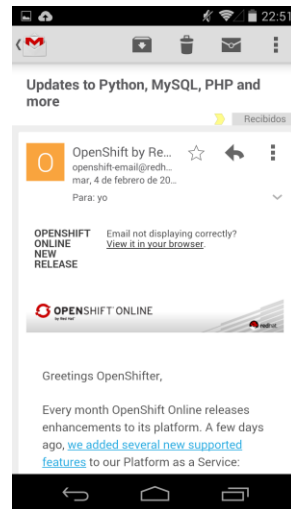
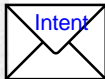


# Intent

*Es un mensaje que se envía al sistema operativo para que éste inicie un componente (Activity, Service, BroadcastReceiver) de una aplicación*

# Intent



# Intent

## Tipos

- 1 Explícitas
- 2 Implícitas

# Intent

## 1 Explícitas

*El intent incluye el nombre del componente que se desea iniciar  
sólo se puede iniciar componentes incluidos en la misma  
aplicación*

Intent intent\_1 = new Intent(this, es.upsa.mimo.app.DetailActivity.class);

Intent intent\_2 = new Intent();  
intent\_2.setClass(this, es.upsa.mimo.app.DetailActivity.class);

Intent intent\_3 = new Intent();  
Intent\_3.setClassName(this, "es.upsa.mimo.app.DetailActivity");

→ Representa android.Content.Context

→ Objecto java.lang.Class que describe la  
activity

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="es.upsa.mimo.app" .... >
    <application .... >
        <activity name=".MasterActivity" . . ./>
        <activity name=".DetailActivity" . . ./>
```

AndroidManifest.xml

# Intent

## 2 Implícitas

El sistema Android determina qué componente iniciar en función de la acción que realice.  
Se puede iniciar componentes de otras aplicaciones.

- 2.1 **ACTION:** cadena de texto que representa la acción que realiza el componente («verbo»)  
Una Activity puede declarar tantos <action> como quiera.

### Standard

### Definida por el usuario

Intent.ACTION\_MAIN

"android.intent.action.ACTION\_MAIN"

Intent.ACTION\_VIEW

"android.intent.action.ACTION\_VIEW"

Intent.ACTION\_PICK "android.intent.action.ACTION\_PICK"

Intent.ACTION\_SEND

"android.intent.action.ACTION\_SEND"

```
<?xml version="1.0" encoding="utf-8"?>
<manifest
  xmlns:android="http://schemas.android.com/apk/res/android"
    package="es.upsa.mimo.app" .... >
  <application .... >
    <activity name=".DetailActivity">
      <intent-filter>
        <action android:name="android.intent.action.ACTION_VIEW">
      </intent-filter>
    </activity>
```

AndroidManifest.xml

"es.upsa.mimo.app.ACTION\_VIEW"

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="es.upsa.mimo.app" .... >
  <application .... >
    <activity name=".DetailActivity">
      <intent-filter>
        <action android:name="es.upsa.mimo.app.ACTION_VIEW">
      </intent-filter>
    </activity>
```

AndroidManifest.xml

# Intent

## 2 Implícitas

- 2.2 *DATA*: representa el dato sobre el que realizar la ACTION.  
Está formado por dos componentes : *Uri* + *MimeType*. Un Activity puede contener muchos <data>

Uri

MimeType



type/subtype

text/html

image/png

application/json

vnd.android.cursor.dir/vnd.mimo.entities.Autobus

```
<activity name=".DetailActivity">
  <intent-filter>
    <action android:name="es.upsa.mimo.app.ACTION_VER"/>
    <data android:scheme="http"
          android:ssp=""
          android:host="www.upsa.es"
          android:port=""
          android:path=""
          android:pathPattern="mimo.*"
          android:mimeType=""/>
  </intent-filter>
</activity>
```

AndroidManifest.xml

# Intent

## 2 Implícitas

### 2.2 DATA

Si un <data> especifica únicamente **scheme**, todas las Uri con ese **scheme** serán válidas

Si un <data> únicamente especifica **scheme** y **authority**, serán válidas aquellas Uri con los mismos **scheme** y **authority** independientemente de su path

Si un <data> especifica **scheme**, **authority** y **path**, sólo serán válidas aquellas Uri que posean los mismos **scheme**, **authority** y **path**.

Si un Intent contiene **Uri** pero **no mimeType**, pasará el test si el <data> **no** contiene **mimeType** y el **Uri** coincide con los componentes fijados

Si un Intent contiene **mimeType** pero **no Uri**, pasará el test si el <data> **sólo** contiene **mimeType** y además coinciden

# Intent

## 2 Implícitas

2.3 **CATEGORY:** es una cadena de caracteres que representa el contexto en el que se va a ejecutar la acción.  
Un Intent puede contener muchas `<category>`.

Estandar

Definida por el usuario

**CATEGORY\_DEFAULT** valor por defecto para los intent implícitos  
"android.intent.category.DEFAULT"

"es.upsa.mimo.app.category.IMPRIMIBLE"

**CATEGORY\_LAUNCHER** debe mostrarse en el launcher "android.intent.category.LAUNCHER"

**CATEGORY\_BROWSABLE** puede lanzarse desde un navegador  
"android.intent.category.BROWSABLE"

AndroidManifest.xml

```
<activity name=".DetailActivity">
  <intent-filter>
    <action android:name="es.upsa.mimo.app.ACTION_VER">
      <data .... />
    <category android:name="android.intent.category.DEFAULT"/>
  </intent-filter>
</activity>
```



# Intent

## 2 Implícitas

`void setAction(String action)`

`String getAction()`

Establece la cadena de texto que referencia el `<action android:name=".....">` del componente a inicializar. Generalmente se expresa mediante contantes `Intent.ACTION_xxxxxxx`

`void setData(Uri data)`

`Uri getData()`

Establece el objeto `android.net.Uri` que describe el dato sobre el que el componente android debe realizar su acción. La invocación a este método implica eliminar el `mimeType` que previamente se hubiera podido fijar.

`void setType(String mimeType)`

`String getType()`

Establece el MIME del dato con el que el componente android debe realizar su acción. La invocación a este método implica eliminar el `Uri` del `data` que previamente se hubiera podido fijar.

`void setDataAndType(Uri data, String mimeType)`

MIME.

`void addCategory(String category)`

`void removeCategory(String category)`

`boolean hasCategory(String category)`

Añade, elimina y comprueba la existencia de una `category` en el `Intent`

# Intent

## 2 Implícitas

```
val intent : Intent = Intent();
intent.action = Intent.ACTION_VIEW
intent.setDataAndType( Uri.parse("http://mimo.com/Entity/a22"),
                      "vnd.mimo.instance/vnd.mimo.Entity");
if (intent.resolveActivity( PackageManager ) != null )
{
    // se puede iniciar la Activity ya que el sistema Android ha
    // encontrado al menos una que cumple con los requisitos
}
```

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="es.mimo.app" .... >
    <application .... >
        <activity name=".MasterActivity" . . ./>
        <activity name=".DetailActivity" . . . >
            <intent-filter>
                <action android:name="android.intent.action.VIEW"/>
                <data android:scheme="http"
                    android:host="mimo.com"
                    android:pathPattern="Entity/.*"
                    android:mimeType="vnd.mimo.instance/vnd.mimo.Entity"/>
                <category android:name="android.intent.category.DEFAULT"/>
            </intent-filter>
        </activity>
```

AndroidManifest.xml

# Intent

## 2 Implícitas

Si se encontrasen varios componentes que cumpliesen los requisitos establecidos por el Intent, el sistema mostraría un diálogo a través del que el usuario podría seleccionar el componente deseado

```
val newIntent: Intent = Intent()
newIntent.action = Intent.ACTION_VIEW
newIntent.setDataAndType( Uri.parse("http://mimo.com/Entity/a22"),
    "vnd.mimo.instance/vnd.mimo.Entity")
if (newIntent.resolveActivity( PackageManager ) != null )
{
    // Hay componentes que cumplen con los requisitos del intent

    val intentChooser = Intent.createChooser(newIntent, "Elige opción")
    // cuando se lance intentChooser siempre se mostrará el dialogo independientemente de si el
    usuario
    // hubiera seleccionado previamente un componente como opción por defecto
```

# Intent

## Extras

Independientemente de su tipo (explícito o implícito) un Intent puede transportar información extra que utilizará el componente destinatario

<code>Intent</code>	<code>Intent.putExtra(String key, boolean value)</code>	<code>boolean</code>	<code>Intent.getBooleanExtra(String key)</code>
<code>Intent</code>	<code>Intent.putExtra(String key, byte value)</code>	<code>byte</code>	<code>Intent.getByteExtra(String key)</code>
<code>Intent</code>	<code>Intent.putExtra(String key, short value)</code>	<code>short</code>	<code>Intent.getShortExtra(String key)</code>
<code>Intent</code>	<code>Intent.putExtra(String key, int value)</code>	<code>int</code>	<code>Intent.getIntExtra(String key)</code>
<code>Intent</code>	<code>Intent.putExtra(String key, long value)</code>	<code>long</code>	<code>Intent.getLongExtra(String key)</code>
<code>Intent</code>	<code>Intent.putExtra(String key, float value)</code>	<code>float</code>	<code>Intent.getFloatExtra(String key)</code>
<code>Intent</code>	<code>Intent.putExtra(String key, double value)</code>	<code>double</code>	<code>Intent.getDoubleExtra(String key)</code>
<code>Intent</code>	<code>Intent.putExtra(String key, CharSequence value)</code>	<code>CharSequence</code>	<code>Intent.getCharSequenceExtra(String key)</code>
<code>Intent</code>	<code>Intent.putExtra(String key, String value)</code>	<code>String</code>	<code>Intent.getStringExtra(String key)</code>
<code>Intent</code>	<code>Intent.putExtra(String key, Serializable value)</code>	<code>Serializable</code>	<code>Intent.getSerializableExtra(String key)</code>
<code>Intent</code>	<code>Intent.putExtra(String key, Bundle value)</code>	<code>Bundle</code>	<code>Intent.getBundleExtra(String key)</code>

# Intent

## Extras

**Intent** Intent.putExtra(String key, **boolean[]** value)

**Intent** Intent.putExtra(String key, **byte[]** value)

**Intent** Intent.putExtra(String key, **short[]** value)

**Intent** Intent.putExtra(String key, **int[]** value)

**Intent** Intent.putExtra(String key, **long[]** value)

**Intent** Intent.putExtra(String key, **float[]** value)

**Intent** Intent.putExtra(String key, **double[]** value)

**Intent** Intent.putExtra(String key, **CharSequence[]** value)

**Intent** Intent.putExtra(String key, **String[]** value)

**boolean[]** Intent.getBooleanArrayExtra(String key)

**byte[]** Intent.getByteArrayExtra(String key)

**short[]** Intent.getShortArrayExtra(String key)

**int[]** Intent.getIntArrayExtra(String key)

**long[]** Intent.getLongArrayExtra(String key)

**float[]** Intent.getFloatArrayExtra(String key)

**double[]** Intent.getDoubleArrayExtra(String key)

**CharSequence[]** Intent.getCharSequenceArrayExtra(String key)

**String[]** Intent.getStringArrayExtra(String key)

**Intent** Intent.putArrayListExtra(String key, **ArrayList<Integer>** value) **ArrayList<Integer>** Intent.getIntegerArrayListExtra(String key)

**Intent** Intent.putArrayListExtra(String key, **ArrayList<String>** value) **ArrayList<String>** Intent.getStringArrayListExtra(String key)

# Intent

## ¿Cómo lanzar y obtener en el destino un Intent?

Actividad origen

Actividad destino

```
val newIntent : Intent = Intent()
newIntent.action = Intent.ACTION_VIEW
newIntent.setDataAndType( Uri.parse("http://mimo.com/Entity/a22"),
                          "vnd.mimo.instance/vnd.mimo.Entity")
newIntent.putExtra("numero", 44)
if (newIntent.resolveActivity( PackageManager ) != null )
{
    startActivity( newIntent )
}
```

```
override fun onCreate(savedInstanceState : Bundle?)
{
    ....
    val theIntent: Intent = intent //from getIntent()/setIntent()
    val dataUri : Uri? = theIntent.data
    val dataMime : String? = theIntent.type
    val numero: Int = intent.getIntExtra("numero", -1);
    ....
}
```

*default value ↑*

# Intent

## ¿Cómo lanzar un Intent para obtener información de otra Activity

Actividad origen

```
class MainActivity : AppCompatActivity() {
    val filmActivityLauncher: ActivityResultLauncher<INPUT> = this.registerForActivityResult(FilmActivityResultContract()) {result->...}
    ...
    viewBinding.btGoFilm.setOnClickListener { val input : INPUT = ...
                                                filmActivityLauncher.launch( input )
                                            }

    inner class FilmActivityResultContract: ActivityResultContract<INPUT, RESULT>()
    {
        override fun createIntent(context: Context, input: INPUT): Intent {
            return Intent(this@MainActivity, FilmActivity::class.java).apply { //putExtra(FilmActivity.EXTRA_XXX, input.xxx)
                                                                    //putExtra(FilmActivity.EXTRA_YYY, input.yyy)
            }
        }

        override fun parseResult(resultCode: Int, intent: Intent?): RESULT {
            when ( resultCode ) {
                Activity.RESULT_OK          -> { //value1 = intent!!.getStringExtra("key1")
                                                //value2 = intent!!.getStringExtra("key2")
                                                // return ok RESULT with value1 & value2
                                                }
                Activity.RESULT_CANCELED -> // return canceled RESULT
            }
        }
    }
}
```

En la actividad debe registrarse un **ActivityResultContract** con el propósito de lanzar una actividad y obtener un resultado de ella. El dato que devuelve es un **ActivityResultLauncher**

Actividad destino

```
val xxx : String = intent.getStringExtra(EXTRA_XXX)
val yyy : String = intent.getStringExtra(EXTRA_YYY)
:::
{
    : : : :
    val intentR = Intent()
    intentR.putExtra("key1", "value_1");
    intentR.putExtra("key2", "value_2");
    if (condicion) setResult(Activity.RESULT_OK, intentR)
    else            setResult(Activity.RESULT_CANCELED)
    finish()
}
```

# Intent

## ¿Cómo lanzar un Intent para obtener información de otra Activity

Actividad origen

```
class MainActivity : AppCompatActivity() {
    val filmActivityLauncher: ActivityResultLauncher<INPUT> = this.registerForActivityResult(FilmActivityResultContract()) { result->... }
    ...
    viewBinding.btGoFilm.setOnClickListener { val input : INPUT = ...
                                                filmActivityLauncher.launch( input )
                                            }

    inner class FilmActivityResultContract: ActivityResultContract<INPUT, RESULT>() {
        override fun createIntent(context: Context, input: INPUT): Intent {
            return Intent(this@MainActivity, FilmActivity::class.java).apply { //putExtra(FilmActivity.EXTRA_XXX, input.xxx)
                                                                    //putExtra(FilmActivity.EXTRA_YYY, input.yyy)
            }
        }

        override fun parseResult(resultCode: Int, intent: Intent?): RESULT {
            when ( resultCode ) {
                Activity.RESULT_OK          -> { //value1 = intent!!.getStringExtra("key1")
                                                //value2 = intent!!.getStringExtra("key2")
                                                // return ok RESULT with value1 & value2
                                                }
                Activity.RESULT_CANCELED -> // return canceled RESULT
            }
        }
    }
}
```

El **ActivityResultContract** precisa dos tipos genéricos **INPUT** y **RESULT**:

- **INPUT**: representa el tipo dato que contiene los datos necesarios para crear el Intent con el que lanzar la actividad destino.
- **RESULT** : representa el tipo de dato del resultado devuelto por aquella actividad

```
val xxx : String = intent.getStringExtra(EXTRA_XXX)
val yyy : String = intent.getStringExtra(EXTRA_YYY)
:::::
{
    : : : :
    val intentR = Intent()
    intentR.putExtra("key1", "value_1");
    intentR.putExtra("key2", "value_2");
    if (condicion) setResult(Activity.RESULT_OK, intentR)
    else setResult(Activity.RESULT_CANCELED)
    finish()
}
```

Actividad destino



# Intent

## ¿Cómo lanzar un Intent para obtener información de otra Activity

Actividad origen

```
class MainActivity : AppCompatActivity() {
    val filmActivityLauncher: ActivityResultLauncher<INPUT> = this.registerForActivityResult(FilmActivityResultContract()) {result->...}
    ...
    viewBinding.btGoFilm.setOnClickListener { val input : INPUT = ...
                                                filmActivityLauncher.launch( input )
                                            }

    inner class FilmActivityResultContract: ActivityResultContract<INPUT, RESULT>()
    {
        override fun createIntent(context: Context, input: INPUT): Intent {
            return Intent(this@MainActivity, FilmActivity::class.java).apply { putExtra(FilmActivity.EXTRA_XXX, input.xxx)
                                                                              putExtra(FilmActivity.EXTRA_YYY, input.yyy)
                                                                              }
        }

        override fun parseResult(resultCode: Int, intent: Intent?): RESULT {
            when ( resultCode ) {
                Activity.RESULT_OK          -> { //value1 = intent!!.getStringExtra("key1")
                                                //value2 = intent!!.getStringExtra("key2")
                                                // return ok RESULT with value1 & value2
                                                }
                Activity.RESULT_CANCELED -> // return canceled RESULT
            }
        }
    }
}
```

Para lanzar la actividad destino se emplea el método `ActivityResultLauncher#launch()` al que como parámetro se le pasa un objeto de tipo `INPUT` con los datos necesarios para crear el Intent.

Actividad destino

```
val xxx : String = intent.getStringExtra(EXTRA_XXX)
val yyy : String = intent.getStringExtra(EXTRA_YYY)
:::::
{
    : : : :
    val intentR = Intent()
    intentR.putExtra("key1", "value_1");
    intentR.putExtra("key2", "value_2");
    if (condicion) setResult(Activity.RESULT_OK, intentR)
    else             setResult(Activity.RESULT_CANCELED)
    finish()
}
```

# Intent

## ¿Cómo lanzar un Intent para obtener información de otra Activity

Actividad origen

```
class MainActivity : AppCompatActivity() {
    val filmActivityLauncher: ActivityResultLauncher<INPUT> = this.registerForActivityResult(FilmActivityResultContract()) {result->...}
    ...
    viewBinding.btGoFilm.setOnClickListener { val input : INPUT = ...
                                                filmActivityLauncher.launch( input )
                                            }

    inner class FilmActivityResultContract: ActivityResultContract<INPUT, RESULT>()
    {
        override fun createIntent(context: Context, input: INPUT): Intent {
            return Intent(this@MainActivity, FilmActivity::class.java).apply { putExtra(FilmActivity.EXTRA_XXX, input.xxx)
                                                                                putExtra(FilmActivity.EXTRA_YYY, input.yyy)
            }
        }

        override fun parseResult(resultCode: Int, intent: Intent?): RESULT {
            when ( resultCode ) {
                Activity.RESULT_OK          -> { //value1 = intent!!.getStringExtra("key1")
                                                //value2 = intent!!.getStringExtra("key2")
                                                // return ok RESULT with value1 & value2
                                                }
                Activity.RESULT_CANCELED -> // return canceled RESULT
            }
        }
    }
}
```

`ActivityResultLauncher#launch()` es quien ejecuta `ActivityResultContract#createIntent()` que recibe como parámetro el objeto `INPUT`. Este último método crea el Intent con el que se lanzará la activity destino.

Actividad destino

```
val xxx : String = intent.getStringExtra(EXTRA_XXX)
val yyy : String = intent.getStringExtra(EXTRA_YYY)
:::::
{
    : : : :
    val intentR = Intent()
    intentR.putExtra("key1", "value_1");
    intentR.putExtra("key2", "value_2");
    if (condicion) setResult(Activity.RESULT_OK, intentR)
    else             setResult(Activity.RESULT_CANCELED)
    finish()
}
```

# Intent

## ¿Cómo lanzar un Intent para obtener información de otra Activity

Actividad origen

```
class MainActivity : AppCompatActivity() {
    val filmActivityLauncher: ActivityResultLauncher<INPUT> = this.registerForActivityResult(FilmActivityResultContract()) {result->...}
    ...
    viewBinding.btGoFilm.setOnClickListener { val input : INPUT = ...
                                                filmActivityLauncher.launch( input )
                                            }

    inner class FilmActivityResultContract: ActivityResultContract<INPUT, RESULT>()
    {
        override fun createIntent(context: Context, input: INPUT): Intent {
            return Intent(this@MainActivity, FilmActivity::class.java).apply { putExtra(FilmActivity.EXTRA_XXX, input.xxx)
                                                                                putExtra(FilmActivity.EXTRA_YYY, input.yyy)
                                                                            }
        }

        override fun parseResult(resultCode: Int, intent: Intent?): RESULT {
            when ( resultCode ) {
                Activity.RESULT_OK          -> { //value1 = intent!!.getStringExtra("key1")
                                                //value2 = intent!!.getStringExtra("key2")
                                                // return ok RESULT with value1 & value2
                                            }
                Activity.RESULT_CANCELED -> // return canceled RESULT
            }
        }
    }
}
```

La actividad destino recupera la información enviada por la actividad origen de su Intent.

Actividad destino

```
val xxx : String = intent.getStringExtra(EXTRA_XXX)
val yyy : String = intent.getStringExtra(EXTRA_YYY)
:::::
{
    : : : :
    val intentR = Intent()
    intentR.putExtra("key1", "value_1");
    intentR.putExtra("key2", "value_2");
    if (condicion) setResult(Activity.RESULT_OK, intentR)
    else             setResult(Activity.RESULT_CANCELED)
    finish()
}
```

# Intent

## ¿Cómo lanzar un Intent para obtener información de otra Activity

Actividad origen

```
class MainActivity : AppCompatActivity() {
    val filmActivityLauncher: ActivityResultLauncher<INPUT> = this.registerForActivityResult(FilmActivityResultContract()) { result->... }
    ...
    viewBinding.btGoFilm.setOnClickListener { val input : INPUT = ...
                                                filmActivityLauncher.launch( input )
                                            }

    inner class FilmActivityResultContract: ActivityResultContract<INPUT, RESULT>() {
        override fun createIntent(context: Context, input: INPUT): Intent {
            return Intent(this@MainActivity, FilmActivity::class.java).apply { putExtra(FilmActivity.EXTRA_XXX, input)
                                                                              putExtra(FilmActivity.EXTRA_YYY, input)
            }
        }

        override fun parseResult(resultCode: Int, intent: Intent?): RESULT {
            when ( resultCode ) {
                Activity.RESULT_OK          -> { //value1 = intent!!.getStringExtra("key1")
                                                //value2 = intent!!.getStringExtra("key2")
                                                // return ok RESULT with value1 & value2
                                                }
                Activity.RESULT_CANCELED -> // return canceled RESULT
            }
        }
    }
}
```

La activity destino crea un Intent el que, a través de EXTRAS, incluye todos los datos a devolver a la activity origen.

Para fijar este intent como resultado a devolver se ejecuta el método **ActivityResult#setResult()**. Su primer parámetro **resultCode** indica:

- **Activity.RESULT\_OK**: la operación destino se ha realizado.
- **Activity.RESULT\_CANCELED**: la operación se ha cancelado.

El resultado se enviará cuando la actividad destino finalice (se fuerza invocando al método **ActivityResult#finish()**)

```
val xxx : String = intent.getStringExtra(EXTRA_XXX)
val yyy : String = intent.getStringExtra(EXTRA_YYY)
:::::
{
    : : : :
    val intentR = Intent()
    intentR.putExtra("key1", "value_1");
    intentR.putExtra("key2", "value_2");
    if (condicion) setResult(Activity.RESULT_OK, intentR)
    else            setResult(Activity.RESULT_CANCELED)
    finish()
}
```

Actividad destino

# Intent

## ¿Cómo lanzar un Intent para obtener información de otra Activity

Actividad origen

```
class MainActivity : AppCompatActivity() {  
    val filmActivityLauncher: ActivityResultLauncher<INPUT> = this.registerForActivityResult(FilmActivityResultContract()) {result->...}  
    ...  
    viewBinding.btGoFilm.setOnClickListener { val input : INPUT = ...  
                                                filmActivityLauncher.launch( input )  
                                            }  
  
    inner class FilmActivityResultContract: ActivityResultContract<INPUT, RESULT>() {  
        override fun createIntent(context: Context, input: INPUT): Intent {  
            return Intent(this@MainActivity, FilmActivity::class.java).apply { putExtra(FilmActivity.EXTRA_XXX, input.xxx)  
                                     putExtra(FilmActivity.EXTRA_YYY, input.yyy)  
            }  
        }  
  
        override fun parseResult(resultCode: Int, intent: Intent?): RESULT {  
            when ( resultCode ) {  
                Activity.RESULT_OK -> { //value1 = intent!!.getStringExtra("key1")  
                                         //value2 = intent!!.getStringExtra("key2")  
                                         // return ok RESULT with value1 & value2  
                                     }  
                Activity.RESULT_CANCELED -> // return canceled RESULT  
            }  
        }  
    }  
}
```

El resultado devuelto por destino se procesa en el método **ActivityResultContract#parseResult()**. El primer parámetro (resultCode) toma el valor **Activity.RESULT\_OK** o **Activity.RESULT\_CANCELED** indicado por la actividad destino. Su segundo parámetro es el Intent devuelto por destino con los datos adjuntos a través de EXTRAS.

A partir de estos EXTRAS, se crea y devuelve el objeto **RESULT** que representará el resultado devuelto por destino.

```
val xxx : String = intent.getStringExtra(EXTRA_XXX)  
val yyy : String = intent.getStringExtra(EXTRA_YYY)  
:::::  
{  
    : : : :  
    val intentR = Intent()  
    intentR.putExtra("key1", "value_1");  
    intentR.putExtra("key2", "value_2");  
    if (condicion) setResult(Activity.RESULT_OK, intentR)  
    else setResult(Activity.RESULT_CANCELED)  
    finish()  
}
```

Actividad destino

# Intent

## ¿Cómo lanzar un Intent para obtener información de otra Activity

Actividad origen

```
class MainActivity : AppCompatActivity() {
    val filmActivityLauncher: ActivityResultLauncher<INPUT> = this.registerForActivityResult(FilmActivityResultContract()) {result->...}
    ...
    viewBinding.btGoFilm.setOnClickListener { val input : INPUT = ...
                                                filmActivityLauncher.launch( input )
                                            }

    inner class FilmActivityResultContract: ActivityResultContract<INPUT, RESULT>()
    {
        override fun createIntent(context: Context, input: INPUT): Intent {
            return Intent(this@MainActivity, FilmActivity::class.java).apply { putExtra(FilmActivity.EXTRA_XXX, input.xxx)
                                                                              putExtra(FilmActivity.EXTRA_YYY, input.yyy)
                                                                              }
        }

        override fun parseResult(resultCode: Int, intent: Intent?): RESULT {
            when ( resultCode ) {
                Activity.RESULT_OK          -> { //value1 = intent!!.getStringExtra("key1")
                                                //value2 = intent!!.getStringExtra("key2")
                                                // return ok RESULT with value1 & value2
                                                }
                Activity.RESULT_CANCELED -> // return canceled RESULT
            }
        }
    }
}
```

El objeto **RESULT** devuelto por **FilmActivityResultContract#parseResult()** llega aquí y a través de esta lambda se procesará.

```
val xxx : String = intent.getStringExtra(EXTRA_XXX)
val yyy : String = intent.getStringExtra(EXTRA_YYY)
:::::
{
    : : : :
    val intentR = Intent()
    intentR.putExtra("key1", "value_1");
    intentR.putExtra("key2", "value_2");
    if (condicion) setResult(Activity.RESULT_OK, intentR)
    else             setResult(Activity.RESULT_CANCELED)
    finish()
}
```

Actividad destino

# Intent

## Ejemplos:

```
public static void invokeWebBrowser(Context context)
{
    val newIntent : Intent = Intent(Intent.ACTION_VIEW)
    newIntent.data = Uri.parse("http://www.google.com")
    context.startActivity(newIntent)
}
```

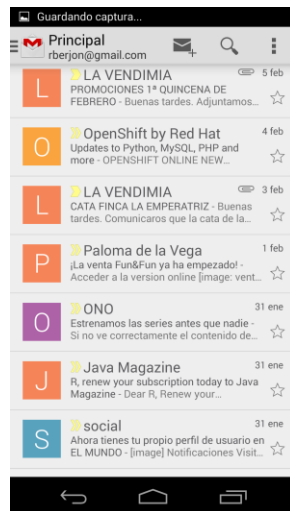
```
public static void call(Context context)
{
    val newIntent : Intent = Intent(Intent.ACTION_CALL)
    newIntent.data = Uri.parse("tel:555-555-555");
    context.startActivity(newIntent);
}
```

```
public static void showMapAtLatLng(Context context)
{
    val newIntent : Intent = Intent(Intent.ACTION_VIEW)
    intent.data = Uri.parse("geo:0,0?z=4&q=restaurantes")
    context.startActivity(newIntent)
}
```

```
fun sendEmail(Context context) : Unit
{
    val to : Array<String> = arrayOf<String>("pepe@gmail.com",
    "paco@me.com")
    val cc : Array<String> = arrayOf("ana@outlook.com")
    val emailIntent : Intent = Intent(Intent.ACTION_SEND)
    emailIntent.setDataAndType(Uri.parse("mailto:"), "text/plain")
    emailIntent.putExtra(Intent.EXTRA_EMAIL, to)
    emailIntent.putExtra(Intent.EXTRA_CC, cc)
    emailIntent.putExtra(Intent.EXTRA_SUBJECT, "Asunto del mensaje")
    emailIntent.putExtra(Intent.EXTRA_TEXT, "Texto del mensaje")
    val sendEmailChooser : Intent = Intent.createChooser(emailIntent, "Elige
    email app")
    context.startActivity( sendEmailChooser )
}
```

# Intent

## Actividades y tareas





# Intent

## Actividades y tareas

*Una tarea es un conjunto de actividades relacionadas que se organizan a través de una pila LIFO («back stack» )*

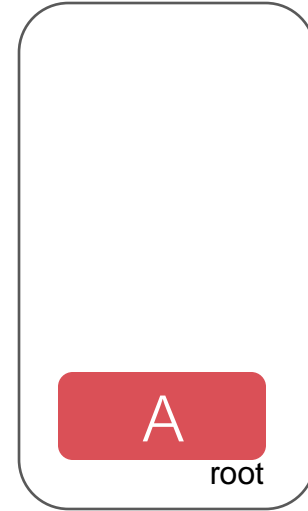
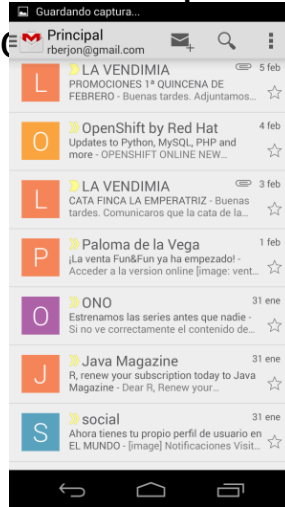
Por defecto, cada aplicación tiene una única tarea

# Intent

## Actividades y tareas

Cuando arranca una aplicación se crea una tarea y se

introduce en la pila una entrada con la actividad mostrada

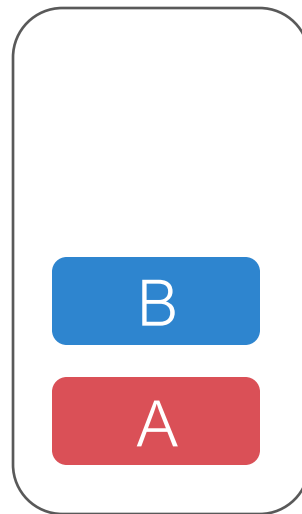
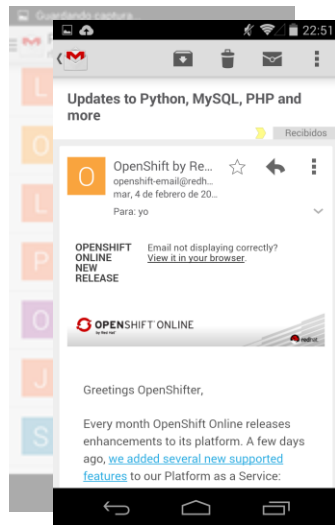


Tarea

# Intent

## Actividades y tareas

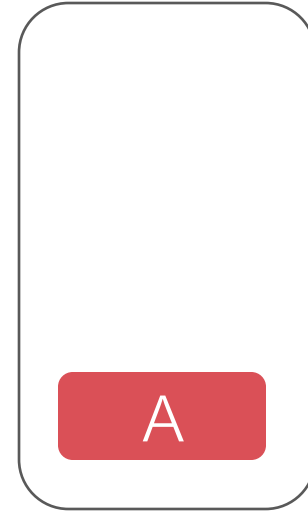
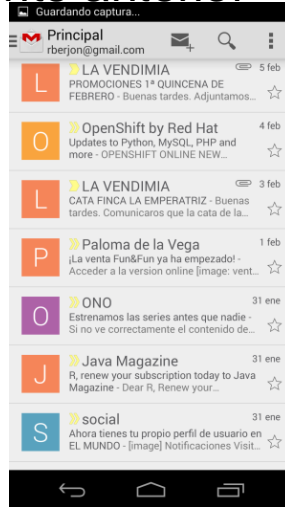
Si desde una actividad A se inicia otra B, por defecto B se ubicará en la misma tarea a la que pertenece A.



# Intent



## Actividades y tareas

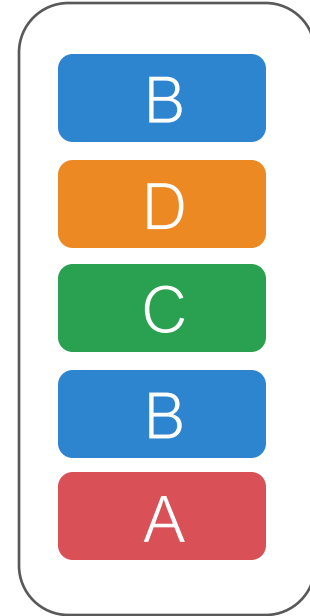
Cuando finaliza una actividad se elimina de la pila. En ese instante pasará a estar en la cima de la pila la actividad inmediatamente anterior



# Intent

## Actividades y tareas

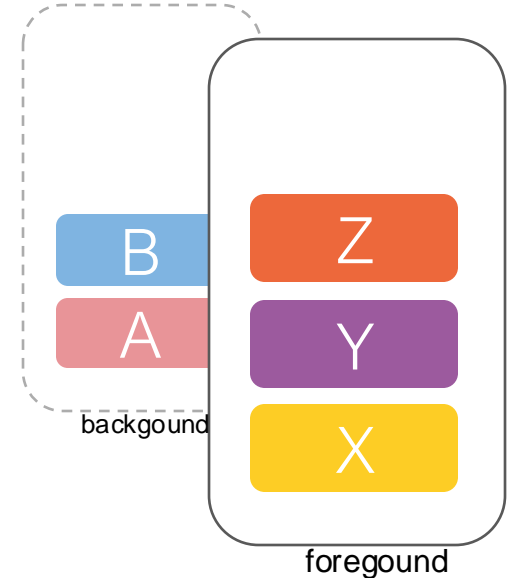
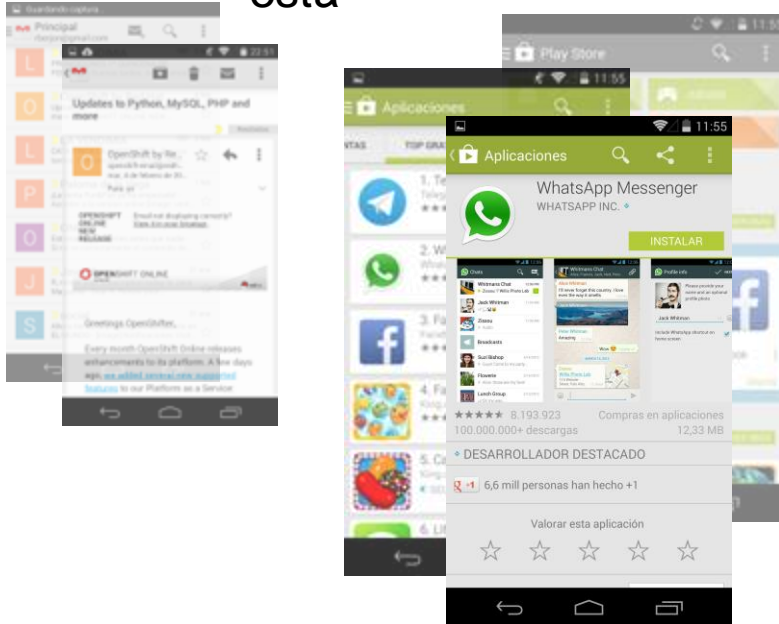
- 1º Arranca la aplicación iniciándose la actividad A
- 2º El usuario inicia la actividad B
- 3º El usuario inicia la actividad C
- 4º El usuario inicia la actividad D
- 5º El usuario inicia la actividad B
- 6º El usuario pulsa el botón 
- 7º El usuario pulsa el botón 



# Intent

## Actividades y tareas

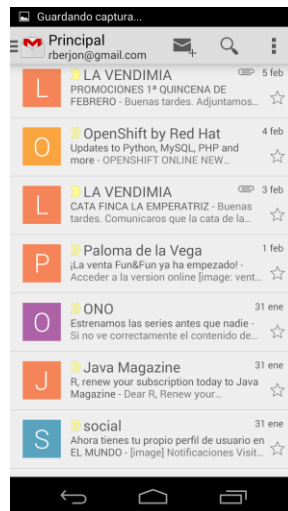
Las tareas, en su conjunto, estarán en primer plano o en segundo, en función de si la aplicación también lo está



# Intent

## Actividades y tareas

Modificación del comportamiento por defecto



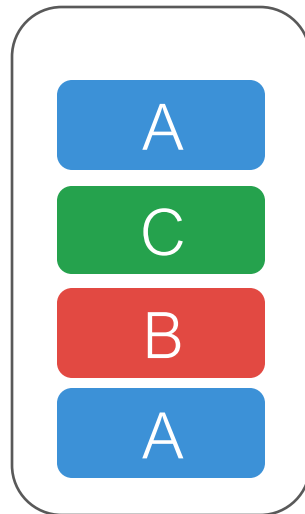
# Actividades y tareas

android:launchMode

# Intent

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android" package="es.mimo.tasks"
.... >
    <application ..... >
        <activity android:name=".A" android:launchMode="standard"> ::: </activity>
        <activity android:name=".B" android:launchMode="standard"> ::: </activity>
        <activity android:name=".C" android:launchMode="standard"> ::: </activity>
    </application>
</manifest>
```

- |                 |              |
|-----------------|--------------|
| ❶ Se accede a A | ❺ Se pulsa ↩ |
| ❷ Se accede a B | ❻ Se pulsa ↩ |
| ❸ Se accede a C | ❼ Se pulsa ↩ |
| ❹ Se accede a A | ❽ Se pulsa ↩ |





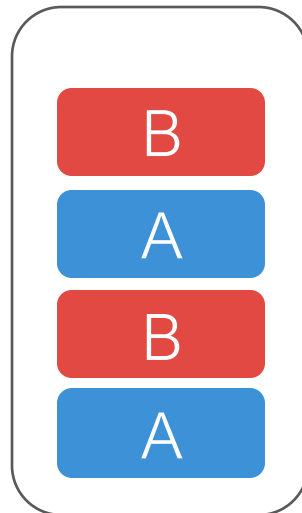
# Actividades y tareas

android:launchMode

# Intent

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android" package="es.mimo.tasks"
.... >
    <application ..... >
        <activity android:name=".A" android:launchMode="standard"> ::: </activity>
        <activity android:name=".B" android:launchMode="singleTop"> ::: </activity>
        <activity android:name=".C" android:launchMode="standard"> ::: </activity>
    </application>
</manifest>
```

- |                 |              |
|-----------------|--------------|
| ❶ Se accede a A | ❹ Se pulsa ↩ |
| ❷ Se accede a B | ❺ Se pulsa ↩ |
| ❸ Se accede a B | ❻ Se pulsa ↩ |
| ❹ Se accede a A | ❼ Se pulsa ↩ |
| ❺ Se accede a B |              |



@Override  
public void onNewIntent(Intent intent)

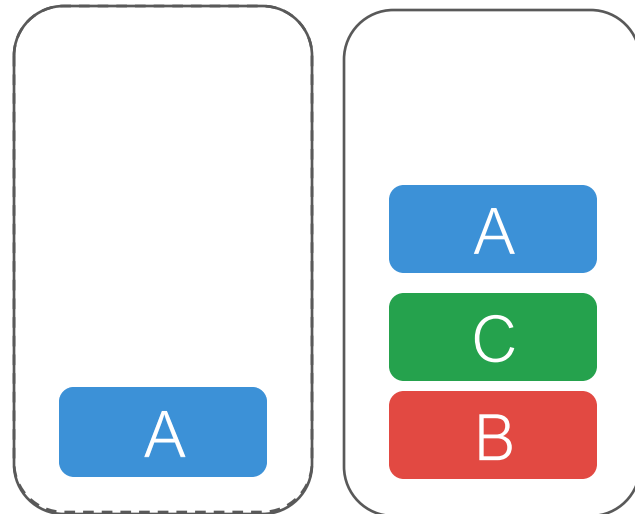
# Actividades y tareas

android:launchMode

# Intent

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android" package="es.mimo.tasks"
.... >
    <application ..... >
        <activity android:name=".A" android:launchMode="standard"> ::: </activity>
        <activity android:name=".B" android:launchMode="singleTask"> ::: </activity>
        <activity android:name=".C" android:launchMode="standard"> ::: </activity>
    </application>
</manifest>
```

- ❶ Se accede a A
- ❷ Se accede a B
- ❸ Se accede a C
- ❹ Se accede a A
- ❺ Se accede a B
- ❻ Se pulsa ↩
- ❼ Se pulsa ↩



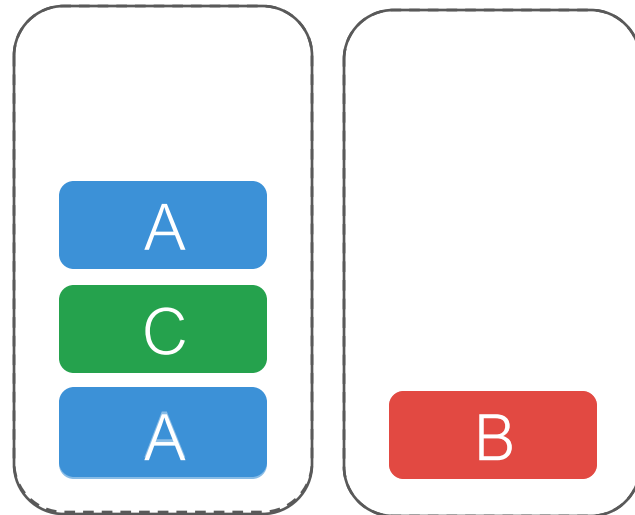
# Actividades y tareas

android:launchMode

# Intent

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android" package="es.mimo.tasks"
.... >
    <application ..... >
        <activity android:name=".A" android:launchMode="standard"> ::: </activity>
        <activity android:name=".B" android:launchMode="singleInstance"> ::: </activity>
        <activity android:name=".C" android:launchMode="standard"> ::: </activity>
    </application>
</manifest>
```

- ❶ Se accede a A
- ❷ Se accede a B
- ❸ Se accede a C
- ❹ Se accede a A



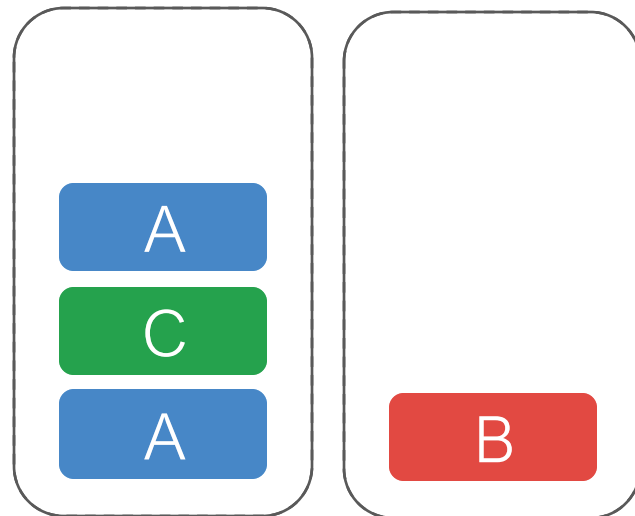
# Actividades y tareas

android:launchMode

# Intent

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android" package="es.mimo.app33"
.... >
    <application ..... >
        <activity android:name=".A" android:launchMode="standard"> ::: </activity>
        <activity android:name=".B" android:launchMode="singleInstance"> ::: </activity>
        <activity android:name=".C" android:launchMode="standard"> ::: </activity>
    </application>
</manifest>
```

- ❶ Se accede a A
- ❷ Se accede a B
- ❸ Se accede a C
- ❹ Se accede a A
- ❺ Se accede a B
- ❻ Se pulsa ↩
- ❼ Se pulsa ↩
- ❽ Se pulsa ↩
- ❾ Se pulsa ↩




# Actividades y tareas

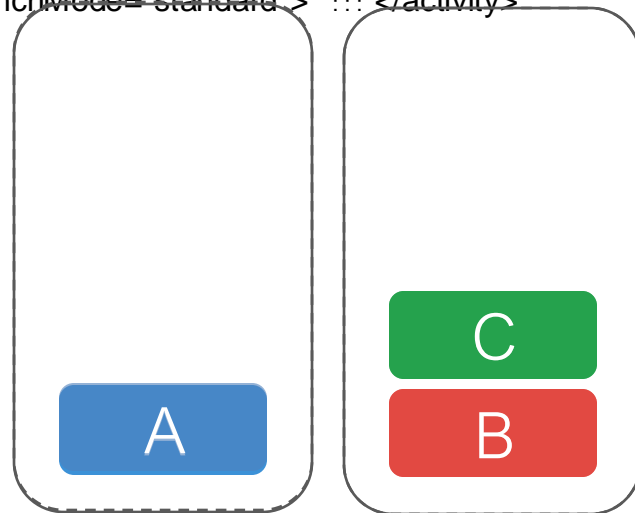
android:taskAffinity

# Intent

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android" package="es.mimo.tasks"
.... >
  <application .... android:taskAffinity="es.mimo.tasks" ... >
    <activity android:name=".A" android:launchMode="standard"> :::</activity>
    <activity android:name=".B" android:launchMode="singleTask"
      android:taskAffinity="es.mimo.tasks.B"> ::: </activity>
    <activity android:name=".C" android:launchMode="standard"> ::: </activity>
  </application>
</manifest>
```

- ❶ Se accede a A
- ❷ Se accede a B
- ❸ Se accede a C
- ❹ Se pulsa 
- ❺ Se accede aplicación

❻ Se pulsa 



# Intent

## Flags

Representa metainformación que se puede añadir a un Intent con el objeto de modificar el comportamiento de la actividad que se pretende arrancar

Intent Intent.addFlags(int flags)

Intent.FLAG\_ACTIVITY\_NEW\_TASK

Crea una nueva tarea si no existiera ya una con una instancia de la Activity.

Intent.FLAG\_ACTIVITY\_CLEAR\_TASK

Debe ir en unión a FLAG\_ACTIVITY\_NEW\_TASK. Si encuentra una tarea con la actividad solicitada, elimina de la tarea el resto de actividades

Intent.FLAG\_ACTIVITY\_CLEAR\_TOP

Se comporta como launchMode="singleTask"

Intent.  
FLAG\_ACTIVITY\_MULTIPLE\_TASK

Crea una nueva tarea y lanza la actividad en ella

Intent.FLAG\_ACTIVITY\_SINGLE\_TOP

Se comporta como launchMode="singleTop"

Intent.FLAG\_ACTIVITY\_PREVIOUS\_IS  
\_TOP

Se obvia el comportamiento launchMode="singleTop" si la actividad que lanza el Intent está en la cima de la pila y a