



Anatomía de la Aplicación

Curso: Desarrollando aplicaciones con Android

Anatomía de la Aplicación

- ▶ Existen una serie de elementos clave que resultan imprescindibles para desarrollar aplicaciones en Android.

- ▶ **Vista (View)**

- ▶ Las *vistas* son los elementos que componen la interfaz de usuario de una aplicación. Son por ejemplo, un botón, una entrada de texto,...

- ▶ **Layout**

- ▶ Un Layout es un conjunto de vistas agrupadas de una determinada forma.

Próxima sesión

- ▶ **Actividad (Activity)**

- ▶ Una aplicación en Android va a estar formada por un conjunto de elementos básicos de visualización. En Android, cada uno de estos elementos, o pantallas de la aplicación, se conoce como *actividad*.

Hoy

- ▶ **Intención (Intent)**

- ▶ Una *intención* representa la voluntad de realizar alguna acción; como realizar una llamada de teléfono, visualizar una página web.

- ▶ **Servicio (Service)**

- ▶ Un *servicio* es un proceso que se ejecuta “detrás”, sin la necesidad de una interacción con el usuario.

- ▶ **Receptor de anuncios (Broadcast receiver)**

- ▶ Un *receptor de anuncios* recibe y reacciona ante anuncios de tipo broadcast.

- ▶ **Proveedores de Contenido (Content Provider)**

- ▶ Con este mecanismo podremos acceder a datos de otras aplicaciones, como la lista de contactos, o proporcionar datos a otras aplicaciones.



Contenidos

- ▶ Aplicaciones
- ▶ Actividades
 - ▶ Ciclo de vida de las actividades
- ▶ Intenciones



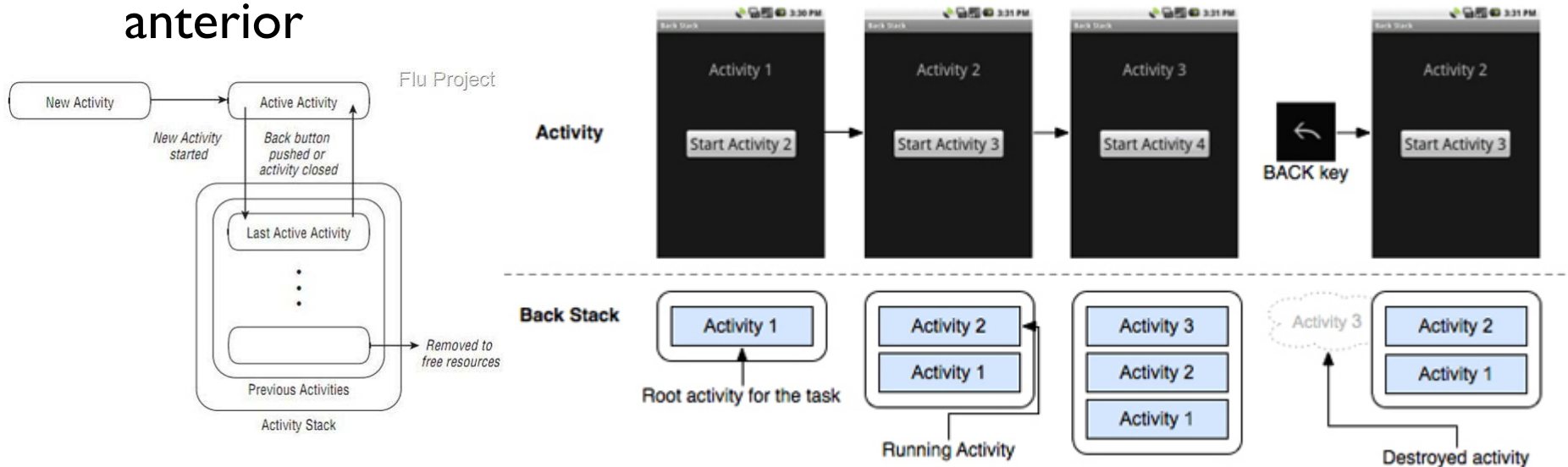
Aplicaciones

- ▶ Una aplicación suele estar formada por una serie de **actividades**, de forma que el usuario puede ir navegando entre actividades.
- ▶ Una aplicación estará formada por un conjunto de actividades independientes, es decir se trata de clases independientes que no comparten variables, aunque todas trabajan para un objetivo común.



Aplicaciones

- ▶ Las aplicaciones en Android sólo tienen un primer plano que ocupa toda la pantalla
- ▶ En un momento dado, una actividad pasa al primer plano y se coloca por encima de otra formando **una pila de actividades**
- ▶ El botón back cierra la actividad y recupera de la pila la anterior



Aplicaciones

- ▶ Las aplicaciones en Android no tienen control de su ciclo de vida
- ▶ Deben estar preparadas para su terminación en cualquier momento
- ▶ Cada aplicación se ejecuta en su propio proceso
- ▶ El *runtime* de Android gestiona el proceso de cada aplicación y por extensión de cada Actividad que contenga.



Actividades

- ▶ **Representa una cosa concreta que puede hacer el usuario**
- ▶ Corresponden con una pantalla de la interfaz de usuario
- ▶ Muestra los controles de la interfaz de usuario y reacciona ante las interacciones del mismo
- ▶ Toda actividad ha de ser una subclase de `Activity`.



Actividades

► Para crear una nueva actividad:

1. Crea un nuevo Layout para la actividad.
2. Crea una nueva clase descendiente de `Activity`. En esta clase tendrás que indicar que el Layout a visualizar es el desarrollado en el punto anterior. ☐
3. Para que nuestra aplicación sea visible será necesario activarla desde otra actividad.
4. De forma obligatoria tendremos que registrar toda nueva actividad en `AndroidManifest.xml`



Actividades

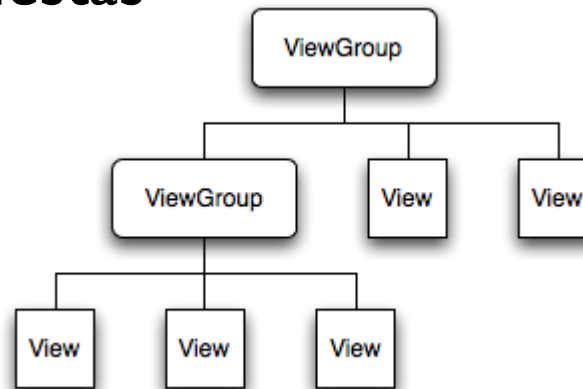
- ▶ **Normalmente una aplicación consta de varias actividades**
- ▶ Cada pantalla se implementa como una actividad
- ▶ Moverse a la siguiente actividad supone llamar al método
 - ▶ **startActivity()**,
 - ▶ **startActivityForResult()**
- ▶ Una aplicación puede reusar actividades de Android o de otras aplicaciones



Actividades

► **View**

- Una actividad se compone de todo tipo de controles o widgets llamados **View** en Android.
- La clase **View** es la clase base de todos los widgets (Button, EditText, TextView...)
- La clase **ViewGroup** es la clase base de los layouts y de otras vistas compuestas



Actividades

► Creando una actividad

```
public class HelloWorld extends Activity {  
    @Override  
    public void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.main);  
    }  
}
```

- onCreate: Se llama a este método cuando se crea la actividad
- setContentView: Asigna a la vista el contenido del recurso layout/esquema
- R.layout.main: Recurso de layout/esquema de la aplicación



Actividades

- ▶ **Moverse a la siguiente actividad**

- ▶ Lanza una nueva actividad sin recibir el resultado

```
startActivity(intent);
```

- ▶ Lanza una nueva actividad y espera el resultado

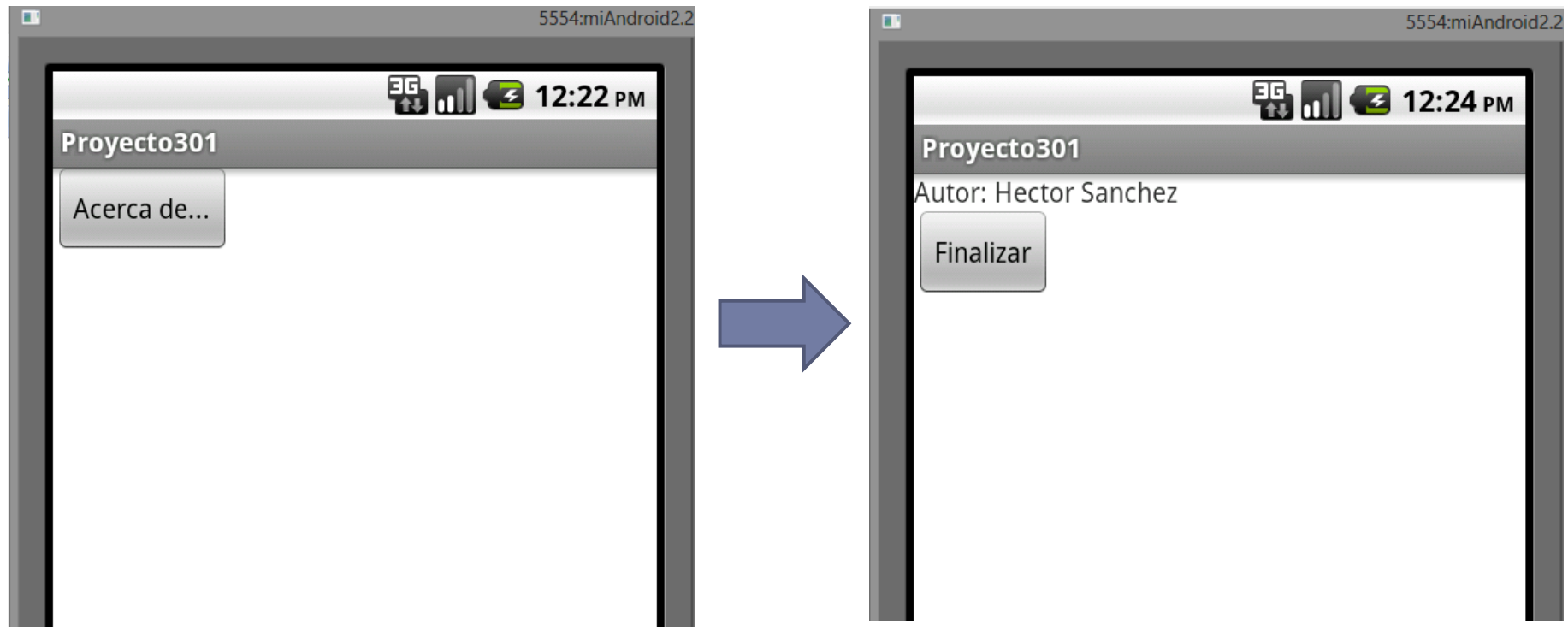
```
startActivityForResult(intent, requestCode);
```

- ▶ `requestCode` es un código, un número entero que elegimos para identificar una actividad cuando nos devuelve un resultado. Su utilidad se manifiesta cuando hemos lanzado varias actividades para un resultado y debemos distinguir entre ellas.



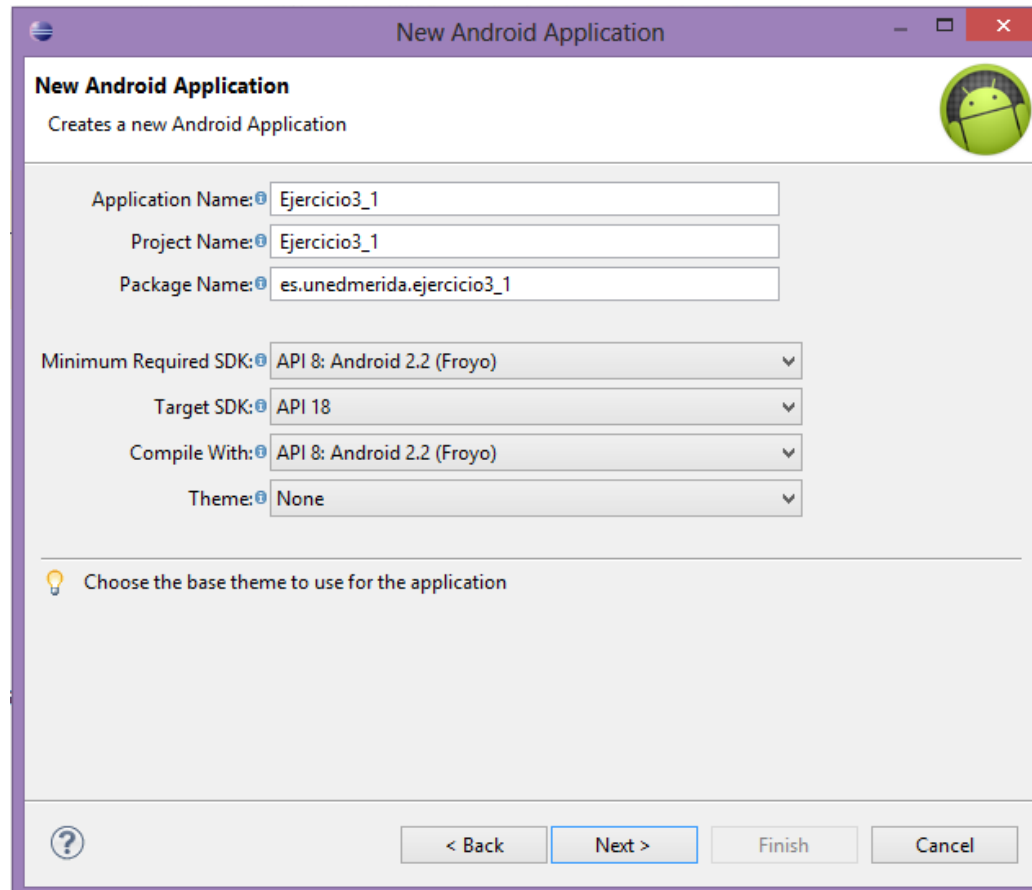
Ejemplo 3.1 – Lanzar una segunda actividad

- ▶ Desarrollar un programa que muestre en la actividad principal un botón que al ser presionado muestre otra actividad con el nombre del programador y un botón para cerrar la actividad



Ejemplo 3.1 – Lanzar una segunda actividad

I. Crea un nuevo proyecto Ejercicio3_1



The screenshot shows the 'New Android Application' dialog box in Android Studio. The dialog has a purple title bar and a white content area. At the top, it says 'New Android Application' and 'Creates a new Android Application'. There is an Android robot icon in the top right corner. The form contains several fields and dropdown menus:

- Application Name: Ejercicio3_1
- Project Name: Ejercicio3_1
- Package Name: es.unedmerida.ejercicio3_1
- Minimum Required SDK: API 8: Android 2.2 (Froyo)
- Target SDK: API 18
- Compile With: API 8: Android 2.2 (Froyo)
- Theme: None

Below these fields, there is a section with a lightbulb icon and the text 'Choose the base theme to use for the application'. At the bottom, there are four buttons: '< Back', 'Next >', 'Finish', and 'Cancel'. The 'Next >' button is highlighted with a blue border.

Ejemplo 3.1 – Lanzar una segunda actividad

1. Un vez creado el proyecto, cambiar Layout a `LinearLayout (Vertical)`, botón derecho sobre el layout.
2. Desde la paleta añade un botón (`Button`), identifícalo y etiquétalo.
 - ▶ Para identificarlo, botón derecho del ratón sobre el widget y `Edit Id`
 - ▶ No se recomienda etiquetarlo directamente sino que habría que añadir el texto al `strings.xml` para futuras traducciones de la interfaz.
 - ▶ Esto lo podemos hacer directamente con el botón derecho sobre el widget ⇨ `Button` y en el menú que aparece crear una nueva cadena de caracteres con `New String`



Ejemplo 3.1 – Lanzar una segunda actividad

4. Para responder a las pulsaciones del ratón, asignamos a la propiedad `onClick` del botón, el método `acercaDe` (¡¡OjO con mayúsculas y minúsculas!!)
5. Dentro de la carpeta “src” encontraremos el `.java` correspondiente a nuestro Activity.
 - ▶ Crear el método `acercaDe` que se llamara desde el botón (debe llevar un `View` como parámetro). Más tarde lo completaremos:

```
public void acercaDe(View view) { ...}
```



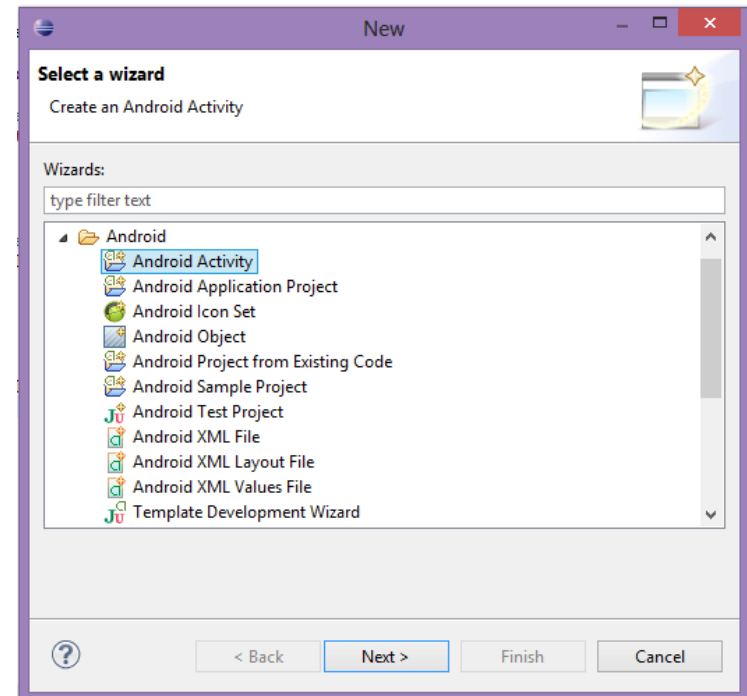
Ejemplo 3.1 – Lanzar una segunda actividad

6. Crear la segunda Activity

- ▶ Pulsamos con el botón derecho sobre el proyecto, seleccionamos New... ⇒ Other y la ventana emergente, bajo Android seleccionamos Android Activity
- ▶ Genera el .java, el layout e incorpora la actividad en AndroidManifest.xml



7. Completar el layout de esta segunda actividad.



Ejemplo 3.1 – Lanzar una segunda actividad

6. Definir la propiedad `onClick` del botón "Finalizar" con el nombre del método que se ejecutará al presionarse el botón (en nuestro caso, lo llamaremos `cerrar`)
7. Codificar el método `cerrar` en el fichero fuente `.java` asociado a la actividad

```
public void cerrar(View view) {  
    finish();  
}
```

- ▶ `finish()` libera el espacio de memoria de esta actividad y pide que se active la actividad anterior



Ejemplo 3.1 – Lanzar una segunda actividad

8. Codificar el método `acercaDe`

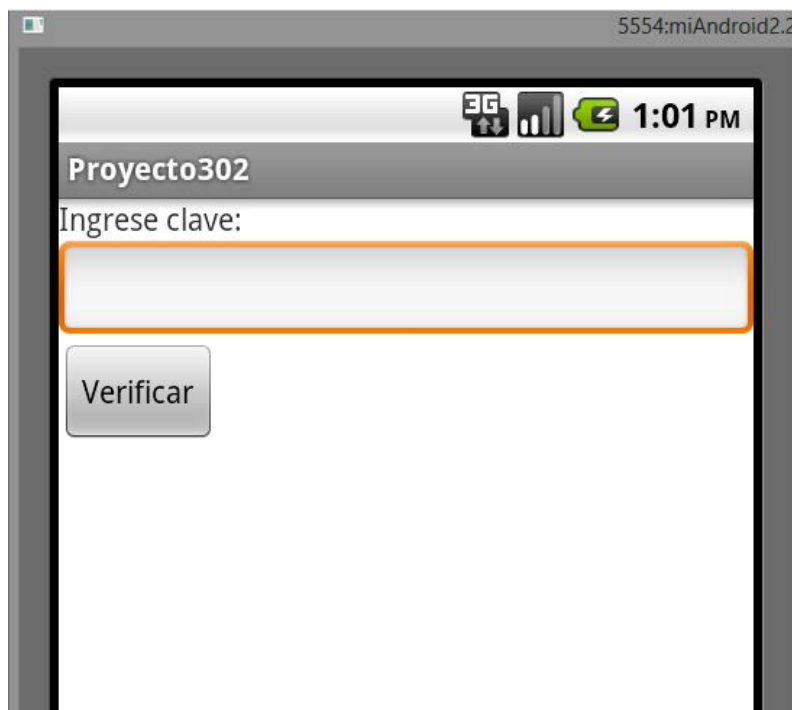
- ▶ Creamos un objeto de la clase `Intent` y le pasamos como parámetros la referencia del objeto de esta clase (`this`) y la referencia de la segunda actividad (`NombreSegundaActividad.class`)
- ▶ Invocamos, a continuación, al método `startActivity` pasando el objeto de la clase `Intent`

```
public void acercaDe(View view) {  
    Intent i = new Intent(this, xxx.class );  
    startActivity(i);  
}
```



Ejemplo 3.2 – Lanzar una segunda actividad

- ▶ Desarrolla un programa que contenga dos actividades
- ▶ En la primera, debes solicitar el ingreso de una contraseña.
- ▶ Si ingresa la contraseña “android” se activa la segunda actividad mostrando un mensaje de bienvenida. Si se ingresa la contraseña incorrecta, debes vaciar el campo de texto para que pueda ingresar una nueva.



Ejemplo 3.2 – Lanzar una segunda actividad

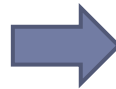
► Pistas

- Mientras introducimos la contraseña, sólo debemos ver puntos. El widget es un `TextView` pero con un `Input Type` concreto.



Ejercicio 3.1 – Lanzar una segunda actividad y pasar parámetros

- ▶ Desarrolla un programa que solicite el ingreso de una dirección de un sitio web y seguidamente abrir una segunda actividad que muestre dicha página.



Ejercicio 3.1 – Lanzar una segunda actividad y pasar parámetros

► Pistas

- Para pasar parámetros, debemos invocar al método `putExtra` de la clase `Intent`. Este método tiene dos parámetros de tipo `String`, en el primero indicamos el nombre del dato y en el segundo el valor del dato:

```
i.putExtra("direccion", editText1.getText().toString());
```

- Para visualizar la web necesitamos un objeto de la clase `WebView` (se encuentra en la pestaña "Composite").
- Para recoger los datos que nos pasa la primera actividad necesitamos un objeto de la clase `Bundle`:

```
Bundle bundle = getIntent().getExtras();
```

```
webView1.loadUrl("http://" + bundle.getString("direccion"));
```



Ejercicio 3.1 – Lanzar una segunda actividad y pasar parámetros

► Pistas

- Como nuestra aplicación debe acceder a Internet debemos hacer otra configuración en el archivo `AndroidManifest.xml`, debemos ir a la pestaña `Permissions` presionar el botón `Add` y seleccionar `Uses Permissions` y fijar en la propiedad `name` el valor `android.permission.INTERNET`.



Actividades

- ▶ **Moverse a la siguiente actividad**

- ▶ Cuando retorna la actividad llamada, se invoca al método `onActivityResult` pasándole el `requestCode` con el que se lanzó desde la actividad

```
onActivityResult(int requestCode, int  
                resultCode, Intent result)
```

- ▶ `resultCode` típicamente toma los valores `RESULT_OK` si el envío del resultado se ha enviado con éxito, o `RESULT_CANCELED` si el envío del resultado ha fallado (por ejemplo, porque el usuario ha cancelado la operación pulsando la tecla Back).



Actividades

- ▶ Cuando lances una actividad usa el siguiente código:

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

- ▶ En la actividad lanzada podemos recoger los datos de la siguiente forma:

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



Actividades

- ▶ Cuando la actividad lanzada termina también podrá devolver datos que podrán ser recogidos por la actividad lanzadora de la siguiente manera.

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

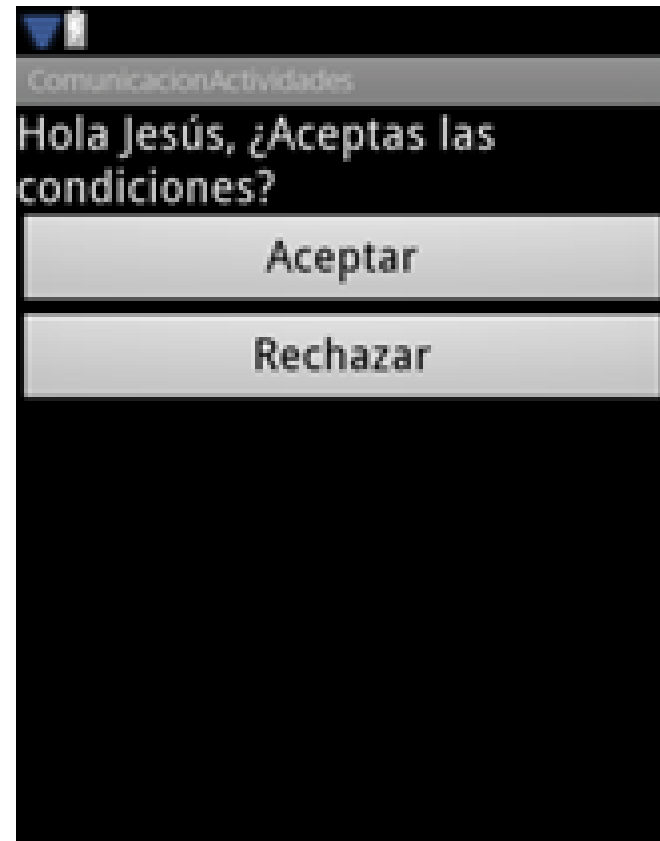
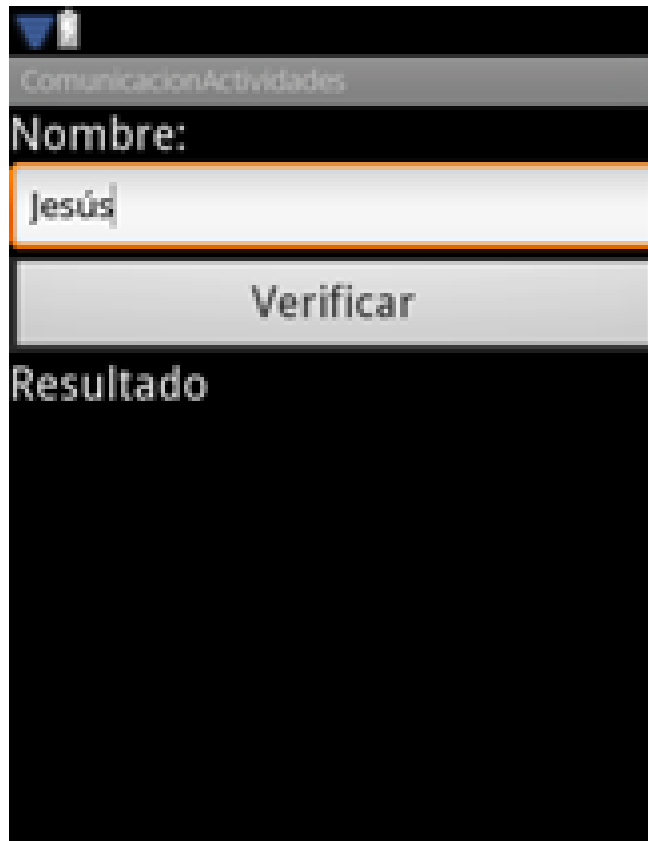
@Override
protected void onActivityResult (int requestCode, int resultCode,
                                   Intent data){
    if (requestCode==1234 && resultCode==RESULT_OK) {
        String res = data.getExtras().getString("resultado");
    }
}
```

- ▶ En la actividad llamada has de escribir:

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



Ejemplo 3.3 – Lanzar una segunda actividad y pasar parámetros



Ejemplo 3.3 – Lanzar una segunda actividad y pasar parámetros

1. Introduce el código para que cuando se pulse el botón `Verificar` se arranque una segunda actividad. A esta actividad se le pasará como parámetro el nombre introducido en el `EditText`.
2. Al arrancar la actividad el texto del primer `TextView` ha de modificarse para que ponga “Hola ”+nombre recibido+” ¿Aceptas las condiciones?”
3. En esta actividad se podrán pulsar dos botones, de forma que se devuelva a la actividad principal la cadena de caracteres `Aceptado` o `Rechazado`, según el botón pulsado. Al pulsar cualquier botón se regresará a la actividad anterior.
4. En la actividad principal se modificará el texto del último `TextView` para que ponga “Resultado: `Aceptado`” o “Resultado: `Rechazado`”, según lo recibido.



Ciclo de vida de las actividades

- ▶ Durante la vida de una actividad esta pasa por una serie de estados
- ▶ En la clase Activity existen métodos para ser redefinidos (override) en sus clases derivadas que incluyen el código a ejecutar en las transiciones entre estados
- ▶ Los métodos redefinidos siempre deben llamar al método de la superclase

```
public class HelloWorld extends Activity {  
    @Override  
    public void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
    }  
}
```



Ciclo de vida de las actividades

► Estados de una actividad

- **Activo (Running):** La actividad está en la cima de la pila, es visible, tiene el foco.
- **Pausado (Paused):** La actividad es visible pero no tiene el foco. Se alcanza este estado cuando pasa a activa otra actividad transparente o que no ocupa toda la pantalla. Cuando una Actividad es tapada por completo pasa a estar parada.
- **Parado (Stopped):** Cuando la actividad no es visible. Se recomienda guardar el estado de la ui, preferencias, etc.
- **Destruído (Destroyed):** Cuando la Actividad termina, o es matada por el runtime de Android. Sale de la Pila de Actividades.



Ciclo de vida de las actividades

▶ Métodos de transición de estados (i)

▶ **onCreate(Bundle)**

- ▶ Se invoca cuando la Actividad se arranca por primera vez.
- ▶ Se utiliza para tareas de inicialización como crear la interfaz de usuario de la Actividad.
- ▶ Su parámetro es `null` o información de estado guardada previamente por `onSaveInstanceState()`

▶ **onStart()**

- ▶ Se invoca cuando la Actividad va a ser mostrada al usuario

▶ **onResume()**

- ▶ Se invoca cuando la actividad va a empezar a interactuar con el usuario



Ciclo de vida de las actividades

▶ Métodos de transición de estados (ii)

▶ **onPause()**

- ▶ Se invoca cuando la actividad va a pasar al fondo porque otra actividad ha sido lanzada para ponerse delante.
- ▶ Se utiliza para guardar el estado de la Actividad

▶ **onStop()**

- ▶ Se invoca cuando la actividad va a dejar de ser visible y no se necesitará durante un tiempo.
- ▶ Si hay escasez de recursos en el sistema, este método podría no llegar a ser invocado y la actividad ser destruida directamente

▶ **onRestart()**

- ▶ Se invoca cuando una actividad parada pasa a estar activa

▶ **onDestroy()**

- ▶ Se invoca cuando la Actividad va a ser destruida.
- ▶ Si hay escasez de recursos en el sistema, este método podría no llegar a ser invocado y la actividad ser destruida directamente.

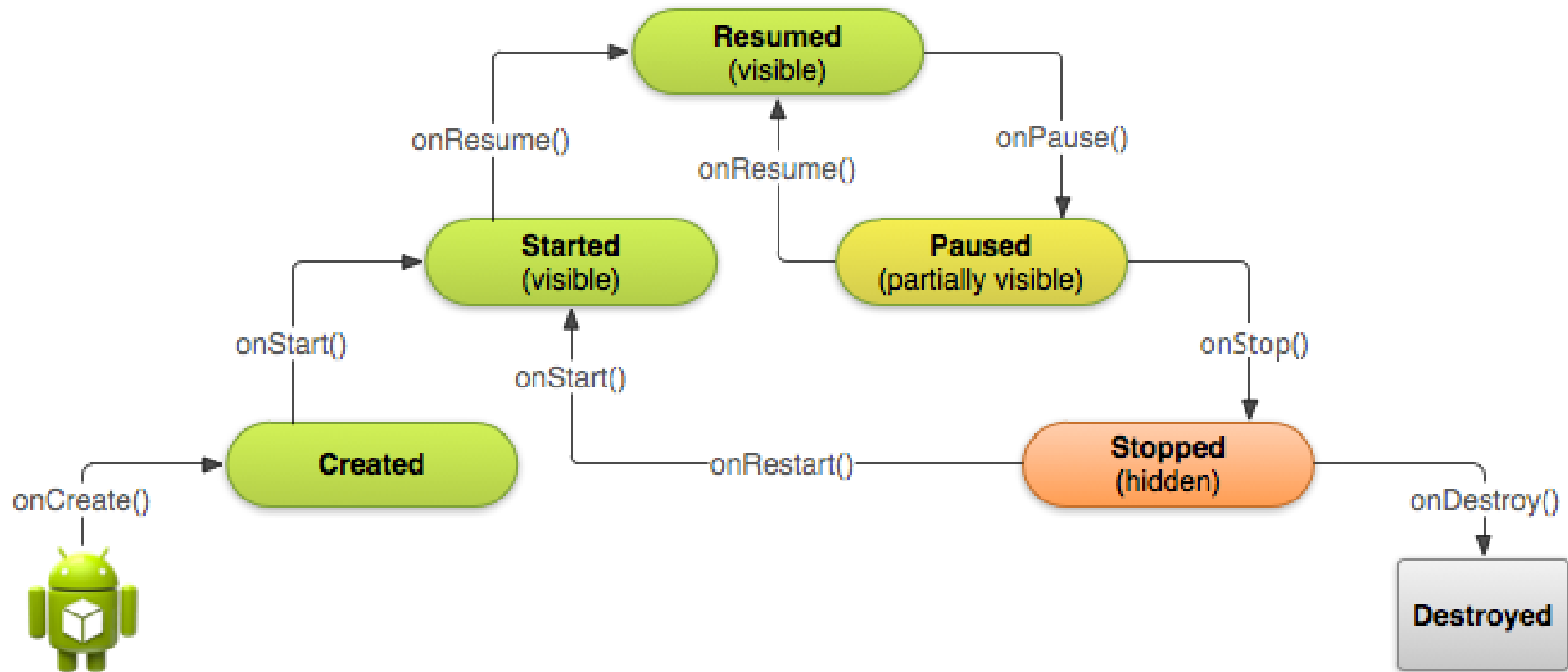


Ciclo de vida de las actividades

- ▶ Métodos de transición de estados (iii)
 - ▶ **onSaveInstanceState(Bundle)**
 - ▶ Se invoca para permitir a la actividad guardar su estado de la ui
 - ▶ Normalmente no necesita ser redefinido
 - ▶ **onRestoreInstanceState(Bundle)**
 - ▶ Se invoca para recuperar el estado guardado por `onSaveInstanceState()`.
 - ▶ Normalmente no necesita ser redefinido



Ciclo de vida de las actividades



Intenciones

- ▶ Representan la “intención” o voluntad de realizar alguna acción o tarea.
- ▶ Utilizaremos intenciones cada vez que queramos:
 - ▶ Lanzar una actividad (`startActivity()`)
 - ▶ Lanzar un servicio (`startService()`)
 - ▶ Lanzar un anuncio de tipo broadcast (`sendBroadcast()`)
 - ▶ Comunicarnos con un servicio (`bindService()`)
- ▶ Las intenciones permiten indicar acciones genéricas, como quiero mandar un mensaje, en cuyo caso se lanzaría la aplicación registrada por defecto para este propósito.



Intenciones

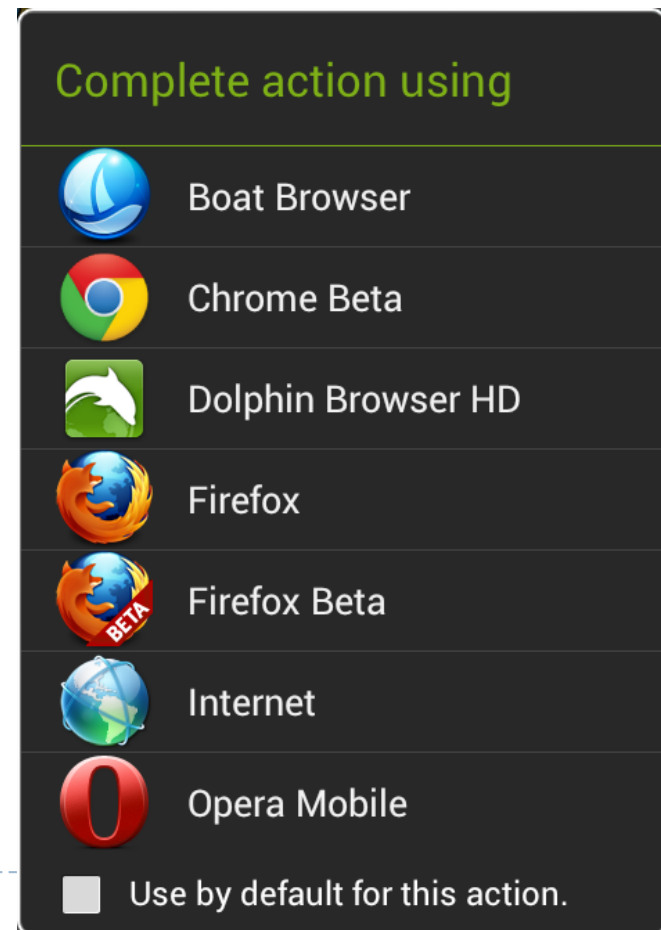
► Tipos de intenciones

- **Intenciones explícitas:** se indica exactamente el componente a lanzar. Su utilización típica es la de ir ejecutando los diferentes componentes internos de una aplicación. Por ejemplo, desde una actividad de nuestra aplicación lanzamos otra.
- **Intenciones implícitas:** pueden solicitar tareas abstractas, como “quiero hacer una foto” o “quiero enviar un mensaje”. Además las intenciones se resuelven en tiempo de ejecución, de forma que el sistema mirará cuantas aplicaciones han registrado la posibilidad de ejecutar este tipo de acción. Si encuentra varias el sistema puede preguntar al usuario la actividad que prefiere utilizar.



Intenciones

- ▶ Por ejemplo, quiero visitar una pagina web. El sistema puede decidir arrancar una aplicación u otra o preguntar si hay varias opciones posibles



Intenciones

▶ Partes de una intención

- ▶ **Nombre del componente:** Con intenciones explícitas: Identificamos el componente que queremos lanzar con la intención. Hay que indicar el nombre de la clase (es `.unex.gexcall.app.android.HelloWorld`)
- ▶ **Acción:** Con intenciones implícitas: Una cadena de caracteres donde indicamos la acción a realizar. La clase Intent define una serie de constantes para acciones. Además de estas, podemos definir nuevas acciones.



Intenciones

► Partes de una intención

Constante	Componente a lanzar	Acción
ACTION_CALL	Actividad	Inicializa una llamada de teléfono.
ACTION_EDIT	Actividad	Visualiza datos para que el usuario los edite.
ACTION_MAIN	Actividad	Arranca como actividad principal de una tarea. (sin datos de entrada y sin devolver datos)
ACTION_SYNC	Actividad	Sincroniza datos en un servidor con los datos de un dispositivo móvil.
ACTION_BATTERY_LOW	receptor de anuncios	Advertencia de batería baja.
ACTION_HEADSET_PLUG	receptor de anuncios	Los auriculares han sido conectados o desconectados.
ACTION_SCREEN_ON	receptor de anuncios	La pantalla es activada.
ACTION_TIMEZONE_CHANGED	receptor de anuncios	Se cambia la selección de zona horaria.



Intenciones

► Partes de una intención

- **Categoría:** Complementa a la acción con información adicional.
 - Por ejemplo: `ACTION_MAIN` y `CATEGORY_LAUNCHER`: primera actividad a lanzar

Constante	Significado
<code>CATEGORY_BROWSABLE</code>	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.
<code>CATEGORY_HOME</code>	La actividad muestra la pantalla de inicio, la primera pantalla que ve el usuario cuando el dispositivo está encendido o cuando la tecla HOME es presionada.
<code>CATEGORY_LAUNCHER</code>	La actividad puede ser la actividad inicial de una tarea y se muestra en el lanzador de aplicaciones de nivel superior.
<code>CATEGORY_PREFERENCE</code>	La actividad a lanzar es un panel de preferencias.



Intenciones

► Partes de una actividad

- **Datos:** Referencia a los datos con los que trabajaremos. Se expresan por medio de URIs (el mismo concepto ampliamente utilizado en Internet). Ejemplos de URIs son: tel:924289300, http://www.unex.es, content://call_log/calls... Para saber el tipo de datos se utilizan los tipos MIME. Ejemplos: text/xml, image/jpeg, audio/mp3...
- **Extras:** Información adicional que será recibida por el componente lanzado. Está formada por un conjunto de pares variable/valor. Estas colecciones de valores se almacenan en un objeto de la clase Bundle.

```
intent.putExtra("usuario", "Pepito Perez");  
intent.putExtra("edad", 27);
```



Intenciones

▶ Ejemplo de uso de intenciones

▶ Intenciones explícitas

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

▶ Intenciones implícitas

```
Intent intent = new Intent(Intent.ACTION_VIEW,  
                           Uri.parse("http://www.unex.es/"));  
  
Intent intent = new Intent(Intent.ACTION_CALL,  
                           Uri.parse("tel:924289300"));  
  
Intent intent = new Intent(Intent.ACTION_VIEW,  
                           Uri.parse("geo:38.915833, -6.333333"));  
  
Intent intent = new Intent("android.media.action.IMAGE_CAPTURE");
```

▶ Para arrancar la actividad: `startActivity(intent)`



Intenciones

► Ejemplo de uso de intenciones

► Para enviar un correo electrónico:

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



Intenciones

- ▶ La etiqueta `<intent-filter>`
 - ▶ Cuando creamos una nueva actividades, servicios o receptor broadcast podemos informar al sistema que tipo de intenciones implícitas pueden ser resultas con nuestro componente. Para conseguir esto utilizaremos la etiqueta `<intent-filter>` de *AndroidManifest.xml*.



Ejercicio 3.2. Intenciones

- ▶ Desarrolla una aplicación cuya interfaz de usuario disponga de 5 botones. La acción asociada a cada uno de estos botones será lanzar una aplicación a través de una intención implícita.
- ▶ Abre `AndroidManifest.xml` e inserta la siguiente línea al final, antes de `</manifest>`:

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

- ▶ Si ejecutas esta aplicación en un emulador es muy posible que el botón mandar Correo o Google Maps no funcione.
 - ▶ La razón es que no hay ninguna aplicación instalada en el emulador que sea capaz de realizar este tipo de acciones. Si tienes estos problemas, Abre el AVD Manager y crea un dispositivo virtual con Google API. Estos dispositivos incorporan además de las API de Android, algunas de las API de Google, como la de Google Maps.



Ejercicio 3.2. Intenciones



Ejercicio 3.3. Intenciones

- ▶ Crea nuevos botones en la aplicación del ejercicio anterior y experimenta con otro tipo de acciones y URIs. Puedes consultar la tabla de la siguiente transparencia.
- ▶ Compara las acciones `VIEW` y `WEB_SEARCH`. ¿Encuentras alguna diferencia?
- ▶ Compara las acciones `CALL` y `DIAL`. ¿Encuentras alguna diferencia?
- ▶ Experimenta con Google Streetview



Ejercicio 3.3. Intenciones

Aplicación	URI	Acción	Resultado
Navegador Web	http://dirección_web https://dirección_web	VIEW	Abre una ventana de navegador con una URL.
	"" (cadena vacía) http://dirección_web https://dirección_web	WEB_SEARCH	Abre el fichero en la ubicación indicada en el navegador.
Teléfono	tel:número_teléfono	CALL	Realiza una llamada de teléfono. Los números validos se definen en IETF RFC 3966 . Ejem. tel:2125551212 .. Necesitamos el permiso android.permission.CALL_PHONE
	tel:número_teléfono voicemail:	DIAL	Introduce un número en la aplicación Teléfono, sin llegar a realizar la llamada. No necesita permiso.
Google Maps	geo:latitud,longitud geo:lat,long?z=zoom geo:0,0?q=dirección geo:0,0?q=búsqueda	VIEW	Abre la aplicación Google Maps para una localización determinada. El campo z especifica el nivel de zoom.
Google Streetview	google.streetview: cbll=latitud,longitud& cbp=l,yaw,,pitch,zoom& mz=mapZoom	VIEW	Abre la aplicación Street View para la ubicación dada. El esquema de URI se basa en la sintaxis que utiliza Google Maps. Solo el campo cbll es obligatorio.

