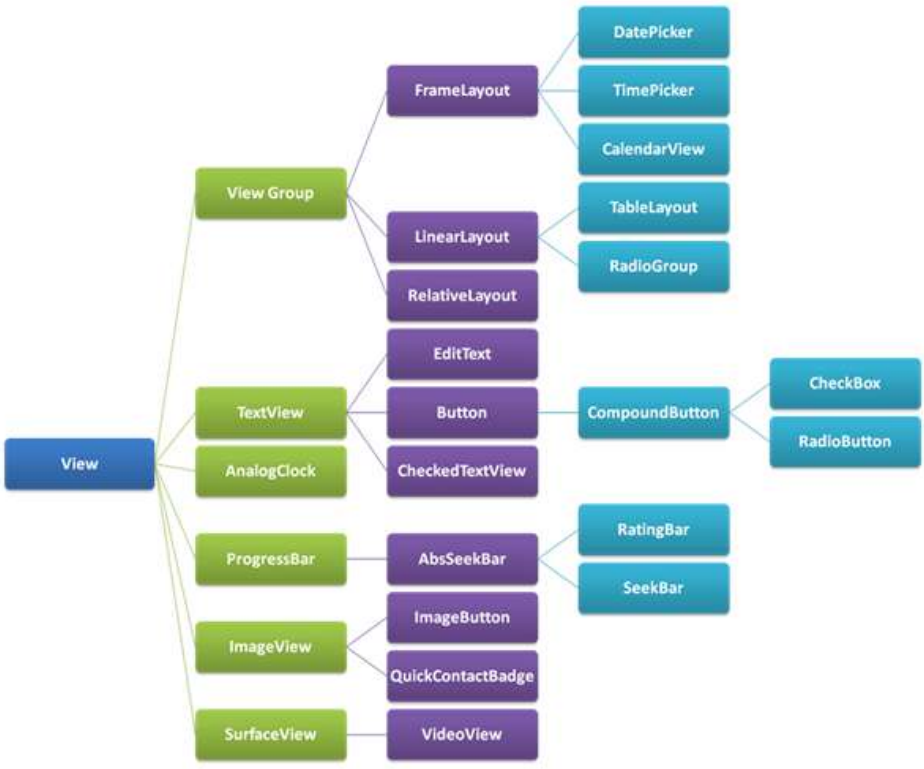
INDICE

1.1. Introducción

- Un control **ImageView** muestra cualquier imagen en la pantalla.
- Permite ser escalada (**android:scaleType**) y tinguida (**android:tint**)
- La clase **ImageView** hereda directamente da clase View



- **Referencias**:
  - ✓ https://developer.android.com/reference/android/widget/ImageView.html

1.2. Caso práctico

- Partimos que ya tenemos creado el proyecto inicial.

  Si no lo tenemos creado antes, crear un paquete de nome **UI** como un subpaquete de tu paquete principal.

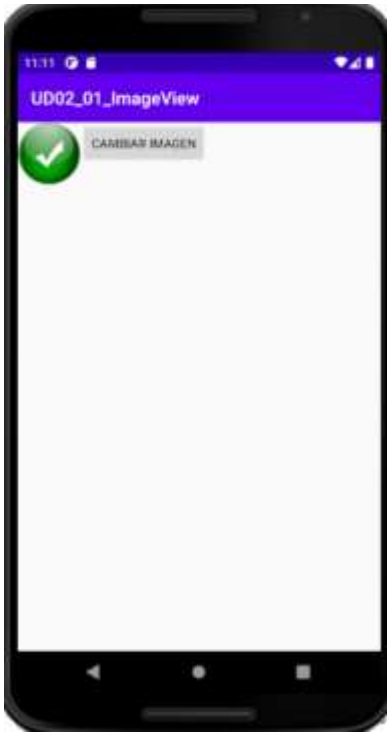
  Dentro do paquete UI, crearemos un nuevo paquete de nombre: **ImagesViews**.

- Dentro del paquete **ImagesViews** crear una nueva 'Empty Activity' de nombre: **UD02_01_ImageView** de tipo Launcher.

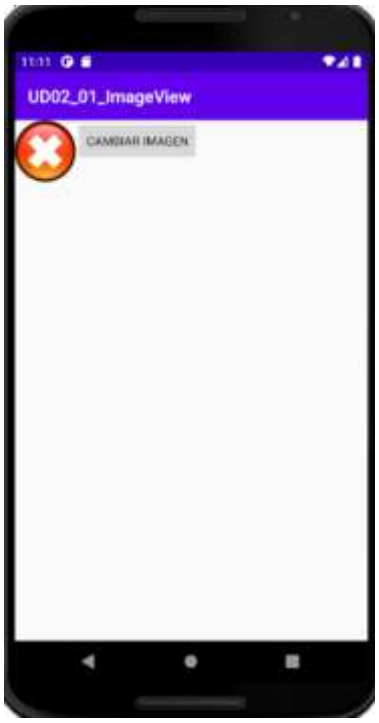
  Modificar el archivo **AndroidManifiest.xml** y añade un label a la activity como ya vimos en la creación del proyecto base.

- Creamos una aplicación que carga una imagen "OK" y un botón por defecto.
- Cuando hacemos clic en el botón o la imagen cambia la imagen a "NO-OK" y así sucesivamente.

ImageView



Inicializada la aplicación



Al hacer clic en el botón cambia la imagen o si hacemos clic en la imagen también la cambia.

Recursos de la imagen

- La carpeta **/res/values/drawable** ya debería estar creada.

  Recuerda que en la vista 'Android' las carpetas físicas están ocultas.

  Si cambiamos la vista de 'Proyecto' o 'Archivos de proyecto' podemos comprobar que tenemos varias carpetas dibujables, una para cada tipo de pantalla.

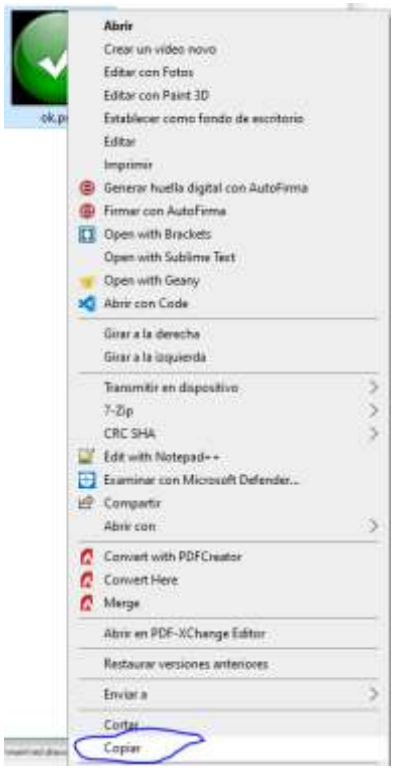
  Cuando agregamos nuevas imágenes a nuestro proyecto, deberíamos haber creado una imagen adaptada a la resolución de cada pantalla.

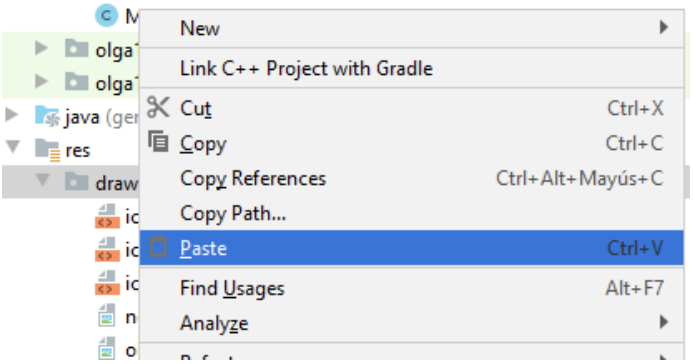
  Importante: Recordar que todos los recursos deben estar en minúsculas y solo se permiten letras minúsculas, números y guiones.

Recursos de imagénes

Descargar el botón **ok.png**

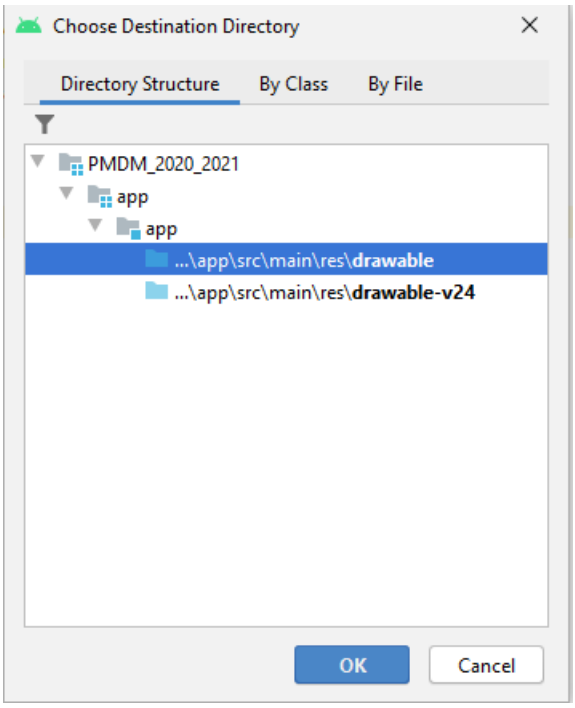
Descargar el botón **no.png**

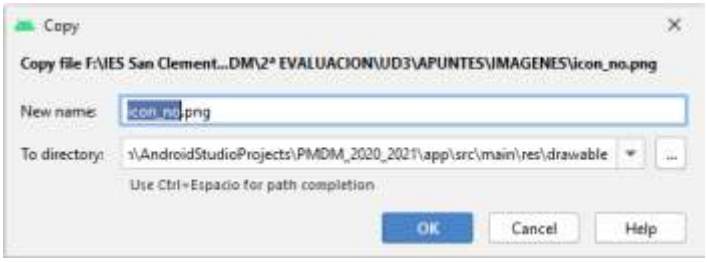




Después la pegamos en la carpeta **drawable**.

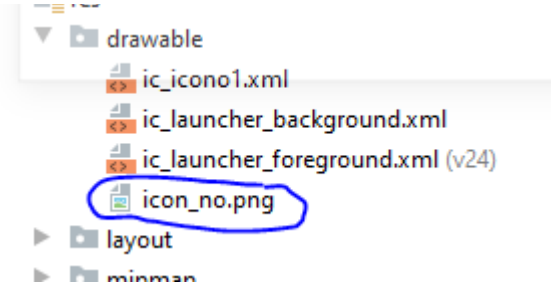
Para añadir una imagen, la copiamos al portapapeles.



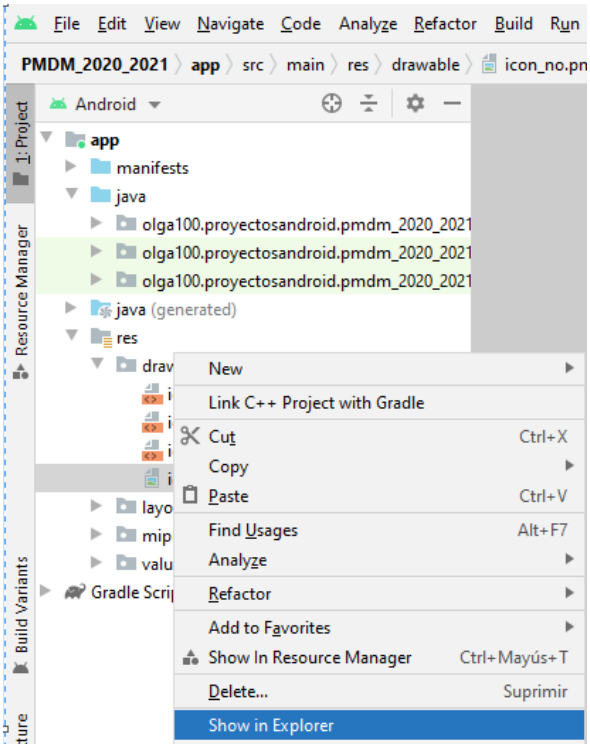


Indicamos el nombre de la imagen.

Elegimos la carpeta. Con drawable, esta imagen se visualizará si no existe una específica en una resolución concreta.

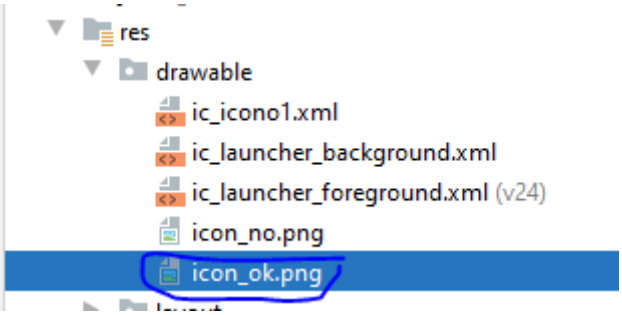


Aparece la imagen en la carpeta indicada.



Otra forma de hacerlo sería escogiendo un recurso que se encuentre en esa carpeta y escoger la opción **Show in explorer**.

Ahora solamente tenemos que copiar as imágenes que queramos.



La imagen aparece automáticamente en AndroidStudio

XML del Layout

- Observar cómo se referencia la imagen que cargará el control (línea 12).
- Los dos controles llaman al mismo método en el caso de hacer clic en ellos.

activity_u_d02_01__image_view.xml | UD02_01_ImageView.java

```
1  <?xml version="1.0" encoding="utf-8"?>
2  <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
3      xmlns:app="http://schemas.android.com/apk/res-auto"
4      xmlns:tools="http://schemas.android.com/tools"
5      android:layout_width="match_parent"
6      android:layout_height="match_parent"
7      tools:context=".UI.ImagesViews.UD02_01_ImageView">
8
9      <ImageView
10         android:id="@+id/img_imagen"
11         android:layout_width="wrap_content"
12         android:layout_height="wrap_content"
13         android:contentDescription="Imagen"
14         android:src="@drawable/icon_ok"
15         android:onClick="onCambiarImagenClick"
16         android:tag="OK" />
17
18     <Button
19         android:id="@+id/btn_cambiar_imagen"
20         android:layout_width="wrap_content"
21         android:layout_height="wrap_content"
22         android:onClick="onCambiarImagenClick"
23         android:text="Cambiar imagen" />
24 </LinearLayout>
25
```

- **Línea 16**: Se pone una etiqueta a la imagen para poder controlar por código cuál es la imagen cargada.

Código Java

UD02_01_ImageView.java

```
1  package olga100.proyectosandroid.pmdm_2020_2021.UI.ImagesViews;
2
3  import androidx.appcompat.app.AppCompatActivity;
4
5  import android.os.Bundle;
6  import android.view.View;
7  import android.widget.ImageView;
8
9  import olga100.proyectosandroid.pmdm_2020_2021.R;
10
11 public class UD02_01_ImageView extends AppCompatActivity {
12
13     public void onCambiarImagenClick(View view) {
14         //ImageView img = (ImageView) view;
15
16         ImageView img = (ImageView) findViewById(R.id.img_imagen);
17         if (img.getTag() == "OK") {
18             img.setTag("NO");
19             img.setImageResource(R.drawable.icon_no);
20         } else {
21             img.setTag("OK");
22             img.setImageResource(R.drawable.icon_ok);
23         }
24     }
25     @Override
26     protected  void onCreate (Bundle savedInstanceState) {
27         super.onCreate(savedInstanceState);
28         setContentView(R.layout.activity_u_d02_01__image_view);
29 }
30 }
```

- **Línea 17:** Comprobamos cuál es la Tag de la imagen cargada.
- **Línea 18, 19:** si la imagen es ok.png, cargamos la de no.png y cambiamos la etiqueta. Lo mismo se hace en las líneas 22 y 23.
- A través de la clase Java R accedemos a las imágenes.
- Línea 13: si se llama al método solo desde la imagen y no desde el botón podríamos hacer uso del parámetro view del método y comentar en la línea 15.
- Uno de los atributos que podremos modificar en un ImageView es ScaleType.

  Este atributo, basado en los valores permitidos, cambiará el tamaño de la imagen para que se ajuste a su contenedor.

- Podéis consultar en este enlace ejemplos de cómo sería una imagen aplicando diferentes valores para este atributo.

1.3. Accediendo a los recursos Drawable

- Como hemos mencionado anteriormente se puede acceder a todos los recursos que se encuentran en /res/ usando el método *getResources()*.
- En el caso de las imágenes utilizadas por ImageView, también podremos obtener una referencia a ellas de la siguiente manera:

```
1 Drawable drawable =  getResources().getDrawable(R.drawable.empresa,getTheme());
2 ImageView img = new ImageView(this);
3 img.setImageDrawable(drawable);
```

El primer argumento es la identificación de las imágenes (**tener en cuenta** que el tipo es R.**drawable** y en vuestro proyecto tendréis que indicar una imagen que exista)

El segundo argumento es el theme de la Activity que se puede obtener llamando al método getTheme().

Nota: Lógicamente, podríamos usar el método setImageResource (int resId) para asignar un recurso drawable directamente al ImageView, pero el ejemplo es para que comprobéis cómo se puede obtener una referencia Drawable de un recurso /res/drawable.

El problema con el método anterior es que solo está permitido si tenemos una API MinSDK 21. De lo contrario, da un error.

- Para evitarlo, podemos hacer uso de la clase ContextCompact y del método getDrawable() da forma:

```
1 Drawable drawable = ContextCompat.getDrawable(getApplicationContext(),R.drawable.empresa);
2 ImageView img = new ImageView(this);
3 img.setImageDrawable(drawable);
```

1.4. Acceso a los recursos Assets

- La carpeta Assets es una carpeta donde podemos guardar recursos de todo tipo (gráficos, audios, bases de datos, ...) y estructurarlos como si se tratara de una carpeta de SO, añadiendo nuevas carpetas y en cada una de ellas guardando diferentes tipos de recursos.
- A diferencia de los recursos utilizados en la carpeta /res/, estos recursos no se compilan y no se puede hacer referencia a ellos mediante la clase R.

  Para hacer referencia a uno de estos recursos debemos llamar al método getAssets() que devuelve un objeto de la clase AssetManager que usamos para obtener el InputStream del recurso a cargar.

- En el caso de las imágenes, el código que nos permite convertir un InputStream en un objeto Drawable es el siguiente:

```
1 ImageView imageView = findViewById(R.id.imageView);    // Una referencia a un ImageView que se
encuentre en el Layout de la Activity
2
3 try {
4     InputStream is =  getAssets().open("nome_imagen.jpg");      // Ejemplo de recurso previa-
mente copiado a la carpeta Assets
5     Drawable d = Drawable.createFromStream(is,null);
6     imageView.setImageDrawable(d);
7 } catch (IOException e) {
8     e.printStackTrace();
9 }
```

- **Nota:** Si la foto está dentro de una carpeta Assets, debemos escribir la ruta relativa desde la raíz de Assets.

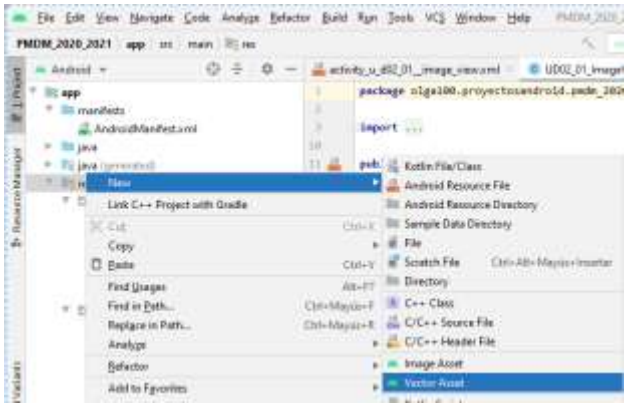
  Por ejemplo, si tenemos las fotos en una carpeta de fotos dentro de Assets, la ruta y el nombre a enviar serían: fotos/nombre_foto.jpg

  No debemos enviar / photos / photo name.jpg

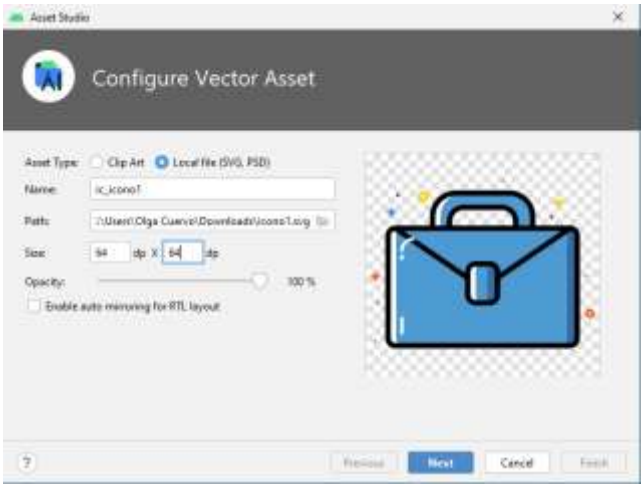
1.5. Usar gráficos SVG con ImageView

- Si queremos usar gráficos vectoriales podemos 'cargarlos' en nuestro proyecto con la herramienta Vector Asset:

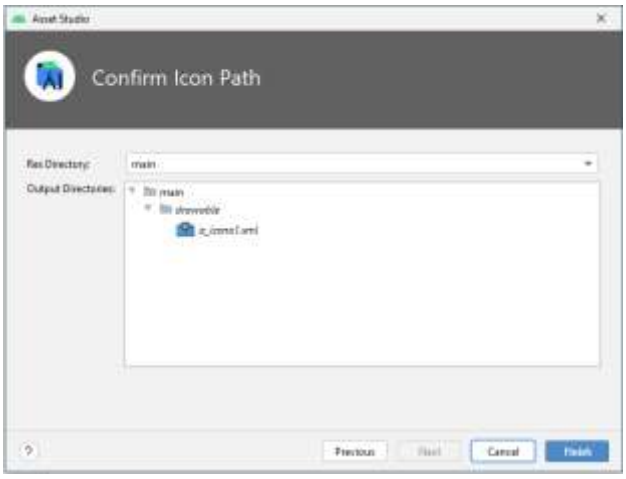
Empleando gráficos svg



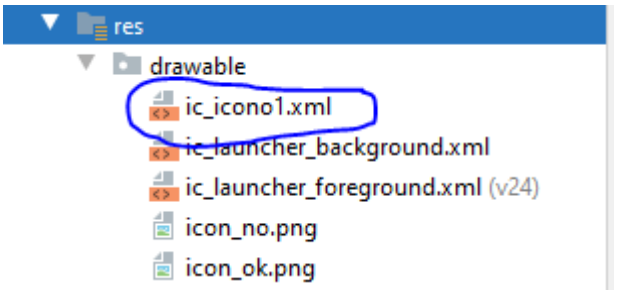
Lanzamos la herramienta **Vector Assets**



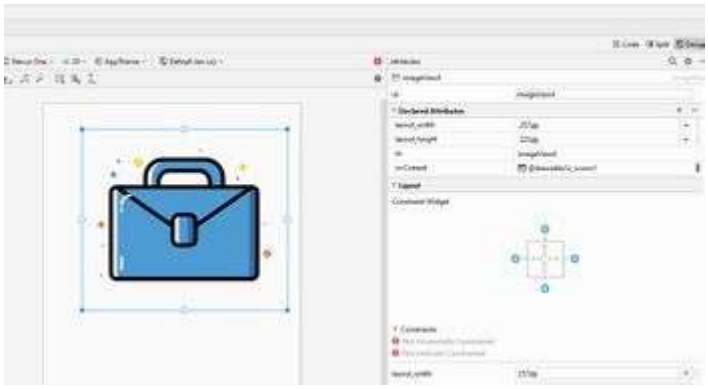
Escogemos el archivo svg descargado. Indicamos el tamaño del gráfico en dp's y el nombre del mismo.



Indicar dónde se va a guardar el gráfico y con qué archivo (fijarse que genera un archivo XML)



El archivo XML generado en la carpeta drawable



Una imagen con el icono asociado

1.6. Métodos mínimos a conocer en el manejo del ImageView

- Referenciar a un ImageView con el método findViewById.
- Modificar propiedades básicas, como ScaleType, color de fondo...
- Administrar el evento Click y saber cómo hacerlo utilizando interfaces anónimas o implementando la interfaz en la Activity.
- Gestionar el evento LongClick.
- Cambiar la imagen cargada usando una imagen Drawable o BitMap.
- Obtener una referencia Drawable a la imagen cargada.

```
ImageView imageView = findViewById(R.id.imageView);  // Referenciamos un ImageView que se encuentre en el Layout de la Activity
imageView.setScaleType(ImageView.ScaleType.FIT_XY);     // Cambiamos el tipo de escala.
imageView.setBackgroundColor(Color.BLUE);               // Cambiamos el color de fondo
imageView.setImageResource(R.drawable.ic_launcher_background);  // Una de las formas de cambiar la imagen es a partir de una imagen que se encuentre en Drawable
//imageView.setImageBitmap(obxecto BitMap);                    // Cambiamos la imagen por un bitmap
// Otra forma de obtener un Drawable y asignarlo a un ImageView
Drawable imagen = getResources().getDrawable(R.drawable.ic_launcher_background);
imageView.setImageDrawable(imagen);
Drawable imagenCargada = imageView.getDrawable();    // Ahora podríamos emplear el Drawable para cargar otro Imageview con esta imagen, por ejemplo.
```