indica que es una activity inicial a la cual no se le va a enviar datos ni devuelve datos.
- <category android:name="android.intent.category.LAUNCHER" />: indica que la activity puede aparecer en el no Launcher del S.O. (el programa que 'lanza' las aplicaciones).

Otro ejemplo sería el siguiente:

```
    <intent-filter >
                <action android:name="android.intent.action.VIEW" />
                <category android:name="android.intent.category.DEFAULT" />
                <category android:name="android.intent.category.BROWSABLE" />
                <data android:scheme="http" android:pathPattern=".*mp3"
android:mimeType="audio/*" />
   </intent-filter>
```

- <action android:name="android.intent.action.VIEW" />: indica que va a poder visualizar algún tipo de información.
- <category android:name="android.intent.category.BROWSABLE" />: Indica que la activity va a poder ser invocada desde el navegador al pulsar en algún enlace
- <category android:name="android.intent.category.DEFAULT" /> : Necesaria para que la activity pueda ser invocada de forma implícita.

En este caso cuando estamos navegando, al pulsar sobre una canción mp3 se abrirá la posibilidad de reproducirla con nuestra activity. Ya dentro de nuestra activity, el dato que podemos obtener es la URL de la canción a reproducir lo podríamos obtener de la siguiente forma:

```
        Intent intent = getIntent();
        // Por si volvemos del navegador web.
        if (intent != null && intent.getData() != null) {
                TextView txturl = (TextView)findViewById(R.id.txtURL);
                txturl.setText(intent.getData().toString());
        }
```

✓ En el caso de ACTION_VIEW es una acción muy genérica y Android tiene que descubrir que activity puede llamar haciendo uso de la composición de la URI.

Para eso mira el esquema que posee la URI (en este caso http) y pregunta a todas las actividades cuál de ellas entiende dicho esquema.

Podríamos tener registrados varios esquemas:

```
<activity ...>
   <intent -filter>
      <action android:name="android.intent.action.VIEW"/>
      <data android:scheme="http" />
      <data android:scheme="https" />
   </intent>
</activity>
```

Más información sobre la etiqueta en https://developer.android.com/guide/topics/manifest/data-element.html

Ejemplos de empleo de Actions: https://developer.android.com/guide/components/intents-common?hl=es-419

1.3. Etiqueta category

- ✓ No que se refiere a la etiqueta <category> esta se utiliza para categorizar las activities y que se puedan encontrar más fácilmente.
- ✓ Más información en: http://developer.android.com/reference/android/content/Intent.html#CATEGORY_ALTERNATIVE
- ✓ Anteriormente vimos un ejemplo en el teníamos: <category android:name="android.intent.category.BROWSABLE" />

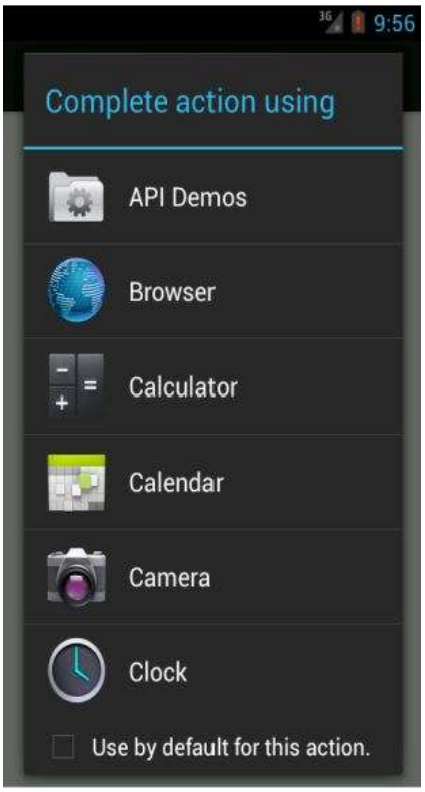
Si vamos al enlace anterior y buscamos a dicha categoría, os informa que dicha categoría es necesaria para que una activity sea llamada desde el navegador.

Por ejemplo, mientras que el sistema se está iniciando busca las activities que tengan la category de launcher: <category android:name="android.intent.category.LAUNCHER" />

Por ejemplo, podríamos lanzar una activity en base a su categoría:

```
        Intent intento = new Intent(Intent.ACTION_MAIN, null);
        intento.addCategory(Intent.CATEGORY_LAUNCHER);
        startActivity(intento);
```

Dará como resultado:



✓ Para lanzar la calculadora, en vez de emplear el nombre del paquete junto con la clase (estos nombres pueden variar de un teléfono a otro debido a la personalización del S.O. Android que hace los diferentes fabricantes) podemos emplear una categoría:

```
1 Intent intent = new Intent();
2 intent.setAction(Intent.ACTION_MAIN);
3 intent.addCategory(Intent.CATEGORY_APP_CALCULATOR);
4 startActivity(intent);
```

Pero también en este caso puede dar problemas, ya que para que una aplicación pueda ser llamada de forma implícita tiene que tener en su AndroidManifiest.xml esta otra categoría: <category android:name="android.intent.category.DEFAULT" />
Se no lo tiene, dará un error.
Veremos a continuación como comprobar si existe una aplicación que atienda a nuestro intento antes de lanzarla.

1.4. Comprobando la disponibilidad

✓ Siempre que hacemos uso de las llamadas de tipo **implícito** deberíamos comprobar si existe alguna Activity que pueda responder al tipo de Action que vamos solicitar.

Esto es necesario ya que se no lo comprobamos, la aplicación dejará de funcionar y se cerrará.

Para comprobarlo tenemos que chamar a método resolveActivity de la clase Intent.

```
1         Intento intent = new Intent(INDICAMOS A ACCIÓN);
```

```
2         if (intent.resolveActivity(getPackageManager())!=null){ // Comprobamos siempre en las llamadas implícitas si existe alguna activity que pueda atender mi petición
3                   startActivity(intent);
4         }
```

✓ Indicar también que los action pueden ser indicados llamando al método setAction de la clase Intent de la forma:

```
1         Intent intent = new Intent();
2         intent.setAction(Intent.ACTION_MAIN);
```

1.5. Caso Práctico 1

✓ Crearemos un nuevo paquete de nombre: **Intents**.

✓ Dentro del paquete **Intents** crear una nueva 'Empty Activity' de nombre: **UD06_02_A1_Intents** de tipo Launcher. Modificar el archivo **AndroidManifiest.xml** y añadir una label a la activity.

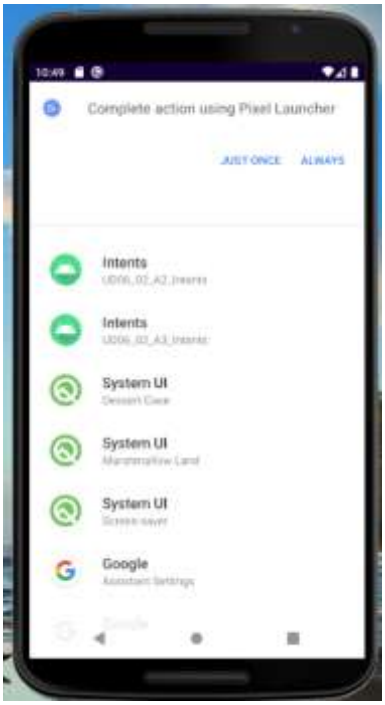
✓ Dentro del paquete **Intents** crear una nueva 'Empty Activity' de nombre: **UD06_02_A2_Intents** de tipo no Launcher. Modificar el archivo **AndroidManifiest.xml** y añadir una label a la activity.

✓ Dentro del paquete **Intents** crear una nueva 'Empty Activity' de nombre: **UD06_02_A3_Intents** de tipo no Launcher. Modifica el archivo **AndroidManifiest.xml** y añadir una label a la activity.

✓ Lo que vamos a hacer es crear tres activities, una principal con botón desde el cual se llamará a otras dos dúas activities creadas, pero en base a su Action:

Lanzando una activity por su Action

Activity principal de tipo Launcher.



Al pulsar sobre el botón el S.O. mostrará todas las activities que tengan el tipo de acción.

Escogemos la segunda Activity.

Si pulsamos el botón Back podemos escoger la otra Activity.

1.5.1. El XML de la Activity Principal

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
```

```
    android:layout_height="match_parent"
    tools:context=".UD06_02_A1_Intents">

    <Button
        android:id="@+id/btnLanzar_UD06_02_A1_Intents"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_marginEnd="8dp"
        android:layout_marginStart="8dp"
        android:layout_marginTop="8dp"
        android:text="Lanzar Activity Implicita"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

</androidx.constraintlayout.widget.ConstraintLayout>
```

1.5.2. Código de la Activity Principal

```
package com.example.intents;

import androidx.appcompat.app.AppCompatActivity;
import android.content.Intent;
import android.os.Bundle;
import android.view.View;

public class UD06_02_A1_Intents extends AppCompatActivity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_u_d06_02__a1__intents);


        findViewById(R.id.btnLanzar_UD06_02_A1_Intents).setOnClickListener(new
View.OnClickListener() {
            @Override
            public void onClick(View v) {
                Intent intent = new Intent(Intent.ACTION_MAIN);
                if (intent.resolveActivity(getPackageManager())!=null){ // Comprobamos siempre
en las llamadas implicitas si existe alguna activity que pueda atender mi petición
                    startActivity(intent);

                }
            }
        });

    }
}
```

- ✓ En este caso estamos utilizando uno de los dos Actions predefinidos del S.O. Android.

1.5.3. El XML de la Segunda Activity

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".UD06_02_A2_Intents">

    <TextView
        android:id="@+id/textView7"
```

```
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginEnd="8dp"
        android:layout_marginStart="8dp"
        android:layout_marginTop="8dp"
        android:text="Segunda Activity"
        android:textSize="24sp"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

</androidx.constraintlayout.widget.ConstraintLayout>
```

1.5.4. El XML de la Tercera Activity

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".UD06_02_A3_Intents">


    <TextView
        android:id="@+id/textView7"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginEnd="8dp"
        android:layout_marginStart="8dp"
        android:layout_marginTop="8dp"
        android:text="Tercera Activity"
        android:textSize="24sp"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />
</androidx.constraintlayout.widget.ConstraintLayout>
```

1.5.5. El Archivo AndroidManifiest.xml

```
<activity android:name=".UD06_02_A1_Intents">
    <intent-filter>
        <action android:name="android.intent.action.MAIN" />
        <category android:name="android.intent.category.LAUNCHER" />
    </intent-filter>
</activity>

<activity
    android:name=".UD06_02_A2_Intents"
    android:label="UD06_02_A2_Intents">
    <intent-filter>
        <action android:name="android.intent.action.MAIN"/>
        <category android:name="android.intent.category.DEFAULT" />

    </intent-filter>
</activity>
<activity
    android:name=".UD06_02_A3_Intents"
    android:label="UD06_02_A3_Intents">
    <intent-filter>
        <action android:name="android.intent.action.MAIN"/>
        <category android:name="android.intent.category.DEFAULT" />
```

```
    </intent-filter>
</activity>
```

✓ Líneas 14,22: Ver cómo debemos incluir la categoría CATEGORY_DEFAULT en el filtro de intents. Si no se pone, startActivity () o startActivityForResult () no funcionarán.

✓ Podemos hacer lo mismo, pero definiendo nuestro propio Action (recordar que en ese caso el nombre del Action tiene que estar precedido por el nombre del paquete):

Modificamos o **AndroidManifiest.xml**

```
<activity
    android:name=".UD06_02_A2_Intents"
    android:label="UD06_02_A2_Intents">
    <intent-filter>
         <!--<action android:name="android.intent.action.MAIN"/>-->
        <category android:name="android.intent.category.DEFAULT" />
        <action android:name="com.example.intents.ACCION_VISUALIZAR"/>
    </intent-filter>
</activity>
<activity
    android:name=".UD06_02_A3_Intents"
    android:label="UD06_02_A3_Intents">
    <intent-filter>
        <action android:name="com.example.intents.ACCION_VISUALIZAR"/>
        <category android:name="android.intent.category.DEFAULT" />
        <!--<action android:name="android.intent.action.MAIN"/>-->
    </intent-filter>
</activity>
```

✓ Y modificamos el código de la Activity Principal:

```
package com.example.intents;

import androidx.appcompat.app.AppCompatActivity;
import android.content.Intent;
import android.os.Bundle;
import android.view.View;

public class UD06_02_A1_Intents extends AppCompatActivity {

    public final String ACCION_VISUALIZAR = "com.example.intents.ACCION_VISUALIZAR";

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_u_d06_02__a1__intents);


        findViewById(R.id.btnLanzar_UD06_02_A1_Intents).setOnClickListener(new
View.OnClickListener() {
            @Override
            public void onClick(View v) {
               /* Intent intent = new Intent(Intent.ACTION_MAIN);
                if (intent.resolveActivity(getPackageManager())!=null){ // Comprobamos siempre
en las llamadas implicitas si existe alguna activity que pueda atender mi petición
startActivity(intent);*/
                Intent intent = new Intent();
```

```
                intent.setAction(ACCION_VISUALIZAR);
                if (intent.resolveActivity(getPackageManager())!=null){ // Comprobamos siempre
en las llamadas implicitas si existe alguna activity que pueda atender mi petición
                    startActivity(intent);
                }
            }
        });

    }
}
```

1.6. Caso Práctico 2

NOTA ACLARATORIA:

- ✓ Llegados a este punto, aunque no vimos el tema de los permisos en Android.

  Hasta ahora estamos, al menos en mi caso, haciendo las pruebas sobre un emulador API 29.

  A partir de la API 23, el sistema de permisos cambia y ahora, ciertos permisos necesitan ser solicitados por la aplicación para poder funcionar.

  Este caso práctico hace uso de ciertos permisos que están dentro de la categoría anterior, por lo que el estudiante necesitaría añadir el código que se encuentra en el PDF '*Permisos AndroidManifiest.xml*', que veremos más adelante, para poder funcionar sobre el emulador API 29.

  En este caso práctico, lo haremos sobre un emulador con una API menor a la 23, por ejemplo API 22 o API 21.

  Nota: Mejor crear un nuevo emulador que cambiar de API en un emulador existente, porque puede dar problemas y no arrancar.

- ✓ Si no lo tenemos creado de antes, crearemos un nuevo paquete de nombre: **Intents**.
- ✓ Dentro del paquete **Intents** crear una nueva 'Empty Activity' de nombre: **UD06_03_A1_Intents** de tipo Launcher.

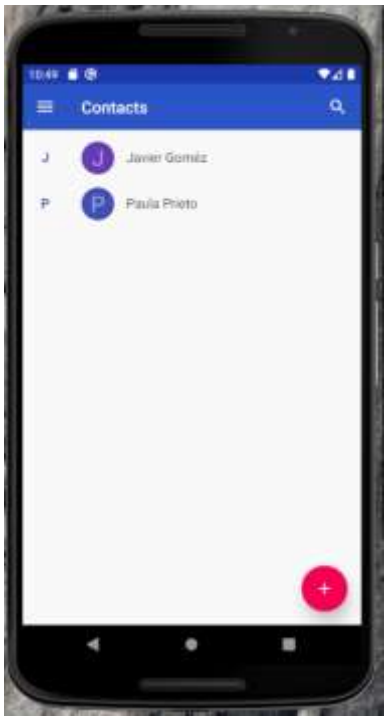
  Modificar el archivo **AndroidManifiest.xml** y añade una label a la activity.

- ✓ Dentro del paquete **Intents** crear una nueva 'Empty Activity' de nombre: **UD06_03_A2_Intents** de tipo no Launcher.

  Modificar el archivo **AndroidManifiest.xml** y añadir una label a la activity.

Intents implícitos

Pulsar en **Mostrar contactos**.
En el fichero AndroidManifest le daremos a la aplicación para que acceda a los contactos. Para lanzar a la siguiente activity de la aplicación contactos el intent tiene el siguiente formato:



La lista de contactos.

intent = new Intent(Intent.ACTION_VIEW,
Uri.parse("content://contacts/people/"));

Pulsar en **Seleccionar contactos**. Se lanza la siguiente intent:
intent = new Intent(Intent.ACTION_PICK,
Uri.parse("content://contacts/people"));
startActivityForResult(intent, COD_CONTACTOS);

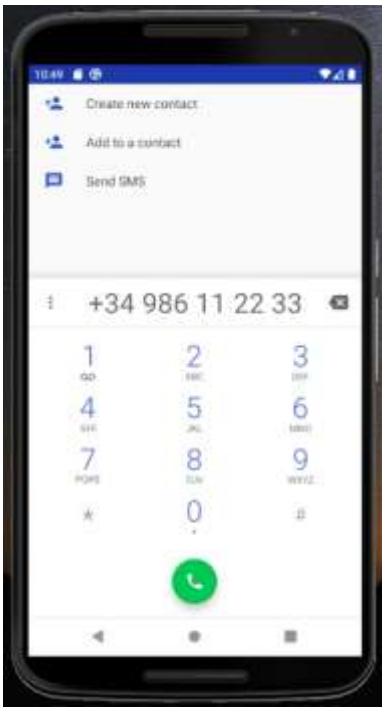
Seleccionamos un contacto y gracias a la Acción con la que se lanzó el intent podemos devolver a la actividad principal los datos de ese contacto.

Cuando volvemos a la actividad principal podemos recuperar el nombre del contacto seleccionado.

Ya inició la marcación al teléfono indicado.

Pulsar en **Llamar por teléfono**. Se va a lanzar el siguiente intent:
intent = new Intent(Intent.ACTION_CALL, Uri.parse("tel:(+34)981445566"));

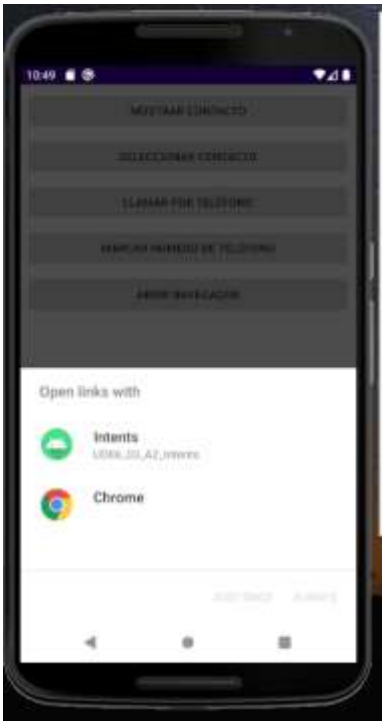
Pulsar en **Marcar nº teléfono**. Se va a lanzar la activity en la que nos decidimos si marcamos el número a marcar. **intent = new Intent(Intent.ACTION_DIAL, Uri.parse("tel:(+34)986112233"));**



Activity en la que el usuario marca el número de teléfono, pero que ya viene cubierto el campo.

Pulsar ahora en **Abrir navegador***. Donde se van ofrecer un par de opciones para abrir la url especificada.

intent = new Intent(Intent.ACTION_VIEW, Uri.parse("http://www.google.es/"));



Activities que tienen definidos filtros de intención en sus ficheros AndroidManifest- para procesar datos de tipo http. En este otro caso, se selecciono nuestra actividad secundaria (RecibirDatos) y lo que hace es mostrar la URL.

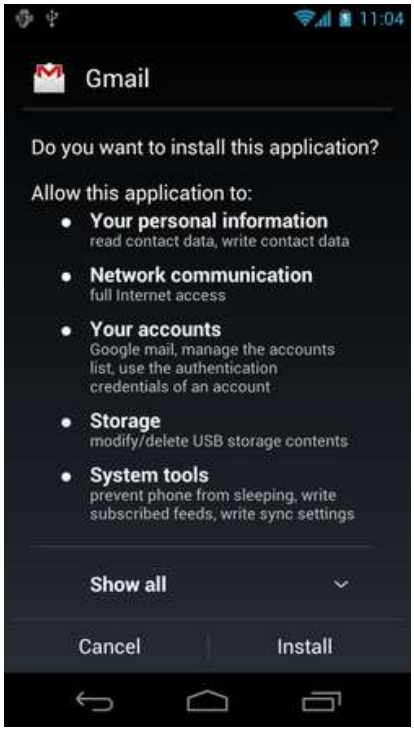
En este caso se seleccionó el navegador web. Pulsar en **Llamar por teléfono**. Se va a lanzar el siguiente intent:

intent = new Intent(Intent.ACTION_CALL, Uri.parse("tel:(+34)981445566"));

En este caso, se seleccionó nuestra actividad secundaria (RecibirDatos) y lo que hace es mostrar la URL.

1.6.1. Permisos

- ✓ Cando instalamos una aplicación en un dispositivo real (no en un AVD) se precisa acceder a características que exigen algún tipo de permiso, o proceso de instalación, pregunta si estamos dispuestos a dar eses permisos para que la aplicación pueda funcionar con todas las funcionalidades.
- ✓ Por ejemplo, instalando Gmail:



✓ El proceso de instalación pregunta al usuario si por ejemplo le deja acceder a esta aplicación a su lista de contactos.

✓ En el fichero **AndroidManifest.xml** es donde se declaran los permisos que precisa la aplicación para poder hacer uso de funcionalidades protegidas: contactos, cámara, memoria usb, gps, etc.

✓ A través de una o varias etiquetas **<uses-permission>** se van a declarar los permisos que precisa la aplicación.

Ejemplos de permisos en el fichero AndroidManifest.xml

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

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

    <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/Theme.Intents">
```

✓ **Línea 6**: Permiso que "permite" realizar una llamada de teléfono sin pasar a través del interface del teléfono para que lo usuario marque.

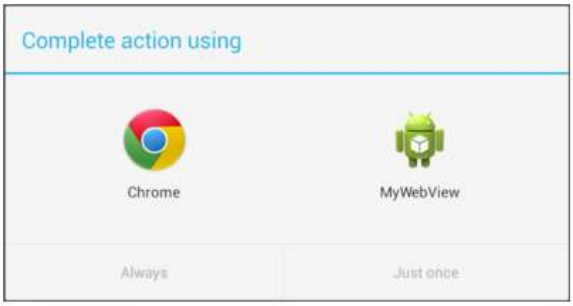
✓ **Línea 7**: Le permite a la aplicación abrir conexiones de red.

✓ **Línea 8**: De su lectura se puede concluir lo que permite.

- ✓ Para conocer los permisos disponibles podemos consultar el siguiente enlace e incluirlos directamente en el xml:
  - http://developer.android.com/reference/android/Manifest.permission.html
- ✓ Si vamos a instalar la aplicación en un dispositivo con una API 23 o superior, dependiendo del tipo de permiso, tendremos que solicitarlo específicamente por programación a mayores de indicarlo en el archivo AndroidManifiest.xml.
- ✓ Recordar que los permisos se asignan en el momento de la instalación a aplicación. En un dispositivo real se pide consentimiento al usuario, no así en un AVD.

1.6.2. Llamadas a intents de modo implícito

- ✓ Como ya se digo, un componente puede ser lanzado de forma implícita cuando se indica la **acción** que se desea realizar y si es el caso los datos sobre los que se desea realizar la acción.
- ✓ No se va a especificar cuál es la activity que va a atender el intent. Incluso si hay varias Activities que puedan atender ese intent, el sistema ofrecerá al usuario las distintas posibilidades para que escoja.
- ✓ Por ejemplo, cuando desde contactos se desea enviar un mensaje a un número de teléfono el sistema ofrece la posibilidad que se envíe a través de sms, Whatsapp, Viber, etc, si se tienen instalados estos últimos.
- ✓ La siguiente imagen muestra un ejemplo no que para abrir una URL hay dos aplicaciones que se pueden hacer.



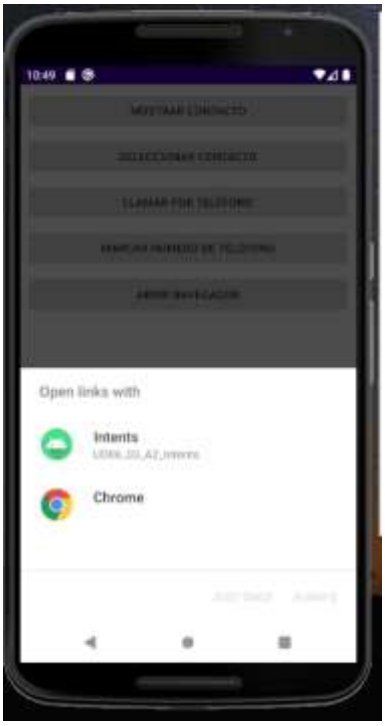
- ✓ Para lanzar un intent de modo implícito se precisa indicar:
  - **Acción (action)**: La acción que se desea llevar a cabo. Por ejemplo,
    - ACTION_VIEW, para mostrar datos al usuario
    - ACTION_EDIT, para editar los datos que nos pasan
    - ACTION_PICK. selecciona un ítem de un conjunto de datos y devolver lo seleccionado.
    - ACTION_WEB_SEARCH. busca en el navegador las palabras indicadas.
    - etc.
  - **Datos (data)**: los datos sobre los que se va a operar, por ejemplo, una url, los datos de un contacto, la cadena de busca, etc.
- ✓ Ejemplo:
  - ACTION_VIEW content://contacts/people/1 -- Muestra la información sobre la persona con identificador 1.

- ✓ **Referencias**:

- http://developer.android.com/reference/android/content/Intent.html

1.6.3. Filtros de intenciones (Intent Filters)

- ✓ Android busca los componentes que pueden responder a una **Acción** en los filtros de intención que se definen en fichero **AndroidManifest.xml** de todas las aplicaciones que están instaladas no dispositivo.
- ✓ Cuando se construye una activity una aplicación, se puede indicar los filtros de intención se manifiestan que esa activity puede dar respuesta las **Acciones** que lanzan otras aplicaciones.
- ✓ En nuestro caso, vamos a modificar el código de la actividad secundaria para que pueda atender a la petición de visionar urls de tipo **http**.
- ✓ Así cuando se lance, por ejemplo, que se desea ver http://www.google.es el sistema va a ofrecer 2 opciones: el navegador de internet y nuestra actividad secundaria del proyecto (RecibirDatos).



- ✓ Para esto es preciso definir que esa Activity puede atender Acciones de visionado de datos tipo http.

  Modificamos el archivo **AndroidManifiest.xml** para la segunda activity.

```
<activity android:name=".UD06_03_A2_Intents"
          android:label="UD06_03_A2_Intents">
    <intent-filter>
        <action android:name="android.intent.action.VIEW" />
        <category android:name="android.intent.category.DEFAULT" />
        <category android:name="android.intent.category.BROWSABLE" />
        <data android:scheme="http" />
    </intent-filter>
</activity>
```

- ✓ **Línea 4:** indica para que tipo de acciones está disponible esta activity.

- ✓ **Líneas 5,6:** La categoría indica las circunstancias en las que se desarrolla la acción.
- ✓ **Línea 7:** el tipo de datos que puede atender esta activity cuando haya una intención preguntando por recursos http.

1.6.4. XML del Layout principal

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".UD06_03_A1_Intents">

    <Button
        android:id="@+id/btnAmosarContact_UD06_03_01_Intents"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_marginEnd="8dp"
        android:layout_marginStart="8dp"
        android:layout_marginTop="8dp"
        android:text="Mostrar Contacto"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

    <Button
        android:id="@+id/btnSelContacts_UD06_03_01_Intents"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_marginEnd="8dp"
        android:layout_marginStart="8dp"
        android:layout_marginTop="8dp"
        android:text="Seleccionar Contacto"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/btnAmosarContact_UD06_03_01_Intents" />

    <Button
        android:id="@+id/btnChamar_UD06_03_01_Intents"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_marginEnd="8dp"
        android:layout_marginStart="8dp"
        android:layout_marginTop="8dp"
        android:text="Llamar por teléfono"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/btnSelContacts_UD06_03_01_Intents" />

    <Button
        android:id="@+id/btnMarcar_UD06_03_01_Intents"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_marginEnd="8dp"
        android:layout_marginStart="8dp"
        android:layout_marginTop="8dp"
        android:text="Marcar número de teléfono"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/btnChamar_UD06_03_01_Intents" />

    <Button
        android:id="@+id/btnAbrirNav_UD06_03_01_Intents"
        android:layout_width="0dp"
        android:layout_height="50dp"
        android:layout_marginEnd="8dp"
```

```
        android:layout_marginStart="8dp"
        android:layout_marginTop="8dp"
        android:text="Abrir Navegador"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.502"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/btnMarcar_UD06_03_01_Intents" />

</androidx.constraintlayout.widget.ConstraintLayout>
```

1.6.5. Código Java de la activity principal

```
package com.example.intents;

import android.app.Activity;
import android.content.Intent;
import android.database.Cursor;
import android.net.Uri;
import android.os.Bundle;
import android.provider.ContactsContract;
import android.view.View;
import android.widget.Toast;

    public class UD06_03_A1_Intents extends Activity implements View.OnClickListener {


        private static final int COD_CONTACTOS = 7;

        @Override
        protected void onCreate(Bundle savedInstanceState) {
            super.onCreate(savedInstanceState);
            setContentView(R.layout.activity_u_d06_03__a1__intents);

            xestionarEventos();
        }

        protected void onActivityResult(int requestCode, int resultCode, Intent data) {

            if (requestCode == COD_CONTACTOS && resultCode == RESULT_OK) {
                Uri contactoData = data.getData();
                Cursor cursor = getContentResolver().query(contactoData, null, null, null,
null);
                if (cursor.moveToFirst()) {
                    String nombre = cur-
sor.getString(cursor.getColumnIndexOrThrow(ContactsContract.Contacts.DISPLAY_NAME));
                    Toast.makeText(this, "Contacto: " + nombre, Toast.LENGTH_LONG).show();
                }
            }
        }

        private void xestionarEventos(){
            findViewById(R.id.btnSelContacts_UD06_03_01_Intents).setOnClickListener(this);
            findViewById(R.id.btnAmosarContact_UD06_03_01_Intents).setOnClickListener(this);
            findViewById(R.id.btnMarcar_UD06_03_01_Intents).setOnClickListener(this);
            findViewById(R.id.btnChamar_UD06_03_01_Intents).setOnClickListener(this);
            findViewById(R.id.btnAbrirNav_UD06_03_01_Intents).setOnClickListener(this);
        }

        @Override
        public void onClick(View v) {
            Intent intent = null;

            switch (v.getId()) {
                case R.id.btnSelContacts_UD06_03_01_Intents:
                    intent = new Intent(Intent.ACTION_PICK,
Uri.parse("content://contacts/people/"));
                    startActivityForResult(intent, COD_CONTACTOS);
                    break;
```

```
                case R.id.btnAmosarContact_UD06_03_01_Intents:
                    intent = new Intent(Intent.ACTION_VIEW,
Uri.parse("content://contacts/people/"));
                    startActivity(intent);
                    break;
                case R.id.btnMarcar_UD06_03_01_Intents:
                    intent = new Intent(Intent.ACTION_CALL, Uri.parse("tel:(+34)981445566"));
                    startActivity(intent);
                    break;
                case R.id.btnChamar_UD06_03_01_Intents:
                    intent = new Intent(Intent.ACTION_DIAL, Uri.parse("tel:(+34)986112233"));
                    startActivity(intent);
                    break;
                case R.id.btnAbrirNav_UD06_03_01_Intents:
                    intent = new Intent(Intent.ACTION_VIEW, Uri.parse("http://www.google.es/"));
                    startActivity(intent);
                    break;

            }
        }
    }
}
```

- ✓ **Línea 17**: Se crea una nueva constante de tipo entero con valor cualquiera para cuando llamemos a la activity de contactos para que nos devuelva un contacto.
- ✓ **Líneas 27-39**: Revisamos cuando se vuelva de una actividad secundaria, si es la de Contactos. En este caso, (se escapa de los objetivos de esta unidad) se creará un cursor para procesar los datos recibidos de la activity contactos.
- ✓ **Líneas 41-47**: Se registra el evento de Click en todos los botones.
- ✓ **Líneas 50-75**: Procesados dos de los distintos botones, creación de los intents en función del tipo de acción y datos que se desean procesar.
- ✓ **Línea 56**: Observar como se llama a la activity de contactos para que luego nos devuelva el contacto seleccionado.

1.6.6. XML del Layout de la Activity Secundaria: UD06_03_A2_Intents

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".UD06_03_A2_Intents">

    <TextView
        android:id="@+id/tvResultado_UD06_03_A2_Intents"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center_horizontal"
        android:text="resultado"
        android:textSize="20sp" />


    <Button
        android:id="@+id/btnPechar_UD06_03_A2_Intents"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center_horizontal"
        android:text="Cerrar" />

</LinearLayout>
```

1.6.7. El código Java de la Activity Secundaria: UD06_03_A2_Intents

```
package com.example.intents;

import androidx.appcompat.app.AppCompatActivity;
import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.widget.TextView;

public class UD06_03_A2_Intents extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_u_d06_03__a2__intents);

        TextView tvResultado = (TextView) findViewById(R.id.tvResultado_UD06_03_A2_Intents);
        Intent intent = getIntent();
        tvResultado.setText("URL: " + intent.getDataString());

        findViewById(R.id.btnPechar_UD06_03_A2_Intents).setOnClickListener(new
View.OnClickListener() {
            @Override
            public void onClick(View v) {
                finish();
            }
        });
    }
}
```

- ✓ **Líneas 18-20:** Se añadió la posibilidad de poder procesar intents cuyo esquema (getSchema) sea de tipo http. En este caso simplemente mostramos la URL a la que desea conectarse el usuario.