Componentes de una aplicación Android

Veremos los distintos tipos de componentes de software con los que podremos construir una aplicación Android.

En aplicaciones desktop estamos acostumbrados a manejar conceptos como ventana, control, eventos o servicios como los elementos básicos en la construcción de una aplicación. En Android vamos a disponer de esos mismos elementos básicos aunque con un pequeño cambio en la terminología y el enfoque. Repasemos los componentes principales que pueden formar parte de una aplicación Android.

- **Activity:** Representan el componente principal de la interfaz gráfica de una aplicación Android. Se puede pensar en una activity como el elemento análogo a una ventana en cualquier otro lenguaje visual.

- **View:** Los objetos view son los componentes básicos con los que se construye la interfaz gráfica de la aplicación. Android pone a nuestra disposición una gran cantidad de controles básicos, como cuadros de texto, botones, listas desplegables o imágenes, aunque también existe la posibilidad de extender la funcionalidad de estos controles básicos o crear nuestros propios controles personalizados.

- **Service:** Los servicios son componentes sin interfaz gráfica que se ejecutan en segundo plano. En concepto, son exactamente iguales a los servicios presentes en cualquier otro sistema operativo. Los servicios pueden realizar cualquier tipo de acciones, por ejemplo actualizar datos, lanzar notificaciones, o incluso mostrar elementos visuales (p.ej. activities) si se necesita en algún momento la interacción con del usuario.

- **Content Provider:** Un content provider es el mecanismo que se ha definido en Android para compartir datos entre aplicaciones. Mediante estos componentes es posible compartir determinados datos de nuestra aplicación sin mostrar detalles sobre su almacenamiento interno, su estructura, o su implementación. De la misma forma, nuestra aplicación podrá acceder a los datos de otra a través de los content provider que se hayan definido.

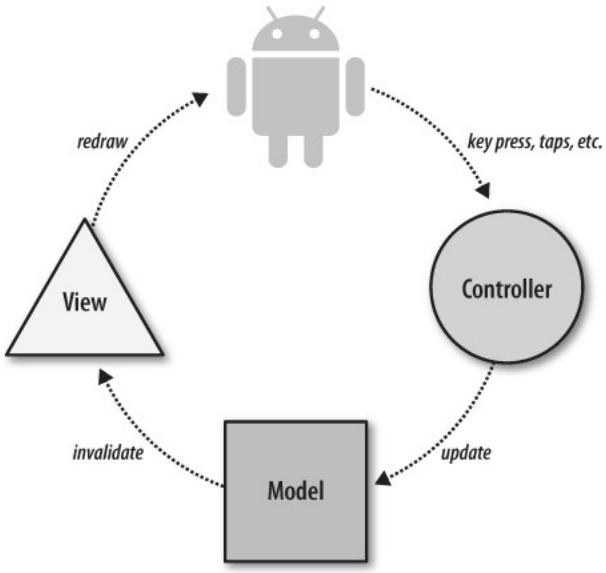
- **Broadcast Receiver:** Un broadcast receiver es un componente destinado a detectar y reaccionar ante determinados mensajes o eventos globales generados por el sistema operativo (por ejemplo: “Batería baja”, “SMS recibido”, “Tarjeta SD insertada”, …) o por otras aplicaciones (cualquier aplicación puede generar “mensajes” no dirigidos a una aplicación concreta sino a cualquiera que quiera escucharlo).

- **Widgets:** Se llama widget o form-widget a todo elemento que podemos colocar en la pantalla de nuestra aplicación.Estos elementos heredan de View. También se conoce como Widget a los elementos que van en el escritorio del sistema operativo y que permiten interacción con el usuario.

- **Intent:** Un intent es el elemento básico de comunicación entre los distintos componentes Android que hemos descrito anteriormente. Se pueden entender como los mensajes o peticiones que son enviados entre los distintos componentes de una aplicación o entre distintas aplicaciones. Mediante un intent se puede lanzar una activity desde cualquier otra, iniciar un servicio, enviar un mensaje broadcast, iniciar otra aplicación, etc.

Concepto Model View Controller

MVC: Patrón de diseño utilizado para aislar la lógica de negocio de la interfaz de usuario. Es muy común en android utilizar un patrón MVC por cada pantalla de nuestra aplicación:

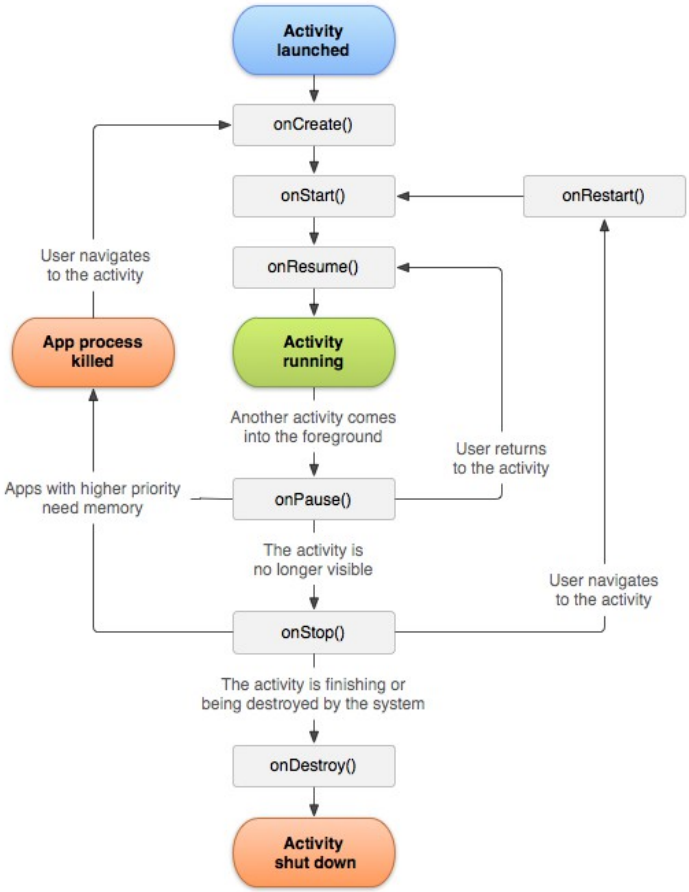


- **Modelo:** Representa la información (data) de la aplicación y las reglas para usarla y manipularla.
- **Vista:** Porción de la aplicación encargada de mostrar en pantalla.
- **Controlador:** Porción de la aplicación que responde a acciones externas.

Comúnmente se utiliza a la Activity como el controller y el modelo (en el caso que sea simple), construyendo una clase aparte para el manejo de la vista.

Ciclo de vida de una aplicación

Existen una gran cantidad de métodos en nuestra activity que podemos sobreescribir, y que se irán ejecutando a medida que la Activity cambie de estado.



Hasta ahora solo reescribimos el método onCreate, existe otro método igual de importante que es el onStop, el cual será ejecutado antes de que la aplicación se cierre o pase a segundo plano.

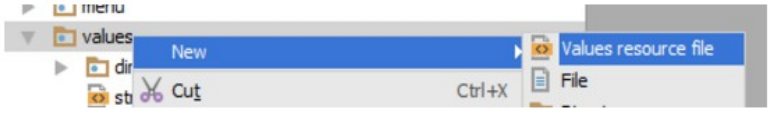
- Si la Activity pasa a segundo plano pero todavía es visible, solo se ejecutará onPause (por ejemplo con un diálogo)
- Si la Activity pasa por completo a segundo plano se ejecutará onPause y onStop.
- Si la Activity se cierra, se ejecutará también onDestroy.

Es importante aclarar que cuando el sistema operativo termina con la aplicación por requerimientos de RAM, el último método que se va a ejecutar con seguridad no es onDestroy sino onStop.

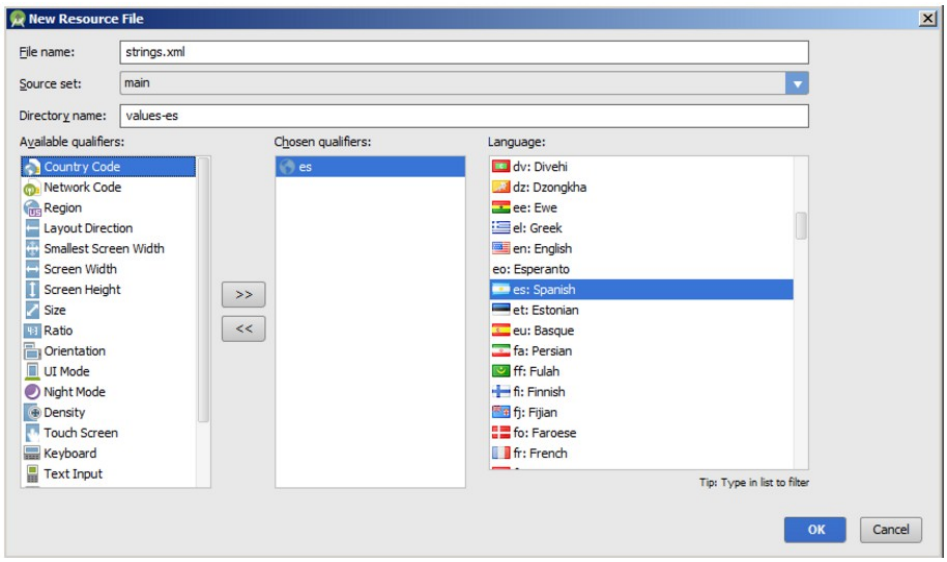
Internacionalización y Localización

Es importante tener la interfaz gráfica de la aplicación en el lenguaje correcto. Para ello, se nombran los directorios values con el idioma correspondiente. Por ej. values-es (español) o values-en para (inglés). Dentro de estos directorios colocaremos un archivos strings.xml en donde para las mismas "claves" definiremos textos en diferentes idiomas.

Para crear un nuevo directorio de values junto con un archivo strings.xml, hacemos click derecho sobre la carpeta value y seleccionamos new Value Resource File



Luego cargamos como nombre strings.xml y seleccionamos como modificador "Language". Veremos que el directorio pasará a llamarse values-es



Ponemos como ejemplo un archivo strings.xml para inglés y para español:

Inglés:

```
<resources>
    <string name="app_name">My Application</string>
    <string name="hello_world">Hello world!</string>
    <string name="action_settings">Settings</string>
</resources>
```

Español:

```
<resources>
    <string name="app_name">Mi Aplicación</string>
    <string name="hello_world">Hola mundo</string>
    <string name="action_settings">Opciones</string>
</resources>
```

Para obtener un string desde el código java:

```
String str = getString(R.string.hello_world);
```

Según el idioma del sistema operativo, el string tomará como valor lo que esta en un archivo o el otro.
Para utilizar estos strings desde un archivo xml, por ejemplo en un textview:

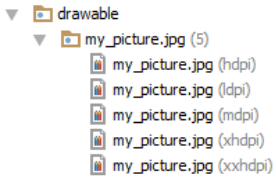
```
android:text:@string/hello_world
```

Además del idioma, podemos colocar modificadores por región, ya que a pesar de tener el mismo idioma diferentes países utilizan diferentes palabras.
Por ej. values-es-rAR para Argentina o values-es-rCO para Colombia.

El values sin modificadores será el por defecto. En el caso que no se encuentre ningún values para una región o idioma específico, se utilizará este xml por defecto.

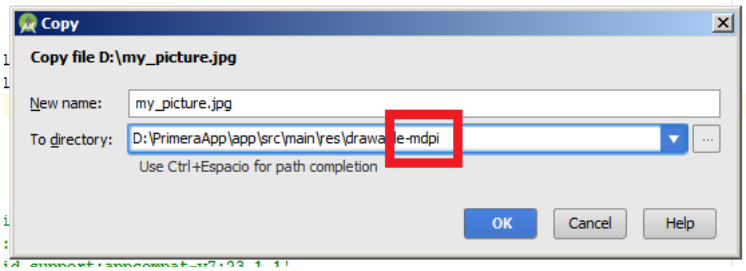
Resources: Strings, Colors y Pictures

De la misma forma que manejamos los textos por medio de strings.xml, podemos definir un archivo de colores colors.xml.
Con respecto a las imágenes, deberán colocarse en la carpeta drawable, las cuales tienen los modificadores de formato según la densidad de la pantalla:



Esto le permite al sistema operativo tomar la imagen del tamaño más parecido al tamaño que debe dibujar en pantalla, para evitar que se pixele frente a un gran rango de densidades de pantalla (variación de los dpi)
El tamaño de las imágenes debe especificarse en la unidad dp y no en pixeles, esta unidad es independiente de la densidad de la pantalla, logrando que la imagen se va del mismo tamaño a pesar de la variación en la densidad:

Para agregar una imagen en nuestro proyecto, en la carpeta drawable, indicando un modificador de formato, debemos seleccionar el archivo de imagen que queremos agregar a nuestro proyecto, presionar CTRL+C y luego hacer click sobre la carpeta drawable y presionar CTRL+V, aparecerá la siguiente ventana:



Agregamos en el nombre del directorio, "-mdpi" o cualquier otro modificador. Luego el IDE indicará con qué modificador se encuentra la imagen: