

CURSO DE TÉCNICO EN DESARROLLO NATIVO SOBRE PLATAFORMAS ANDROID

EOI – Escuela de Organización Industrial

TEMA 3. Esqueleto desarrollo Android

CONTENIDOS

1. Conceptos básicos de una aplicación Android
2. El proyecto de una aplicación Android
3. Contenidos de un proyecto Android
4. Gradle y Android Manifest
5. Recursos en un proyecto Android

ANDROID APPLICATION

- Es un programa que el usuario final podrá instalar desde Google Play o descargar desde otra fuente.
- Contiene una interfaz de usuario (UI)
- Puede contener código para ejecutar tareas en segundo plano (multi-tasking)

COMPONENTES DE UNA APP

- Punto de entrada:
 - Diferente a un programa plano en Java, donde el punto de entrada es el método *main*
 - En Android, se hereda de clases ofrecidas por el Sistema Operativo

```
public class JavaApp {  
    public static void main(String[] args) {  
    }  
}
```

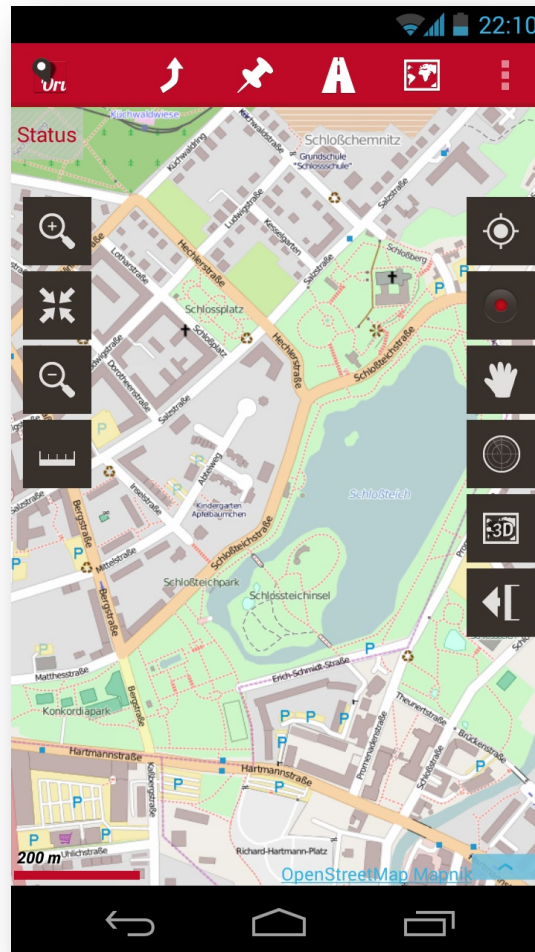
COMPONENTES DE UNA APP

- Hay cuatro tipos de componentes de entrada en una aplicación Android
 - Activity
 - Service
 - Content Provider
 - Broadcast Receiver

ACTIVITY

- Componente de responsabilidad única
 - Contiene interfaz de usuario
 - Puede ser un punto de entrada a la aplicación
 - Normalmente, ocupan toda la pantalla del dispositivo
- Por analogía, una Activity podría ser:
 - Una ventana de un programa en Windows
 - Un dialogo de un programa en Windows
 - Una simple página de una Web clásica

ACTIVITY



Fuente: Klumbumbus. Wikimedia

SERVICES

- Componente de responsabilidad única
 - **No** contiene interfaz de usuario
 - Están diseñados para seguir ejecutándose independientemente de cualquier interfaz
- Pueden utilizarse para
 - Comprobar actualizaciones en un servicio de RSS
 - Reproducir un audio en mientras el usuario utiliza otra aplicación

CONTENT PROVIDERS

- Ofrecen a procesos independientes, es decir, otras aplicaciones, acceso a los datos de la propia aplicación
- Por ejemplo
 - Si tu aplicación es un gestor de descargas de ficheros, puedes crear un *Content Provider* para hacer accesibles esos ficheros desde fuera de tu aplicación (por ejemplo, un lector de PDF, o un editor de texto, o una galería de imágenes)

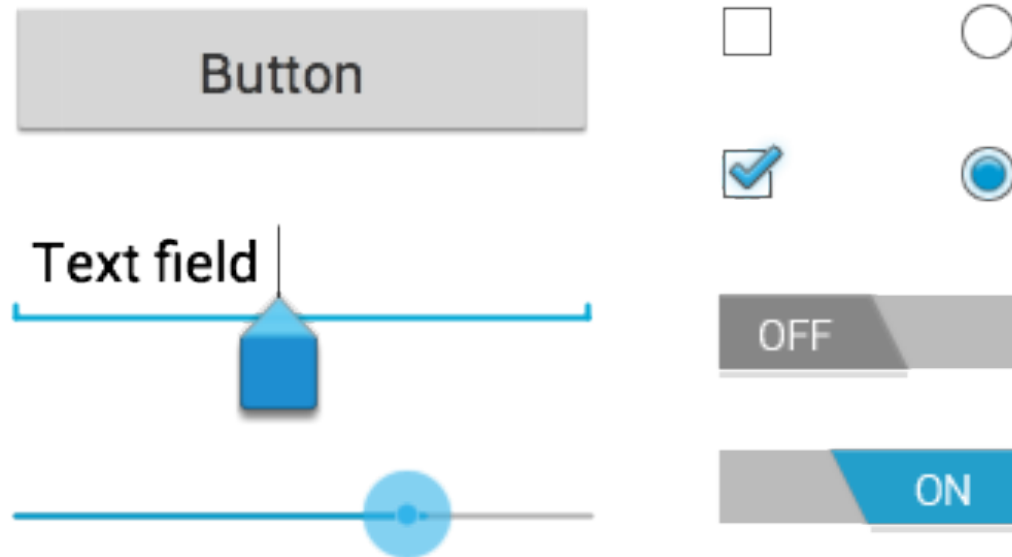
BROADCAST RECEIVERS

- Android (u otras aplicaciones) puede enviar mensajes *broadcast*, es decir, a todos los receptores interesados
 - Por ejemplo: la batería es baja, la pantalla se ha bloqueado, se ha conectado a una red WiFi, se han conectado unos auriculares, etc...
- Un *Broadcast Receiver* sirve para interceptar dichos mensajes y actuar en consecuencia

WIDGETS

- Un **widget** es la unidad básica de interfaz de usuario
 - Por ejemplo: un botón, una etiqueta, un campo de texto, etc..
- Nuestra interfaz de usuario está construida con uno o más *widgets*

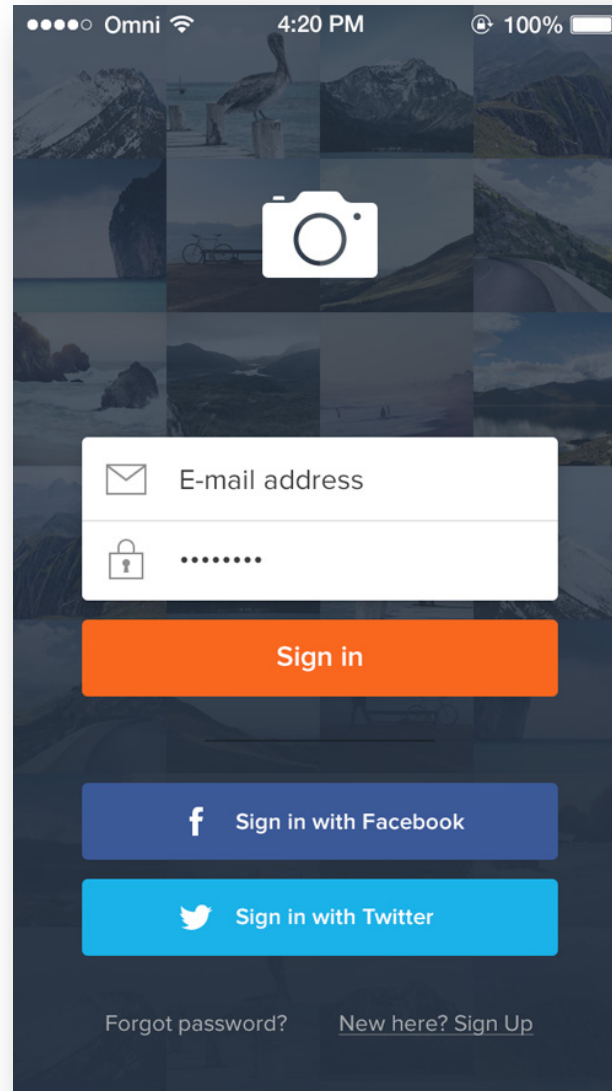
WIDGETS



CONTAINERS

- Cuando la interfaz de usuario contiene varios *widgets*, estos deben organizarse en la pantalla siguiendo unas reglas
- Estas reglas las definen los **contenedores**
- Los contenedores permiten ubicar los *widgets* en filas, columnas e incluso organizaciones más complejas

CONTAINERS



RESOURCES

- Los **recursos** son elementos tales como imágenes, cadenas de caracteres u otro material que la aplicación utiliza pero que no están definidas en ningún lenguaje de programación
- Muchos de ellos se especifican en formato XML

PACKAGE

- Toda aplicación Android tiene un *package name*, un identificador de la aplicación. Este paquete debe cumplir:
 - Debe ser un paquete Java válido
 - No pueden existir dos aplicaciones con el mismo nombre de paquete instaladas en un dispositivo
 - No pueden existir dos aplicaciones con el mismo nombre de paquete en Google Play
 - Por conveniencia, se utiliza el *reverse domain name*

PROCESOS Y THREADS

- Una aplicación se ejecuta en su propio proceso
- Cada proceso puede tener diferentes *threads* (hilos de ejecución)
 - Al *thread* principal se le llama **UI Thread** o **Main Thread**

EL PROYECTO ANDROID

- Un proyecto Android no es más que un directorio en el entorno de desarrollo que contiene:
 - Código fuente en Java
 - Recursos (imágenes, texto, etc...)

CONCEPTOS GLOBALES

- **Application Name**
 - Nombre de vuestra aplicación tal y como la verán los usuarios finales
- **Project Name**
 - Nombre del proyecto tal y como se presenta en Android Studio
- **Package Name**
 - Nombre del paquete Java e identificador único de la aplicación

CONCEPTOS GLOBALES

- **Minimum required SDK**
 - Versión de Android mínima que la aplicación soporta
- **Target SDK**
 - Versión de Android para la que la aplicación ha sido creada
- **Theme**
 - Definición general del diseño de la aplicación (colores, tamaños, etc...)

CONCEPTOS GLOBALES

- **Version Code**
 - Código de la versión. Valor entero positivo e incremental. Sirve para identificar la versión en Google Play y Android. P.E: **30**
- **Version Name**
 - “Nombre” de la versión que el usuario final verá en Google Play. P.E: **“3.0”**

PRÁCTICA

1. Crear un proyecto Android vacío con:
 - Nombre de la aplicación: **Mi Primera App**
 - Nombre del paquete Java (identificador único):
com.teaching.android.miprimeraapp
 - Activity vacía (*Empty Activity*)
 - Versión del SDK de Android 26
2. Crear repositorio en GitHub y hacer el primer *commit* con el proyecto Android

CONTENIDOS DE UN PROYECTO

- Android Manifest
 - **AndroidManifest.xml**
 - Describe la aplicación y qué componentes tiene (activities, services, broadcast receivers...)
 - Es como una *tabla de contenidos* o un *manifiesto* indicando **qué contiene** la aplicación

CONTENIDOS DE UN PROYECTO

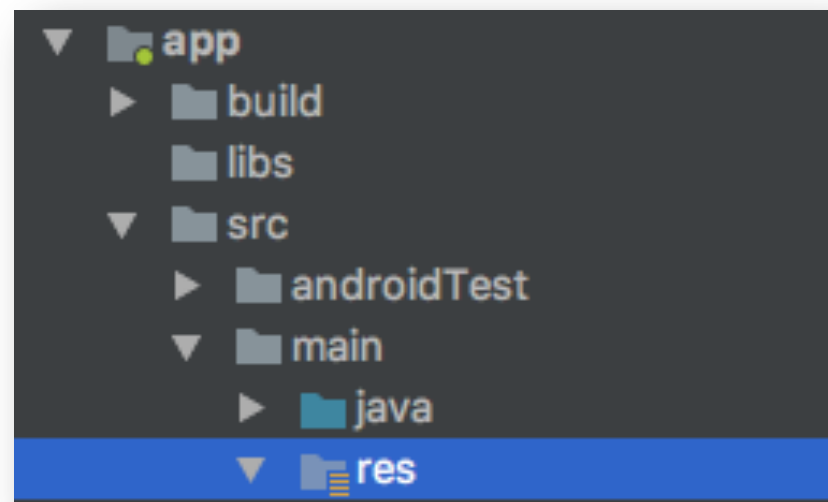
- Código Java
 - Al crear el primer proyecto, se crea una clase en el directorio correspondiente al nombre del paquete Java:
app/src/main/java/com/teaching/android/miprimeraapp
 - Se puede añadir, borrar o modificar código Java dentro de dicho directorio

CONTENIDOS DE UN PROYECTO

- Código Java: ***R.java***
 - **R** es una clase autogenerada por Android que permite acceder a los **recursos** del proyecto desde el **código Java**
 - No se debe modificar la clase ***R.java***, ya que Android la regenera cada vez que se ejecuta la aplicación

CONTENIDOS DE UN PROYECTO

- Recursos
 - Los recursos se almacenan en el directorio **res** del proyecto
 - Son ficheros estáticos que se empaquetan junto al código fuente para crear la aplicación Android



CONTENIDOS DE UN PROYECTO

- Recursos

- Algunos subdirectorios dentro de **res** pueden ser:

- res/drawable -> para imágenes (PNG, JPEG)
 - res/layout -> para definir la UI de la aplicación
 - res/menu -> para definir menús
 - res/raw -> para ficheros de uso genérico
 - res/values -> para strings, tamaños, etc...
 - res/xml -> para ficheros XML de uso genérico

CONTENIDOS DE UN PROYECTO

- Recursos
 - Algunos subdirectorios pueden contener **sufijos**
 - `res/drawable-hdpi`
 - Indican que los recursos alojados en dicho directorio solo se utilizarán en circunstancias concretas. En el ejemplo, se utilizarán en dispositivos cuya densidad de pantalla sea *hdpi*

CONTENIDOS DE UN PROYECTO

- Fichero APK
 - APK (Android Application Package) es un fichero comprimido con el código Java compilado y los recursos estáticos de la aplicación
 - Al ejecutar la aplicación desde Android Studio, el fichero APK que se genera se encuentra en:
build/outputs/apk

GRADLE

- Gradle
 - Es un *software* para construir *software*, es decir, un *sistema de construcción*
 - Se encarga de compilar, enlazar y empaquetar todo lo necesario para construir una aplicación
 - Se pueden especificar instrucciones para la construcción a través del lenguaje Groovy en los ficheros ***build.gradle***

GRADLE

- Gradle
 - Un proyecto Android normalmente dispone de dos ficheros ***build.gradle***:
 - A nivel de proyecto: define la configuración de Gradle para todos los módulos del proyecto.
 - A nivel de módulo (o aplicación): define la configuración de Gradle para el módulo en concreto

GRADLE Y MANIFEST

- Algunos valores definidos en ***AndroidManifest.xml*** pueden sobrescribirse en ***build.gradle***:
 - Nombre del paquete e ID de la aplicación
 - Version del SDK
 - Version Code
 - Version Name

ANDROID MANIFEST

- Android Manifest TAGs
 - <manifest>
 - <application>
 - » *android:icon*
 - » *android:label*
 - <activity>
 - » *android:name*
 - <intent-filter>
 - <uses-permission>

PRÁCTICA

1. Cambiar el nombre de la aplicación a un nombre de vuestra elección
2. Cambiar el icono de la aplicación por una imagen cualquiera

ACTIVITY

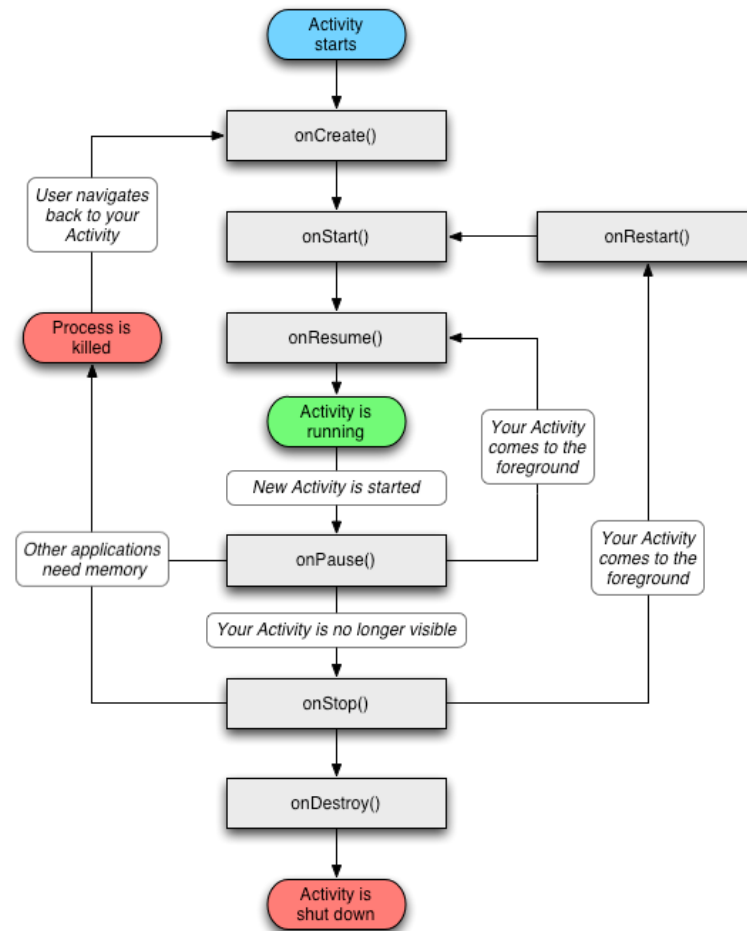
- Una **Activity** lleva asociada una interfaz de usuario, normalmente definida en un fichero XML.
- Dicha asociación se define a través del método **setContentView(R.layout.activity)**

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

ACTIVITY

- **Ciclo de vida.** Métodos de nuestra Activity que Android ejecuta según el estado actual.
 - onCreate()
 - onStart()
 - onResume()
 - onPause()
 - onStop()
 - onDestroy()

ACTIVITY



PRÁCTICA

1. Escribir todos los métodos del ciclo de vida en **MainActivity**
2. Imprimir por pantalla un mensaje con el nombre de cada método.

```
Log.d("MainActivity", "onCreate");
```

INTENT

- Objeto de acción que puedes utilizar para solicitar una acción de otro componente.
- Tres casos de uso fundamentales:
 - Para comenzar una Activity
 - Para iniciar un Service
 - Para entregar un mensaje (broadcast)

docs: <https://developer.android.com/reference/android/content/Intent>

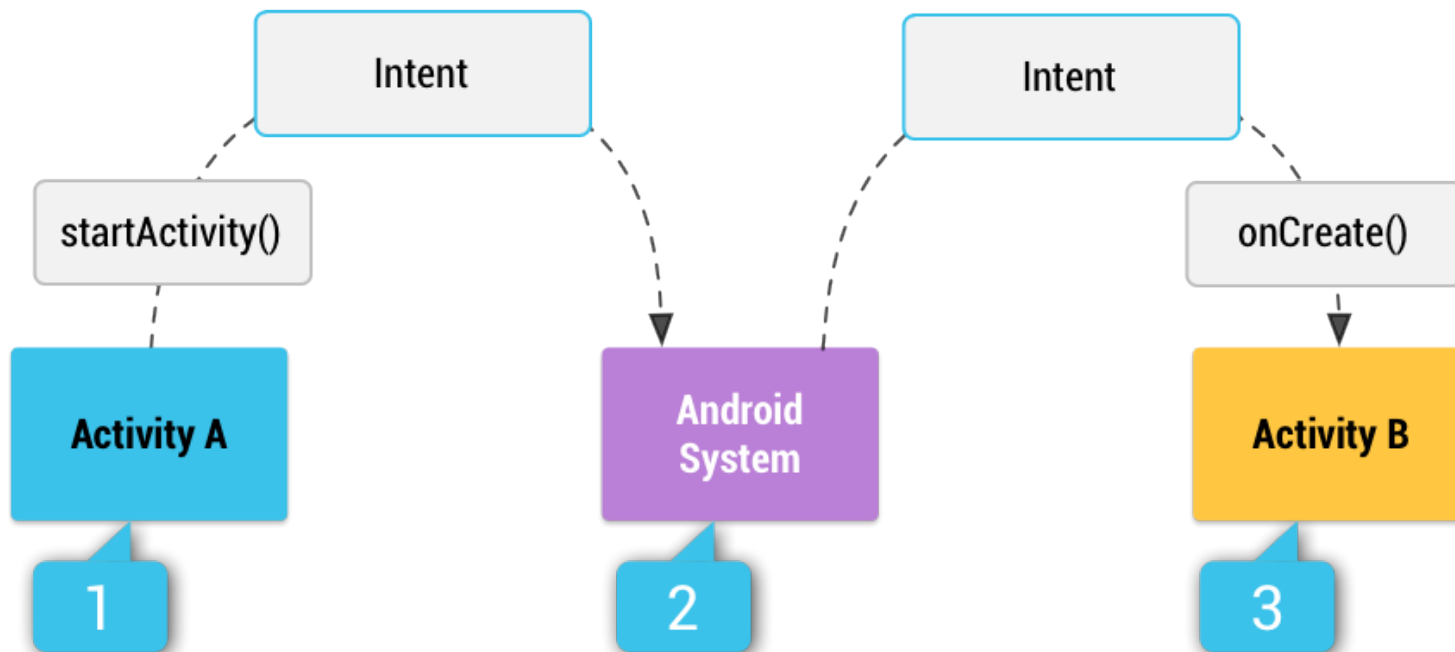
INTENT

- **Explícitos:** especifica qué componente se debe iniciar mediante su nombre.
- Normalmente se utiliza para iniciar un componente de nuestra aplicación (*activity* o *service*), ya que conocemos el nombre de dicho componente.

INTENT

- **Explícito**

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



INTENT

- **Implícitos:** se declara una acción general y Android se encarga de encontrar el componente adecuado para manejarla.
- Ejemplo:
 - Si se quiere mostrar una página web, pero nuestra aplicación no contiene un navegador. Se puede crear un intent implícito con la URL y Android abrirá el navegador con dicha página web.

INTENT

- **Implícito**

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

- Otros intents implícitos frecuentes
 - ACTION_DIAL
 - ACTION_EDIT
 - ACTION_SEND

PREPARANDO PRÁCTICA

1. Cambiar en **activity_main.xml** el contenedor **ConstraintLayout** por **LinearLayout** y añadir el atributo `android:orientation="vertical"`

```
<LinearLayout
    android:orientation="vertical"
    android:layout_width="match_parent"
    android:layout_height="match_parent">


    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Hello World!"
        . . .
```

PRÁCTICA

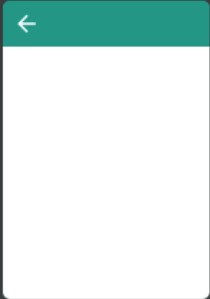
1. Crear una segunda Activity en el proyecto:
SecondActivity
 - *Right click on package name*
 - *New > Activity > Empty Activity*
2. Añadirla al fichero **AndroidManifest.xml**
3. Crear un botón en **MainActivity** que, al pulsarlo, abra la segunda Activity

PRÁCTICA

New Android Activity

 **Configure Activity**
Android Studio

Creates a new empty activity



Activity Name
SecondActivity

☒ Generate Layout File

Layout Name
activity_second

☐ Launcher Activity

☒ Backwards Compatibility (AppCompat)

Package name

The name of the activity class to create

Cancel Previous Next Finish

- Para crear un botón y el código que se ejecuta al hacer click:
 - Cambiar en **activity_main.xml** el **TextView** por **Button**
 - Añadir la propiedad **onClick="onClick"** al widget **Button**
 - Crear el método **public void onClick(View view) {}** en **MainActivity.java**

```
<Button  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content"  
    android:text="Abrir segunda Activity"  
    android:onClick="onClick"/>
```

```
public void onClick(View view) {  
  
}
```

PRÁCTICA

1. Crear otro botón en **MainActivity** que, al pulsarlo, abra el navegador con la URL <http://www.google.com>
 - Utilizar intent implícito con ACTION_VIEW
2. Crear un nuevo botón en MainActivity que, al pulsarlo, abra la aplicación de teléfono y escriba el teléfono **1234** para llamar.
 - Utilizar intent implícito con ACTION_DIAL

docs: [*https://developer.android.com/reference/android/content/Intent*](https://developer.android.com/reference/android/content/Intent)

EXTRAS

- Los intents pueden contener información **extra** que se envía desde una Activity a otro componente. Por ejemplo:
 - La URL a abrir en un navegador
 - El número de teléfono a marcar
 - Datos de nuestra aplicación que nos interesa enviar de una Activity a otra

docs: <https://developer.android.com/guide/components/intents-filters>

EXTRAS

- La información extra se envía como **parejas de clave-valor:**
 - Un identificador (valor tipo String)
 - El valor que queremos enviar

```
Intent intent = new Intent(this, SecondActivity.class);  
intent.putExtra("IDENTIFICADOR_VALOR_EXTRA", "Android Rool");  
startActivity(intent);
```

EXTRAS

- Leer información extra en Activity de destino
 - Se obtiene el intent con **getIntent**
 - Se obtiene el valor que queremos mediante el identificador y un valor por defecto (si no existe)

```
Intent intent = getIntent();  
String valor = intent.getStringExtra("IDENTIFICADOR_VALOR_EXTRA");
```

PRÁCTICA

1. Añadir un valor **extra** al intent que utilizamos para abrir **SecondActivity**
 - Identificador: "id"
 - Valor: "Android Roolz"
2. Leer el valor **extra** en el método **onCreate** de **SecondActivity**, e imprimirlo en Logcat

```
Intent intent = ...  
String valor = ...  
Log.d("Second Activity", valor);
```

INTENT FILTER

- Definen, en Android, los componentes que pueden **hacerse cargo de los intents implícitos**
- Si varios componentes pueden hacerse cargo de dicho Intent, el sistema muestra un dialogo para que el usuario seleccione la aplicación que desee.

docs: <https://developer.android.com/guide/components/intents-filters>

INTENT FILTER

- Pueden contener tres atributos:
 - **<action>**: declara la acción del intent que acepta este filtro
 - **<data>**: declara el tipo de datos que acepta este filtro
 - **<category>**: declara la categoría del intent que acepta
- Con estos tres atributos, Android puede identificar qué componentes instalados en el dispositivo son capaces de manejar determinados intents implícitos

docs: <https://developer.android.com/guide/components/intents-filters>

INTENT FILTER

- Se definen dentro del tag **<activity>** en **AndroidManifest.xml**
- Ejemplo: *Intent filter* para recibir un intent con acción ACTION_SEND y el tipo de dato es texto

```
<intent_filter>  
  <action android:name="android.intent.action.SEND"/>  
  <category android:name="android.intent.category.DEFAULT"/>  
  <data android:mimeType="text/plain"/>  
</intent_filter>
```

INTENT FILTER

- Intent filter para definir el punto de entrada de la aplicación

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


INTENT FILTER

- Los filtros que defináis en vuestra aplicación son **públicos**
- Otras aplicaciones pueden crear **intents** para abrir vuestras activities (como hacemos nosotros con ACTION_VIEW, por ejemplo)