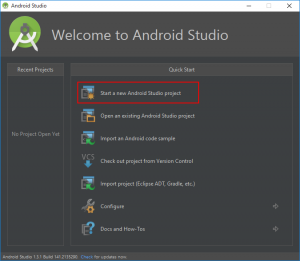
Para empezar a comprender cómo se construye una aplicación Android vamos a crear un nuevo proyecto en Android Studio y echaremos un vistazo a la estructura general del proyecto creado por defecto.

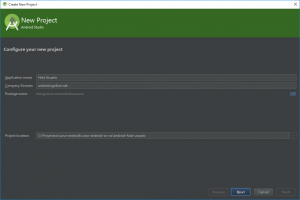
Para crear un nuevo proyecto ejecutaremos Android Studio y desde la pantalla de bienvenida pulsaremos la opción “Start a new Android Studio project” para iniciar el asistente de creación de un nuevo proyecto.



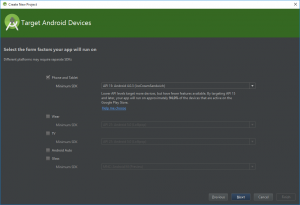
Si ya habíamos abierto anteriormente Android Studio es posible que se abra directamente la aplicación principal en vez de la pantalla de bienvenida. En ese caso accederemos al menú “File / New project…” para crear el nuevo proyecto.

El asistente de creación del proyecto nos guiará por las distintas opciones de creación y configuración de un nuevo proyecto Android.

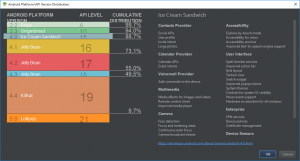
En la primera pantalla indicaremos, por este orden, el nombre de la aplicación, el dominio de la compañía, y la ruta donde crear el proyecto. El segundo de los datos indicados tan sólo se utilizará como paquete de nuestras clases java. Así, si por ejemplo indicamos como en mi caso *com.digitalhouse.android.helloworld*, el paquete java principal utilizado para mis clases *com.digitalhouse.android.helloworld*. En tu caso puedes utilizar cualquier otro dominio.



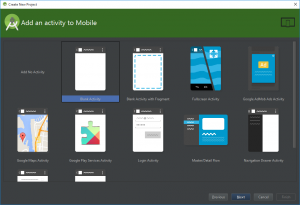
En la siguiente pantalla del asistente configuraremos las plataformas y APIs que va a utilizar nuestra aplicación. Nosotros nos centraremos en aplicaciones para teléfonos y tablets, en cuyo caso tan sólo tendremos que seleccionar la API mínima (es decir, la versión mínima de Android) que soportará la aplicación. API 16



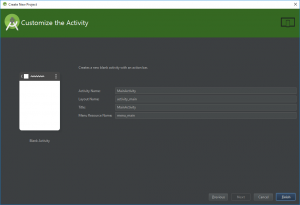
La versión mínima que seleccionemos en esta pantalla implicará que nuestra aplicación se pueda ejecutar en más o menos dispositivos. De esta forma, cuanto menor sea ésta, a más dispositivos podrá llegar nuestra aplicación, pero más complicado será conseguir que se ejecute correctamente en todas las versiones de Android. Para hacernos una idea del número de dispositivos que cubrimos con cada versión podemos pulsar sobre el enlace “Help me choose”, que mostrará el porcentaje de dispositivos que ejecutan actualmente cada versión de Android. Por ejemplo, en el momento de escribir este artículo, si seleccionamos como API mínima la 15 conseguiríamos cubrir un 94,0% de los dispositivos actuales. Como información adicional, si pulsamos sobre cada versión de Android en esta pantalla podremos ver una lista de las novedades introducidas por dicha versión.



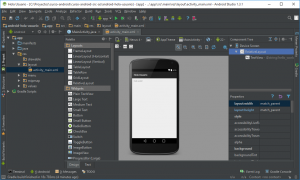
En la siguiente pantalla del asistente elegiremos el tipo de *actividad* principal de la aplicación. Entenderemos por ahora que una *actividad* es una “ventana” o “pantalla” de la aplicación. Para empezar seleccionaremos *BlankActivity*, que es el tipo más sencillo.



Por último, en el siguiente paso del asistente indicaremos los datos asociados a esta actividad principal que acabamos de elegir, indicando el nombre de su clase java asociada (*Activity Name*) y el nombre de su *layout xml* (algo así como la interfaz gráfica de la actividad, lo veremos más adelante), su título, y el nombre del recurso XML correspondiente a su menú principal. No nos preocuparemos mucho por ahora de todos estos datos por lo que podemos dejar todos los valores por defecto. Más adelante en el curso explicaremos cómo y para qué utilizar estos elementos.

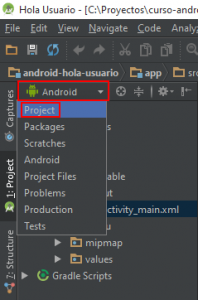


Una vez configurado todo pulsamos el botón *Finish* y Android Studio creará por nosotros toda la estructura del proyecto y los elementos indispensables que debe contener. Si todo va bien aparecerá la pantalla principal de Android Studio con el nuevo proyecto creado.

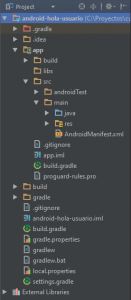


En ocasiones Android Studio no realiza correctamente esta primera carga del proyecto y es posible que nos encontremos con un error del tipo “*Rendering Problems…*“. Para solucionarlo no tenés más que cerrar la ventana del editor gráfico y volverla a abrir pulsando sobre el fichero “*activity\_main.xml*” (se encuentra en la parte izquierda del explorador).

En la parte izquierda, podemos observar todos los elementos creados inicialmente para el nuevo proyecto Android, sin embargo por defecto los vemos de una forma un tanto peculiar que podría llevarnos a confusión. Para entender mejor la estructura del proyecto vamos a cambiar momentáneamente la forma en la que Android Studio nos la muestra. Para ello, pulsaremos sobre la lista desplegable situada en la parte superior izquierda, y cambiaremos la vista de proyecto al modo “Project”.

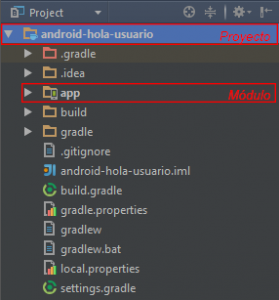


Tras hacer esto, la estructura del proyecto cambia un poco de aspecto y pasa a ser como se observa en la siguiente imagen:



En los siguientes apartados describiremos los elementos principales de esta estructura.

Lo primero que debemos distinguir son los conceptos de *proyecto* y *módulo*. La entidad *proyecto* es única, y engloba a todos los demás elementos. Dentro de un proyecto podemos incluir varios *módulos*, que pueden representar aplicaciones distintas, versiones diferentes de una misma aplicación, o distintos componentes de un sistema (aplicación móvil, aplicación servidor, librerías, …). En la mayoría de los casos, trabajaremos con un proyecto que contendrá un sólo módulo correspondiente a nuestra aplicación principal. Por ejemplo en este caso que estamos creando tenemos el proyecto “android-hola-usuario” que contiene al módulo “app” que contendrá todo el software de la aplicación de ejemplo.



A continuación describiremos los contenidos principales de nuestro módulo principal.

Carpeta /app/src/main/java

Esta carpeta contendrá todo el código fuente de la aplicación, clases auxiliares, etc. Inicialmente, Android Studio creará por nosotros el código básico de la pantalla (*actividad* o *activity*) principal de la aplicación, que recordemos que en nuestro caso era MainActivity, y siempre bajo la estructura del paquete java definido durante la creación del proyecto.

carpeta-java

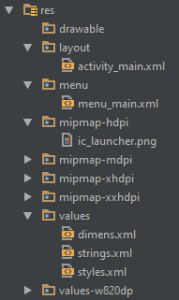
Carpeta /app/src/main/res/

Contiene todos los ficheros de recursos necesarios para el proyecto: imágenes, *layouts*, cadenas de texto, etc. Los diferentes tipos de recursos se pueden distribuir entre las siguientes subcarpetas:

|  |  |
| --- | --- |
| Carpeta | Descripción |
| /res/drawable/ | Contiene las imágenes y otros elementos gráficos usados por la aplicación. Para poder definir diferentes recursos dependiendo de la resolución y densidad de la pantalla del dispositivo se suele dividir en varias subcarpetas:   * /drawable (recursos independientes de la densidad) * /drawable-ldpi (densidad baja) * /drawable-mdpi (densidad media) * /drawable-hdpi (densidad alta) * /drawable-xhdpi (densidad muy alta) * /drawable-xxhdpi (densidad muy muy alta) |
| /res/mipmap/ | Contiene los iconos de lanzamiento de la aplicación (el icono que aparecerá en el menú de aplicaciones del dispositivo) para las distintas densidades de pantalla existentes. Al igual que en el caso de las carpetas /drawable, se dividirá en varias subcarpetas dependiendo de la densidad de pantalla:   * /mipmap-mdpi * /mipmap-hdpi * /mipmap-xhdpi * … |
| /res/layout/ | Contiene los ficheros de definición XML de las diferentes pantallas de la interfaz gráfica. Para definir distintos *layouts* dependiendo de la orientación del dispositivo se puede dividir también en subcarpetas:   * /layout (vertical) * /layout-land (horizontal) |
| /res/anim/  /res/animator/ | Contienen la definición de las animaciones utilizadas por la aplicación. |
| /res/color/ | Contiene ficheros XML de definición de listas de colores según estado. |
| /res/menu/ | Contiene la definición XML de los menús de la aplicación. |
| /res/xml/ | Contiene otros ficheros XML de datos utilizados por la aplicación. |
| /res/raw/ | Contiene recursos adicionales, normalmente en formato distinto a XML, que no se incluyan en el resto de carpetas de recursos. |
| /res/values/ | Contiene otros ficheros XML de recursos de la aplicación, como por ejemplo cadenas de texto (*strings.xml*), estilos (*styles.xml*), colores (*colors.xml*), arrays de valores (*arrays.xml*), tamaños (*dimens.xml*), etc. |

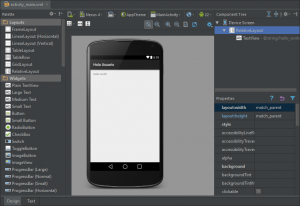
No todas estas carpetas tienen por qué aparecer en cada proyecto Android, tan sólo las que se necesiten. Iremos viendo durante el curso qué tipo de elementos se pueden incluir en cada una de ellas y cómo se utilizan.

Como ejemplo, para un proyecto nuevo Android como el que hemos creado, tendremos por defecto los siguientes recursos para la aplicación:

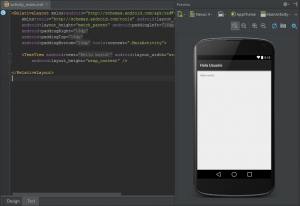


Como se puede observar, existen algunas carpetas en cuyo nombre se incluye un sufijo adicional, como por ejemplo “values-w820dp”. Estos, y otros sufijos, se emplean para definir recursos independientes para determinados dispositivos según sus características. De esta forma, por ejemplo, los recursos incluidos en la carpeta “values-w820dp” se aplicarían sólo a pantallas con más de 820dp de ancho, o los incluidos en una carpeta llamada “values-v11” se aplicarían tan sólo a dispositivos cuya versión de Android sea la 3.0 (API 11) o superior. Al igual que estos sufijos “-w” y “–v” existen otros muchos para referirse a otras características del terminal, puede consultarse la lista completa en la [documentación oficial del Android](http://developer.android.com/guide/topics/resources/providing-resources.html).

Entre los recursos creados por defecto cabe destacar los *layouts*, en nuestro caso sólo tendremos por ahora el llamado “*activity\_main.xml*”, que contienen la definición de la interfaz gráfica de la pantalla principal de la aplicación. Si hacemos doble click sobre este fichero Android Studio nos mostrará esta interfaz en su editor gráfico, y como podremos comprobar, en principio contiene tan sólo una etiqueta de texto con el mensaje “Hello World!”.



Pulsando sobre las pestañas inferiores “Design” y “Text” podremos alternar entre el editor gráfico (tipo arrastrar-y-soltar), mostrado en la imagen anterior, y el editor XML que se muestra en la imagen siguiente:



Durante el curso no utilizaremos demasiado el editor gráfico, sino que modificaremos la interfaz de nuestras pantallas manipulando directamente su fichero XML asociado. Esto en principio puede parecer mucho más complicado que utilizar el editor gráfico [no es nada complicado en realidad], pero nos permitirá aprender muchas de las complicaciones de Android más rápidamente.

Fichero /app/src/main/AndroidManifest.xml

Contiene la definición en XML de muchos de los aspectos principales de la aplicación, como por ejemplo su identificación (nombre, icono, …), sus componentes (pantallas, servicios, …), o los permisos necesarios para su ejecución. Veremos más adelante más detalles de este fichero.

Fichero /app/build.gradle

Contiene información necesaria para la compilación del proyecto, por ejemplo la versión del SDK de Android utilizada para compilar, la mínima versión de Android que soportará la aplicación, referencias a las librerías externas utilizadas, etc. Más adelante veremos también más detalles de este fichero.

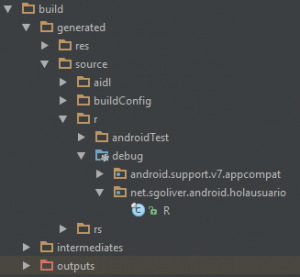
En un proyecto pueden existir varios ficheros *build.gradle*, para definir determinados parámetros a distintos niveles. Por ejemplo, en nuestro proyecto podemos ver que existe un fichero*build.gradle* a nivel de proyecto, y otro a nivel de módulo dentro de la carpeta /app. El primero de ellos definirá parámetros globales a todos los módulos del proyecto, y el segundo sólo tendrá efecto para cada módulo en particular.

Carpeta /app/libs

Puede contener las librerías java externas (ficheros *.jar*) que utilice nuestra aplicación. Normalmente no incluiremos directamente aquí ninguna librería, sino que haremos referencia a ellas en el fichero *build.gradle* descrito en el punto anterior, de forma que entren en el proceso de compilación de nuestra aplicación. Veremos algún ejemplo más adelante.

Carpeta /app/build/

Contiene una serie de elementos de código generados automáticamente al compilar el proyecto. Cada vez que compilamos nuestro proyecto, la maquinaria de compilación de Android genera por nosotros una serie de ficheros fuente java dirigidos, entre otras muchas cosas, al control de los recursos de la aplicación. Importante: dado que estos ficheros se generan automáticamente tras cada compilación del proyecto es importante que no se modifiquen manualmente bajo ninguna circunstancia.



A destacar sobre todo el fichero que aparece desplegado en la imagen anterior, llamado “R.java”, donde se define la clase R. Esta clase R contendrá en todo momento una serie de constantes con los identificadores (ID) de todos los recursos de la aplicación incluidos en la carpeta /app/src/main/res/, de forma que podamos acceder fácilmente a estos recursos desde nuestro código java a través de dicho dato. Así, por ejemplo, la constante R.layout.activity\_main contendrá el ID del layout “*activity\_main.xml*” contenido en la carpeta /app/src/main/res/layout/.