INDICE

1.1 Introducción

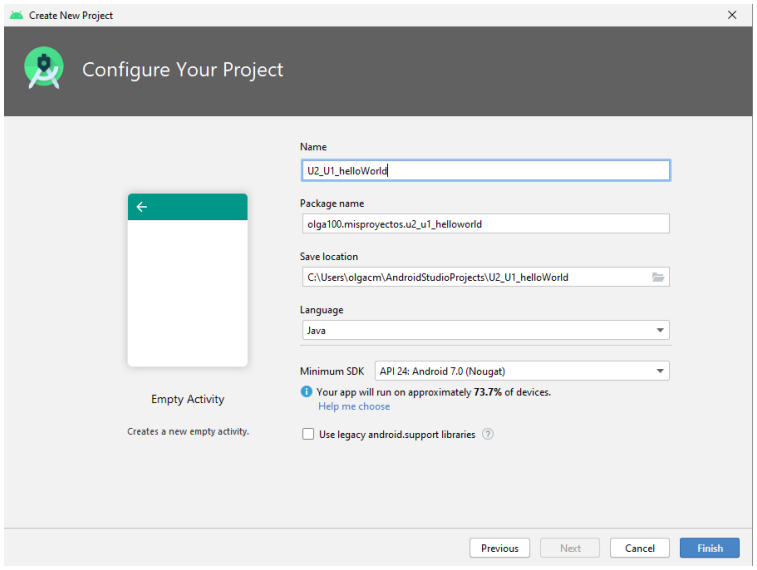
1 En esta sección se describirán las carpetas y archivos más esenciales que componen un proyecto de Android.

2 Empezaremos creando un proyecto: **Hello World**, y estudiaremos las partes en las que se divide.

3 Los proyectos se nombrarán y numerarán de la siguiente manera: **U2_XX_ApplicationName**, para facilitar el seguimiento. U2 es la unidad 2 y XX el número de proyecto dentro de la unidad.

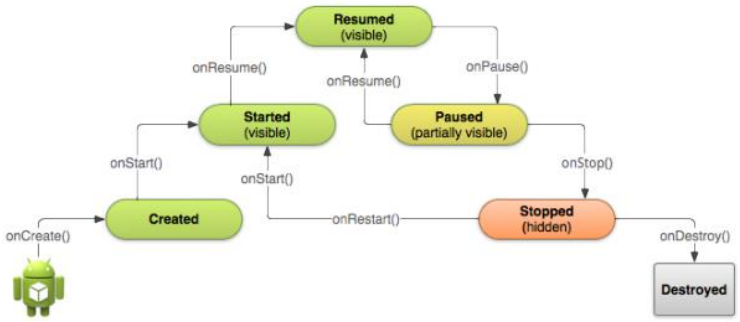
1.2 Crear el proyecto: Hello World

En este punto no me voy a detener demasiado, porque ya sabemos crear un proyecto a partir de la plantilla Empty Activity. Luego vamos a crear la actividad Hello World!!!.



1.3 Activities

- En el anexo de la unidad anterior ya explicamos qué era una **Activity**. Más adelante las veremos con más detalle.
- Una **Activity** es una pantalla única con la que el usuario puede interactuar a través de los elementos disponibles en ella.
- Una **aplicación** constará de una o más actividades.
- Incluso se puede pasar de la **Activity** de una aplicación a cualquier **Activity** de otra aplicación. Por ejemplo, al usar WhatsApp y acceder a los detalles de un contacto.
- Se puede llamar a una **Activity** cuando se inicia una aplicación, desde otra aplicación o cuando se recupera la aplicación.
- Una **Activity** puede ser destruida por la propia aplicación, cuando se presiona el botón **Back** del dispositivo o por el sistema porque está en la pila de aplicaciones en segundo plano y se necesitan los recursos que se están consumiendo.
- Las **Activities** tienen un ciclo de vida como se muestra de forma sencilla en la siguiente imagen:

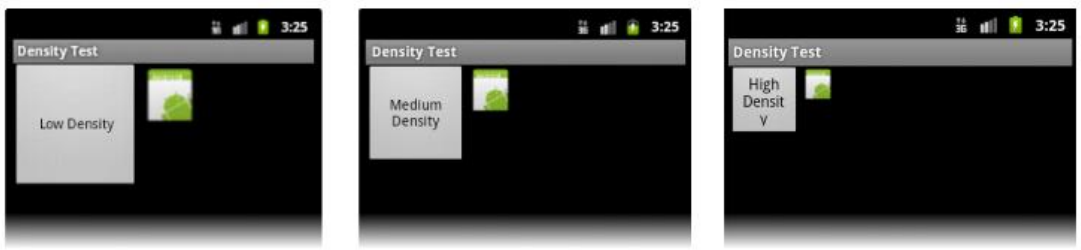


Referencias:

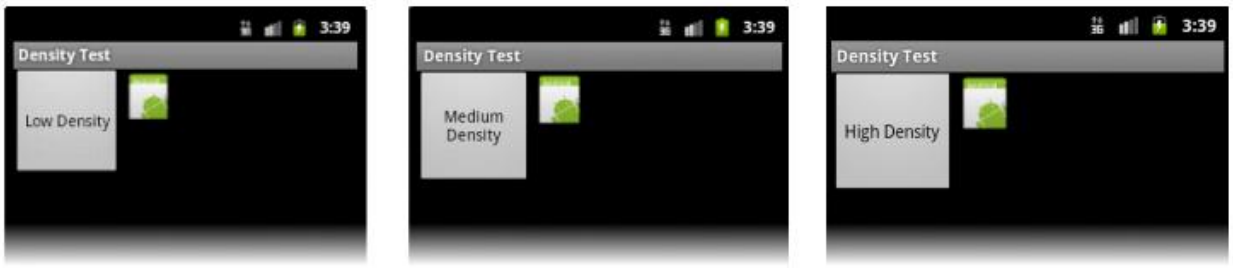
- http://developer.android.com/reference/android/app/Activity.html
- http://developer.android.com/guide/components/activities.html
- http://developer.android.com/training/basics/activity-lifecycle/index.html

1.4 Soporte múltiples pantallas.

- Como sabemos en el mercado existen diferentes dispositivos: teléfonos, tabletas, etc. y ahora ... con las nuevas versiones también relojes, pulseras, televisores, etc.
- Cada uno tiene sus propios tamaños y resoluciones.
- Android proporciona herramientas para adaptarse a toda esa variedad. Por ejemplo, podríamos diseñar una aplicación cuya interfaz gráfica fuera diferente de un teléfono a una tableta, o si el teléfono esta en horizontal o vertical.
- El sistema también intenta escalar la aplicación para que se adapte a diferentes tipos (tamaños de pantalla / resoluciones), pero si usamos píxeles para definir los elementos que componen una pantalla puede suceder lo siguiente:



- Los dispositivos tienen el mismo tamaño de pantalla.
- La resolución del dispositivo varía de menor a mayor (de izquierda a derecha).
- Observar cómo al configurar un objeto para que ocupe X * Y **píxeles**, estos objetos son más pequeños a medida que la resolución de la pantalla es mayor.
- Tenga en cuenta que los tipos de recursos gráficos se ven afectados:
    - Los creados por el usuario en la aplicación: por ejemplo el botón.
    - Los añadidos por el usuario a la aplicación (dibujos / imágenes): por ejemplo la imagen de Android.
- El objetivo sería que, independientemente de la resolución de la pantalla, se obtuviera el siguiente resultado: se ven objetos del mismo tamaño incluso con diferentes resoluciones.



- Vamos a ver antes de nada una serie de conceptos:

Tamaños de pantalla y densidad (puntos por pulgada: dpi)

- **Tamaño de pantalla:** el tamaño diagonal de la pantalla física del dispositivo.
    - Para simplificar, Android divide las pantallas, hasta la versión 3.2 de Android, en: **pequeñas (small), normales**, **grandes (large)** y **extras grandes (extra large)**.
- **Densidad de pantalla (Screen density):** puntos por pulgada (**ppp / (dot per inch (dpi))**). Una pantalla de baja resolución tendrá pocos dpis en comparación con una pantalla normal o de alta resolución.
- Por simplicidad, Android agrupó las densidades:
    - **ldpi**: bajo (low 120 ppp)
    - **mdpi**: medio (medium, 160 ppp)
    - **hdpi**: alta (hig, 240 ppp)
    - **xhdpi**: extra alto (extra high, 320 ppp)
    - **xxhdpi**: extra extra alto (extra-extra high, 480 ppp)
    - **xxxhdpi**: extra extra extra alto (extra extra-extra-extra high, 640 ppp)

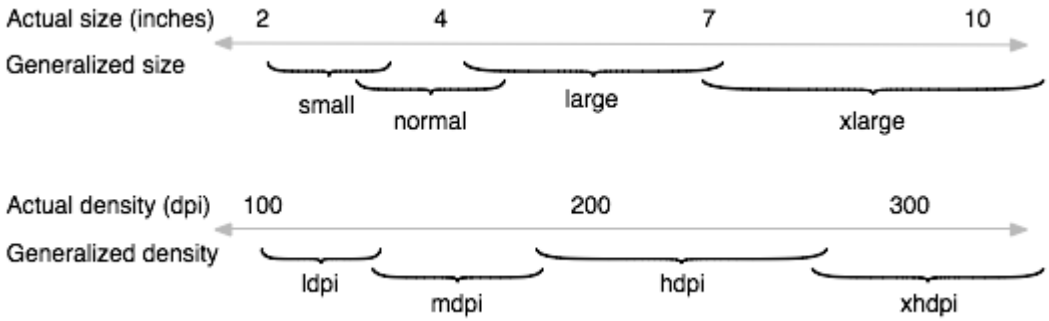
Orientación y resolución

- **Orientación**: Desde el punto de vista del usuario, indica si el dispositivo es horizontal o vertical. Al girar la pantalla, cambia la relación de altura y ancho.
- **Resolución**: indica el número total de píxeles en la pantalla. Al desarrollar aplicaciones para diferentes pantallas, se debe trabajar con la densidad y el tamaño de la pantalla y no con la resolución total.

Píxeles independientes de la densidad (dp)

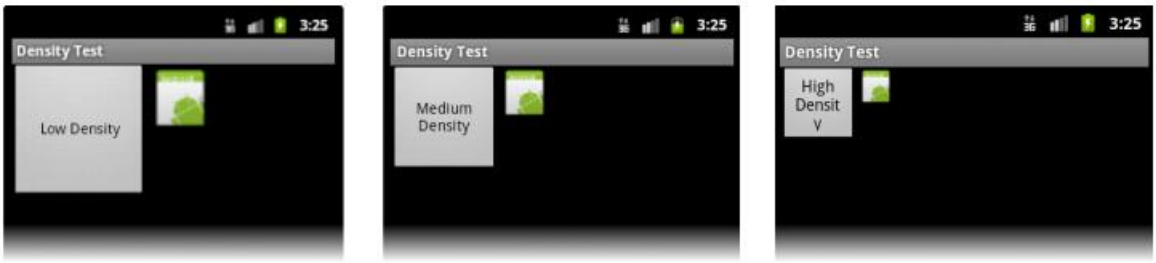
- **Píxel independiente de la densidad (Density-indepent pixel (dp)):** es un píxel virtual que se debe utilizar al definir elementos gráficos en un diseño (layout), tanto para expresar el tamaño de un objeto como su posición.
- Una pantalla mediana (medium) tiene 160 ppp.
- En una pantalla media 1 dp = 1 dpi. Es decir, en una pantalla media, el píxel virtual coincide con el píxel real.
- El tiempo de ejecución es cuando se convierten unidades dp en píxeles físicos (px).
- El factor de conversión es: **px = dp (dpi / 160)**.
- Por ejemplo, 1 dp en una pantalla de densidad:
    - 120 ppp: sería igual a 0,75 px (píxeles físicos).
    - 160 ppp: sería igual a 1 px (píxeles físicos).
    - 240 ppp: equivaldría a 1,5 px (píxeles físicos).
    - 320 ppp: sería igual a 2 px (píxeles físicos).
    - 480 ppp: sería igual a 3 px (píxeles físicos).
    - 640 ppp: sería igual a 4 px (píxeles físicos).
- De esta manera, si siempre se definen las dimensiones y posiciones en dp, siempre realizará una conversión en tiempo de ejecución al número equivalente de píxeles físicos.
- **Scale-indepent Píxels (sp)**: esta unidad es igual a dp, pero se utiliza para tamaños de texto. Estos tamaños se pueden ajustar según la densidad de la pantalla y las preferencias del usuario.

La siguiente imagen muestra los rangos entre los que se mueven los distintos tamaños y los distintos tipos de densidad



1.5 Las imágenes

- Pero aún necesitamos saber qué hacer con las **imágenes que un usuario puede incrustar en una aplicación**.
  - Se puede proporcionar un archivo diferente adaptado a las diferentes densidades para cada imagen.
  - Si no se proporcionan archivos para las diferentes densidades, el sistema escalará el archivo proporcionado para la densidad media (mdpi) y lo adaptará a la densidad de la pantalla. Pero siempre será mejor proporcionar diferentes versiones de la imagen.
  - Para proporcionar versiones de diferente densidad de una imagen debemos seguir la regla: 3 (ldpi): 4 (mdpi): 6x (hdpi): 8 (xhdpi). Es decir, si tomamos como referencia una imagen de densidad media (mdpi) de tamaño X * Y píxeles, debemos crear nuevas imágenes que mantengan la siguiente relación en tamaño (número de píxeles): ldpi (x0,75), mdpi (x1), hdpi (x1,5) y xhdpi (x2).
  - Por ejemplo, si tenemos una imagen de 48x48 píxeles (tamaño del icono de inicio de una aplicación) y esto es para una densidad mdpi, deberíamos crear versiones con los siguientes tamaños en píxeles:
    - ldpi: 36 x 36
    - mdpi: 48 x 48
    - hdpi: 72 x 72
    - xhdpi: 96 x 96
- Visto todo o anterior ahora volvamos a ver de nuevo las imágenes donde se usan los píxeles para los tamaños y las posiciones:



- Es la que usa **dp** y diferentes versiones de la misma imagen:

- Observar como coinciden para los dos casos en la densidad media.
- Referencias:
  - http://developer.android.com/guide/practices/screens_support.html
  - http://developer.android.com/design/style/iconography.html
  - Para as dimensións: http://developer.android.com/guide/topics/resources/more-resources.html#Dimension

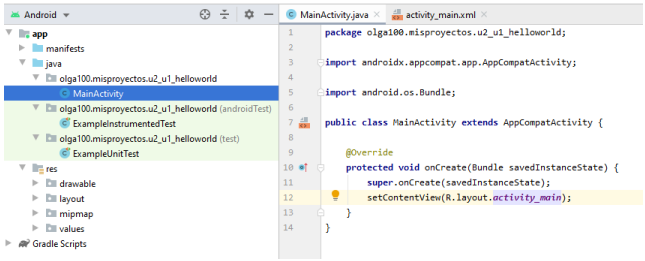
1.6 Ficheros y carpetas de un proyecto Android

Lo primero que debemos conocer es que un proyecto en Android Studio puede contener varios módulos. Cada módulo corresponde a una aplicación o ejecutable diferente. Disponer de varios módulos en un mismo proyecto nos será muy útil cuando queramos crear varias versiones de nuestra aplicación, para dispositivo móvil, Wear, TV, Things, etc. También si queremos crear varias versiones de **nuestra** aplicación con nivel mínimo de SDK diferentes.

En este tema solo vamos a desarrollar aplicaciones para móviles, por lo que no vamos a necesitar un módulo. Este módulo ha sido creado con el nombre ***app***.

Cada módulo en Android está formado por un descriptor de la aplicación (***manifests***), el código fuente en Java (***java***), una serie de ficheros con recursos (***res***) y ficheros para construir el módulo (***Gradle Scripts***). Cada elemento se almacena en una carpeta específica, indicada entre paréntesis. Aprovecharemos el proyecto que acabamos de crear para estudiar la estructura de un proyecto en Android Studio. No debemos asustarnos con el exceso de información. Más adelante se dará más detalles sobre la finalidad de cada fichero.

/src



Carpeta **src:** contiene el código fuente de la aplicación, la interfaz gráfica, clases auxiliares. El código fuente asociado con la actividad principal (Actividad / Pantalla) de la aplicación se encuentra dentro del paquete Java. Este código inicia la pantalla principal (actividad) y el menú.

```
public class MainActivity extends AppCompatActivity {

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

La clase **U2_01_MainActivity** hereda de la clase **AppCompatActivity** sus atributos y métodos. En este caso, se sobrescribe el método **onCreate**, que es el primero en ejecutarse cuando se lanza una actividad (ver imagen superior de los estados de una actividad). Este método recibe un parámetro **Bundle**, en el que recibe el estado de la actividad:

(**-Null**: porque acabamos de empezar

-o **valores del estado** anterior si acaba de detenerse. Mirar la imagen superior del ciclo de vida).

Llama al método **onCreate** de la clase principal.

A continuación se muestra en pantalla el contenido del layout: **activity_main.** Para ello utilizar la clase R como veremos a continuación.

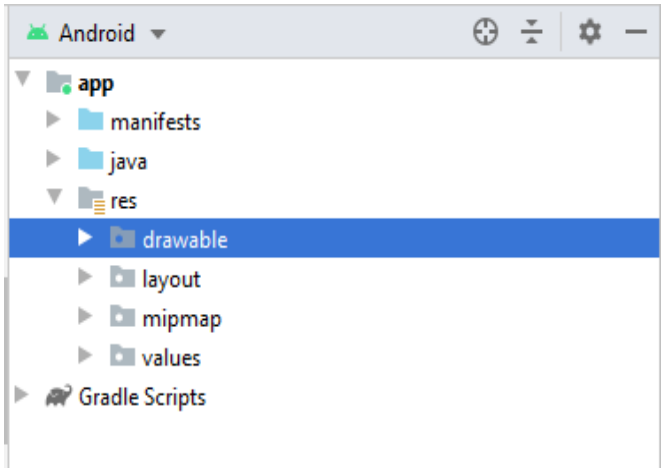
https://developer.android.com/reference/android/os/Bundle.html

https://developer.android.com/reference/android/app/Activity.html#setContentView(int)

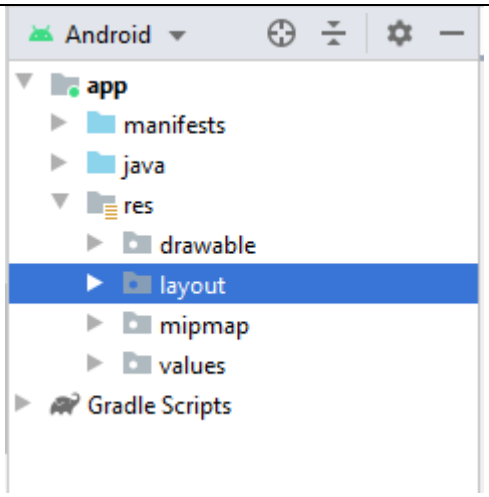
/res

Carpeta res (Recursos/Resources): almacena todos los archivos de recursos necesarios para la aplicación: imágenes, videos, ficheros xml, animaciones, ficheros para internacionalización, etc.

/res



***/res/drawable*:** En esta carpeta se almacenan los ficheros de imágenes (JPG o PNG) y descriptores de imágenes en XML.

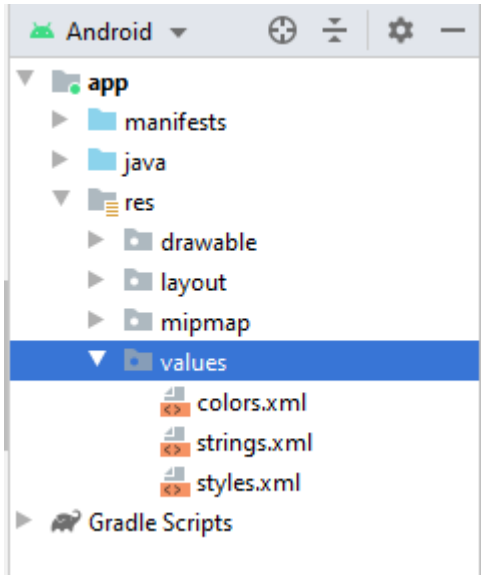


***layout*:** Contiene ficheros XML con vistas de la aplicación. Las vistas nos permitirán configurar las diferentes pantallas que compondrán la interfaz de usuario de la aplicación. Se utiliza un formato similar al HTML usado para diseñar páginas web.

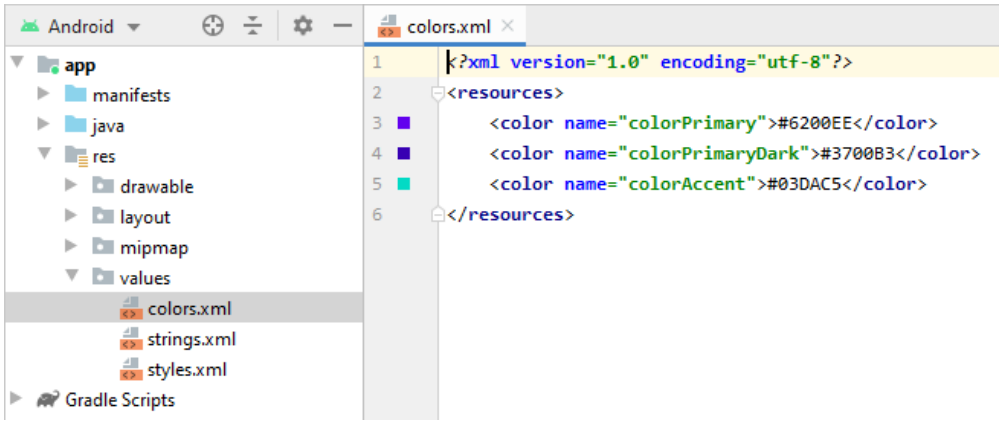
***menu*:** Ficheros XML con los menús de cada actividad. En el proyecto no hay ningún menú por lo que no se muestra esta carpeta.

```xml
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/android"
    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=".MainActivity">

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Hello World!"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintRight_toRightOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

</androidx.constraintlayout.widget.ConstraintLayout>
```

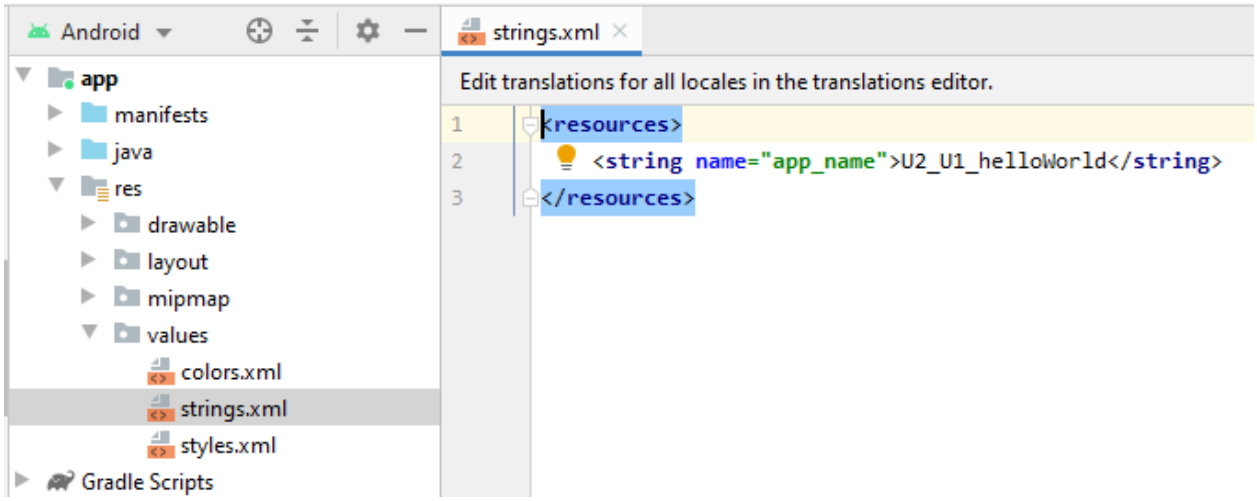
Este es el contenido del archivo XML de diseño.



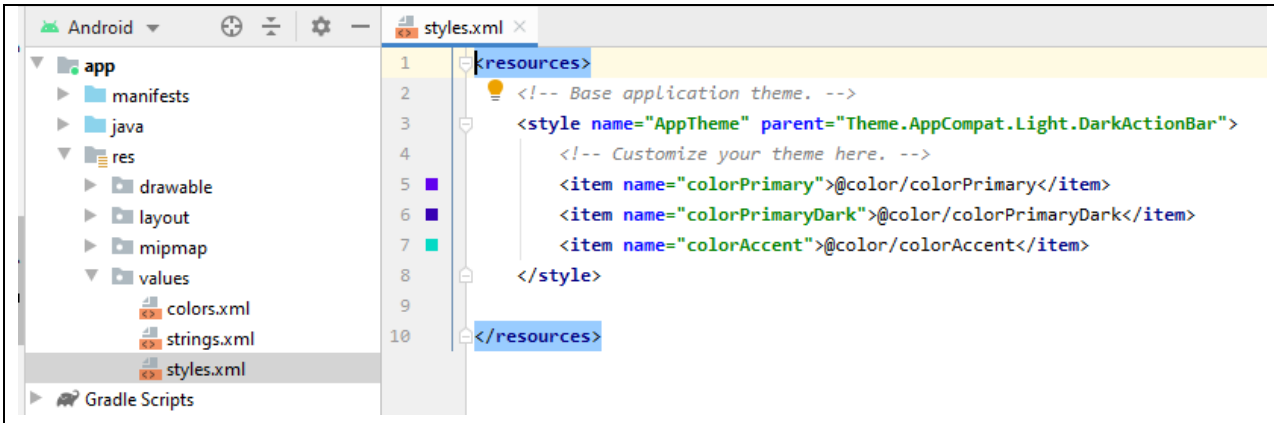
/res/values: También utilizaremos ficheros XML para indicar valores usados en la aplicación, de esta manera podremos cambiarlos desde estos ficheros sin necesidad de ir al código fuente.



En ***colors.xml*** se definen los tres colores primarios de la aplicación.

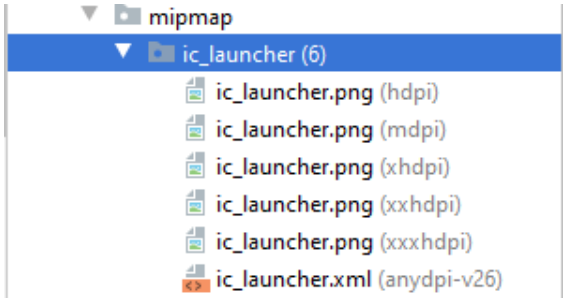


En el fichero ***strings.xml***, tendrás que definir todas las cadenas de caracteres de tu aplicación. Creando recursos alternativos resultará muy sencillo traducir una aplicación a otro idioma.

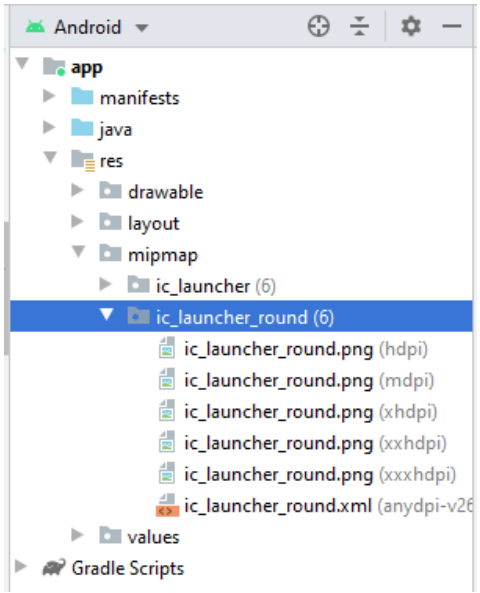


Finalmente en ***styles.xml***, podrás definir los estilos y temas de tu aplicación.

En ***dimens.xml***, se pueden definir dimensiones como el margen por defecto o el ancho de los botones. En este ejemplo no existe el fichero.



/res/mipmap: En una carpeta guardaremos el icono de la aplicación. En el proyecto se ha incluido el fichero ***ic_launcher.png*** que será utilizado como icono de la aplicación. Observa cómo este recurso se ha añadido en seis versiones diferentes. Usaremos un sufijo especial si queremos tener varias versiones de un recurso, de forma que solo se cargue al cumplirse una determinada condición. Por ejemplo: (*hdpi*) significa que solo ha de cargar los recursos contenidos en esta carpeta cuando el dispositivo donde se instala la aplicación tenga una densidad gráfica alta (180- dpi); (*mdpi*) se utilizará con densidad gráfica alta (180- dpi). 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.



El fichero ***ic_launcher_round.png*** es similar pero para cuando se quieren usar iconos redondos.

Otros ficheros que pueden aparecer en la carpeta **/res** son:

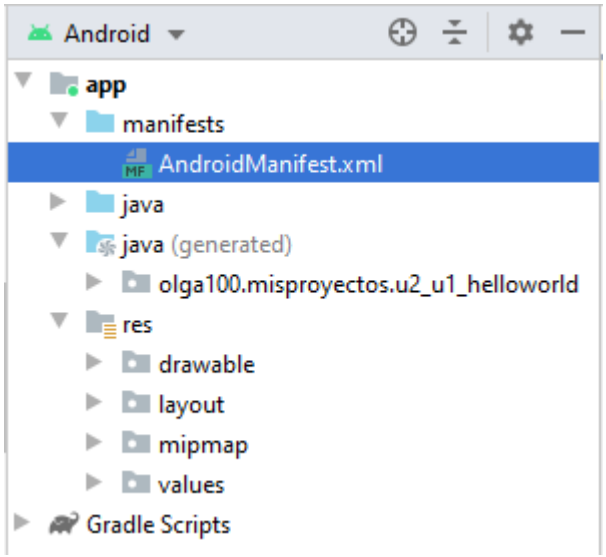
anim: Contiene ficheros XML con animaciones de vistas (Tween). Las animaciones se describen al final del capítulo 4.

animator: Contiene ficheros XML con animaciones de propiedades.

xml: Otros ficheros XML requeridos por la aplicación.

raw: Ficheros adicionales que no se encuentran en formato XML.

Gradle Scripts: En esta carpeta se almacenan una serie de ficheros *Gradle* que permiten compilar y construir la aplicación. Observar como algunos hacen referencia al módulo app y el resto son para configurar todo el proyecto. El fichero más importante es *build.gradle (Module:app)* que es donde se configuran las opciones de compilación del módulo:



***AndroidManifest.xml*:** Este fichero describe la aplicación Android. Se define su nombre, paquete, icono, estilos, etc. Se indican las *actividades*, las *intenciones*, los *servicios* y los *proveedores de contenido* de la aplicación. También se declaran los permisos que requerirá la aplicación. Se indica el paquete Java, la versión de la aplicación, etc.

Para finalizar este primer tema de unidad de trabajo 2, recomiendo se visualice el videotutorial (Estructura de un proyecto Android) indicado en las referencias. Ayudará a entender mejor la estructura de un proyecto Android.

Referencias:

VideoTutorial: Estructura de un proyecto Android