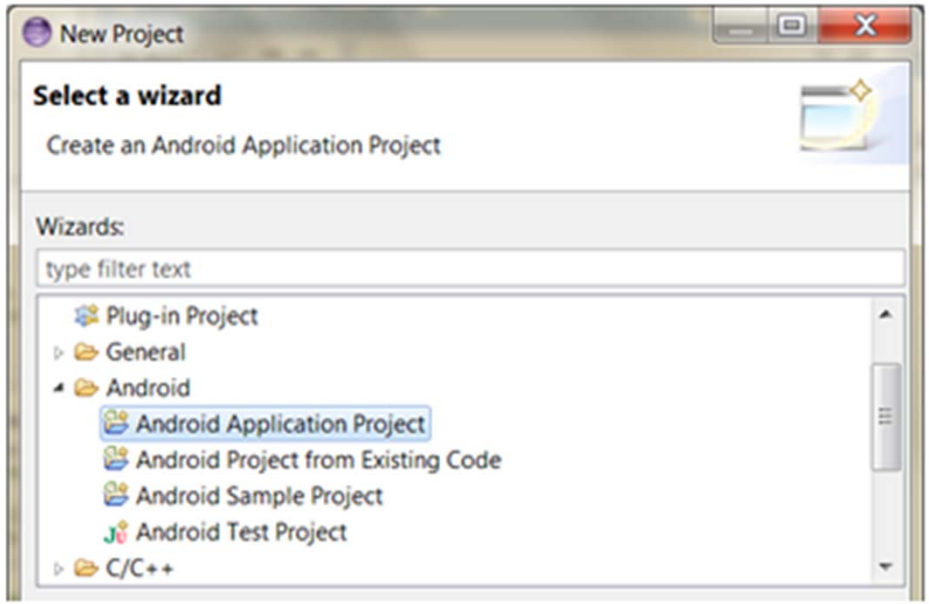
Nuestro primer programa: Hello Android!

Curso: Desarrollando aplicaciones con Android

Crear, construir y ejecutar la aplicación Android "HelloWorld"

- **Paso 1. Crea un nuevo proyecto Android**
  - Selecciona `File -> New -> Project...` Si el Plug-in de Android se ha instalado correctamente, el diálogo que aparece debe tener un directorio llamado `Android` que debe contener `Android Project`.



Crear, construir y ejecutar la aplicación Android "HelloWorld"

- **Paso 1. Crea un nuevo proyecto Android**
  - Rellena los detalles del proyecto
    - Como **`Application name`**, introduce **`Hello, Android`** (o el que más te guste)
    - Para el campo de texto **`Project name`**, introduce **`HelloAndroid`** (o el que más te guste)
    - En el campo de texto **`Package name`**, introduce es.uned.**`merida.app.android.helloandroid`**
    - En los campos de texto **`Minimun Required SDK`**, **`Target SDK`** y **`Compile With`** selecciona **`API 16: Android 4.1.2 (Jelly Bean)`**
    - En **`Theme`** selecciona **`None`**.
    - Pulsa sobre el botón **`Next >`**

Crear, construir y ejecutar la aplicación Android "HelloWorld"

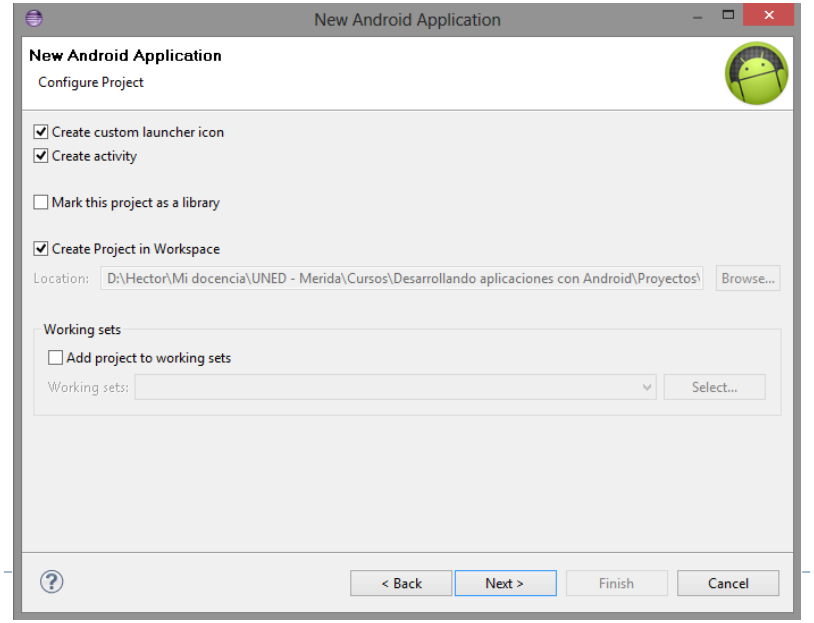
- A continuación vemos una descripción para cada campo:
  - `Application Name`: Es el nombre de la aplicación que aparecerá en el dispositivo Android. Tanto en la barra superior cuando esté en ejecución, como en el icono que se instalará en el menú de programas.
  - `Project Name`: Es el nombre del proyecto. Se creará una carpeta con el mismo nombre que contendrá los ficheros del proyecto.
  - `Package Name`: Indicamos el paquete con el espacio de nombres utilizado por nuestra aplicación. Hay que usar las reglas de los espacios de nombre en el lenguaje de programación Java. Las clases que creemos estarán dentro de él. Esto también establece el nombre del paquete donde se almacenará la aplicación generada.
  - `Minimum requiered SDK`: Este valor especifica el mínimo nivel del API que requiere tu aplicación. Por lo tanto, la aplicación no podrá ser instalada en dispositivos con una versión inferior. Procura escoger valores pequeños para que tu aplicación pueda instalarse en la mayoría de dispositivos. Un valor adecuado puede ser nivel de API 7 (v2.1), dado que cubriría más del 99,8% de los dispositivos. O nivel de API 8 (v2.2), que cubriría más del 98,4% de los dispositivos. Escoger valores pequeños para este parámetro tiene un inconveniente. No podremos utilizar ninguna de las mejoras que aparezcan en los siguientes niveles de API. Por ejemplo, si queremos que nuestra aplicación se configure de forma diferente cuando el usuario seleccione modo automóvil tendremos que indicar en este campo la versión 2.2, dado que este modo no aparece hasta esta versión.

Crear, construir y ejecutar la aplicación Android "HelloWorld"

- A continuación vemos una descripción para cada campo:
  - `Target SDK`: Indica la versión más alta de Android con la que se han puesto a prueba la aplicación. Cuando salgan nuevas versiones del SDK, tendrás que probar la aplicación con estas versiones y actualizar el valor.
  - `Compile With`: Es la versión de la plataforma con la que compila la aplicación. Se recomienda indicar la versión más reciente que haya aparecido. Las nuevas versiones no solo añaden funcionalidades al API, también añaden mejoras en el desarrollo. Por ejemplo, en algunas versiones se asigna un tema a la aplicación, o a partir de la versión 3.0 se verifica que un acceso a un servidor Web se haga en un hilo auxiliar. Utilizar un *Target SDK* alto no está reñido con usar un *Minimum requiered SDK*pequeño.
  - `Theme`: Estilo que se utiliza en la aplicación. Un tema define los colores, fuentes y otros aspectos visuales de nuestra aplicación.

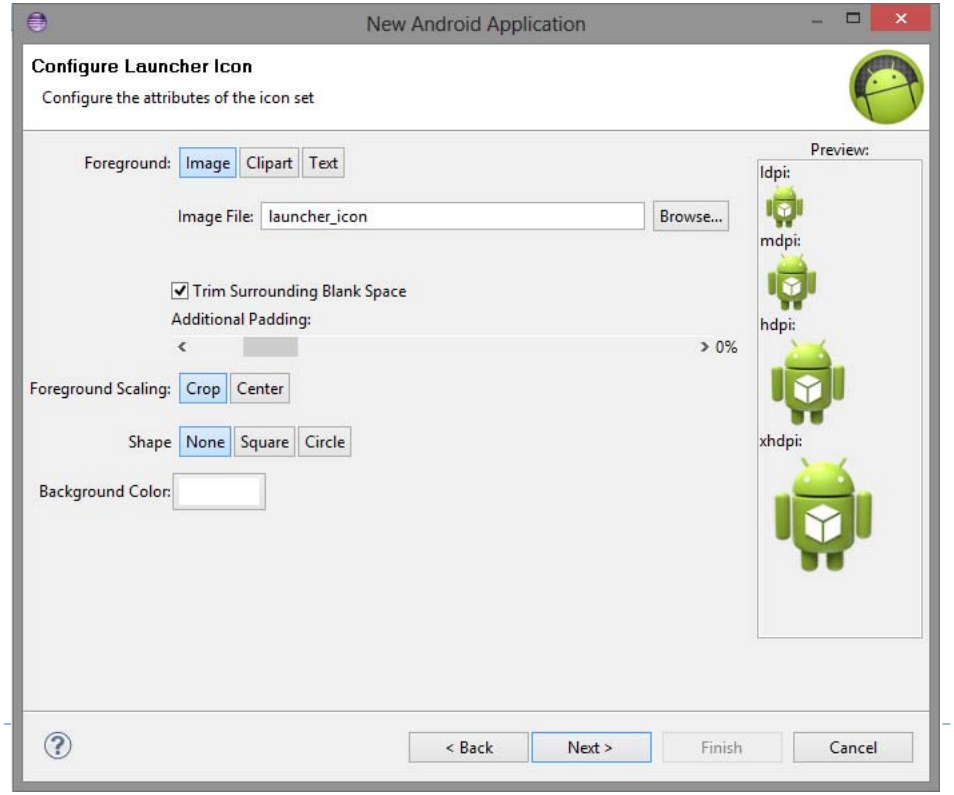
Crear, construir y ejecutar la aplicación Android "HelloWorld"

- **Paso 1. Crea un nuevo proyecto Android**
  - Puedes dejar los valores por defecto. Los dos primeros checkbox te permiten saltarte las pantallas que veremos a continuación. Los tres siguientes permiten crear una librería, indicar donde se almacena el proyecto y asignar una serie de parámetros de configuración. Pulsa sobre el botón **`Next >`**



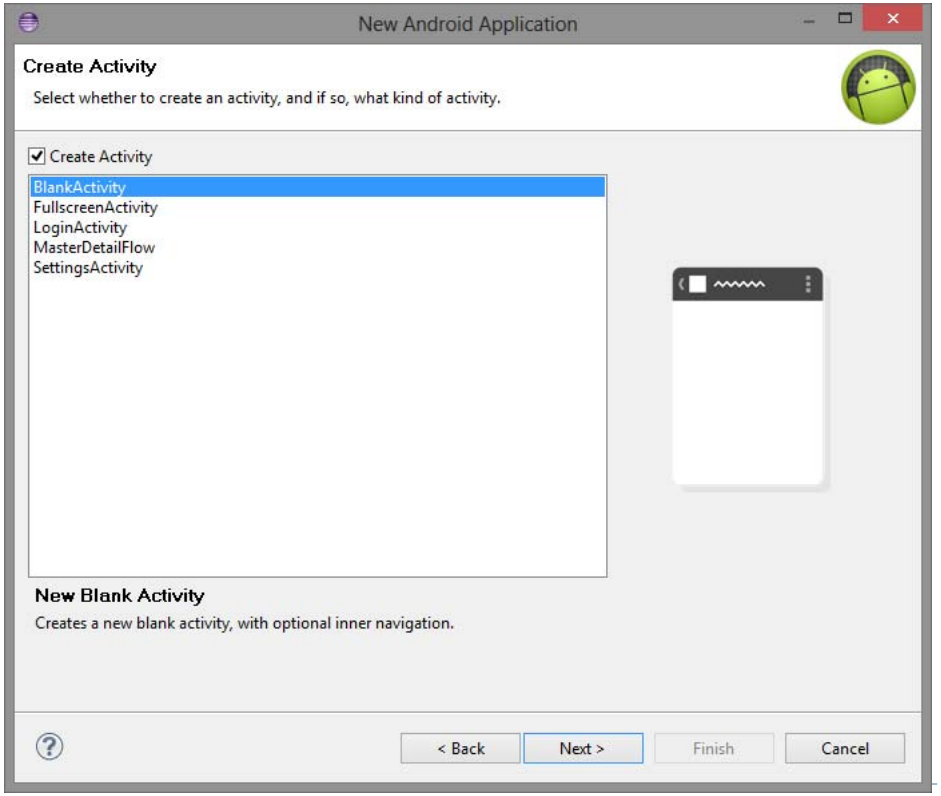
Crear, construir y ejecutar la aplicación Android "HelloWorld"

- **Paso 1. Crea un nuevo proyecto Android**
  - Aquí podrás configurar el icono de la aplicación. Resulta sencillo de utilizar. Más adelante explicaremos más detalles sobre el diseño de iconos. Pulsa sobre el botón **`Next >`**



Crear, construir y ejecutar la aplicación Android "HelloWorld"

- **Paso 1. Crea un nuevo proyecto Android**
  - En la siguiente pantalla especificamos que queremos crear una Actividad vacía. Pulsamos **`Next >`**.

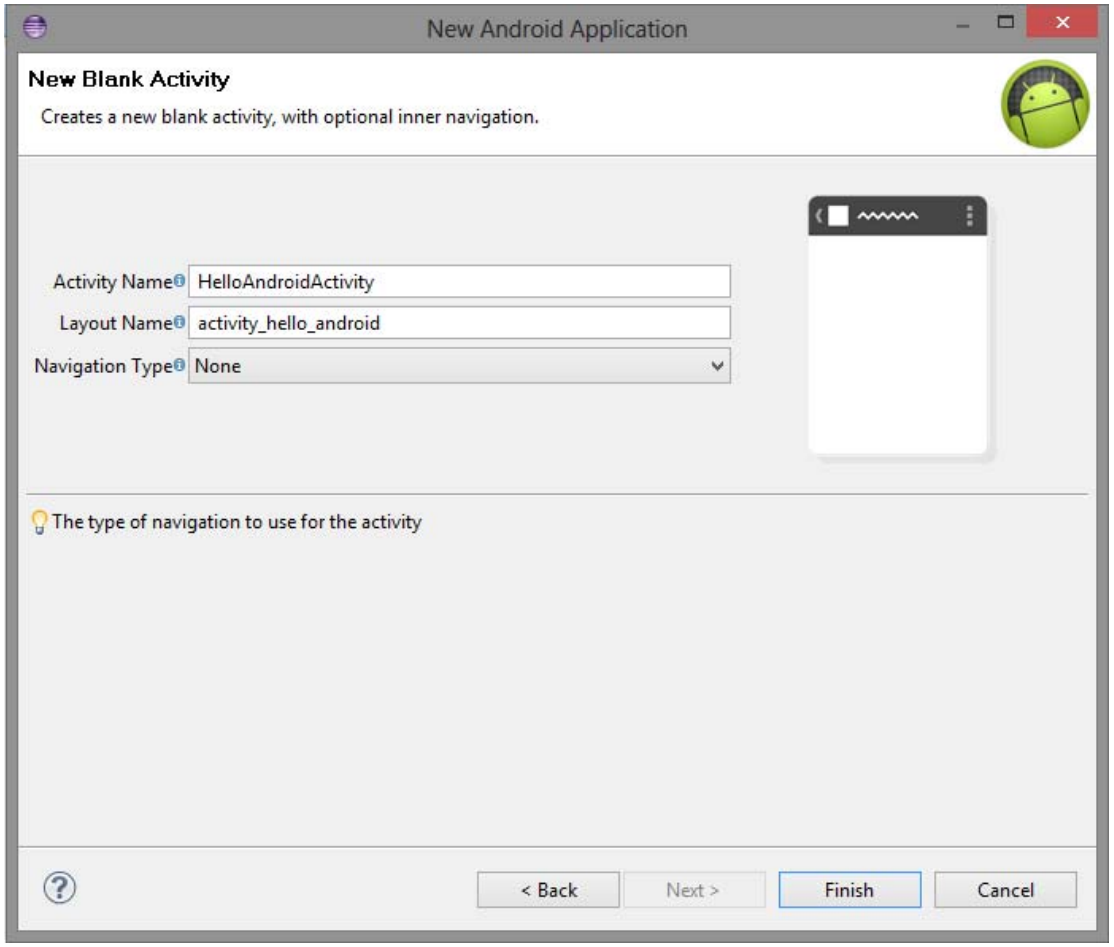


Crear, construir y ejecutar la aplicación Android "HelloWorld"

- **Paso 1. Crea un nuevo proyecto Android**
  - En el campo de texto **`Activity Name`** introduce **`HelloAndroidActivity`**. El campo de texto **`Layout Name`** lo dejamos tal y como nos lo rellena Eclipse: **`activity_hello_android`**.
  - Pulsamos sobre el botón **`Finish`**.

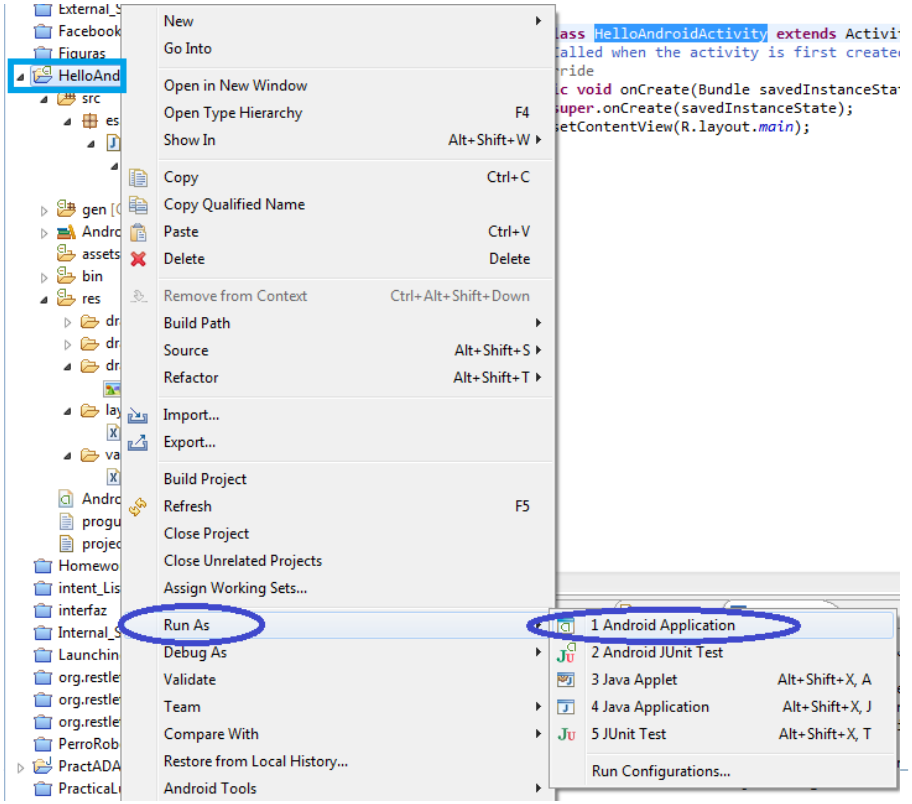
Crear, construir y ejecutar la aplicación Android "HelloWorld"

- **Paso 1. Crea un nuevo proyecto Android**



Crear, construir y ejecutar la aplicación Android "HelloWorld"

- **Paso 2. Construir y ejecutar la aplicación**
  - Pulsa con el botón derecho del ratón sobre el proyecto `HelloAndroid` y selecciona `Run As -> Android Application`. Paciencia.

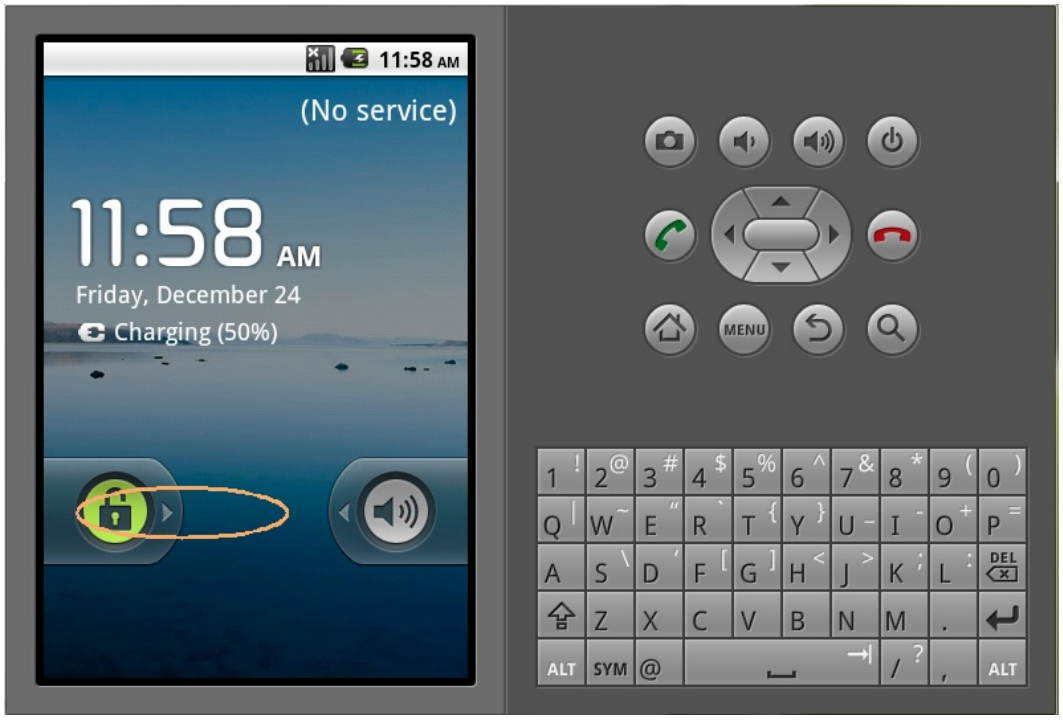


Crear, construir y ejecutar la aplicación Android "HelloWorld"

- **Paso 2. Construir y ejecutar la aplicación**
  - Observa que el emulador del dispositivo Android se visualiza.

Crear, construir y ejecutar la aplicación Android "HelloWorld"

- **Paso 3**
  - En el emulador, desliza la flecha ubicada en la izquierda hacia la derecha (para desbloquear)



Crear, construir y ejecutar la aplicación Android "HelloWorld"

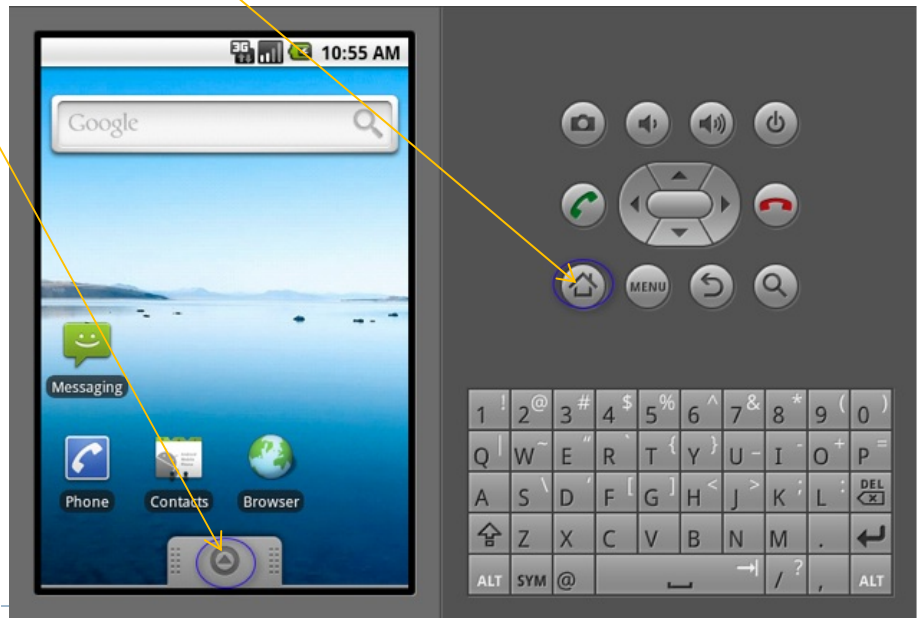
- **Paso 3**
  - Observa el resultado.

Crear, construir y ejecutar la aplicación Android "HelloWorld"

- **Paso 4**. La aplicación también puede lanzarse del siguiente modo:
  - Haz clic en Home (Inicio)
  - Haz clic en esa especie de panel táctil centrado en la parte inferior de la pantalla
  - Haz clic en tu aplicación.

Crear, construir y ejecutar la aplicación Android "HelloWorld"

- **Paso 4**. La aplicación también puede lanzarse del siguiente modo:
  - Haz clic en Home (Inicio)
  - Haz clic en esa especie de panel táctil centrado en la parte inferior de la pantalla



Crear, construir y ejecutar la aplicación Android "HelloWorld"

- **Paso 4**. La aplicación también puede lanzarse del siguiente modo:
  - Haz clic en tu aplicación.

Crear, construir y ejecutar la aplicación Android "HelloWorld"

- **Paso 5**. Ejecución en un terminar real
  - No tienes más que usar un cable USB para conectarlo al PC.
  - Abre `Android SDK and AVD Manager` y asegúrate que está instalado el paquete `USB Driver`. En caso contrario instálalo.

Posiblemente este driver genérico no sea adecuado para tu terminal y tengas que utilizar el del fabricante. Si no dispones de el, puedes buscarlo en: http://developer.android.com/sdk/oem-usb.html

Crear, construir y ejecutar la aplicación Android "`HelloWorld`"

- **Paso 5**. Ejecución en un terminar real
  - En el terminal accede al menú `Ajustes > Opciones de desarrollador` y asegúrate que la opción `Depuración de USB` está activada.

  - Conecta el cable USB.
  - Se indicará que hay un nuevo *hardware* y te pedirá que le indiques el controlador.

La aplicación HelloAndroid en detalle

- Un proyecto Android está formado básicamente por un descriptor de la aplicación (`AndroidManifest.xml`), el código fuente en Java y una serie de ficheros con recursos. Cada elemento se almacena en una carpeta específica.

- `src`: Carpeta que contiene el código fuente de la aplicación. Como puedes observar los ficheros Java se almacenan en un espacio de nombres.
- `gen`: Carpeta que contiene el código generado de forma automática por el SDK. Nunca hay que modificar de forma manual estos ficheros. Dentro encontraremos:
  - `BuildConfig.java`: Define la constante DEBUG para que desde Java puedas saber si tu aplicación está en fase de desarrollo.
  - `R.java`: Define una clase que asocia los recursos de la aplicación con identificadores. De esta forma los recursos podrán ser accedidos desde Java.

La aplicación HelloAndroid en detalle

- `Android x.x`: Código JAR, el API de Android según la versión seleccionada.
- `Android Dependencies`: Librerías asociadas al proyecto.
- `assets`: Carpeta que puede contener una serie arbitraria de ficheros o carpetas que podrán ser utilizados por la aplicación (ficheros de datos, fuentes,…). A diferencia de la carpeta `res`, nunca se modifica el contenido de los ficheros de esta carpeta ni se les asociará un identificador.
- `bin`: En esta carpeta se compila el código y se genera el .apk, fichero comprimido que contiene la aplicación final lista para instalar.
- `libs`: Código JAR con librerías que quieras usar en tu proyecto.

La aplicación HelloAndroid en detalle

- `res`: Carpeta que contiene los recursos usados por la aplicación. Las subcarpetas pueden tener un sufijo si queremos que el recurso solo se cargue al cumplirse una condición. Por ejemplo `-hdpi` significa que solo ha de cargar los recursos contenidos en esta carpeta cuando el dispositivo donde se instala la aplicación tiene una densidad gráfica alta (>180dpi); `-v11` significa que el recurso solo ha de cargarse en un dispositivo con nivel de API 11 (v3.0).
  - `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.
  - `values`: También utilizaremos ficheros XML para indicar valores del tipo string, color o estilo. De esta manera podremos cambiar los valores sin necesidad de ir al código fuente. Por ejemplo, nos permitiría traducir una aplicación a otro idioma.
  - `anim`: Contiene ficheros XML con animaciones por interpolacion.
  - `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.

La aplicación HelloAndroid en detalle

- `AndroidManifest.xml`: Este fichero describe la aplicación Android. En él se indican las *actividades*, *intenciones*, *servicios* y *proveedores de contenido* de la aplicación. También se declaran los permisos que requerirá la aplicación. Se indica la versión mínima de Android para poder ejecutarla, el paquete Java, la versión de la aplicación, etc.
- `ic_launcher-web.png`: Icono de la aplicación de gran tamaño para ser usado en páginas Web. El nombre puede variar si se indicó uno diferente en el proceso de creación del proyecto. Ha de tener una resolución de 512x512 (con alfa).
- `proguard-project.txt`: Fichero de configuración de la herramienta *ProGuard*, que te permite optimizar y ofuscar el código generado. Es decir, se obtiene un .apk más pequeño y donde resulta más difícil hacer ingeniería inversa.
- `default.properties`: Fichero generado automáticamente por el SDK. Nunca hay que modificarlo. Se utiliza para comprobar la versión del API y otras características cuando se instala la aplicación en el terminal.

La aplicación HelloAndroid en detalle

- Paso 1: Estudiar la clase `HelloAndroidActivity` definida en `HelloAndroidActivity.java`
  - Observa que la clase `HelloAndroid` extiende la clase `Activity`. Una actividad es una entidad de aplicación que se utiliza para realizar acciones. Una aplicación puede tener múltiples actividades, pero el usuario interactuará con ellas de una en una. El método `onCreate()` será invocado por el sistema Android cuando se inicia la actividad – es, por lo tanto, donde se debe realizar toda la inicialización y configuración de la interfaz de usuario. Una actividad no requiere tener una interfaz de usuario, pero, por lo general, la tendrá.
    - Haz doble clic en `HelloAndroidActivity.java` bajo `HelloAndroid -> es.uned.merida.app.android.helloandroid`.
    - Observa que el código de `HelloAndroidActivity.java` se muestra en la ventana de edición.
    - Mueve el cursor sobre `Activity` y espera un segundo aproximadamente. Observa como emerge la documentación de la clase `Activity` (Utilice esta característica del IDE Eclipse para estudiar cualquier API de Android del que necesites tener más información.)

La aplicación HelloAndroid en detalle

- Paso 2: Estudiar el archivo de recursos de diseño en modo gráfico
  - Cada pantalla tiene su archivo de recursos de diseño correspondiente (a menos que se cree programáticamente). Como la aplicación `HelloAndroid` sólo tiene una pantalla, sólo hay un archivo de recursos de diseño. En esta aplicación se nombra como `hello_android_activity.xml`. La actividad especifica que archivo de recursos de diseño utilizar para cada pantalla que representa.
    - Expande `HelloAndroid-> res-> layout`.
    - Haz doble clic sobre `hello_android_activity.xml` haga clic. En función del sistema, el archivo de diseño `hello_android_activity.xml` se visualiza, por defecto, en modo gráfico. Si no es así, haz clic en la ficha Diseño gráfico (*Graphical Layout*) en la parte inferior del editor para verlo en modo gráfico.

La aplicación HelloAndroid en detalle

- Paso 3: Estudiar el archivo de recursos de diseño en modo XML
  - Haz clic sobre la ficha `hello_android_activity.xml`.
  - Observa el archivo de recursos de diseño `hello_android_activity.xml` ahora en modo XML.

La aplicación HelloAndroid en detalle

- Paso 4: Estudiar el archivo `AndroidManifest.xml`
  - Antes de que el sistema Android puede iniciar un componente de aplicación, por ejemplo, una actividad, debe saber que existe dicho componente. Por ello, las aplicaciones declaran sus componentes en un archivo manifiesto que se incluye en el paquete Android, el archivo `.apk`, que también incluye el código de la aplicación, archivos y recursos.
  - El manifiesto es un archivo XML estructurado y, para todas las aplicaciones Android, siempre se denomina `AndroidManifest.xml`. Además de declarar los componentes de la aplicación, hace otras cosas como nombrar las librerías con las que la aplicación debe estar vinculada (además de la librería Android por defecto) e identificar los permisos que la aplicación espera sean concedidos.
  - Pero la tarea principal del manifiesto es informar a Android sobre los componentes de la aplicación. En este ejemplo, tenemos sólo un componente de aplicación, la actividad `HelloAndroidActivity`.
    - Haz doble clic en `AndroidManifest.xml`
    - Observa que el archivo `AndroidManifest.xml` se muestra en modo asistente.

La aplicación HelloAndroid en detalle

- Paso 4: Estudiar el archivo `AndroidManifest.xml`
  - Haz clic en la ficha `AndroidManifest.xml` y observa como el archivo se muestra en formato XML.
  - Observa que esta aplicación contiene una sola actividad llamada `HelloAndroidActivity`.

La aplicación HelloAndroid en detalle

- Paso 4: Estudiar el archivo `AndroidManifest.xml`
  - El atributo `name` del elemento `<activity>` nombra a la subclase `Activity` que implementa la actividad. En el ejemplo, se establece `.HelloAndroidActivity`. El paquete, `es.uned.merida.app.android`, y el nombre de la actividad `.HelloAndroidActivity`, constituyen la ruta completa a la clase `Activity`, `es.uned.merida.app.android.HelloAndroidActivity` en este caso. Los atributos `icon` y `label` apuntan a archivos de recursos que contienen el icono y la etiqueta que se pueden mostrar a los usuarios para representar la actividad.
  - Una actividad que contiene el sub-elemento `<intent-filter>`, con sus propios `sub-elementos <action android:name="android.intent.action.MAIN" />` y `<category android:name="android.intent.category.LAUNCHER" />` especifica que esa actividad es la actividad de inicio. Buscando una analogía con una aplicación Java, indica que el método `onCreate()` de esa actividad funciona igual que el método "`main`" de la aplicación Java.

La aplicación HelloAndroid en detalle

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="“es.uned.merida.app.android"
    android:versionCode="1"
    android:versionName="1.0" >

    <uses-sdk android:minSdkVersion="“16" />

    <application
        android:icon="@drawable/ic_launcher"
        android:label="@string/app_name" >
        <activity
            android:name=".HelloAndroidActivity"
            android:label="@string/app_name" >
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>

</manifest>
```

La aplicación HelloAndroid en detalle

- Paso 5: Estudiar el archivo de recursos `strings.xml`
  - En el archivo de recursos `strings.xml` defines todas las cadenas de caracteres para tu interfaz de usuario. Así, en lugar de codificar una cadena de caracteres en el archivo de recursos de diseño o en las clases Java, definirás la cadena en el archivo `strings.xml` y, entonces, te referirás a ella utilizando el nombre de la cadena.
    - Expandir `HelloAndroid-> res-> values`.
    - Haz doble clic sobre `strings.xml`.
    - Observa que las entradas del archivo `strings.xml` se muestran en modo asistente.

La aplicación HelloAndroid en detalle

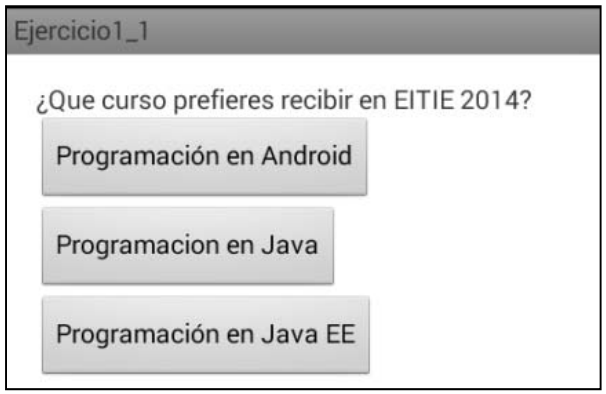
- Paso 5: Estudiar el archivo de recursos `strings.xml`
  - Haz clic en la ficha `strings.xml` de la parte inferior.
  - Observa que el archivo `strings.xml` se muestra en modo XML.
  - Observa que hay dos elementos de cadena de caracteres – "`hello`" y "`app_name`" - definidos en el archivo `strings.xml` de esta aplicación.

La aplicación HelloAndroid en detalle

- Paso 6: Estudiar el fichero `R.java`
  - El archivo `R.java` de un proyecto es un índice a todos los recursos definidos en los archivos de recursos de la aplicación. Ejemplos de archivos de recursos son el archivo de recursos de diseño - `main.xml` en este ejemplo - o el archivo de recursos `strings.xml`. Utiliza esta clase en el código fuente como una especie de atajo para hacer referencia a los recursos que hemos incluido en el proyecto. Esto es particularmente potente junto a las características de auto-completado de código de entornos de desarrollo como Eclipse, ya que permite localizar rápida e interactivamente la referencia específica que estás buscando.
  - Es posible que tu archivo `R.java` tenga un aspecto ligeramente diferente al que se te mostrará a continuación (tal vez los valores hexadecimales son diferentes). Por ahora, observa la clase interna denominada "`layout`", y su atributo de clase "`main`". El plugin de Eclipse advierte el archivo de diseño XML denominado `main.xml` y genera una clase para él en el archivo `R.java`. Cuando añadas otro recurso a tu proyecto (por ejemplo, una cadena de caracteres en el fichero `res/values/strings.xml` o dibujables en el directorio `res/drawable/`) verás como el archivo `R.java` cambia mantenerlo.
  - Nunca debes editar este fichero manualmente.

Ejercicio 1.1.

- Crear la aplicación HelloAndroidVersion2
  - La ejecución en el emulador debe mostrar lo siguiente:



  - Cuando hagas clic en los botones, debe cambiar su texto, el color del texto y/o el color del botón del siguiente modo:
    - De "Programación en Android" a "Prefiero Android". El color del texto a Color.GREEN.
    - De "Programación en Java" a "Prefiero Java". El color del botón a Color.YELLOW.
    - De "Programación avanzada en Java EE" a "Prefiero Java EE". El color del texto a Color.BLUE y el color del botón a Color.CYAN.