

# Intents

# Intents (I)

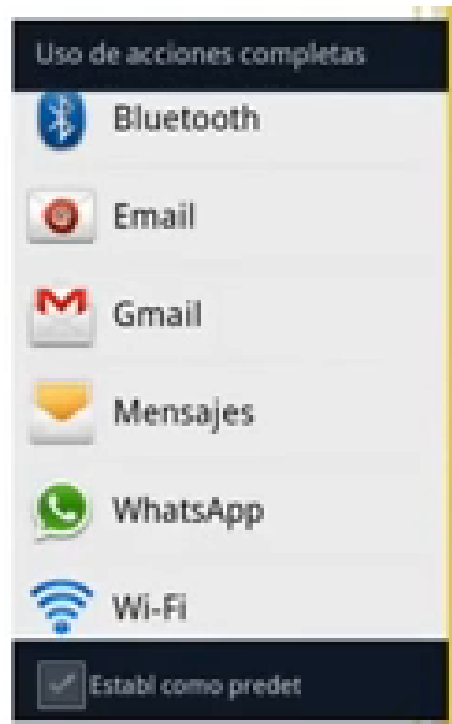
- ▶ Representan la **voluntad de realizar una acción o tarea.**
- ▶ Permiten lanzar una actividad, o **servicio de nuestra aplicación o de una aplicación diferente.**
- ▶ Usaremos Intents cada vez que queramos:
  - \*  
\*\* **Lanzar una actividad:**
    - *startActivity()*
    - *startActivityForResult()*
  - \*  
\*\* **Lanzar un servicio (*startService()*) y comunicarnos con un servicio (*bindService()*).**
  - \*  
\*\* **Lanzar un anuncio de tipo broadcast:**
    - *sendBroadcast()*
    - *sendOrderedBroadcast()*
    - *sendStickyBroadcast()*

# Intents (II)

- ▶ Los **Intents** también **permiten indicar acciones genéricas (Intents implícitos)**:

- Como quiero mandar un mensaje,

En cuyo caso **se lanzaría la aplicación registrada por defecto para este propósito**, de las que tenga instaladas en mi dispositivo.



# Tipos de Intents

## 1. Intents explícitos:

- ▶ Indicamos exactamente qué componente queremos lanzar: una *activity*, un *servicio*,...

## 1. Intents implícitos:

- ▶ Indicamos acciones de tipo **abstracto**, como “*quiero tomar una foto*”, “*quiero enviar un mensaje*”, *elegir un contacto*,...
- ▶ Se **resuelven en tiempo de ejecución**.
- ▶ Su **ejecución dependerá del software que el usuario tenga instalado en su dispositivo**:
  - El sistema mirará cuántas aplicaciones han registrado la posibilidad de ejecutar ese tipo de acción.
  - Si **encuentra varias** el sistema puede preguntar al usuario qué **actividad prefiere** usar.

# Partes de un Intent (I)

1. **Nombre del componente** (INTENTS EXPLÍCITOS)
2. **Acción** (INTENTS IMPLÍCITOS)
3. **Datos**
4. **Categoría** (INTENTS IMPLÍCITOS)
5. **Tipo**
6. **Extras**
7. **Flags**

# Partes de un Intent (II)

## 1. Nombre del componente: (INTENTS EXPLÍCITOS)

- ▶ Identifica el componente exacto que queremos lanzar con el Intent.
- ▶ Hay que indicar el **nombre de la clase**. Puede ser una **clase nuestra** o **del sistema**.
- ▶ El **nombre de la clase** puede ir precedido por el nombre del paquete donde está la clase:

*org.example.ejercicio1.AcercaDe*

## 2. Acción: (INTENTS IMPLÍCITOS)

- ▶ Es una **cadena de texto** que indica la acción que se va a realizar o **en caso de un *Receptor de anuncios (Broadcast receiver)*** la acción que tuvo lugar y queremos reportar.
- ▶ La clase **Intent** **define constantes** representando varias acciones predeterminadas.
- ▶ Se pueden **inventar acciones propias**, pero deben llevar prefijado el nombre del paquete de la aplicación:

*org.example.ejer1.MUESTRA\_PUNTUACIONES.*

## Partes de un Intent (III)

### Constantes predeterminadas para acciones

Constante	Componente a lanzar	Acción
ACTION_CALL	Actividad	Abre una activity que <i>Inicia una llamada de teléfono</i>
ACTION_DIAL	Actividad	Abre el dialer con el número de tfno marcado, pero NO LLAMA.
ACTION_EDIT	Actividad	Abre una activity que <i>visualiza datos para que el usuario los edite</i> . Si son texto, abre un editor de texto, si son gráfico abre un editor gráfico,...
ACTION_MAIN	Actividad	Permite arrancar actividades, normalmente como <b>actividad principal de una tarea</b> (sin datos de entrada y sin devolver datos)
ACTION_SYNC	Actividad	Sincroniza datos en un servidor con los datos de un dispositivo móvil

## Partes de un Intent (IV)

### Constantes predeterminadas para acciones

Constante	Componente a lanzar	Acción
<b>ACTION_BATTERY_LOW</b>	Receptor de anuncios	Advertencia de batería baja.
<b>ACTION_HEADSET_PLUG</b>	Receptor de anuncios	Los auriculares han sido conectados o desconectados
<b>ACTION_SCREEN_ON</b>	Receptor de anuncios	La pantalla es activada
<b>ACTION_TIMEZONE_CHANGED</b>	Receptor de anuncios	Se cambia la selección de zona horaria.



# Partes de un Intent (V)

## ¿Cómo se indica la ACTION?

```
Intent miIntent = new  
    Intent(Intent.ACTION_DIAL);
```

*O también:*

```
Intent miIntent = new Intent();  
miIntent.setAction(Intent.ACTION_DIAL);
```

# Partes de un Intent (VI)

## 3. Datos:

- ▶ Indicamos los datos con los que va a trabajar el Intent.
- ▶ Se expresan por medio de **URIs**.
- ▶ Ejemplos de URIs son:

*tel:963228525,*

*http://www.google.com,*

*content://call\_log/calls* → registro de llamadas

*Uri.parse("geo:0,0?*

*q=1600+Pennsylvania+Ave+Washington+DC")*

carga los datos indicados en un mapa

*Uri.parse("tel:+15555555");* → número para marcar en el DIALER

# Partes de un Intent (VII)

Algunos ejemplos de parejas ACTION/DATA son:

- ▶ ACTION\_VIEW *content://contacts/people/1* – Muestra información acerca de la persona cuyo identificador es 1.
- ▶ ACTION\_DIAL *content://contacts/people/1* – Muestra el DIALER con esa persona rellena
- ▶ ACTION\_VIEW *tel:123* – Muestra el DIALER con ese número marcado.  
Fijaros que ACTION\_VIEW realiza acciones razonables para una URI dada.
- ▶ ACTION\_DIAL *tel:123* – Hace lo mismo que el ejemplo anterior.
- ▶ ACTION\_EDIT *content://contacts/people/1* – Permite editar la información de la persona cuyo identificador es 1.
- ▶ ACTION\_VIEW *content://contacts/people/* -- Muestra una lista de personas, y el usuario puede navegar por esa lista.  
Es el típico listado de contactos que vemos. Si elegimos uno de ellos, lanzaremos un nuevo intent { ACTION\_VIEW *content://contacts/N* } que arrancará una nueva activity para mostrar esa persona.

# Partes de un Intent (VIII)

- ▶ Cada acción está asociada con diferentes especificaciones de datos (URIs).

Por ejemplo:

- ✖ si la acción es **ACTION\_EDIT**, “*data*” contendrá la *URI del documento a editar*.
- ✖ Si la acción es **ACTION\_CALL**, el campo “*data*” sería una *URI con tel: con el número de teléfono a llamar*.
- ✖ Si la acción es **ACTION\_VIEW** y la **URI es de tipo http:** , *la actividad receptora descargará y visualizará aquello a lo que se refiera la URI*.

## Partes de un Intent (IX)

### ¿Cómo se especifican los DATOS en el Intent?

```
Intent miIntent = new  
    Intent(Intent.ACTION_DIAL,  
        Uri.parse("tel:+34985621245");
```

*O también:*

```
Intent miIntent = new Intent(Intent.ACTION_DIAL);  
miIntent.setData(  
    Uri.parse("tel:+34985621245");
```

## 4. Categoría: (INTENTS IMPLÍCITOS)

- ▶ Es un String con información adicional sobre el tipo de componente que debería ser lanzado o que gestiona el Intent (*complementa a la acción*). *Quién va a llevar a cabo esa acción.*
- ▶ Se pueden usar **varias categorías en un mismo Intent**.
- ▶ Hay varias **CATEGORÍAS PREDEFINIDAS** como:

Categoría	Significado
CATEGORY_BROWSABLE	La actividad lanzada puede ser con seguridad invocada por el navegador para mostrar los datos referenciados por un enlace – por ejemplo, una imagen o un mensaje de correo electrónico.
CATEGORY_GADGET	La actividad puede ser embebida dentro de otra actividad que aloja gadgets.
CATEGORY_HOME	Es la Activity HOME, la primera pantalla que ve el usuario cuando el dispositivo está encendido o cuando la tecla HOME es presionada.

# Partes de un Intent (XI)

Categoría	Significado
CATEGORY_LAUNCHER	La actividad DEBERÍA ser mostrada en el lanzador de aplicaciones de nivel superior.
CATEGORY_PREFERENCE	La actividad a lanzar es un panel de preferencias.

► Estas categorías tiene sentido cuando se asocian a una ACCIÓN en concreto.

► Por ejemplo:

**ACTION\_MAIN** y **CATEGORY\_LAUNCHER**: primera actividad a lanzar

**ACTION\_MAIN** y **CATEGORY\_HOME**: lanza la pantalla HOME

### ¿Cómo se especifican la CATEGORÍA en el Intent?

```
Intent miIntent = new  
    Intent(Intent.ACTION_VIEW,  
        Uri.parse("http://www.google.com");  
  
miIntent.addCategory ( CATEGORY_BROWSABLE);
```



## 5. Tipo:

- ▶ Especifica el **tipo de datos MIME** asociado a los datos de un **INTENT**.
- ▶ Normalmente, el tipo es inferido por android desde los datos suministrados.
- ▶ Si lo especificamos explícitamente, desactivamos esa evaluación de Android y fuerza al tipo explícito.
- ▶ Es **MUY IMPORTANTE**, ya que si no, corremos el riesgo de intentar abrir un archivo de texto con un visor de imágenes, o situaciones similares.
- ▶ Ejemplos de tipos MIME son:

*text/xml,*

*image/jpeg,*

*audio/mp3*

*image/\**

*text/plain*

## Partes de un Intent (XIV)

- ▶ Hay muchas situaciones en las que **el contenido se puede inferir sólo con el URI**.
  - En particular, las **URI “content:”** indican que los **datos están en el dispositivo** y son **controladas por un Provider**.
- ▶ El **tipo MIME** se puede especificar explícitamente (y **se debe cuando no se pueda deducir**).
- ▶ Para esto hay varios **métodos**:
  - **setType** especifica sólo el tipo **MIME**
  - **setDataAndType** especifica los datos y el tipo **MIME**.
  - **getData** lee la **URI** y **getType** lee el tipo **MIME**

### ¿Cómo se especifican el TIPO en el Intent?

```
Intent miIntent = new  
    Intent(Intent.ACTION_SEND);  
miIntent.putExtra(Intent.EXTRA_TEXT, "Esto es mi  
    texto a enviar.");  
  
miIntent.setType ( "text/plain");
```

# Partes de un Intent (XVI)

## 6. Extras:

- ▶ Es un **BUNDLE** con cualquier información adicional que será recibida por el componente lanzado.
- ▶ Está formada por un **conjunto de pares clave-valor**. *(es un BUNDLE)*
  - *De hecho, se pueden pasar Bundles enteros al Intent mediante **putExtras** y leerlo mediante **getExtras**.*
  - También se **pueden pasar de forma independiente** con **putExtra** y leerlos con **getExtra**.

Ejemplo:

```
intent.putExtra ("usuario", "Pepito Pérez");
```

```
intent.putExtra ("edad",27);
```

- ▶ Al igual que algunas ACCIONES están ligadas a ciertas URIs, también están ligadas a ciertos EXTRAS.

# Partes de un Intent (XVII)

## Por ejemplo:

- ✧ **ENVIAR UN EMAIL** lleva **ASOCIADO** varios **EXTRAS** para indicar: direcciones destinatarios, asunto,...

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

```
miIntent.putExtra(Intent.EXTRA_MAIL,
```

```
    new String[]{
```

```
        "alu1@educastur.es", "alu2@educastur.es",
```

```
        "alu3@google.com"} );
```

```
emailIntent.putExtra(Intent.EXTRA_CC, cc); → cc es un String[]
```

```
emailIntent.putExtra(Intent.EXTRA_SUBJECT, asunto); → asunto String
```

```
emailIntent.putExtra(Intent.EXTRA_TEXT, mensaje); → mensaje  
                                                       CharSequence
```

# Partes de un Intent (XVIII)

## Por ejemplo:

- ※ **ACTION\_TIMEZONE\_CHANGED** tiene un extra llamado “**time-zone**” que identifica la **nueva zona horaria**.

Ejemplo: `miIntent.putExtra(“time-zone”, nuevaZona);`

- ※ **ACTION\_HEADSET\_PLUG** tiene un extra llamado “**state**” que **indica el estado del conector para los cascos** (*enchufado o desenchufado*) y otro llamado “**name**” que indica el **tipo de cascos**.
- ※ Si creásemos una acción **MOSTRAR\_COLOR**, deberíamos definir un **extra** con clave “**color**”.

## 6. Flags :

- ▶ Son flags de varios tipos. Muchos **indican al sistema cómo tiene que lanzar una actividad**. (relacionado con BackStack). (*ver la documentación*).

## 7. Flags :

- ▶ Especifican cómo debería ser gestionado el Intent.
- ▶ Son flags de varios tipos. Muchos **indican al sistema cómo tiene que lanzar una actividad**. (relacionado con BackStack). (*ver la documentación*).
- ▶ *EJEMPLOS:*
  - ▶ **FLAG\_ACTIVITY\_NO\_HISTORY**  
No pone esta Activity en la pila History
  - ▶ **FLAG\_DEBUG\_LOG\_RESOLUTION**  
Imprime información extra en el logging cuando es procesado este Intent. **MUY ÚTIL PARA VER CÓMO LO RESUELVE.**

### ¿Cómo se especifican los FLAGS en el Intent?

```
Intent miIntent = new  
    Intent(Intent.ACTION_SEND);  
miIntent.setFlags (  
    Intent.FLAG_ACTIVITY_NO_HISTORY);
```



# Ejemplo de USO DE INTENTS (I)

## ▶ INTENTS EXPLÍCITOS:

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

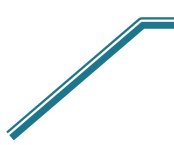
## ▶ INTENTS IMPLÍCITOS:

```
Intent i = new Intent(Intent.ACTION_VIEW,  
    Uri.parse("http://www.androidcurso.com/"));
```


```
Intent i = new Intent(Intent.ACTION_VIEW,  
    Uri.parse("geo:41.656313,-0.877351"));
```

```
Intent intent = new Intent(Intent.ACTION_CALL,  
    Uri.parse("tel:962849347"));
```

```
Intent i= new Intent("MediaStore.ACTION_IMAGE_CAPTURE");
```



Intenta visualizar:  
una web, un  
mapa,...



Toma una foto y  
nos devuelve un  
bitmap pequeño

Para arrancar las actividades: **startActivity(i);**

# Ejemplo de USO DE INTENTS (II)

## ► Enviar un correo electrónico:

```
Intent i = new Intent(Intent.ACTION_SEND);  
i.setType("text/plain");  
i.putExtra(Intent.EXTRA_SUBJECT, "asunto");  
i.putExtra(Intent.EXTRA_TEXT, "texto del correo");  
i.putExtra(Intent.EXTRA_EMAIL, new String[]  
    { "alejandra@gmail.es" });  
startActivity(i);
```

# Ejemplo de USO DE INTENTS (III)

- Tabla de INTENTS que podemos usar con APLICACIONES de GOOGLE:

APLICACIÓN	URI	ACCIÓN	RESULTADO
Browser	<a href="http://dirección_web">http://dirección_web</a> <a href="https://dirección_web">https://dirección_web</a> <a href="tel:123">tel:123</a> Content://contacts/people/1	VIEW	Muestra datos al usuario: si es una url muestra el navegador, si es un contacto, muestra sus datos,...
	"" (cadena vacía) <a href="http://direccion_web">http://direccion_web</a> <a href="https://dirección_web">https://dirección_web</a>	WEB_SEARCH	Realiza una búsqueda web. Se indica la cadena de búsqueda en el extra SearchManager.QUERY
Dialer	<a href="tel:número_tfno">tel:número_tfno</a>	CALL (necesitamos el permiso <b>android.permission.CALL_PHONE</b> ).	Realiza una llamada de tfno. Los números válidos se definen en <a href="#">IETF RFC 3966</a> . Entre estos están: <a href="tel:2125551212">tel:2125551212</a> y <a href="tel:(212)5551212">tel:(212)5551212</a> .

# Ejemplo de USO DE INTENTS (IV)

APLICACIÓN	URI	ACCIÓN	RESULTADO
Dialer	<u>tel:número_tfno</u> voicemail:	DIAL	Introduce un número sin llegar a realizar la llamada.
Google Maps	geo:latitud,longitud geo:lat,long?z=zoom geo:0,0?q=dirección geo:0,0?q=búsqueda	VIEW	Abre Google Maps para una localización determinada. El campo z especifica el nivel de zoom.
Google StreetView	google.streetview:cbll =latitud,longitud&cbp =a,b,c,d,e	VIEW	Abre la aplicación Street View para la ubicación dada. El esquema de URI se basa en la sintaxis que usa Google Maps. Sólo el campo cbll es obligatorio.

# Android – ShareCompat is new way to send content

Before in order to send content using SMS or e-mail app, we would write following code:

```
String message = "This is a message to send";  
String chooser = "Choose app to send message";  
  
Intent sendIntent = new Intent(android.content.Intent.ACTION_SEND);  
sendIntent.setType("text/plain");  
sendIntent.putExtra(android.content.Intent.EXTRA_SUBJECT, "");  
sendIntent.putExtra(android.content.Intent.EXTRA_TEXT, message);  
startActivity(Intent.createChooser(sendIntent, chooser));
```

Now we can send using ShareCompat:

```
String message = "This is a message to send";  
String chooser = "Choose app to send message";  
  
ShareCompat.IntentBuilder  
    .from(this)  
    .setText(message)  
    .setType("text/plain")  
    .setChooserTitle(chooser)  
    .startChooser();
```

# Resolución de Intents (I)

## 1. INTENTS EXPLÍCITOS:

- ▶ Cuando enviamos un Intent explícito, el **Intent se enviará al componente designado por el nombre.**
- ▶ **No se comprobarán otras cosas:** el intent irá **directamente** a ese componente.

## 2. INTENTS IMPLÍCITOS:

- ▶ Para los intent implícitos, se necesita una estrategia distinta y más compleja.
- ▶ Como **no hay un objetivo explícito**, **Android debe encontrar el mejor componente (o componentes)** para que pueda encargarse del Intent.
- ▶ Esto se hace comparando el contenido del Intent con **intent-filters.**

# Resolución de Intents (II)

- ▶ Estos **filtros** están asociados en el **MANIFEST** con **componentes** (*activities*, sobre todo) que pueden recibir potencialmente intents implícitos.
- ▶ Los **intent filters** anuncian las capacidades que tiene un **componente** y al mismo tiempo **delimitan** qué tipo de intents **puede aceptar**.
- ▶ Si un **componente** (*activity, por ejemplo*) **no tiene intent-filter, sólo podrá recibir intents explícitos**.
- ▶ Sin embargo, **el poseer un intent-filter no impide que el componente reciba intents explícitos**, pudiendo recibir ambos tipos.
- ▶ A la hora de evaluar qué intents acepta un filtro, sólo se evalúan tres cosas (en realidad 4):
  - ▶ **action**
  - ▶ **data (URI y tipo MIME)**
  - ▶ **category**

**Los extras no se usan en esta comparación.**

# Intents e Intent-filter (III)

- ▶ Los **componentes**, por otro lado, **deben anunciar sus capacidades, qué tipo de intents pueden aceptar** mediante “**intent-filters**”.
- ▶ Android necesita saber **qué intents maneja cada componente** antes de lanzarlo, por tanto deben estar definidos en el Manifest.
- ▶ Para ello se usa la **etiqueta <intent-filter>**
- ▶ Un componente puede tener cualquier número de intent filters, **cada uno describiendo una capacidad** del componente.
- ▶ Un **intent** puede **referirse a un componente específico** por su nombre (**en ese caso los filters no influyen**).



# Intent-Filter (I)

- ▶ **Un Intent implícito sólo llegará a su destinatario si pasa por el filtro.**
- ▶ Cada **filtro describe una capacidad del componente**, un **conjunto de intents que el componente puede recibir**.
- ▶ **Un componente puede tener varios filtros separados** para cada trabajo que puede hacer
  - Un ejemplo sería por ejemplo que una misma actividad tuviese un *filtro para crear un nuevo mail* y *otro para editar un mail ya existente*.
  - La actividad sería la misma, pero se *aceptarían intents diferentes y la actividad tendría un comportamiento diferente en función del intent*.

### ¿Cómo se especifican los IntentFilters?

- ▶ Los filtros son una instancia de la clase **IntentFilter**
  - Sin embargo como el sistema necesita saber la información antes de que exista el componente, **se declaran en el manifest.**
- ▶ Se declaran en el xml mediante la etiqueta **<intent-filter>**, **excepto en el caso de los BroadcastReceivers, que se crean en código.**

## ¿Cómo se especifican los IntentFilters?

```
<activity ...>
```

```
    <intent-filter ...>
```

```
        ...
```

```
        <action android:name="actionName" />
```

```
        ...
```

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

```
    ...
```

```
</activity>
```

## ¿Cómo se especifican los IntentFilters?

- En el XML no se usan las constantes definidas en la clase Intent, **sino que se usan los nombres completos (incluyendo el nombre de paquete).**
  - **Ejemplo:**

```
<activity class=".NoteEditor" android:label="@string/title_note">
  <intent-filter android:label="@string/resolve_edit">
    <action android:name="android.intent.action.VIEW" />
    <action android:name="android.intent.action.EDIT" />
    <category android:name="android.intent.category.DEFAULT" />
    <data android:mimeType="vnd.android.cursor.item/vnd.google.note" />
  </intent-filter>
```

## AÑADIENDO DATOS AL IntentFilter

- ▶ En el XML los intent-filter tendrán campos similares a su representación en Java.
  - Estos campos son category, data y action.

```
<intent-filter ...>
...
<data
  android:mimeType="string"
  android:scheme="string"
  android:host="string"
  android:port="string"
  android:path="string"
  android:pathPattern="string"
  android:pathPrefix="string"
/>
...
</intent-filter>
```

## CÓMO RESUELVE ANDROID LOS Intents implícitos

- ▶ Los intents implícitos **se comprueban para los tres campos: category, data y action.**
  - El componente **debe pasar las tres comprobaciones.**
  - Si falla una de las tres, el componente no recibirá el intent.
  - Eso no impide que se reciba el intent mediante otro filtro.

# Intent filters

- Test de acciones
  - Un elemento <intent-filter> lista sus acciones como hijos del tipo <action>, cada uno definiendo su android:name
  - El filtro puede mencionar varias acciones.
    - Debe haber una al menos, sino el filtro lo bloqueará todo.
  - El objeto intent debe coincidir con una de las acciones del filtro.
    - Si el filtro no especifica acciones, ningún intent pasará el test
    - Si especifica al menos una y el intent no especifica ninguna, automáticamente pasa el test.

# Intent filters

- Ejemplo

```
<intent-filter . . . >
  <action android:name="com.example.project.SHOW_CURRENT" />
  <action android:name="com.example.project.SHOW_RECENT" />
  <action android:name="com.example.project.SHOW_PENDING" />
  . . .
</intent-filter>
```



# Intent filters

- Test de categorías
  - El filter debe listar categorías como subelementos de tipo <category> con un atributo android:name
  - Para pasar el test, todas las categorías en el intent deben coincidir con una de las categorías en el filtro.
  - Esto quiere decir que un Intent sin categorías debe pasar el filtro siempre
    - Pero hay excepciones en este comportamiento

# Intent Filters

- Todos los intents pasados mediante `startActivity` llevan siempre la categoría *`android.intent.category.DEFAULT`*
  - Por tanto, las actividades que quieran recibir intents implícitos deben poner esta categoría en el filtro.
  - Se **exceptúan** de esta regla los intent con acción “**MAIN**” y categoría “**LAUNCHER**”.

# Intent Filters

- Test de datos
  - Las especificaciones de datos también son elementos hijo de <intent-filter>, con la etiqueta <data>
  - Cada elemento <data> puede especificar una URI y un tipo de datos MIME.
    - Para las URI, hay varios atributos distintos, cada uno haciendo referencia a una parte de la URI

# Intent Filters

- Para definir una URI los atributos son **scheme**, **host**, **port** y **path**

- **scheme://host:port/path**

- Por ejemplo, la URI

`content://com.example.project:200/folder/subfolder/etc`

- **scheme** sería “**content**”
  - **host** sería “**com.example.project**”
  - **port** sería “**200**”
  - **path** sería “**folder/subfolder/etc**”
- El host y el puerto juntos son la *autoridad* de la URI. Si no se especifica host, el puerto se ignora.

# Intent Filters

- Los atributos de la URI son opcionales, pero dependientes unos de otros. Si no tenemos un esquema, no tiene sentido especificar una autoridad. Deben estar ambos.
- La comparación en el filtro se hace parte por parte.
  - Si un filtro especifica sólo esquema, todas las URI con ese esquema pasarán el filtro.
  - Si especifica esquema y autoridad, todas las URI que coincidan en ambos pasarán el filtro.
  - Igualmente con el resto de atributos y combinaciones.
- El atributo “path” en el filtro, puede especificar caracteres comodín.

# Intent Filters

- El atributo type de un elemento <data> especifica el tipo MIME.
- Es más habitual que las URI en los filtros.
- Recordemos que un tipo mime es de la forma tipo/subtipo
- Tanto el intent como el filtro, pueden especificar el subtipo como “\*” para indicar que les vale cualquier subtipo.

# Intent Filters

- El test de datos intenta comparar ambas cosas, la URI y el tipo MIME.
  - Si un intent no especifica ninguna, sólo pasa el test si el filtro no define ninguna.
  - Si el intent contiene sólo URI, y el tipo no se puede deducir a partir de la misma, pasa el test sólo si su URI coincide con una URI del filtro y el filtro tampoco especifica tipo MIME.
    - Esto ocurre con URIs tipo mailto: y tel: que no usan datos.
  - Si el intent contiene tipo MIME pero no URI, sólo pasa el test si coincide con un tipo MIME del filtro y no especifica URI

# Intent Filters

- Si el intent contiene ambos (o el tipo MIME se puede inferir con la URI):
  - La parte de datos la pasa sólo si coincide con un tipo listado en el filtro
  - Para pasar la parte de la URI deberá o bien coincidir con una URI del filtro, o bien si el filtro no especifica URI pero el intent tiene URI content: o file:
  - Esto quiere decir que si el filtro especifica sólo un tipo MIME, se supone que el componente soportará content: y file:



# Intent Filters

- Si el intent pasa los filtros de más de una actividad o servicio, se le preguntará al usuario qué componente usar.
- Si no se encuentra ninguno, se lanzará una excepción.

# Ejemplo de una aplicación que muestra MAPAS

```
<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 = "geo"/>  
</intent-filter>
```

# Casos típicos

- Como vimos antes, **los filtros que sólo especifiquen un tipo de datos, se espera que puedan soportar content: o file:**
- Esto quiere decir que **si especificamos sólo un tipo MIME, se debe poder coger los datos desde un DataProvider o desde un archivo.**
  - Un ejemplo podría ser:
    - `<data android:mimeType="image/*" />`
    - Esto dice que el componente aceptará imágenes en diversos formatos de un proveedor o de un archivo.

# Casos típicos

- Otro **caso típico es especificar el esquema y el tipo.**
  - Aquí se estaría especificando la URI (una gama de ellas).
- Ejemplo:
  - `<data android:scheme="http" android:type="video/*" />`
- **Esto quiere decir que se aceptará video vía red**

# Búsqueda de componentes

- **Los filtros también son usados por el sistema** para saber características sobre los componentes publicados en el mismo (en el propio sistema).
- Por ejemplo, para la pantalla de lanzamiento de aplicaciones, **se buscan las actividades que tengan de acción `android.intent.action.MAIN` y de categoría `android.intent.category.LAUNCHER`.**
- La actividad que ejecuta la pantalla “home” el sistema puede buscarla porque su categoría es **“`android.intent.category.HOME`”**

# Búsqueda de componentes

- **Podemos usar el mismo tipo de búsquedas en nuestra aplicación.**
- La clase **PackageManager** tiene un conjunto de métodos query que devuelven todos los componentes que pueden aceptar un intent específico
  - También hay una serie de métodos resolve que pueden determinar el mejor componente para responder a un intent.
- Estos métodos no activan ningún componente, sólo dan la información para usarla en las aplicaciones.

# Cómo usar INTENTS para comunicar ACTIVITIES

# Pasar datos a la actividad lanzada

- ▶ En la **actividad que lanza a otra**:

```
Intent intent = new Intent(this, MI_CLASE.class);  
intent.putExtra("usuario", "Pepito Perez");  
intent.putExtra("edad", 27);  
startActivity(intent);
```

- ▶ En la **actividad lanzada**:

```
Bundle extras = getIntent().getExtras();  
String s = extras.getString("usuario");  
int i = extras.getInt("edad");
```



# Devolver datos desde una Activity lanzada a quién nos lanzó (I)

- ▶ En la **actividad que lanza a otra** (*y que luego esperará los resultados*):

```
Intent intent = new Intent(this, MI_CLASE.class);
startActivityForResult(intent, 1234);
```

```
...
```

```
@Override
```

```
protected void onActivityResult (int requestCode,
                                   int resultCode,
                                   Intent data) {
    if (requestCode==1234 && resultCode==RESULT_OK)
    {
        String res =
            data.getExtras().getString("resultado");
    }
}
```

## Devolver datos desde una Activity lanzada a quién nos lanzó (II)

- ▶ En la **actividad lanzada**:

```
Intent intent = new Intent();  
intent.putExtra("resultado", "valor");  
setResult(RESULT_OK, intent);  
finish();
```

# Cómo configurar Button UP en activity hija

# Configurar Button UP en activity HIJA

- ▶ En el **AndroidManifest** de la app, en la etiqueta de la Activity hija, pondremos:

```
<activity
    android:name=".ActivityHija"
    android:parentActivityName=".ActivityPadre">
    <meta-data
        android:name="android.support.PARENT_ACTIVITY"
        android:value=".ActivityPadre" />
</activity>
```