

[**Duración**](http://hackspace.la/android.html)

4 Semanas

[**Entrega Final**](http://hackspace.la/android.html)

14 Feb 2016

**Sobre los Entrenadores**

* 

*Brian Castillo*

Brian es un Desarrollador Android con varios años de experiencia Java y otros lengujaes de programacion.

* 

*Victor Casas*

Developer y geek, ha desarrollado videojuegos para Inmotion Chile y la Fundacion Peruana de Cáncer; dos aňos de experiencia haciendo android, actualmente da clases en una ONG a jovenes de bajos recursos y por las noches trata de dominar el mundo.

**Programas Necesarios**

[Java JDK 8](http://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html), [Android Studio](http://developer.android.com/intl/es/sdk/index.html" \t "_blank), [Genymotion](https://www.genymotion.com/" \l "!/download" \t "_blank)

**Comparte el Curso**

#### Semana 1 : Primeros pasos

### ****Teoría****Configurando el entorno.

### Instalando lo necesario

Antes de empezar debemos tener las herramientas necesarias para poder programar para Android

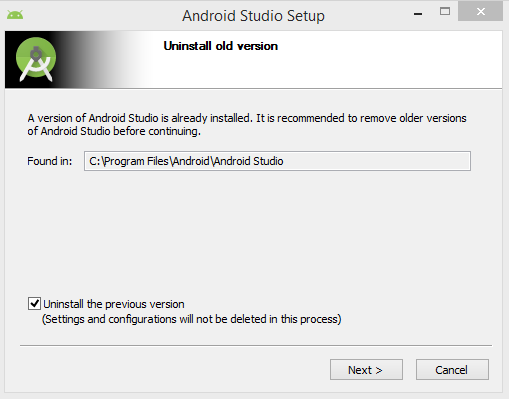
Primero debemos instalar el compilador de java y la maquina virtual:<http://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html>

1. Le damos aceptar.
2. Seleccionamos nuestro sistema operativo.
3. Descargamos e instalamos.
4. Next, next, next.



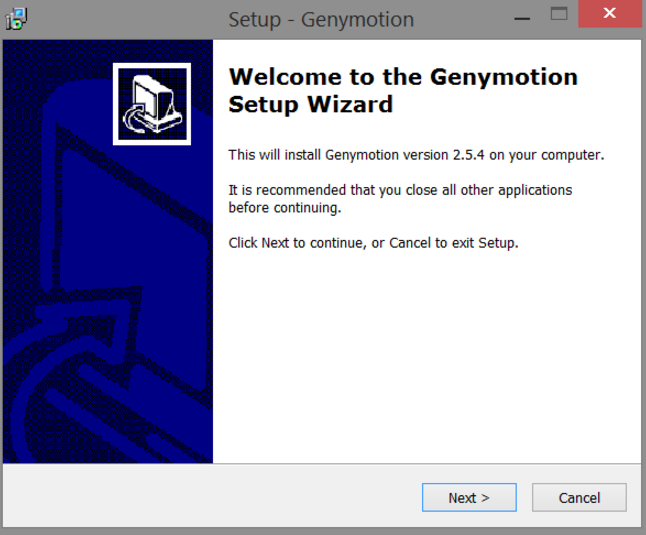
Segundo, descargamos el Android Studio: <http://developer.android.com/intl/es/sdk/index.html>

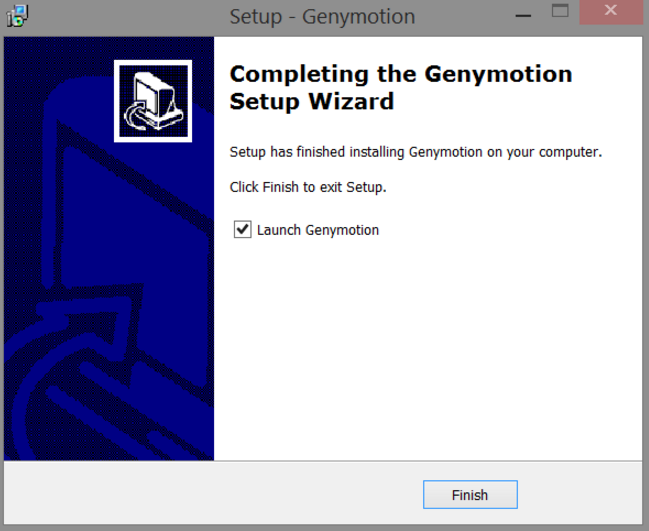
1. Descargamos e instalamos
2. Next, next, next

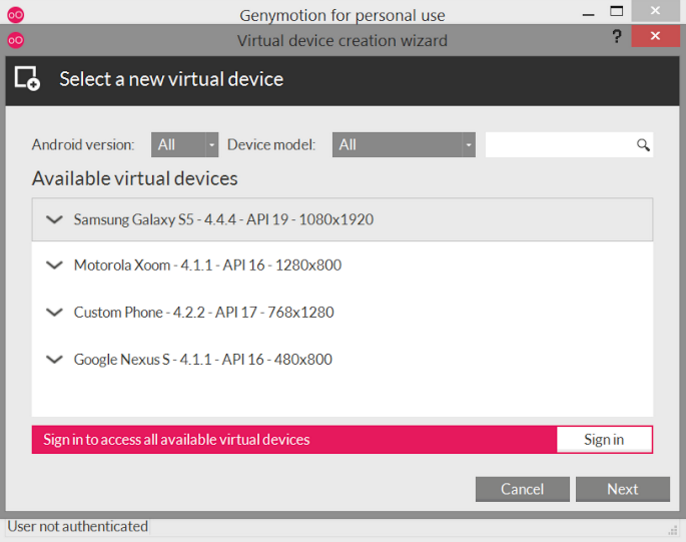


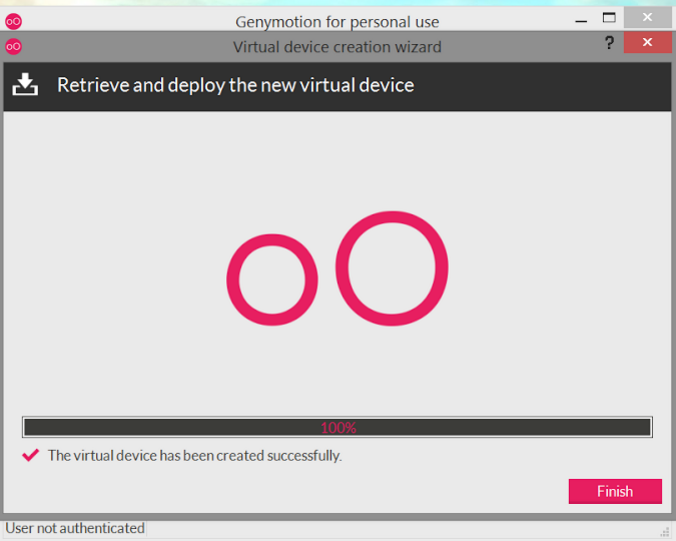
**Opcional**, si no tenemos un dispositivo movil para probar nuestra aplicación es recomendable instalar Genymotion <https://www.genymotion.com/#!/download>

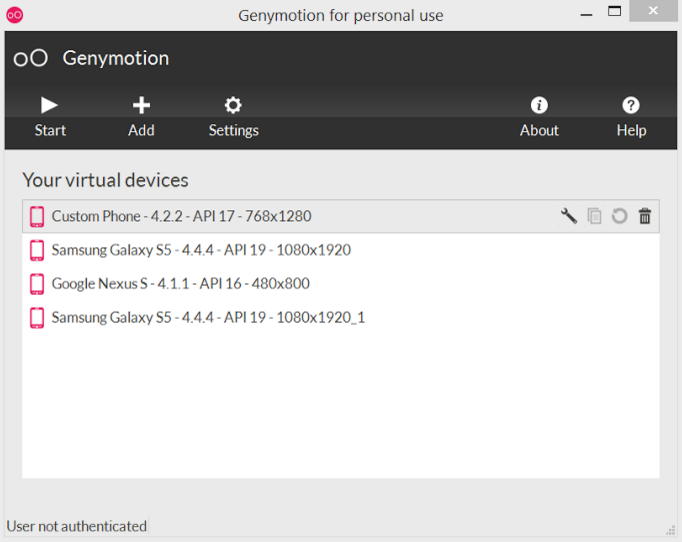
1. Creamos una cuenta
2. Seleccionamos nuestro sistema operativo.
3. Iniciamos sesión.
4. Descargamos el emulador que nos guste.

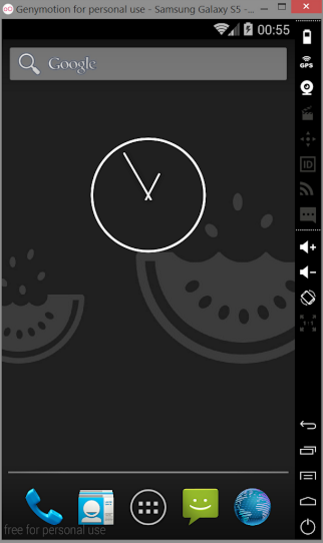












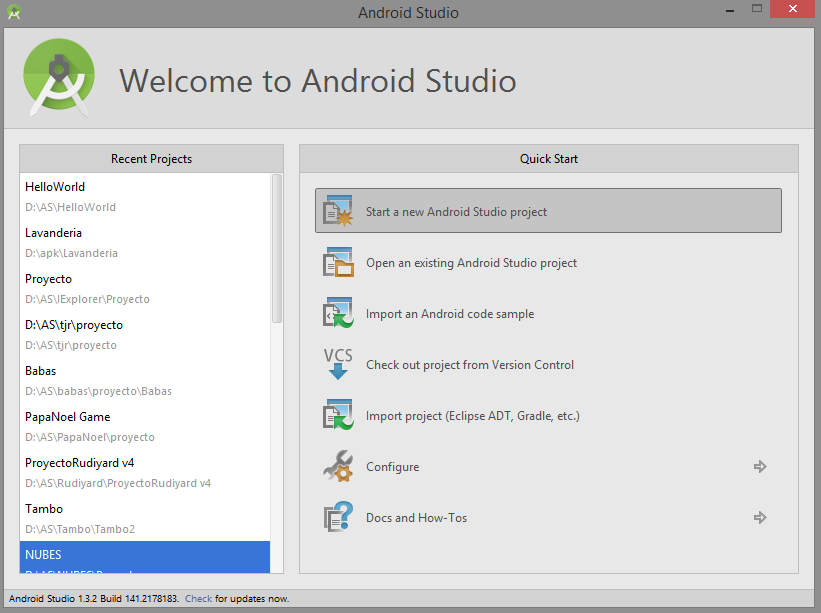
Con esto tenemos todo lo necesario para poder empezar a programar para Android.

### ****Teoria****Mi primer "Hola Mundo".

### Empezando a utilizar Android Studio

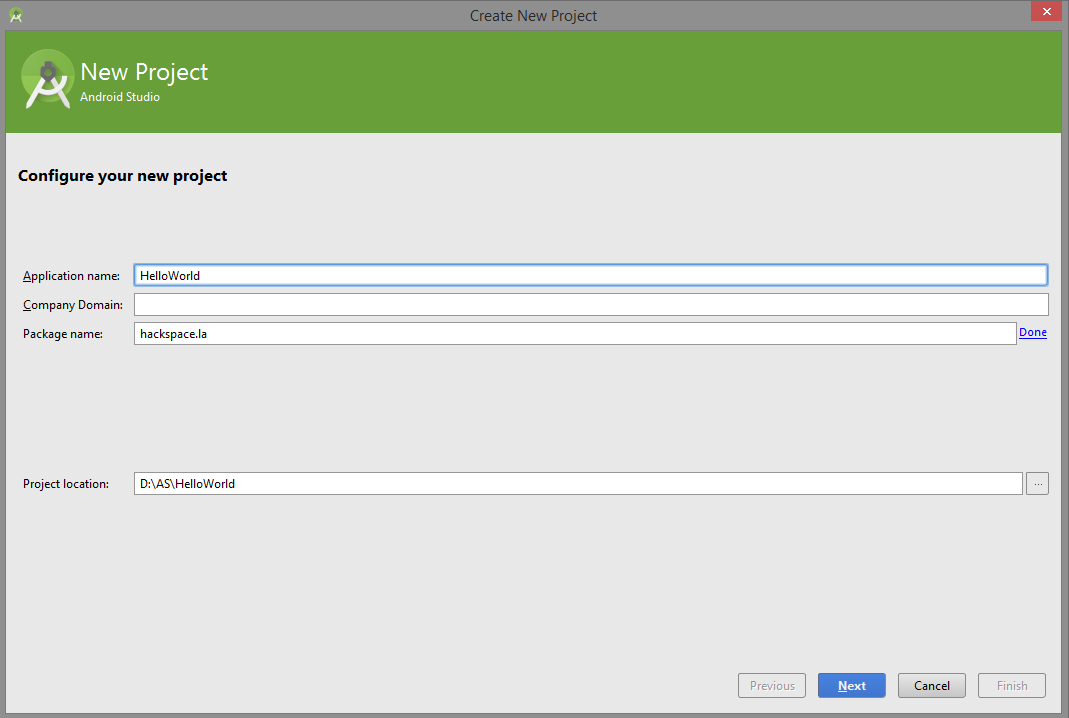
Una vez instalado el Android Studio nos sale nuestra ventana principal.

Elegimos la opción "Start a New Android Studio project":

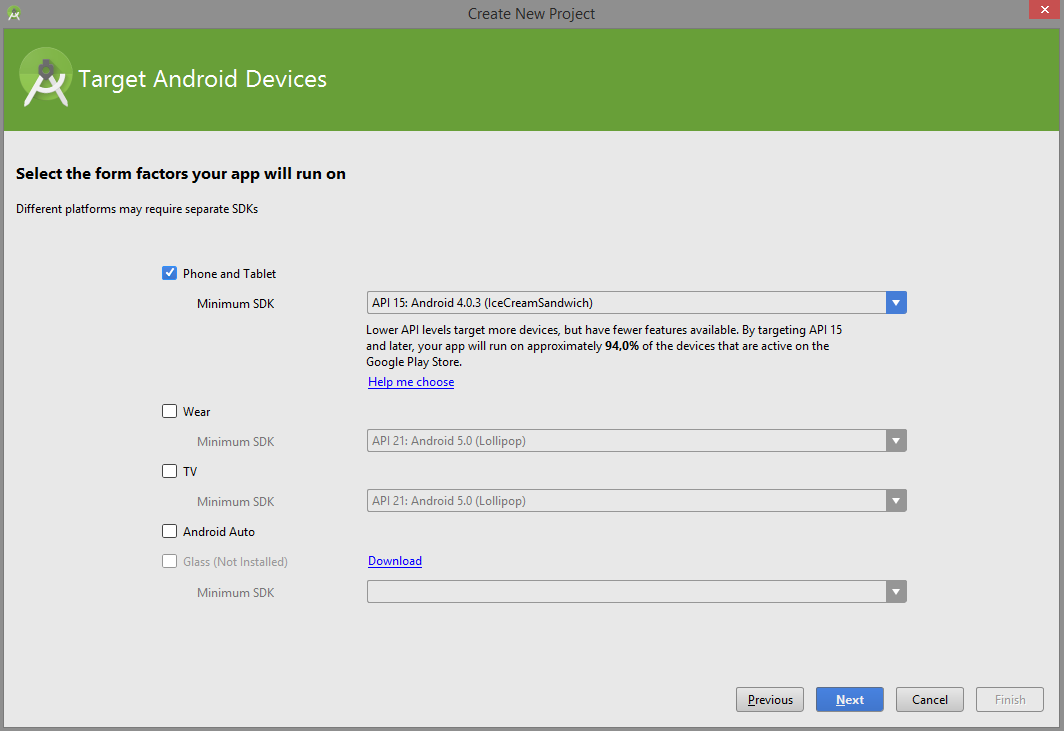


Ahora en la nueva ventana llenaremos las configuraciones principales de nuestra aplicación.

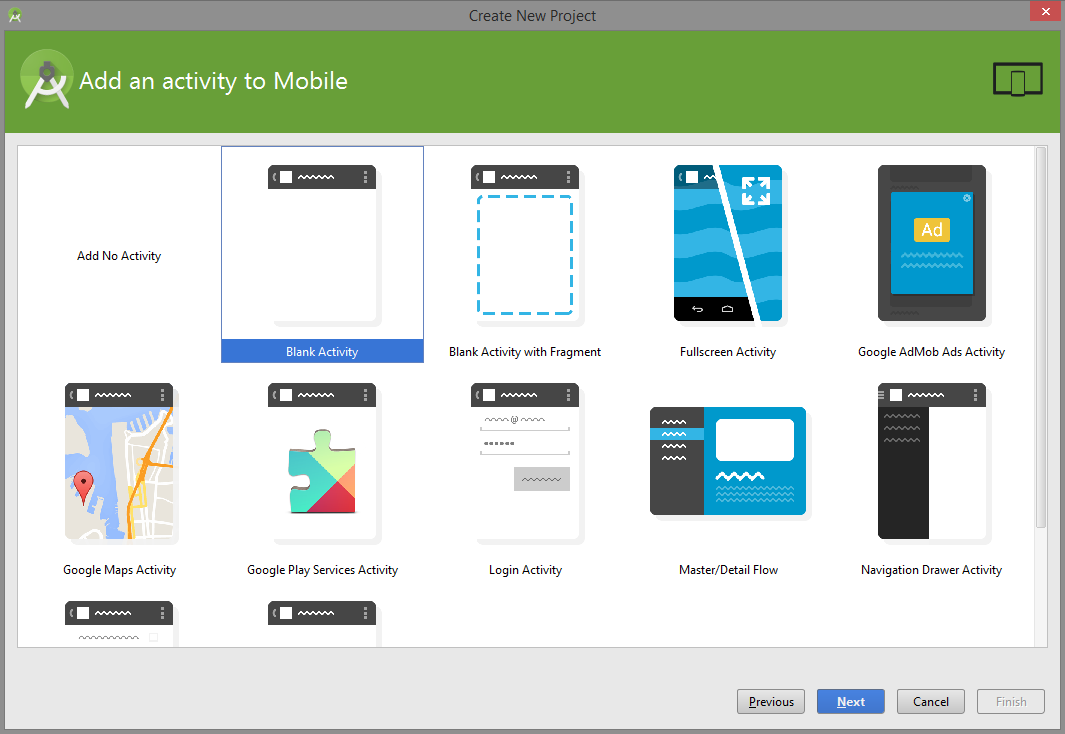
Especificamos el nombre de nuestra aplicación **HelloWorld**. El dominio **hackspace.la** y por defecto se crea el Package name. Este vendrá a ser el identificador de nuestra app, si deseamos lo podemos cambiar. Y por último, escogemos donde se guardará nuestro proyecto.



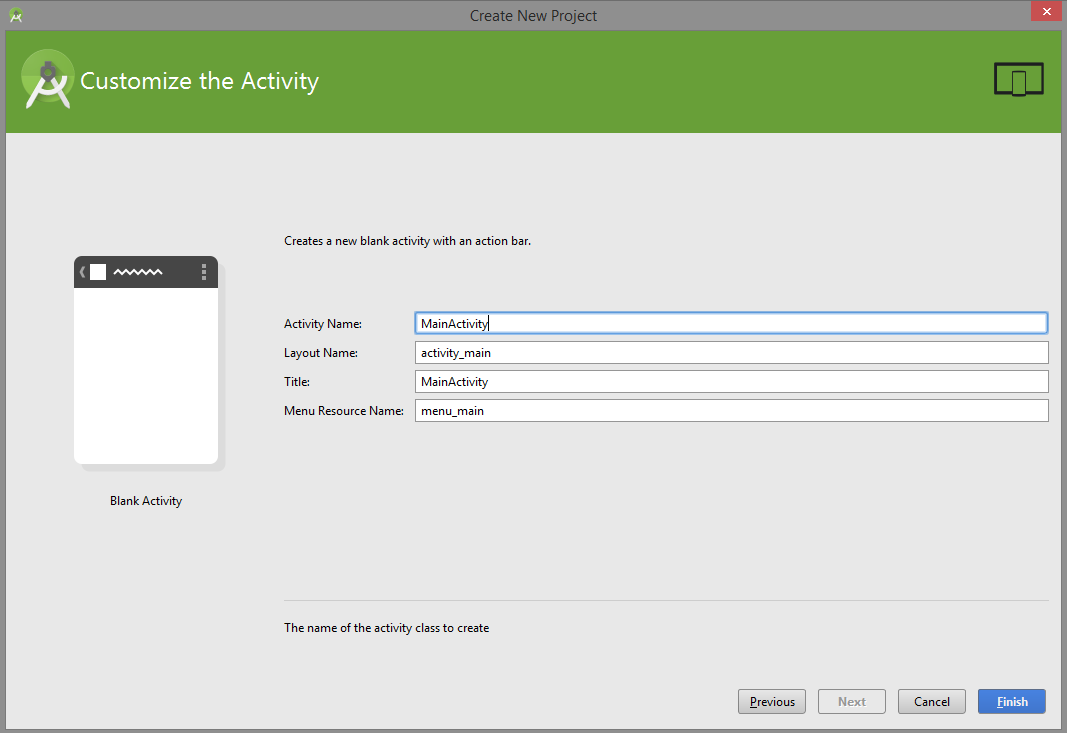
En la siguiente ventana especificamos que tipo de aplicación vamos a desarrollar. Lo dejamos tal y como está.



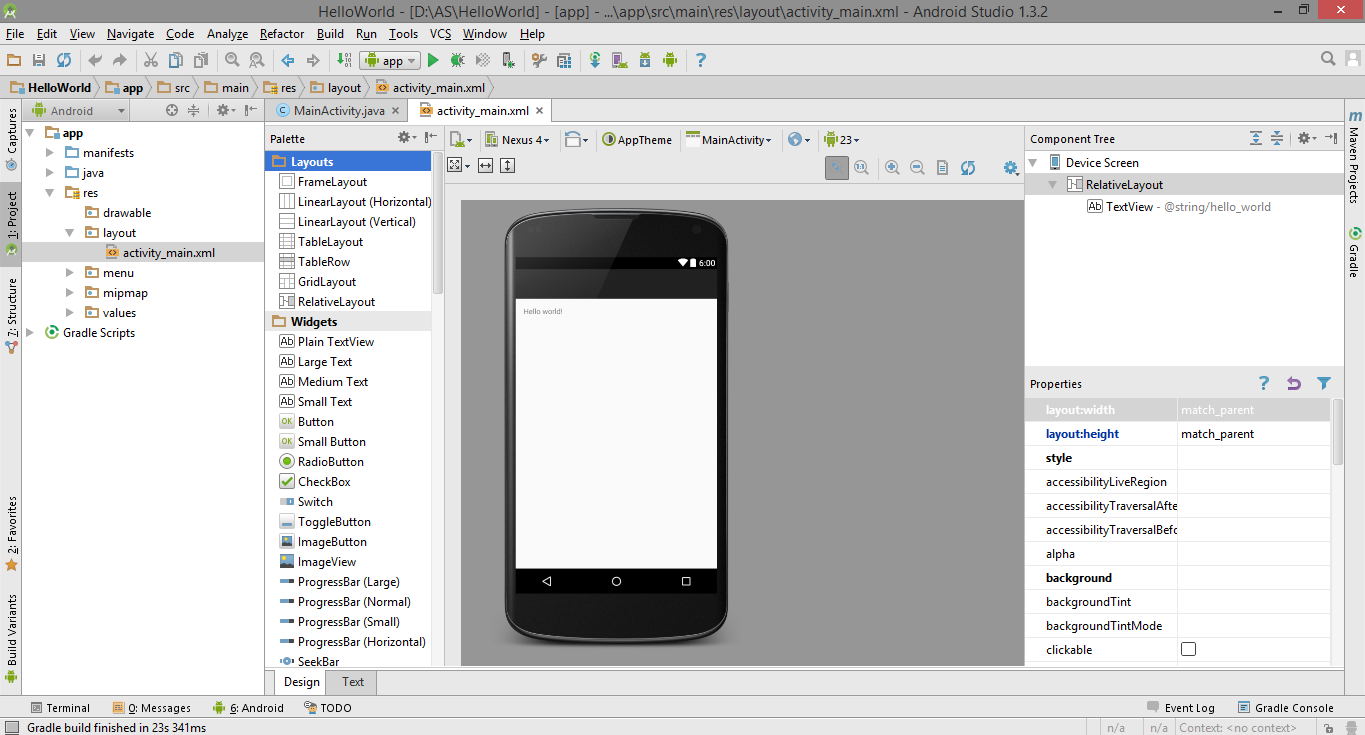
En la tercera ventana especificaremos como queremos que se vea la primera pantalla de la aplicación. Lo dejamos en **Blank Activity**.



Por último, tenemos que indicar como se llamará nuestra pantalla principal. Por ahora dejaremos todo igual.



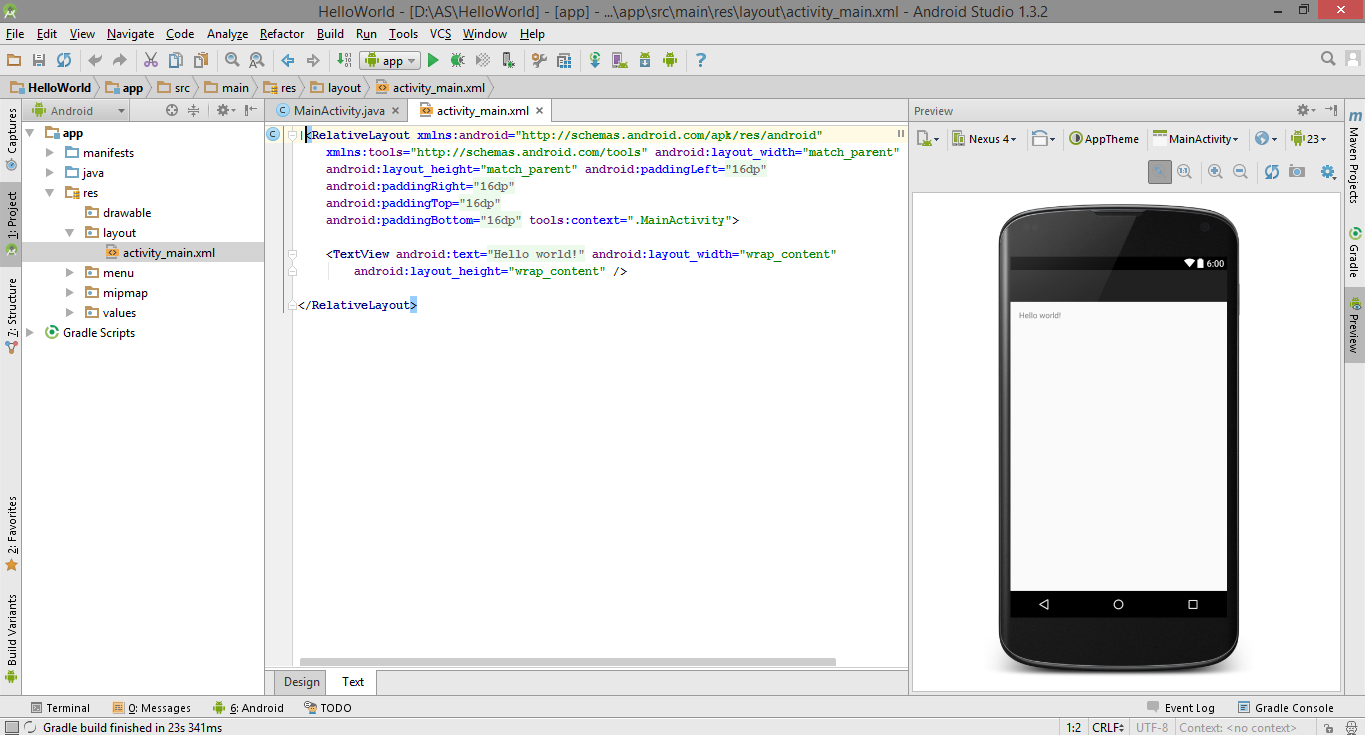
Le damos **finish** y se autogenerará nuestro primer código y se verá el entorno del Android Studio.



A la izquierda vemos el explorador de nuestro proyecto creado. No haremos en este momento un análisis del significado y objetivo de cada uno de estas secciones y archivos generados, sino a medida que avancemos con este curso iremos viendo en forma puntual y profunda.

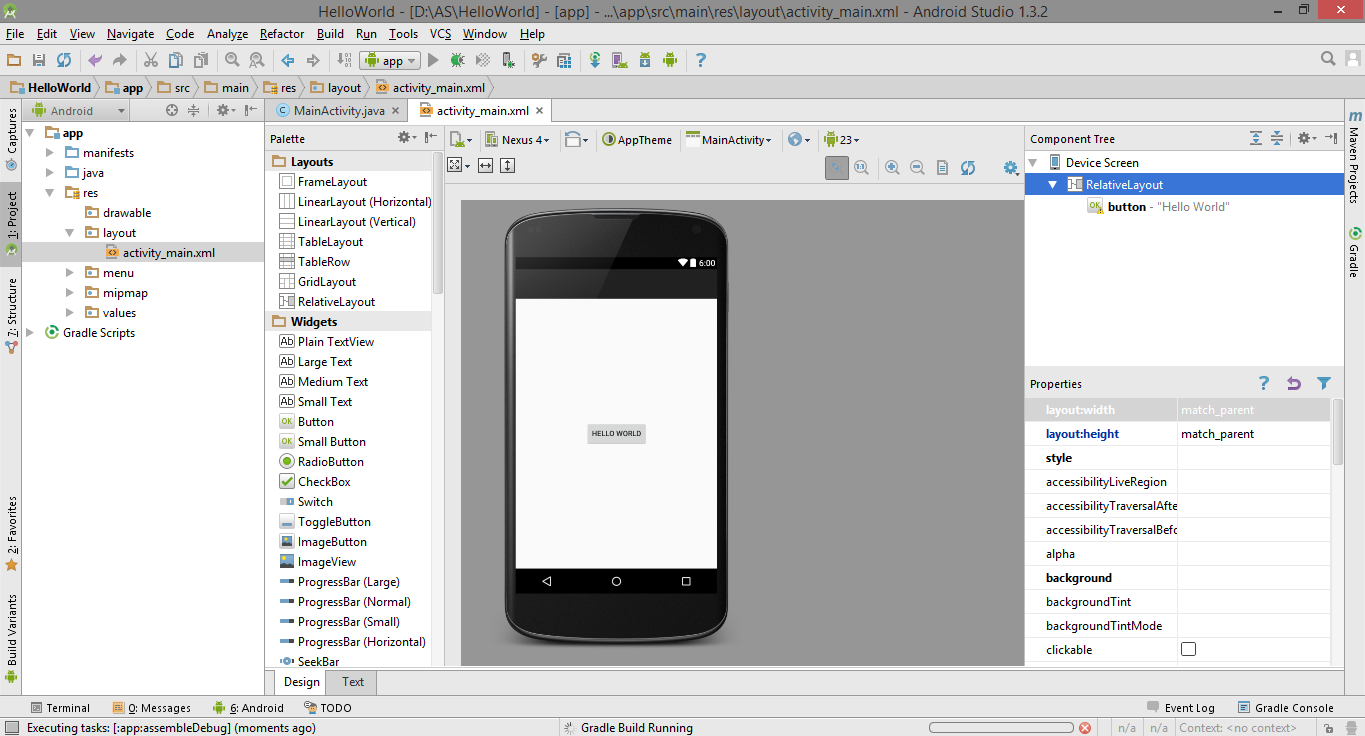
La interfaz visual de nuestro programa para Android se almacena en un archivo **XML** en la carpeta res, subcarpeta layout y el archivo se llama activity\_main.xml. En esta carpeta tenemos creada nuestra primera pantalla.

Al seleccionar este archivo, Android Studio nos permite visualizar el contenido en **Design** o **Text** (es decir en vista de diseño o en vista de código):

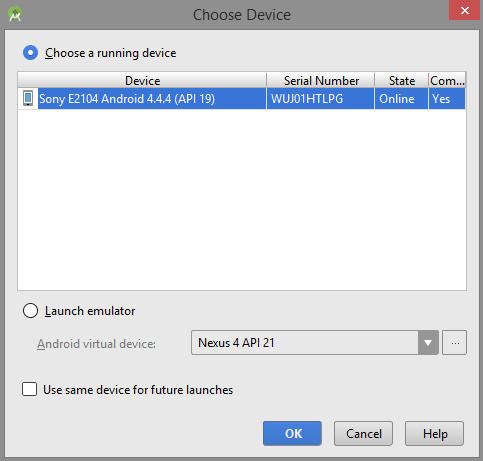


A lo largo de este curso iremos viendo los objetivos de cada una de las secciones que cuenta el Android Studio para implementar la interfaz, codificar en java las funcionalidades de la aplicación etc.

Antes de probar la aplicación procederemos a hacer un pequeño cambio a la interfaz que aparece en el celular: borraremos el widget de tipo **TextView** que dice **Hello World** (simplemente seleccionando con el mouse dicho elemento y presionando la tecla delete) y de la **Palette** arrastraremos un **Button** al centro del celular y en la ventana **Properties** estando seleccionado el **Button** cambiaremos la propiedad **text** por la cadena **Hola Mundo**.

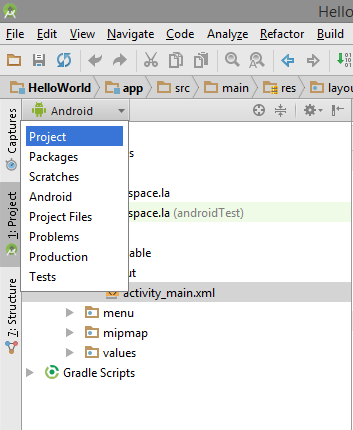


Para ejecutar la aplicación presionamos el triángulo verde o seleccionamos del menú de opciones **Run -> Run app**. En la ventana que se abre podremos ver nuestros dispositivos conectados, como aún no tenemos alguno esta lista aparece vacía. Entonces vamos a abrir nuestro Genymotion y le damos doble click en nuestro emulador. Una vez listo, aparecerá en nuestra lista. Lo seleccionamos y presionamos **Ok**.

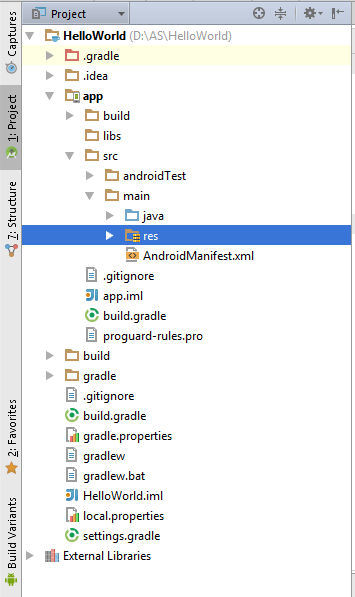


Luego de un rato podremos ver en nuestro emulador nuestra primera aplicación. Es IMPORTANTE tener en cuenta que una vez que el emulador se ha iniciado no lo debemos cerrar cada vez que hacemos cambios en nuestra aplicación, sino que volvemos a ejecutar la aplicación con los cambios y estos se ven reflejados en nuestro emulador.

En la parte izquierda, podemos ver el explorador de los elementos creados inicialmente para el nuevo proyecto Android. Para explicar 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:



### ****Teoria****Mi primer "Hola Mundo".

### Estructura del proyecto

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 **módulo**s, que pueden representar aplicaciones distintas, versiones diferentes de una misma aplicación, o distintos componentes de un sistema (aplicación móvil, aplicación servidor, librerías, etc.). 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 HelloWorld 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** | **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 de pantalla)   * /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 * /mipmap-xxhdpi |
| /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/color/ | Contiene ficheros XML de definición de listas de colores según estado. |
| /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. |

#### 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.

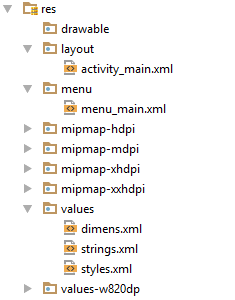
http://hackspace.la/img/ejemplos/android1.2.12.png

#### 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.

No todas estas carpetas tienen por qué aparecer en cada proyecto Android, tan sólo las que se necesiten.

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.

Entre los recursos creados por defecto cabe destacar los **layouts**, en nuestro caso sólo tendremos por ahora el llamado activity\_main.xml.

#### 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, etc.), sus componentes (pantallas, servicios, etc.), o los permisos necesarios para su ejecución.

#### 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.

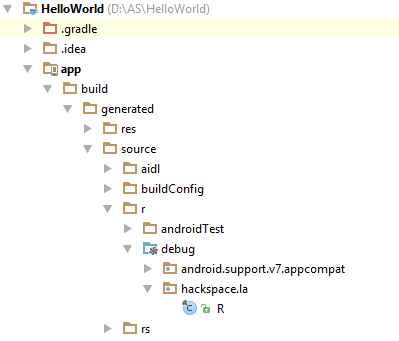
En un proyecto pueden existir varios ficheros build.gradle, para definir determinados parámetros a distintos niveles. Por ejemplo, en nuestro proyecto podemos ver que existe un fichero build.gradle a nivel de proyecto, y otro a nivel de módulo dentro de la carpeta app/. El primero de ellos definirá parámetros 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 ficherobuild.gradle descrito en el punto anterior, de forma que entren en el proceso de compilación de nuestra aplicación.

#### 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/.

### ****Reto 1****Primer reto del entrenamiento de Android.

### Reto 1 de Android

Ya hemos visto como de manera sencilla podemos colocar un botón.

Para este primer reto vamos a jugar con la interfaz de diseño de Android Studio. No hay problema si algunas cosas aún no entendemos, a lo largo de curso vamos a ir viendolas. Ahora concentremonos en crear una **interfaz de Login**:



Para crear la interfaz, primero tenemos que eliminar el boton que creamos. Ahora a la mano izquerda podremos observar todas las clases de widgets que Android nos provee. Buscamos la seccion **"Text Fields"**y arrastramos los dos campos que necesitamos.

Como ya lo dije, el objetivo es jugar, asi que no tengan miedo de jugar con las propiedades de los widgets como el texto entre otras. No se olviden de probar su aplicación en su emulador o en algun dispositivo Android.

Nota: Si quieren saber más sobre los widgets que nos proporciona Android les dejamos una pequeña lectura que más adelante nos servirá:[Click para ir a la Lectura](https://s3-us-west-1.amazonaws.com/udacity-content/PDFs/Common+Android+Views+Cheat+Sheet+%281%29.pdf)

#### Semana 2 : Jugando con layouts y Java

### ****video****Un video donde haremos una grilla con pokemons en donde veremos las propiedades de los LinearLayouts y RelativeLayouts.

**https://youtu.be/Kpy4HI8jMbc**

### ****Video****Explicamos cómo vincular widgets xml con java para crear un formulario de login funcional

**<https://youtu.be/wfWUlVC1eL0>**

### ****Reto 1****Primer reto del entrenamiento de Android.

### Reto 2 de Android

Hasta ahora hemos visto como crear una interfaz y como hacer un login simple.

Para esta semana vamos a jugar con lo que hemos aprendido. Vamos a usar nuestra imaginación y darle un estilo al login. Una página que me gusta mucho es[www.materialup.com](http://hackspace.la/semana2/android/www.materialup.com" \t "_blank). Aqui podran encontrar toda la inspiracion e ideas que necesitan.

El reto en concreto es crear un login más trabajado y que tenga algo de estilo. Para esto utilizaremos los contenedores o "layouts". Les dejo una lectura sobre este tema[http://developer.android.com/intl/es/guide/topics/ui/declaring-layout.html](http://developer.android.com/intl/es/guide/topics/ui/declaring-layout.html" \t "_blank).

Para tu calificación, solo tienes que enviarnos el pantallazo (o screenshot) del login que haz realizado. Y siempre preguntar si tienen alguna duda.

PD: Pueden encontrar en ejemplos aqui <http://www.materialup.com/search?q=login>.

[Entregar Reto](https://hackspaceper.typeform.com/to/hteLz6" \t "_blank)