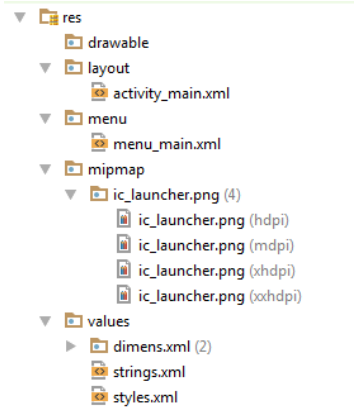
Recursos

Recursos

- Los recursos son archivos que fijan el comportamiento de nuestra aplicación Android: nos referimos a imágenes, "strings", colores, estilos, diseños de pantallas, etc.
- Siempre se deben externalizar dichos comportamientos en ficheros para que puedan **mantenerse de forma independiente** al código java/kotlin/xml y **reutilizarse** las veces que sean necesarias.
- Además, separar los recursos permite proporcionar alternativas para **dar soporte a las distintas configuraciones** de dispositivos como idiomas, tamaños de pantalla u orientación.

Carpeta **res**

- Contiene todos los ficheros de recursos necesarios para el proyecto: imágenes, vídeos, cadenas de texto, etc.
- Los diferentes **tipos de recursos** se deberán distribuir entre las siguientes carpetas (no todas estas carpetas tienen por qué aparecer en cada proyecto Android, tan sólo las que se necesiten):



- Los recursos que se guardan en los subdirectorios definidos en la tabla anterior son los recursos "predeterminados" . Es decir, estos recursos definen el diseño y el contenido predeterminados de la aplicación.

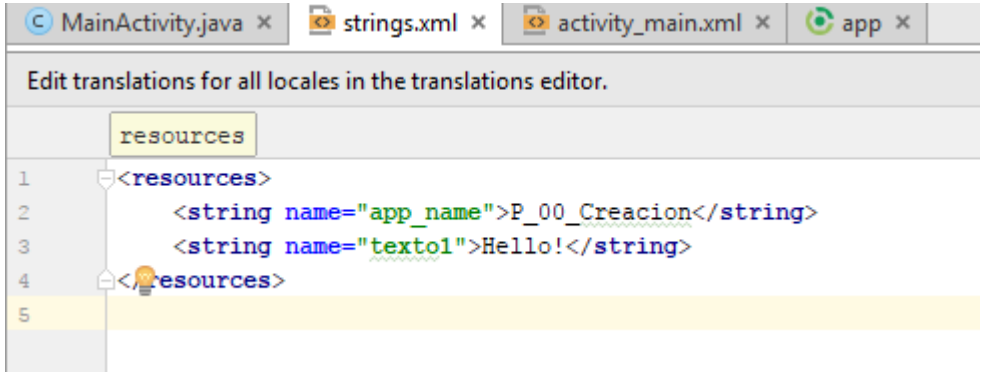
Nombre del subdirectorio	Contenido
`animator/`	Alberga archivos XML con animaciones de propiedades para objetos.
`anim/`	Contiene archivos XML que representan animaciones especiales para views.
`color/`	Aquí encontrarás archivos XML con definiciones de listas de estados de color (**Color State List**). Estas listas determinan el color de un componente dependiendo del estado en que se encuentre.
`drawable/`	Recursos gráficos que puedan ser proyectados en pantalla. Generalmente encontrarás archivos de imagen como **.png**, **.jpg** o **.gif**, sin embargo es posible usar otros como: archivos nine-patch, listas de estado, drawables con múltiples niveles, drawables con figuras 2D definidas en XML, y muchas más.
`mipmap/`	Contiene el icono de la aplicación para evitar distorsión entre varias densidades de pantalla.
`layout/`	Archivos XML que contienen definiciones de la interfaz de usuario.
`menu/`	Archivos XML que establecen las características para los menús usados en la interfaz. Normalmente contienen definiciones sobre los ítems albergados en un menú y las agrupaciones entre ellos.
`raw/`	Almacena aquí todos aquellos archivos que deseas leer directamente como un flujo de caracteres estándar (`InputStream`).
`values/`	Archivos XML que contienen datos simples como enteros, strings, booleanos, colores. Puedes agrupar estos elementos en las siguientes convenciones de archivos: ▪ **strings.xml** para cadenas ▪ **colors.xml** para definiciones de colores en hexadecimal. ▪ **dimens.xml** para dimensiones de márgenes, padding, tamaños, etc. ▪ **styles.xml** para los estilos de interfaz que tendrá la aplicación. ▪ **arrays.xml** para arreglos de elementos.
`xml/`	Contiene archivos XML con definiciones especiales requeridas en acciones con el framework de Android. O para usos que requieras en tus requerimientos, como parsear una estructura XML directamente en la aplicación.

Soporte por configuración

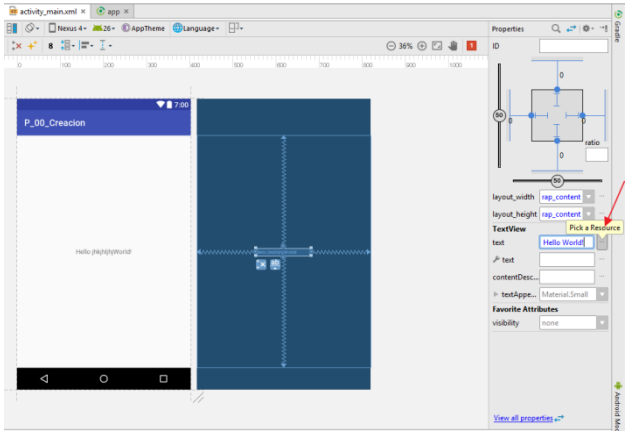
- A fin de brindar compatibilidad con configuraciones diferentes (idioma, tamaño, orientación, etc.), hay que organizar los recursos del directorio /res del proyecto, para lo cual debe usarse varios subdirectorios que agrupen recursos por tipo y configuración.
- Para cualquier tipo de recurso, hay que especificar un recurso predeterminado (OBLIGATORIO) y además varios recursos alternativos (OPTATIVOS):
  - Un recurso **predeterminado** es el que se usa sin importar la configuración del dispositivo o cuando no hay recursos alternativos que coincidan con la configuración encontrada el ejecutar la app. Se guarda según su tipo en las carpetas citadas anteriormente. (Ej.: strings.xml en carpeta values)
  - Un recurso **alternativo** es una variación de un recurso, que se ajusta a una característica de configuración en el dispositivo móvil donde se ejecuta la aplicación. A fin de especificar que un grupo de recursos es para una configuración específica, se agregan calificadores de configuración apropiados al nombre del directorio. (Ej.: strings.xml en carpeta values-es)
- En tiempo de ejecución, Android utilizará el recurso adecuado en función de la configuración encontrada.

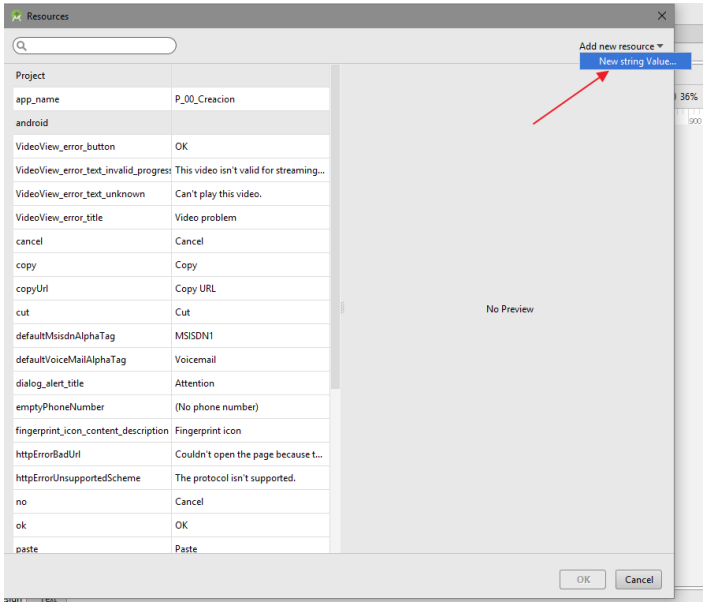
Archivo strings.xml

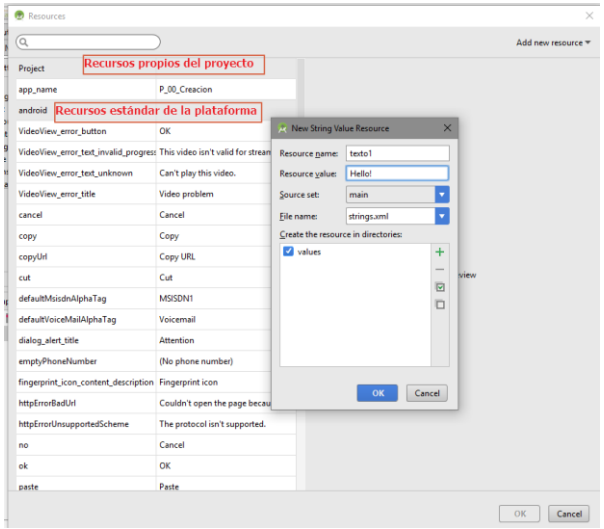
- El archivo strings.xml se utiliza para almacenar **todas las constantes de cadenas de caracteres que se necesitan en un programa**.
- La idea fundamental es tener todos los "mensajes" que muestra nuestra aplicación en un archivo XML y no en el código Java para facilitar la implementación de aplicaciones en múltiples idiomas.

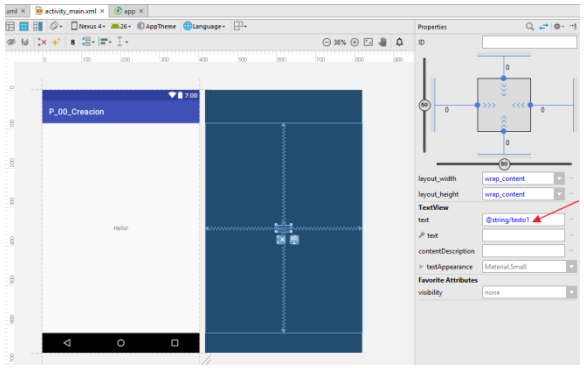


Práctica con strings.xml



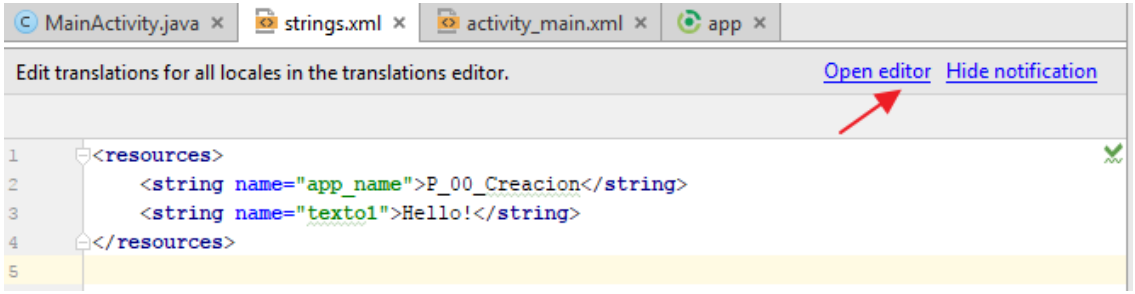


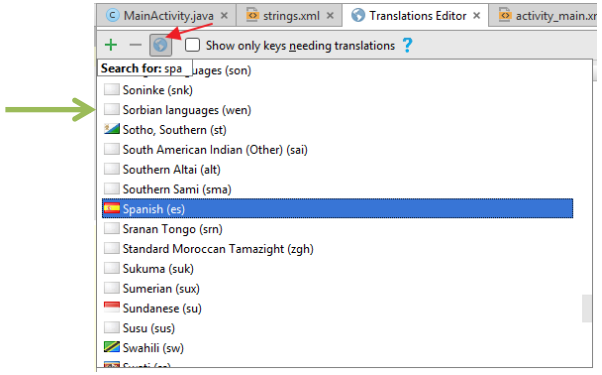


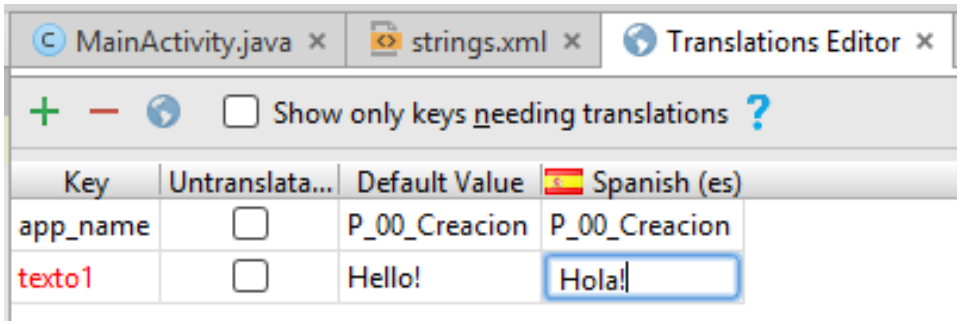


Ejemplo de soporte para idiomas

- Creando recursos alternativos resultará muy sencillo **traducir** una aplicación a otro idioma.
- AS proporciona asistente para la creación





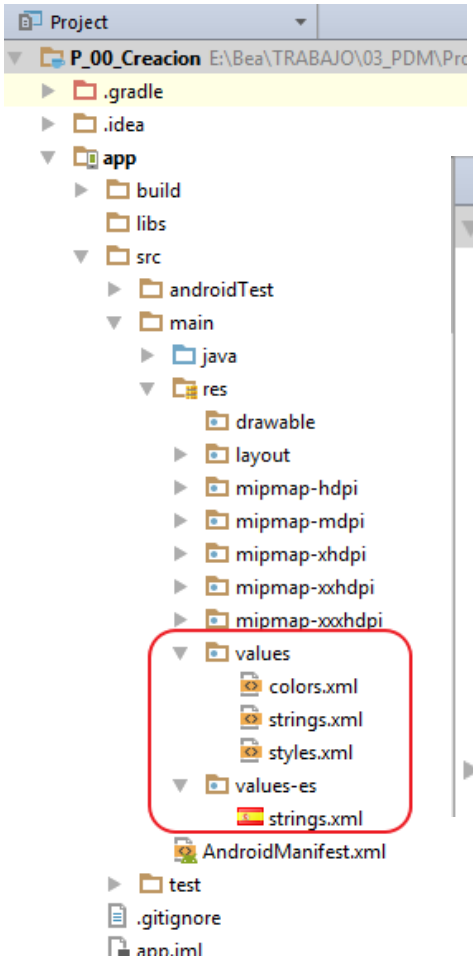


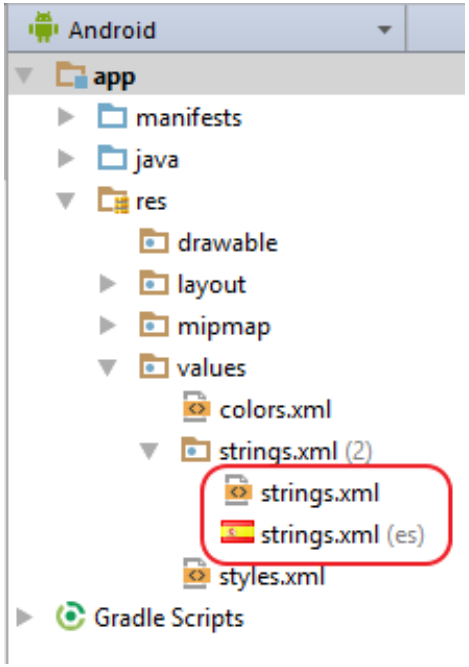
Calificadores para recursos alternativos

- Un recurso alternativo es una variación de un recurso, que se ajusta a una característica de configuración en el dispositivo móvil donde se ejecuta la aplicación.
- Esto se logra a través de una tabla de **calificadores** creada por Google que estandariza todas las configuraciones posibles que se puedan presentar.
- Un calificador es un mecanismo gramatical que especifica el propósito de un recurso. Su sintaxis es la siguiente:

  <nombre_recurso>-<calificador>

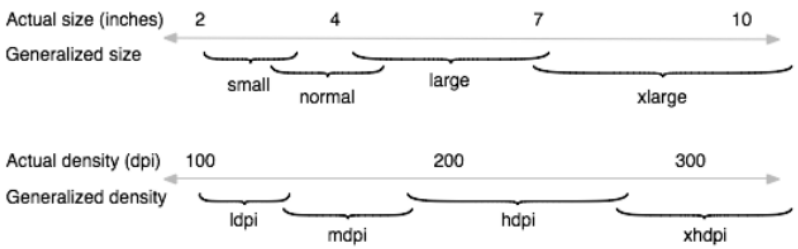
  Ej.: values-es especifica que el recurso se usará en dispositivos con ajuste de idioma Español
- Todas las variaciones responden a un mismo identificador, por eso tienen el mismo nombre de archivo "strings.xml".



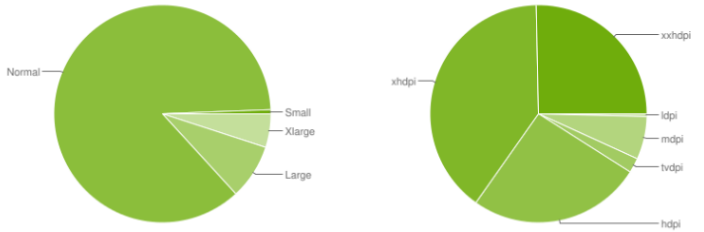


Soporte para diversas pantallas

- Las aplicaciones se ejecutan en una variedad de dispositivos que ofrecen diferentes tamaños de pantalla y densidades.
- **Tamaño** de pantalla: Tamaño físico real, medido como la diagonal de la pantalla. Se agrupan en 4 tipos: small, normal,large, xlarge..
- **Densidad**: La cantidad de pixeles dentro de un área física de la pantalla; normalmente se conoce como ppp (puntos por pulgada). Se simplifican en 6: ldpi (low) ~120dpi, mdpi (medium) ~160dpi, hdpi (high) ~240dpi, xhdpi (extra-high) ~320dpi, xxhdpi (extra-extra-high) ~480dpi, xxxhdpi (extra-extra-extra-high) ~640dpi
- **Orientación**: en general, vertical para teléfonos y horizontal para tablets
- Lectura obligatoria: [enlace 1]
- Hay que diseñar para el máximo nº de combinaciones de tamaño y densidad (recuerda: imágenes, iconos, IU's, etc.), pero puede resultar abrumador; siempre hay que recordar la cuota de mercado☺!



	ldpi	mdpi	tvdpi	hdpi	xhdpi	xxhdpi	Total
Small	0.4%				0.1%	0.1%	0.6%
Normal		0.9%	0.3%	24.0%	37.7%	23.6%	86.5%
Large		2.4%	1.9%	0.6%	1.6%	1.7%	8.2%
Xlarge		3.1%		1.3%	0.6%		5.0%
Total	0.4%	6.4%	2.2%	25.9%	40.0%	25.4%	



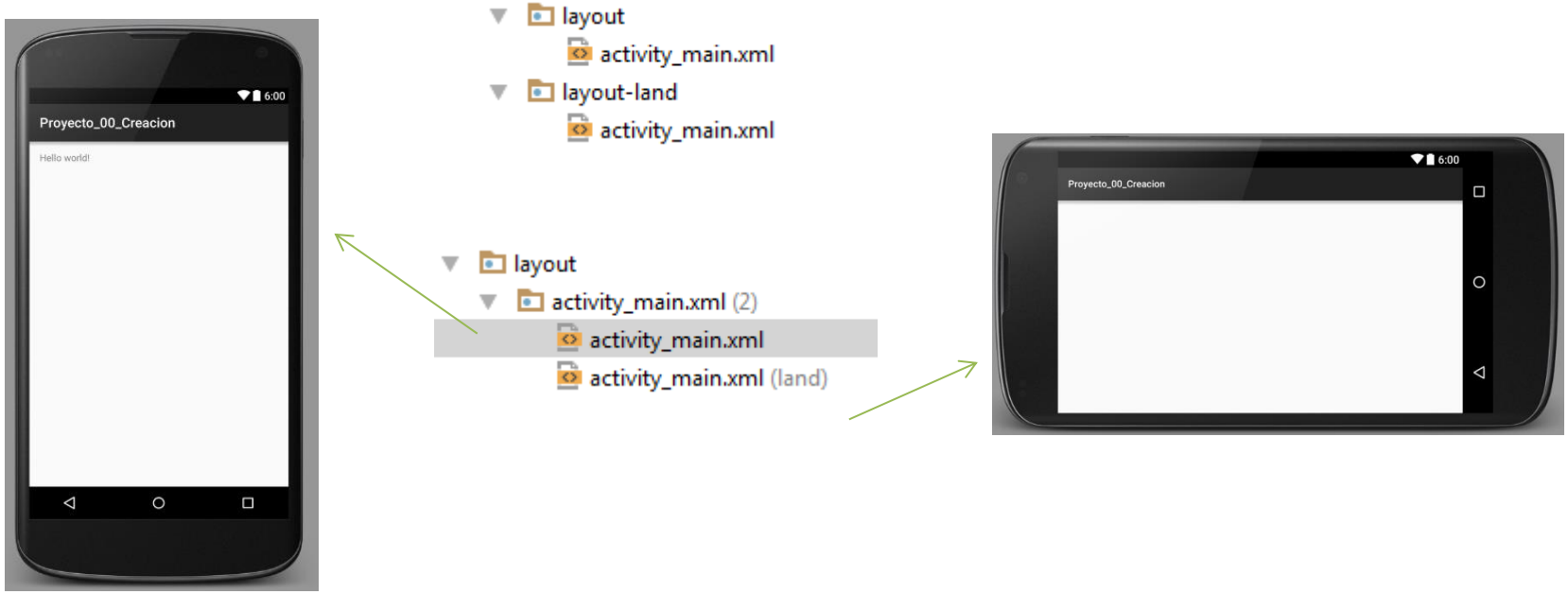
Data collected during a 7-day period ending on May 7, 2019.
Any screen configurations with less than 0.1% distribution are not shown.

Calificadores para pantallas

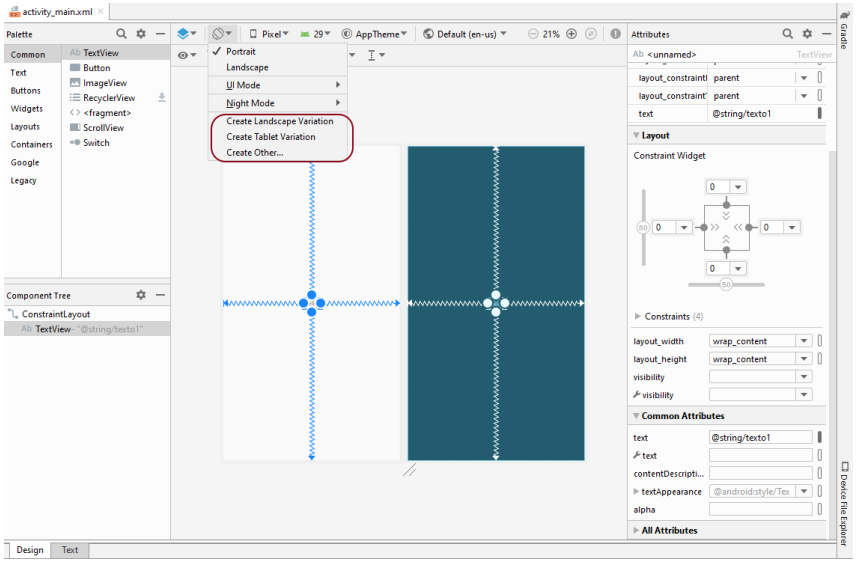
Características de la pantalla	Calificador	Descripción
Tamaño	`small`	Recursos para pantallas de tamaño *pequeño*.
	`normal`	Recursos para pantallas de tamaño *normal*. (Este es el tamaño de referencia).
	`large`	Recursos para pantallas de tamaño *grande*.
	`xlarge`	Recursos para pantallas de tamaño *extragrande*.
Densidad	`ldpi`	Recursos para pantallas de densidad baja (*ldpi*) (~120 dpi).
	`mdpi`	Recursos para pantallas de densidad media (*mdpi*) (~160 dpi). (Esta es la densidad de referencia).
	`hdpi`	Recursos para pantallas de densidad alta (*hdpi*) (~240 dpi).
	`xhdpi`	Recursos para pantallas de densidad extraalta (*xhdpi*) (~320 dpi).
	`xxhdpi`	Recursos para pantallas de densidad extra extraalta (*xxhdpi*) (~480 dpi).
	`xxxhdpi`	Recursos para usos de densidad extra extra extraalta (*xxxhdpi*) (~640 dpi). Usa esto únicamente para el ícono lanzador, consulta la nota anterior.
	`nodpi`	Recursos para todas las densidades. Estos son recursos independientes de la densidad. El sistema no ajusta recursos que reciben etiquetas con este calificador, independientemente de la densidad de pantalla actual.
	`tvdpi`	Recursos para pantallas entre mdpi y hdpi; aproximadamente 213 dpi. Este no se considera un grupo de densidad "primario". Se usa principalmente para televisiones, y la mayoría de las aplicaciones no lo necesitarán; con recursos mdpi y hdpi será suficiente para la mayoría de las apps y el sistema los ajustará según corresponda. Si crees necesario proporcionar recursos tvdpi, debes ajustar su tamaño con un factor de 1,33*mdpi. Por ejemplo, una imagen de 100 x 100 px para pantallas mdpi debe ser de 133 x 133 px para tvdpi.
Orientación	`land`	Recursos para pantallas con orientación horizontal (relación de aspecto ancha).
	`port`	Recursos para pantallas con la orientación vertical (relación de aspecto alta).
Relación de aspecto	`long`	Recursos para pantallas con una relación de aspecto considerablemente más alta o ancha (con orientación vertical u horizontal, respectivamente) que la configuración de pantalla de referencia.
	`notlong`	Recursos para pantallas con una relación de aspecto similar a la configuración de pantalla de referencia.

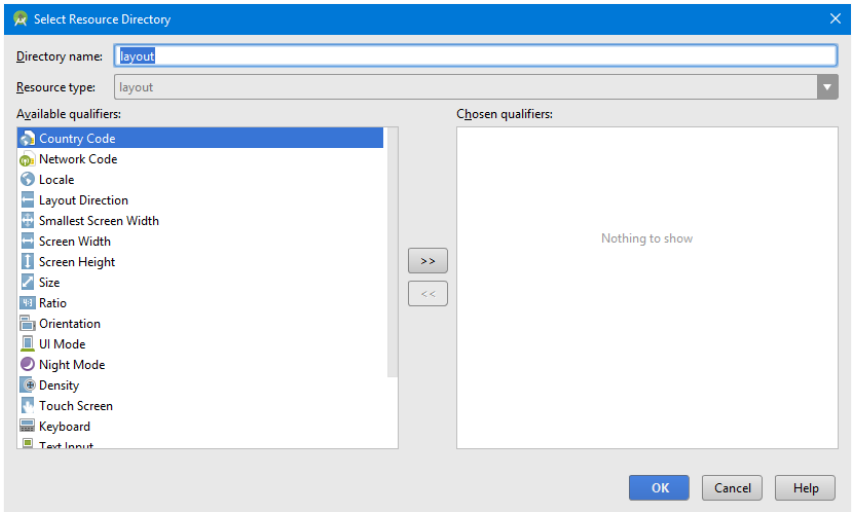
Ejemplo de soporte para UI según orientación

- Se usan los sufijos ***port*** para vertical (es el de por defecto para tfnos.) y ***land*** para horizontal



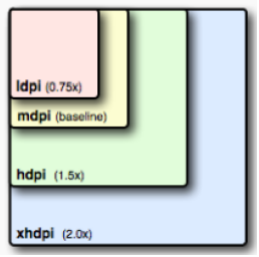
Asistentes de creación de variaciones de UI



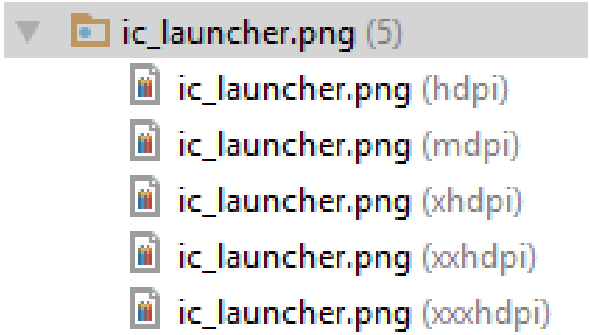


Ejemplos de soporte para imágenes

- Para lograr una buena calidad gráfica y rendimiento en todas las densidades de pantalla, siempre hay que proporcionar recursos de que se ajustan correctamente a cada una de las densidades.
- Obligatoriamente para cuatro: hdpi, mdpi, xdpi y xxhdpi
- Si se trata de un icono lanzador también xxxhdpi
- Por la baja cuota de mercado, se obvía ldpi
- Ejemplo: En el proyecto se ha incluido el fichero ***ic_launcher.png*** que será utilizado como icono de la aplicación. Si pulsas sobre las diferentes versiones del recurso observarás como se trata del mismo icono pero con más o menos resolución, de forma que en función de la densidad gráfica del dispositivo se ocupe un tamaño similar en la pantalla.
- Esto significa que si se usa una imagen 200x200 para dispositivos xhdpi, debe generarse el mismo recurso en 150x150 para hdpi, 100x100 para mdpi y 75x75 para dispositivos ldpi.

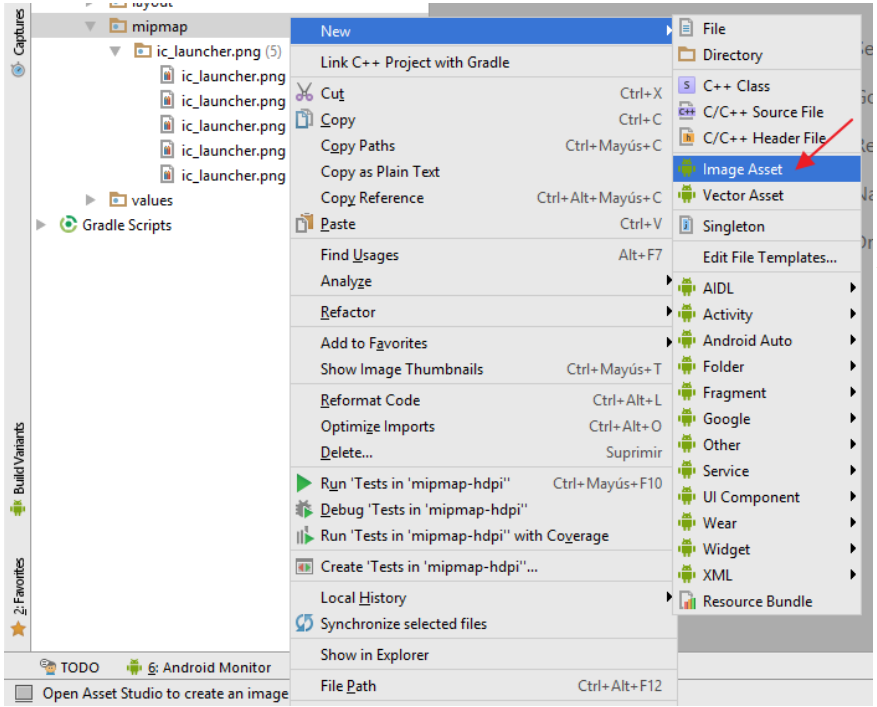


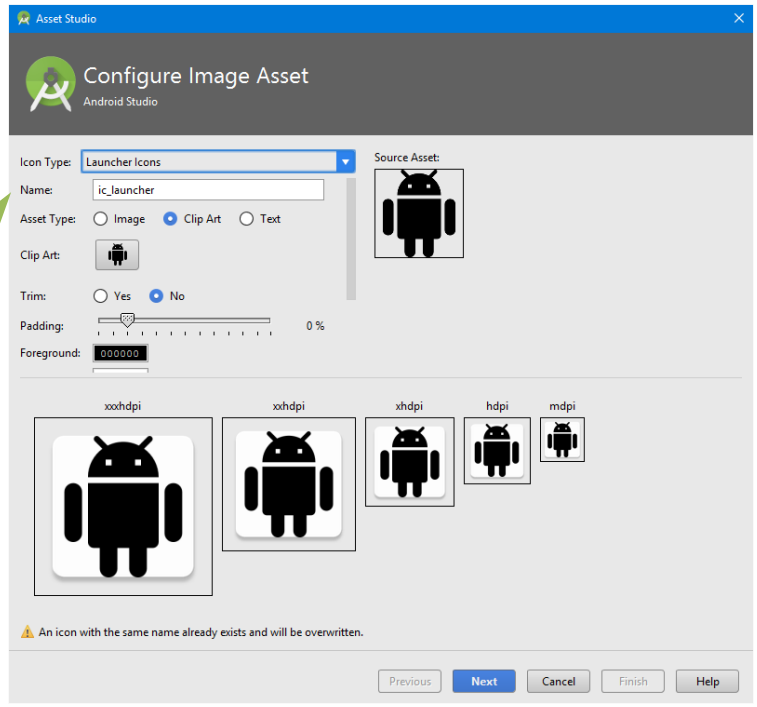
- *ldpi* (low) ~120dpi
- *mdpi* (medium) ~160dpi
- *hdpi* (high) ~240dpi
- *xhdpi* (extra-high) ~320dpi
- *xxhdpi* (extra-extra-high) ~480dpi
- *xxxhdpi* (extra-extra-extra-high) ~640dpi



Asset Studio

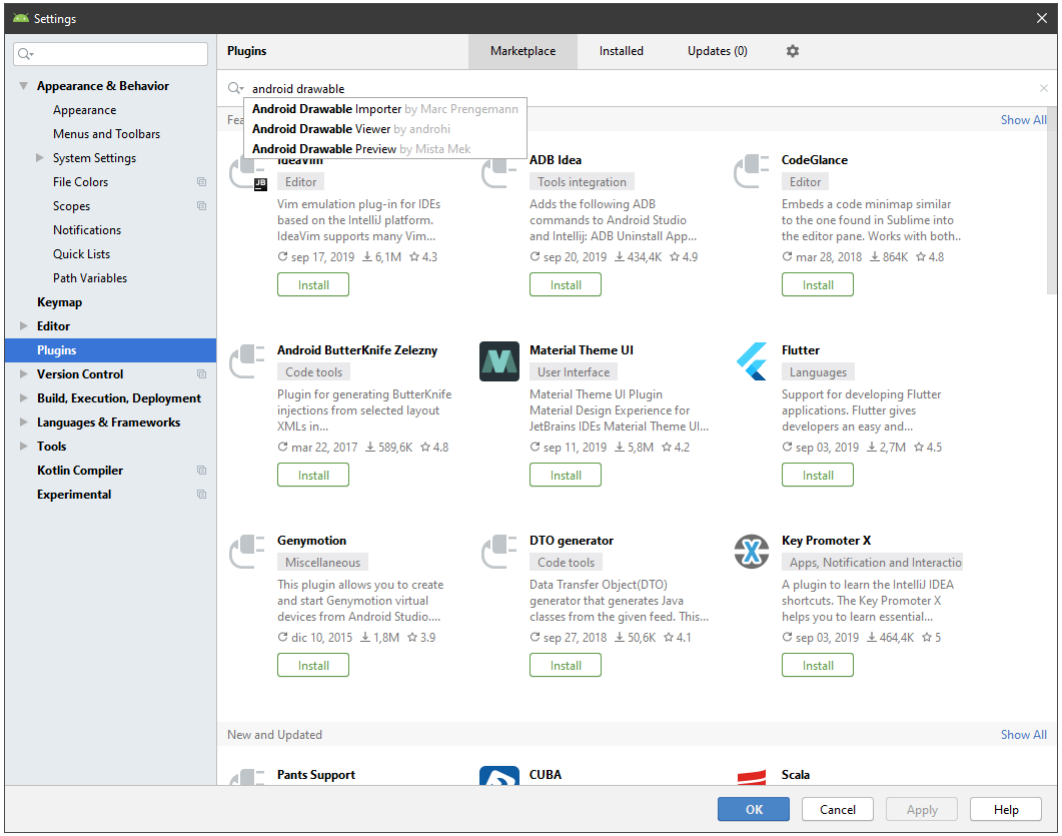
- La creación de iconos (mapas de bits) de diferentes densidades puede ser un poco tedioso.
- Por ello AS nos ofrece el asistente Asset Studio



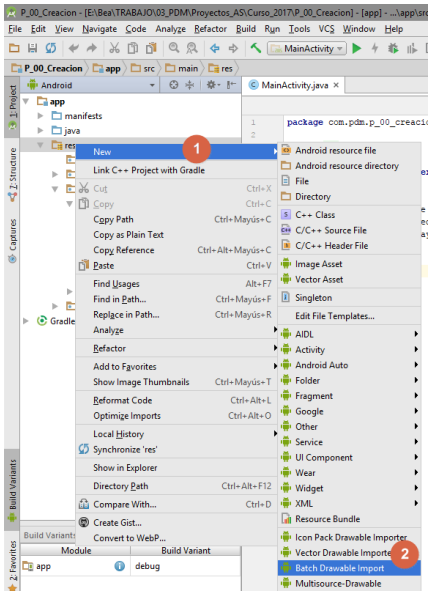


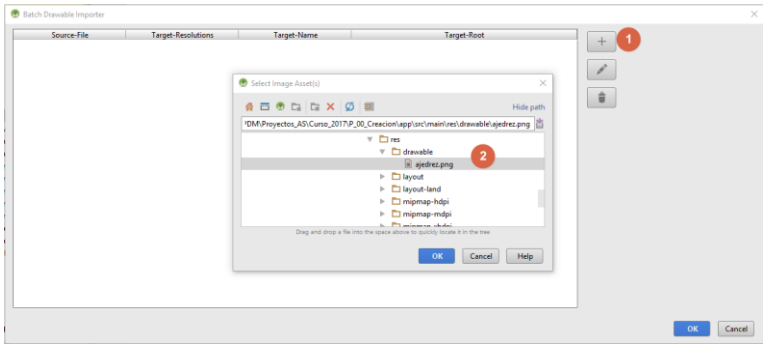
Plugin recomendable

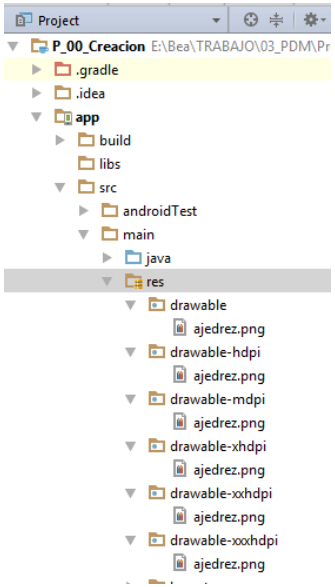
- Android Drawable Import: [información]

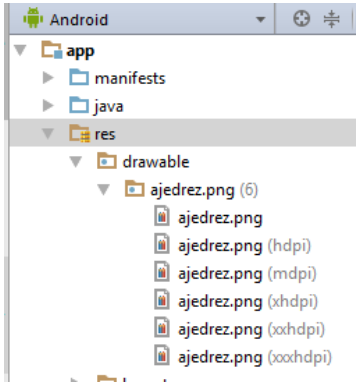


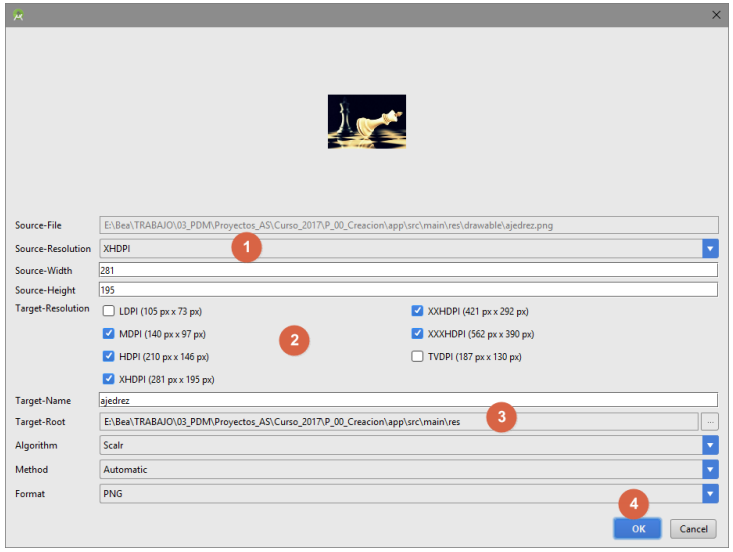
Usando el plugin











Soporte para versiones

- Por ejemplo, los recursos incluidos en la carpeta llamada “values-**v11**” se aplicarían tan sólo a dispositivos cuya versión de Android sea la 3.0 (API 11) o superior.

Lista de calificadores

- Enlace

Tabla 2: Nombres de calificadores de configuración.

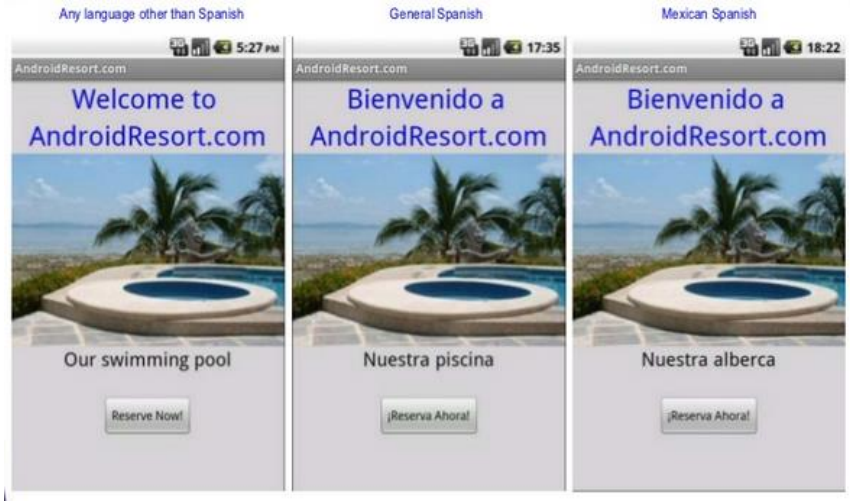
Configuración	Valores de calificadores	Descripción

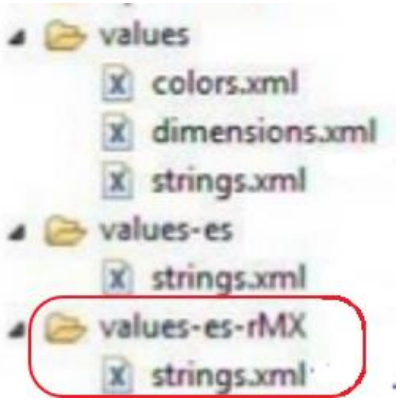
MCC y MNC	Ejemplos:
	`mcc310`
	`mcc310-mnc004`
	`mcc208-mnc00`
	etc.
		El código de país de móvil (MCC), opcionalmente seguido del código de red móvil (MNC) de la tarjeta SIM del dispositivo. Por ejemplo, `mcc310` se refiere a los Estados Unidos con cualquier operador, `mcc310-mnc004` se refiere a los Estados Unidos con Verizon, y `mcc208-mnc00` se refiere a Francia con Orange.
		
		Si el dispositivo utiliza una conexión de radio (teléfono GSM), los valores del MCC y el MNC provienen de la tarjeta SIM.
		
		También puedes utilizar el MCC solo (por ejemplo, para incluir recursos legales de países específicos en tu aplicación). Si necesitas especificar solamente en función del idioma, utiliza el calificador de *idioma y región* (se trata a continuación). Si decides utilizar el calificador de MCC y MNC, debes hacerlo con cuidado y comprobar que funcione de la forma esperada.
		
		También observa los campos de configuración `mcc` y `mnc`, que indican el código móvil de país y el código de red móvil actuales, respectivamente.

Idioma y región	Ejemplos:
	`en`
	`fr`
	`en-rUS`
	`fr-rFR`
	`fr-rCA`
	`b+en`
	`b+en+US`
	`b+es+419`
		El idioma se define mediante un código de idioma ISO 639-1 de dos letras, opcionalmente seguido de un código de región ISO 3166-1-alfa-2 de dos letras (precedido de "`r`" en minúscula).
		
		Los códigos *no* distinguen mayúsculas de minúsculas; el prefijo `r` se utiliza para distinguir la parte de la región. No se puede especificar una región sola.
		
		Android 7.0 (nivel de API 24) presentó compatibilidad con las etiquetas de idioma BCP 47, que puedes usar para la calificación de recursos de idiomas y regiones específicos. Una etiqueta de idioma se compone de una secuencia de una o más subetiquetas. Cada una acota o afina la variedad del idioma que identifica la etiqueta general. Para obtener más información sobre etiquetas de idioma, consulta Etiquetas para identificar idiomas.
		
		Para usar una etiqueta de idioma BCP 47, concatena `b+` y un código de idioma ISO 639-1 de dos letras, al que puedes agregar dos subetiquetas separadas con `+`.
		
		La etiqueta de idioma puede cambiar mientras se usa tu aplicación si los usuarios cambian el idioma en la configuración del sistema. Consulta la sección Manejo de cambios en tiempo de

Reglas de calificadores

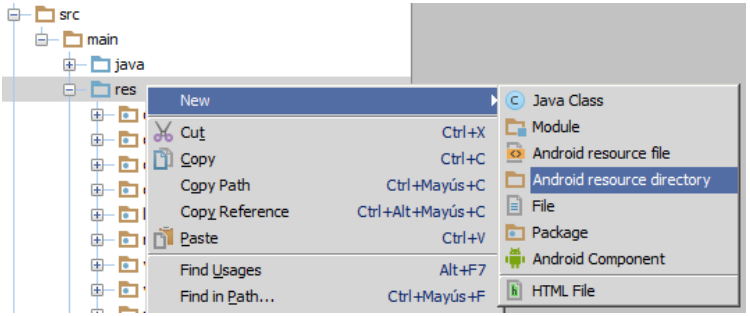
- Se pueden especificar varios calificadores a un mismo conjunto de recursos, separándolos por guiones.
- Los cualificadores deben estar en el orden de la lista anterior. Por ejemplo:
  - Incorrecto: drawable-hdpi-port
  - Correcto: drawable-port-hpdi
- Las carpetas de recursos alternativos no pueden ser anidadas. Por ejemplo:
  - Incorrecto: res/drawable/drawable-en
  - Correcto: res/drawable-en
- Los nombres de calificadores no consideran las diferencias entre mayúsculas y minúsculas.
- Sólo es válido un valor para cada tipo de calificador. Por ejemplo:
  - Incorrecto: drawable-es-fr
  - Correcto: drawable-es y otra carpeta drawable-fr.
- Lo que sí se puede hacer es crear alias entre recursos.

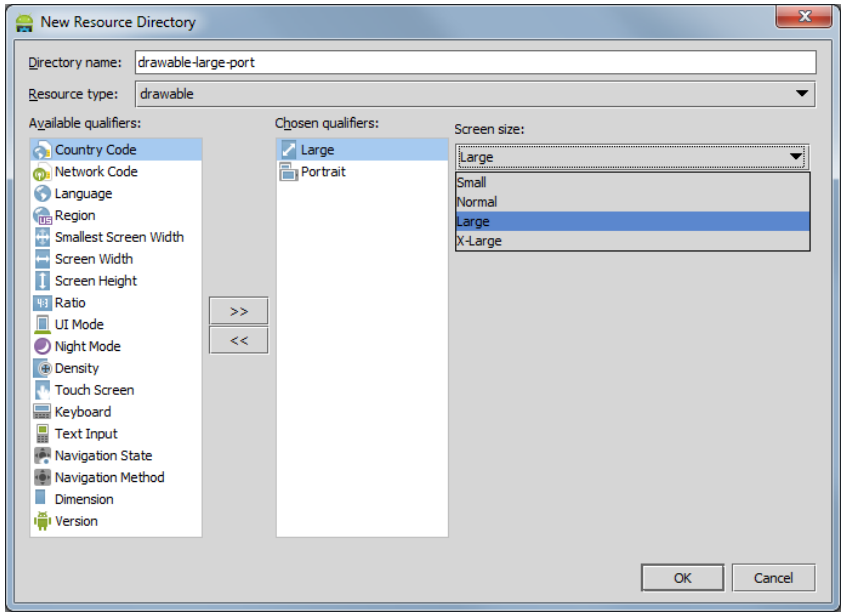


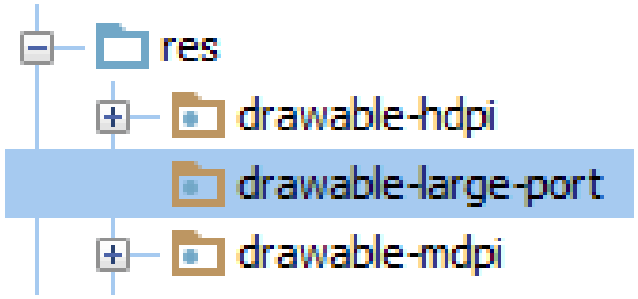


Creación de recursos alternativos

- Android Studio proporciona asistentes:







Prioridad de los recursos alternativos

- Android utilizará automáticamente en la aplicación los recursos basados en la configuración actual del dispositivo.
- Cada vez que se solicita un recursos, Android comprueba si hay carpetas de recursos alternativos que contengan el recurso solicitado y entonces localiza el **recurso que mejor se adapta** a la configuración actual.
- Si no hay recursos alternativos para la configuración actual, Android usará los recursos correspondientes **por defecto** (los que no incluyen un cualificador).
- No todos los calificadores están en todas las API's. Para evitar problemas siempre hay que incluir un conjunto de recursos por defecto

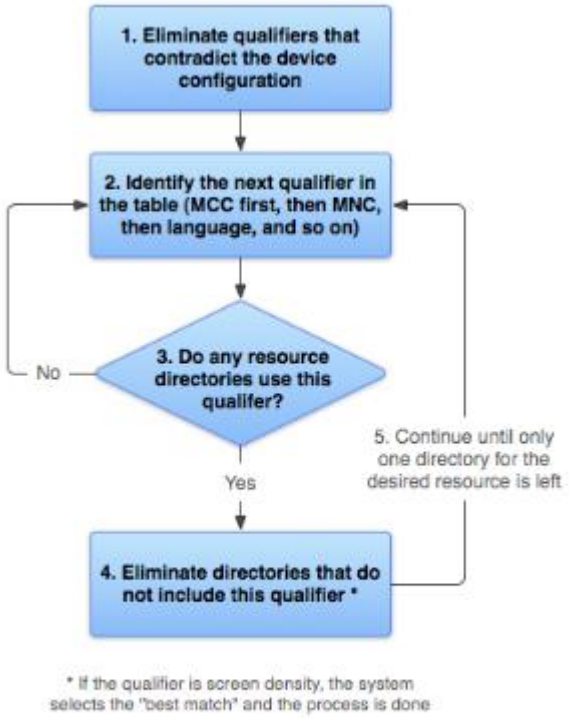
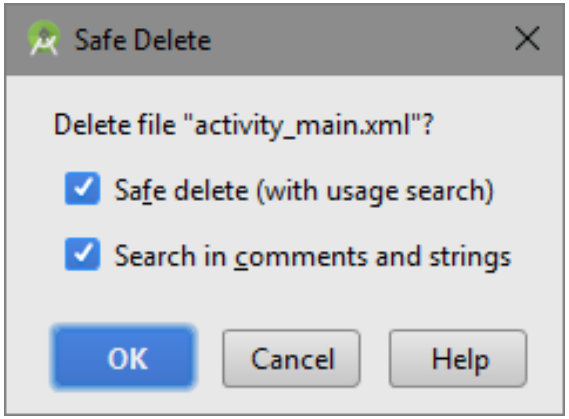
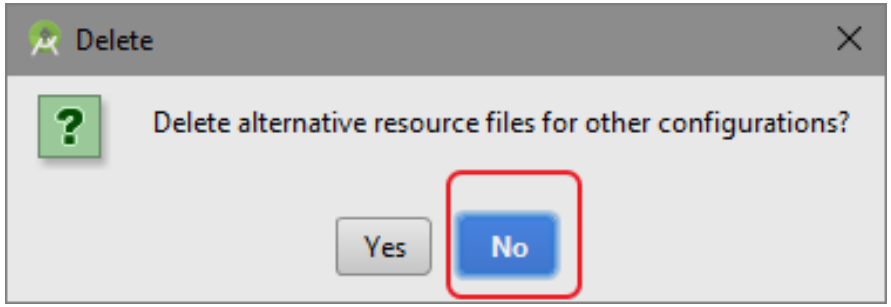


Figura 2: Diagrama de flujo del método que usa Android para encontrar el recurso de coincidencia óptima

Ojo al borrar recursos alternativos

- Si se borra un recurso alternativo (por ejemplo, el layout en horizontal) hay que tener cuidado de no borrar también todos los demás, incluido el de por defecto.
- AS solicita confirmación al borrar, pero es fácil equivocarse por las prisas!





Acceso a los recursos

- Una vez que se externalizan los recursos para la aplicación, se puede acceder a ellos mediante los ID de recursos que se generan en la clase R del proyecto.
- El ID de recurso siempre está compuesto por tipo (que coincide con el de la carpeta) y nombre (que es el nombre sin extensión si es un fichero o el contenido del atribute android:name si es un valor simple como una string).
- Existen dos maneras de acceder a un recurso:
  - **En código:** Usando un valor entero de una subclase de tu clase R. Sintaxis:

    [<*package_name*>.]R.<*resource_type*>.<*resource_name*>

    Ej.: R.string.hello, string es el tipo de recurso y hello es el nombre del recurso. Hay muchas API de Android que pueden acceder a los recursos cuando proporcionas un ID de recurso en este formato. El editor inteligente nos facilita la tarea.
  - **En XML:** Sintaxis:

    @[<*package_name*>:]<*resource_type*>/<*resource_name*>

    Ej.:@string/hello, donde string es el tipo de recurso y hello es el nombre del recurso. Los asistentes suelen facilitarnos la tarea.

```
// Load a background for the current screen from a drawable resource
getWindow().setBackgroundDrawableResource(R.drawable.my_background_image) ;

// Set the Activity title by getting a string from the Resources object, because
//  this method requires a CharSequence rather than a resource ID
getWindow().setTitle(getResources().getText(R.string.main_title));

// Load a custom layout for the current screen
setContentView(R.layout.main_screen);

// Set a slide in animation by getting an Animation from the Resources object
mFlipper.setInAnimation(AnimationUtils.loadAnimation(this,
        R.anim.hyperspace_in));

// Set the text on a TextView object using a resource ID
TextView msgTextView = (TextView) findViewById(R.id.msg);
msgTextView.setText(R.string.hello_message);
```

MainActivity.java × activity_main.xml ×

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.constraint.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/an
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context="com.pdm.p_00_creacion.MainActivity">

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/texto1"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintRight_toRightOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

</android.support.constraint.ConstraintLayout>
```

Acceso a recursos de la plataforma

- Android contiene una cantidad de recursos estándar, como strings, estilos, temas y diseños.
- Para acceder a esos recursos, califica la referencia a tu recurso con el nombre de paquete android.
- En código:

  android.R.layout.simple_list_item_1
- En XML:

  "@android:string/cut"