

SERVICIOS EN DISPOSITIVOS INALÁMBRICOS

**Tema II. Desarrollo de Servicios
y Aplicaciones para Dispositivos
Inalámbricos**

Índice

- Introducción a la programación de aplicaciones sobre plataformas móviles basadas en Java.2.2
- Programación con Android Studio y Android SDK2.3
- Desarrollo de aplicaciones básicas para móviles con Android Studio2.4
- Introducción a la programación de sensores

INTRODUCCIÓN A LA PROGRAMACIÓN DE APLICACIONES SOBRE PLATAFORMAS MÓVILES BASADAS EN JAVA.2.2

Introducción

- Actualmente es necesario el trabajo en el campo de la programación para dispositivos móviles
- Empresas adaptándose a tendencias del mercado y necesidades de sus clientes
- Posibilitar acceso a la información en cualquier lugar y en cualquier momento
- Este acceso debe proporcionarse tanto dentro de la administración de la propia empresa desde dispositivos móviles como a través de aplicaciones de escritorio
- Parte esencial del funcionamiento de los procesos empresariales

Dispositivo Móvil

- Puede definirse como cualquier arquitectura hardware con características similares a las máquinas de escritorio
- Con la salvedad de que todos los recursos son más reducidos
- Suelen estar integrados en una sola pieza

Entornos de Programación

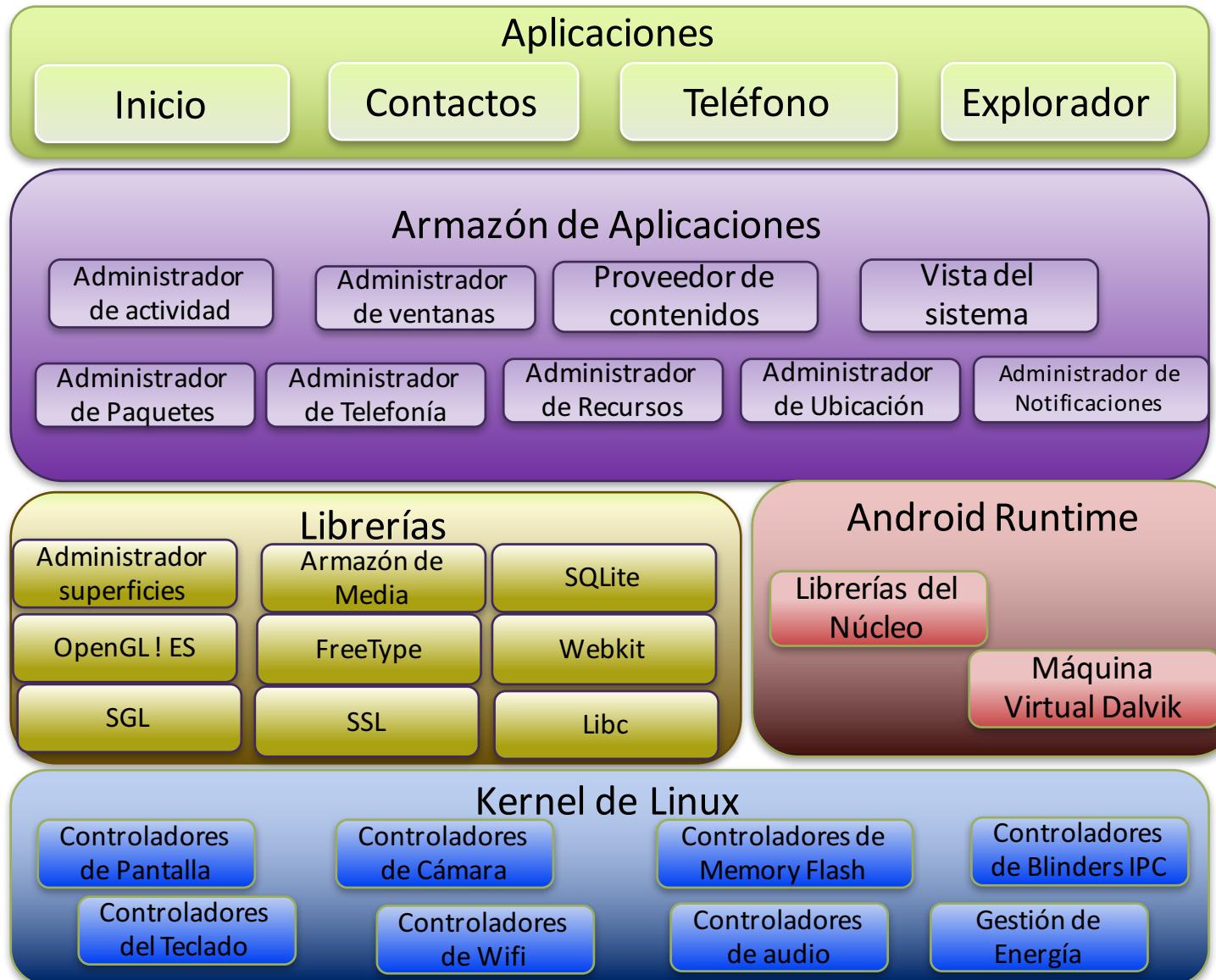
- Sistema Operativo para móvil:
 - Symbian
 - Windows Mobile
 - IOS
 - Linux
 - Android

PROGRAMACIÓN CON ANDROID STUDIO Y ANDROID SDK2.3

Entornos de Desarrollo

- Compilador
- Emulador
- Depurador
- Perfiles de rendimiento
 - Plugin para Eclipse
 - Orientado a comandos

Entornos de Desarrollo



Arquitectura: Aplicaciones

- Incluidas por defecto de Android
- Añadidas por el usuario
 - Sin diferenciación
 - De terceras empresas
 - O de su propio desarrollo
- Usan API, servicios y librerías por igual

Arquitectura: Entorno de Aplicaciones

- Conjunto de herramientas de desarrollo de cualquier aplicación:
 - Activity Manager: Gestión del Ciclo de Vida
 - Window Manager: Gestión de las ventanas de aplicaciones (usa la librería surface manager)
 - Telephone Manager: Incluye todas las API relacionadas con las funcionalidades propias del teléfono (llamadas, mensajes, etc)

Arquitectura: Entorno de Aplicaciones

- Content Provider: Compartición de datos entre aplicaciones. Acceso a los contactos, agenda....
- View System: Elementos de construcción de (GUI): botones, entradas de texto, junto con la gestión de estos elementos.
- Location Manager: Información de posicionamiento
- Notification Manager: Comunicación de eventos al usuario: llamadas entrantes, SMS, conexiones Bluetooth, etc. Pueden asociarse a una acción (intent).
- XMPP Service: API de acceso para el intercambio de mensajes XML.

Arquitectura: Librerías

- Libc: librería básica de C (se basa en BSD)
- Surface Manager: Construcción y gestión de los elementos visuales y views de una aplicación
- OpenGL/SL y SGL: librería gráfica con soporte 3D (OpenGL/SL) y 2D (SDL)
- Media Libraries: librería de códecs y formatos multimedia
- FreeType: librería de fuentes
- SSL: Librería de comunicaciones seguras
- SQLite: Librería para la creación y gestión de BBDD relacionales
- WebKit: Núcleo del navegador Web

Arquitectura: Entorno de Ejecución

- Core Libraries: librerías Java
- Máquina Virtual Dalvik

Arquitectura: Núcleo Linux

- Basado en el núcleo de linux 2.6
 - HAL: capa de abstracción para el hardware disponible en los dispositivos móviles
 - Drivers necesarios para que cualquier componente hardware se pueda usar mediante las llamadas correspondientes
 - Fabricantes de Hardware:
 - Crean librerías de control
 - Crean los drivers

Dalvik

- VM: arquitectura basada en registros
- Optimizada para:
 - Bajo consumo de memoria
 - Recolección de basura
 - Ejecución múltiples instancias simultáneas
 - Delega en el núcleo la gestión (procesos, memoria e hilos)
- Librerías escritas en C/C++ con wrappers de acceso

Dalvik

- Los programas de Android se compilan en .dex (Dalvik Executable) y se empaquetan en un .apl (Android Package)
 - Reutiliza la información duplicada por múltiples archivos .class
 - Menos memoria que un .jar
 - Bytecode propio (no compatible JVM)
- Ingeniería Inversa .dex: www.retrodev.com

Bloques de construcción

- Android Manifest.xml
- Actividades
- Vistas
- Intents
- Servicios
- Notificaciones
- Proveedores de Contenido

Actividades

- Las actividades se apilan
- Sólo una es visible
- Sólo una está activa
- Se van apilando encima según van apareciendo

Estado de las Actividades

- Activa
 - Encima de la Pila
- Pausada
 - Sin foco, todavía está visible
- Parada
 - No visible
 - Pueden eliminarse en caso de pocos recursos
- En destrucción
 - Se ha notificado su destrucción

Vistas

- Bloques de construcción básicos
- Saben cómo dibujarse
- Responden a eventos
- Se organizan en forma de árbol para construir el GUI
- Están definidos en el XML en los recursos layout

Views y ViewGroups

- Las Views y Viewgroups construyen GUIs complejos
- El Framework de Android se encarga de:
 - Las Medidas
 - Los emplazamientos
 - Dibujado

Patrón: carga de layout

- Android compila el código XML que describe el layout
- Después se carga al crearse la Actividad

```
Public void onCreate (Bundle savedInstanceState)
{
    ...
    setContentView (R.layout.filename)
    ...
}
```

Patrón: identificadores

- Identificadores únicos en la definición de la View en XML
 - Permite su uso posterior en Java

```
private View name;  
  
public void onCreate (Bundle savedInstanceState)  
{  
    ...  
    name = (View) findViewById(R.id.name)  
    ...  
}
```

Intents

- Descripción abstracta de una operación
 - Permite especificar la Actividad a invocar
- Describe lo que la aplicación pretende hacer
- Se asocian una acción, unos datos y una categoría
 - Las actividades declaran las acciones que pueden llevar a cabo y los tipos de datos que pueden gestionar.
 - La categoría indica si la actividad se arranca desde el lanzador de aplicaciones, desde el menú de otra aplicación o desde otra actividad

Ejemplo de Intent

```
<!-- Este filtro nos permite ver y editar los datos de una nota
<intent-filter android:label="@string/resolve_edit">
    <action android:name="android.intent.action.VIEW" />
    <action android:name="android.intent.action.EDIT" />
    <action android:name="com.android.notepad.action.EDIT_NOTE" />
    <action android:name="android.intent.category.DEFAULT" />
    <data android:mimeType="vnd.android.cursor.item/.vnd.google.note" />
</intent-filter>
```

Servicios

- Se ejecutan en segundo plano
- No interactúan con el usuario
- Se ejecutan en el hilo principal del proceso
- Se mantienen en ejecución mientras:
 - Se arranque
 - Tengan conexiones

Notificaciones

- Alertan al usuario de eventos
- Se envían a través del NotificationManager
- Tipos
 - Iconos persistentes
 - LEDs encendidos
 - Sonidos o vibraciones

Proveedores de Contenidos

- Los ContentProviders son objetos para:
 - Obtener datos
 - Almacenar datos
- Datos disponibles para todas las apps
- UNICO modo de compartir información entre paquetes
- Normalmente con SQLite
- Poco acople con clientes

Estados de los Procesos

- Procesos en primer plano (críticos)
- Procesos visibles (alta prioridad)
- Procesos de servicio (alta prioridad)
- Procesos en segundo plano (baja prioridad)
- Procesos vacíos (baja prioridad)
 - Cacheados para posterior uso

AndroidManifest.xml

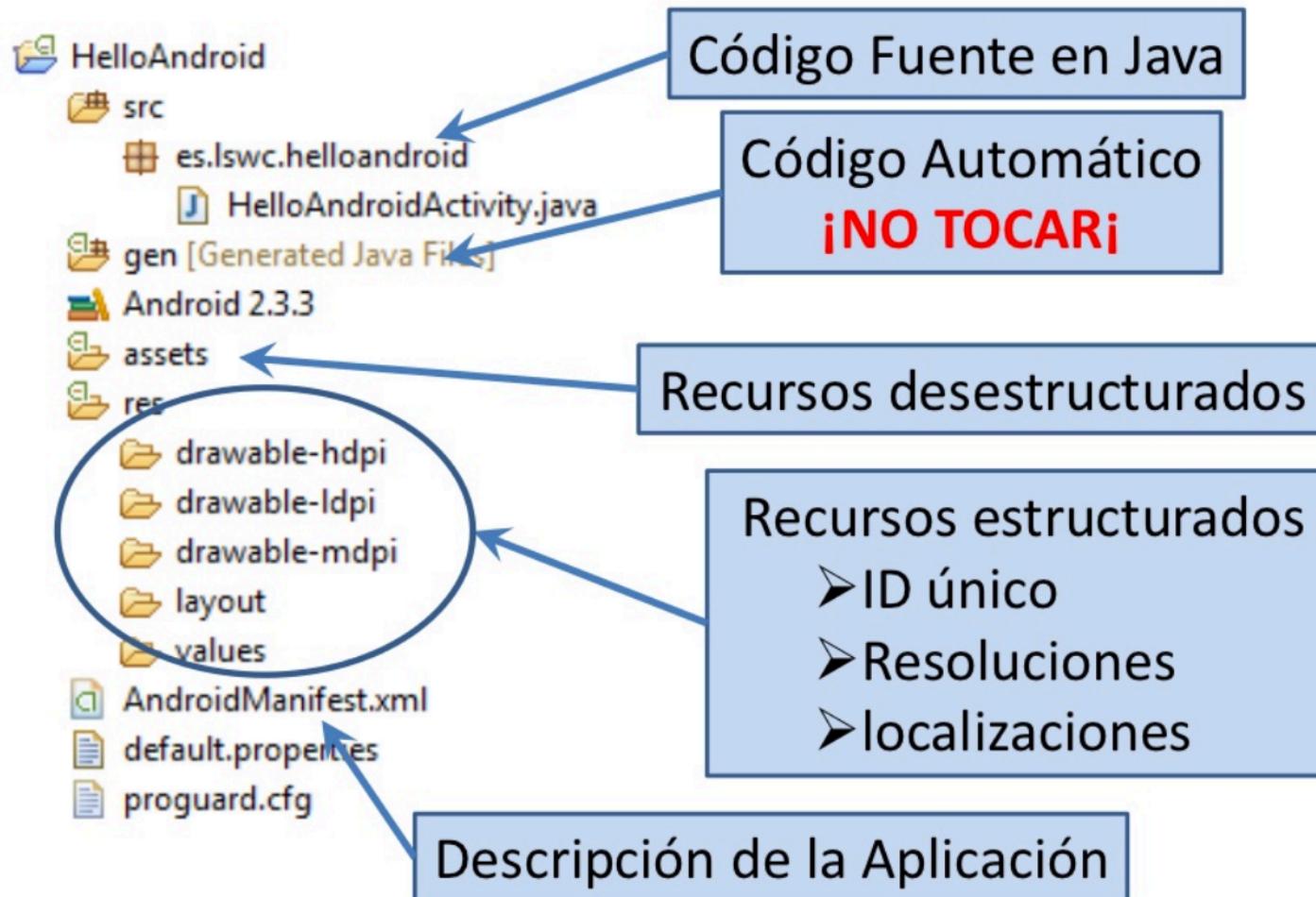
- Fichero de control que le dice al sistema qué hacer y cómo se relacionan los componentes de alto nivel
- Lo que une los **intents** con las actividades
- Especifica los permisos

AndroidManifest.xml

- Todas las aplicaciones deben tener un AndroidManifest.xml
- Almacenado en el directorio raíz
- Información esencial:
 - Nombra el paquete Java para la aplicación
 - Describe los componentes, actividades, servicios, contenedores, etc
 - Determina los procesos que alojan los componentes
 - Declara los permisos
 - Enumera las clases Instrumentation (perfil y otra información de ejecución)
 - Declara el nivel mínimo de Android API
 - Enumera las librerías que necesita la aplicación

DESARROLLO DE APLICACIONES BÁSICAS PARA MÓVILES CON ANDROID STUDIO

Organización del código en Eclipse



Resource Manager

- Elementos no relacionados con programación
 - Imágenes, vídeos, audios, textos,.....
- Almacenados en /res /asssets de la aplicación
 - /res accesible mediante la clase R
 - Compilada por el entorno android
 - Estructurado: localización, resolución pantalla,
 - R.carpeta.recurso
 - » R.layout.main
 - /assets no se generan Ids para el contenido
 - Especificar path relativo y nombre de los ficheros

Proguard.cfg

- Modifica el código para
 - Minimizar el tamaño
 - Optimizarlo
 - Ofuscarlo: Más difícil de desensamblar por ingeniería inversa
 - Resulta un .apk más pequeño y eficiente

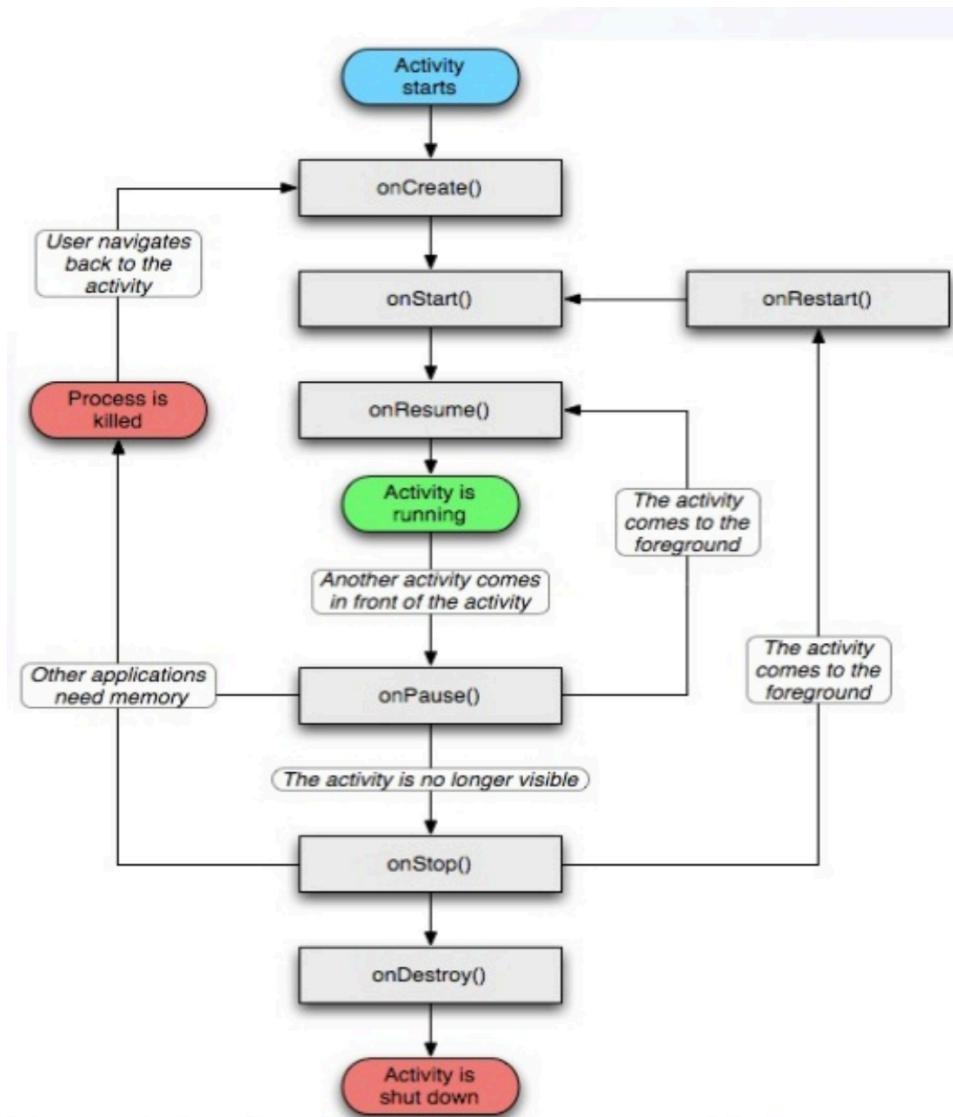
Consejos prácticos

- Prestar especial atención a los recursos (no estamos en un entorno de escritorio y las prestaciones son limitadas)
- Evitar crear objetos (en la medida de lo posible)
- Usar métodos nativos
- Mejor virtual que interfaces
- Mejor static que virtual (en la medida de lo posible, no son necesarios los objetos)
- Sin getters/setters internos
- Declarar constantes (final)
- Evitar enums

Métodos principales de las Activity

- Void onCreate(): se llama al crearse la actividad o al restaurarse si esta estaba pausada
- Void onStart(): se hace la actividad visible al usuario
- Void onRestart(): como la anterior, aunque con la salvedad de que la actividad proviene del background (fondo de pantalla)
- Void onResume(): la actividad viene de estar parcialmente escondida a pasar a estar delante
- Void onPause(): otra actividad se pone al frente aunque la actual está visible
- Void onStop(): la actividad deja de estar visible para el usuario
- Void onDestroy(): para cerrar la actividad

Ciclo de vida de las Activity



Ciclo de vida de los Services

- Se pueden utilizar de dos formas:
 - Context.startService(): se ejecutará el servicio y se queda a la espera
 - Context.bindService(): o bien una actividad, otro servicio o cualquier otro cliente conecta con el servicio [si este no existe lo crea]
- Si hay un cliente conectado al servicio, hasta que no se desconecte el servicio no se puede parar

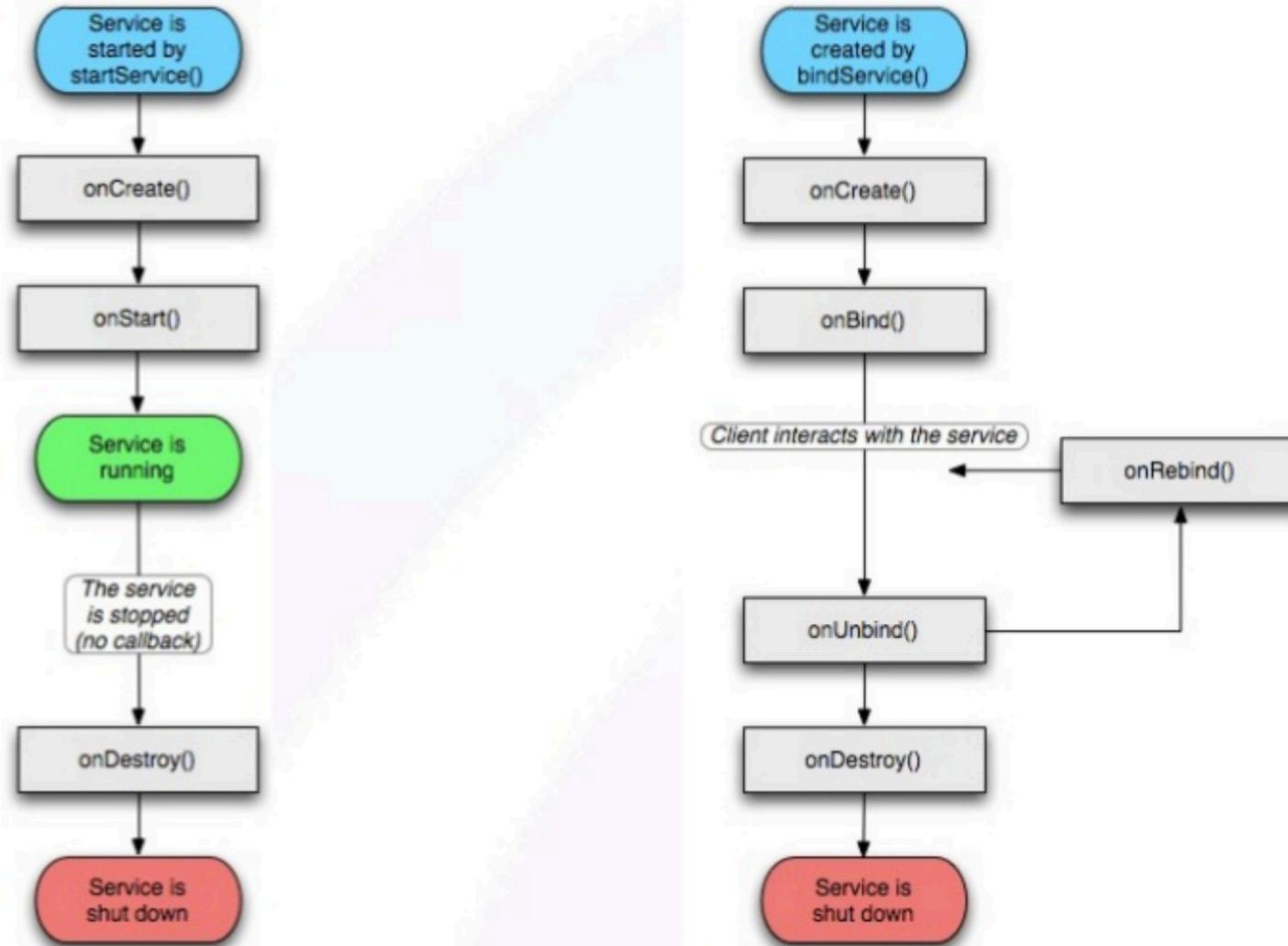
Métodos principales de los Services

- void onCreate(): para la creación del servicio
- Void onStart(Intent intent): Se inicia el servicio, el intent tiene información adicional que puede necesitar el servicio para crearse
- Void onDestroy(): para finalizar el servicio

Métodos si el Service permite conexiones

- Ibinder onBind(Intent intent): cuando un cliente inicia una conexión, se devuelve un canal de comunicaciones para la interacción
- Boolean onUnbind(Intent intent): para desconectar a un cliente de un servicio
- Void onRebind(Intent intent): cliente solicita reconectarse tras haberse desconectado

Ciclo de vida de los Services



Ciclo de vida de los componentes (Broadcast Receiver)

- Como punto de entrada principal tenemos el método
 - Void onReceive(Context curContext, Intent broadcastMsg)
- El Broadcast Receiver estará activo mientras se esté ejecutando ese método (después pasa a inactivo)
- NOTA: el método onReceive debe consumir poco tiempo (sino el sistema automáticamente lo marcará como inactivo)
- Si es necesario más tiempo para procesar el método, se recomienda lanzar un Service para realizar el trabajo

Ejemplo completo de cómo crear una aplicación en Android (Hello World!!!)



Ejemplo completo de cómo crear una aplicación en Android (Hello World!!!)

- El SDK contiene un emulador para testear las aplicaciones (ojo emula no simula!!)
- Emula tanto hardware como software
- El emulador soporta configuraciones AVD(Android Virtual Devices) para el testeo sobre diferentes plataformas de Android
- Entre las limitaciones que presenta no puede enviar ni recibir llamadas ni tampoco simular la cámara

El emulador de Android

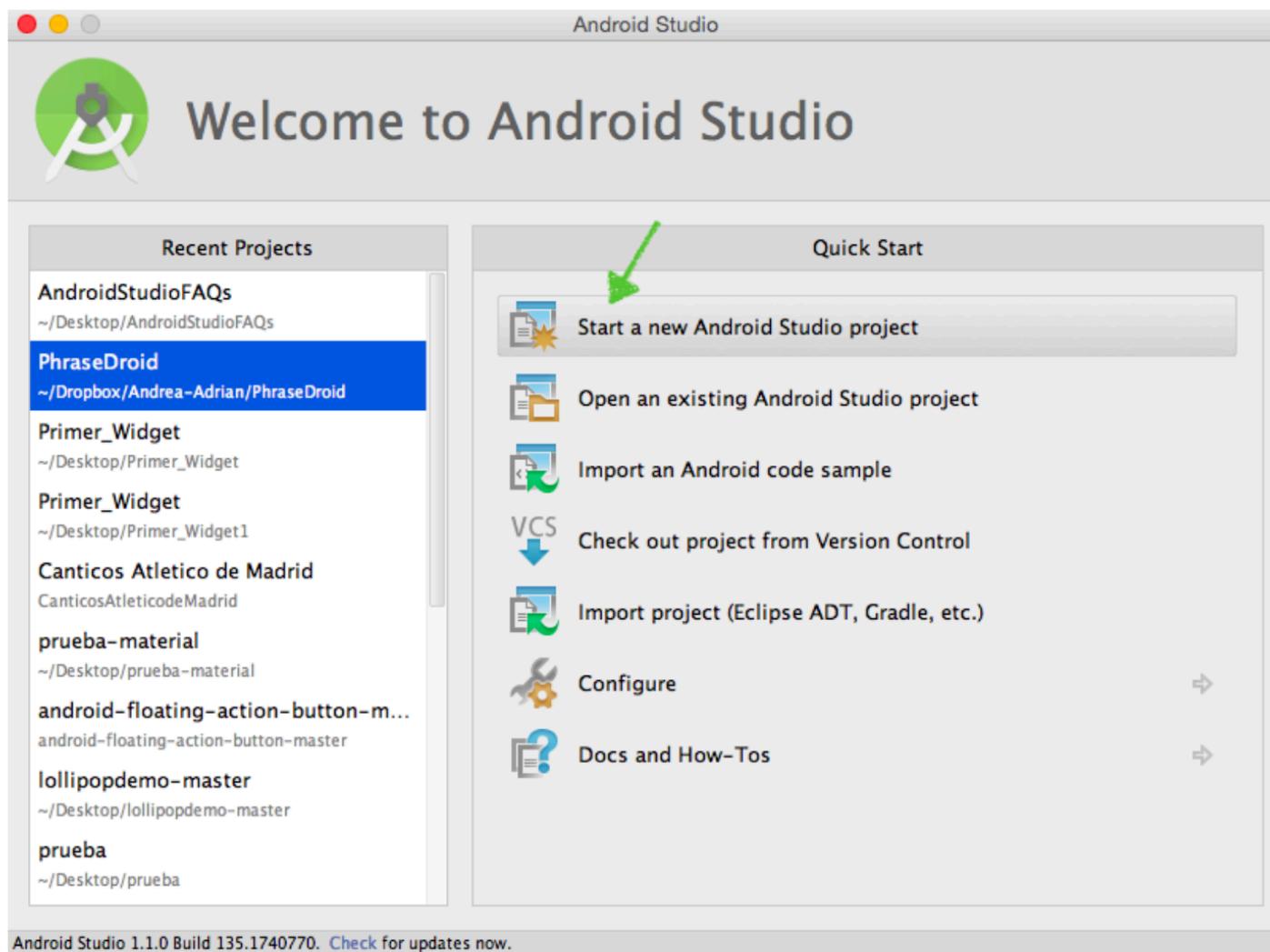
- Emulator -avd<avd_name> inicializa el emulador y carga una configuración AVD
- Se puede conectar a través de consola con:
 - telnet localhost 5554 (puerto por defecto)
- Desde la consola se pueden simular diferentes eventos de hardware (ej: geo fix 41.12)
- Para finalizar el emulador basta con cerrar ventana

Debugger

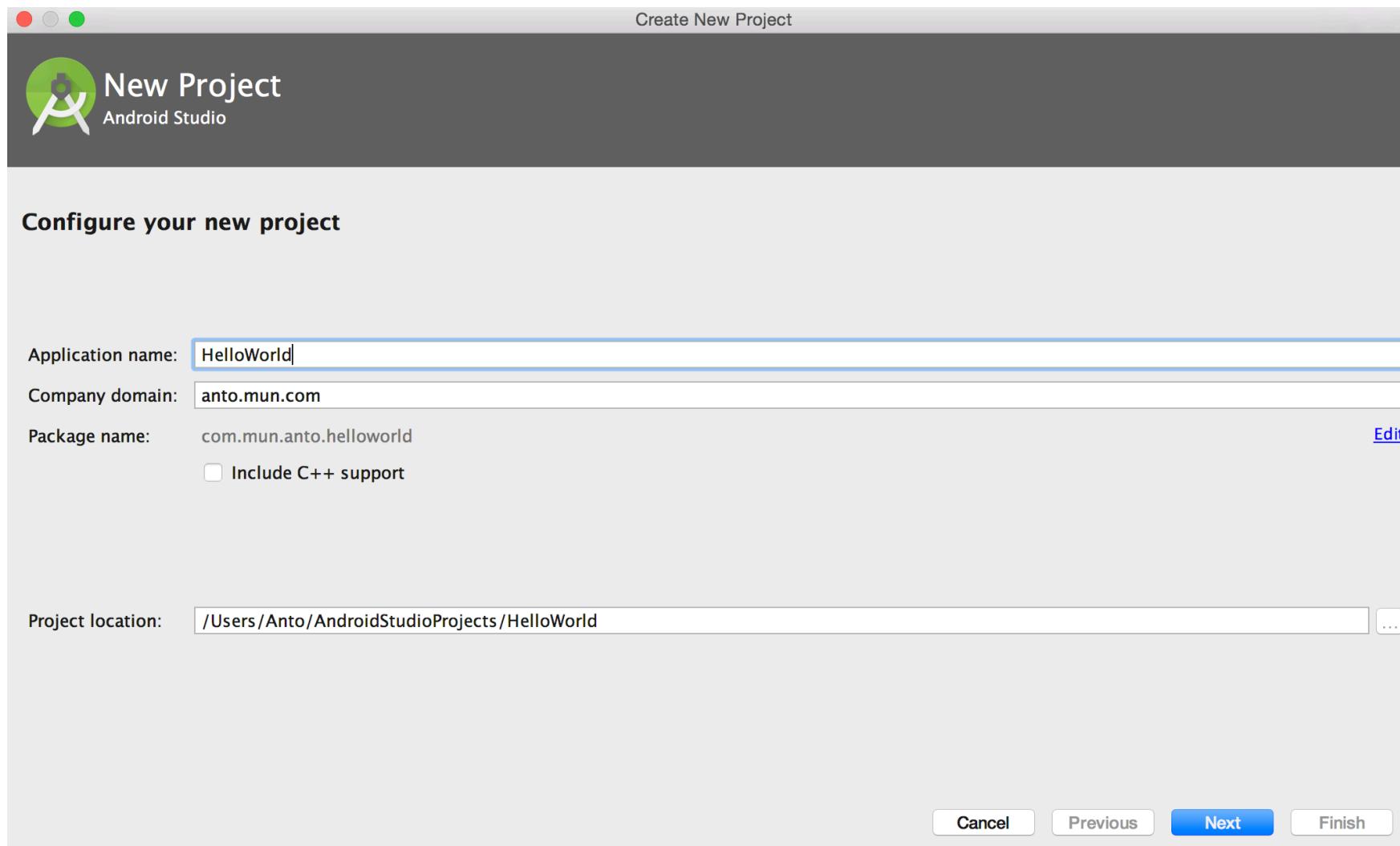
- DDMS – Dalvik Debug Monitor Server
- Incluido en el SDK
- Nos proporciona información del estado de una instancia de Android
- Se conecta al ADB (Android Debugger) y lleva a cabo la monitorización de una máquina virtual

MI PRIMERA APLICACIÓN CON ANDROID STUDIO

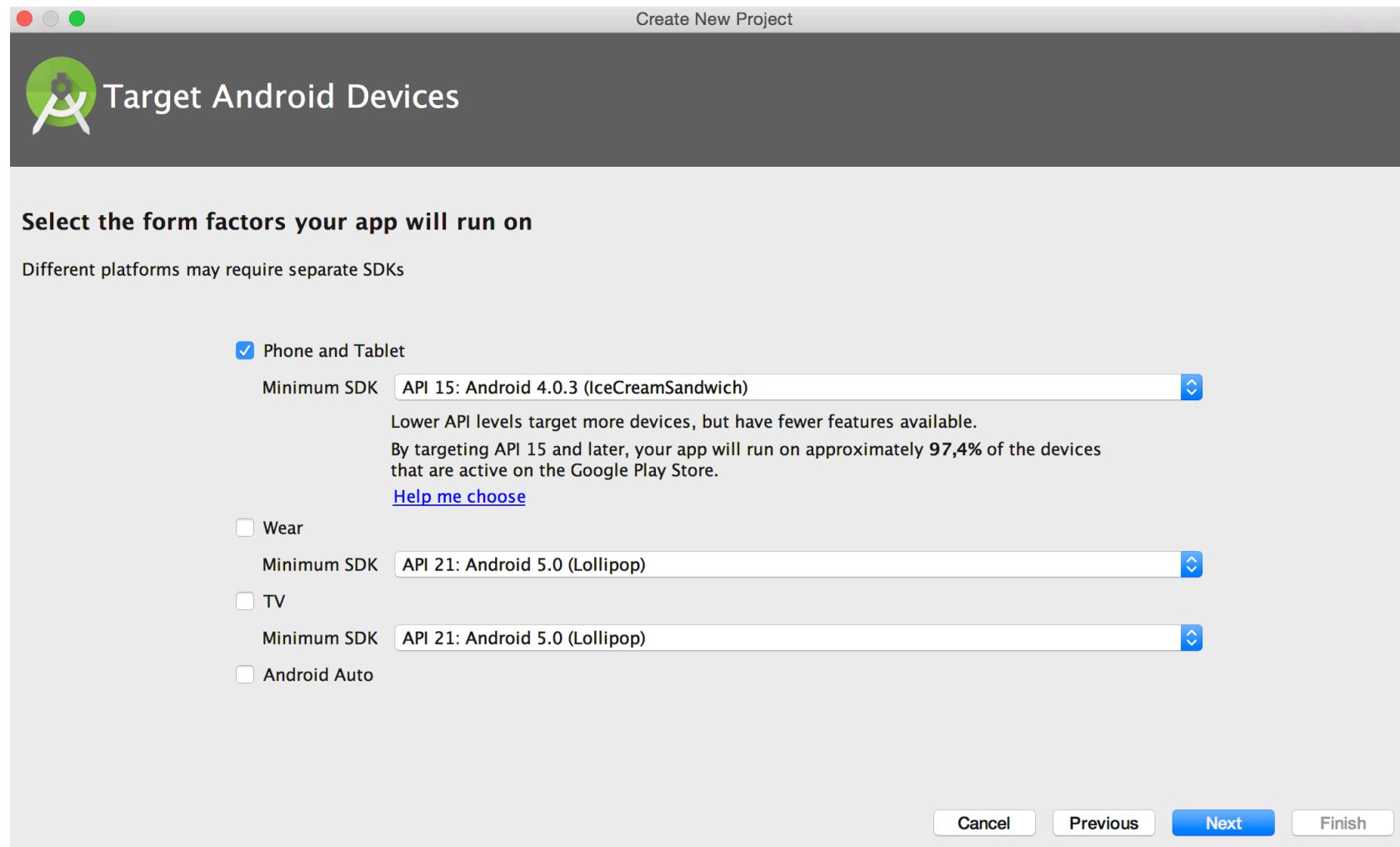
Bienvenido a Android Studio



Creamos el proyecto

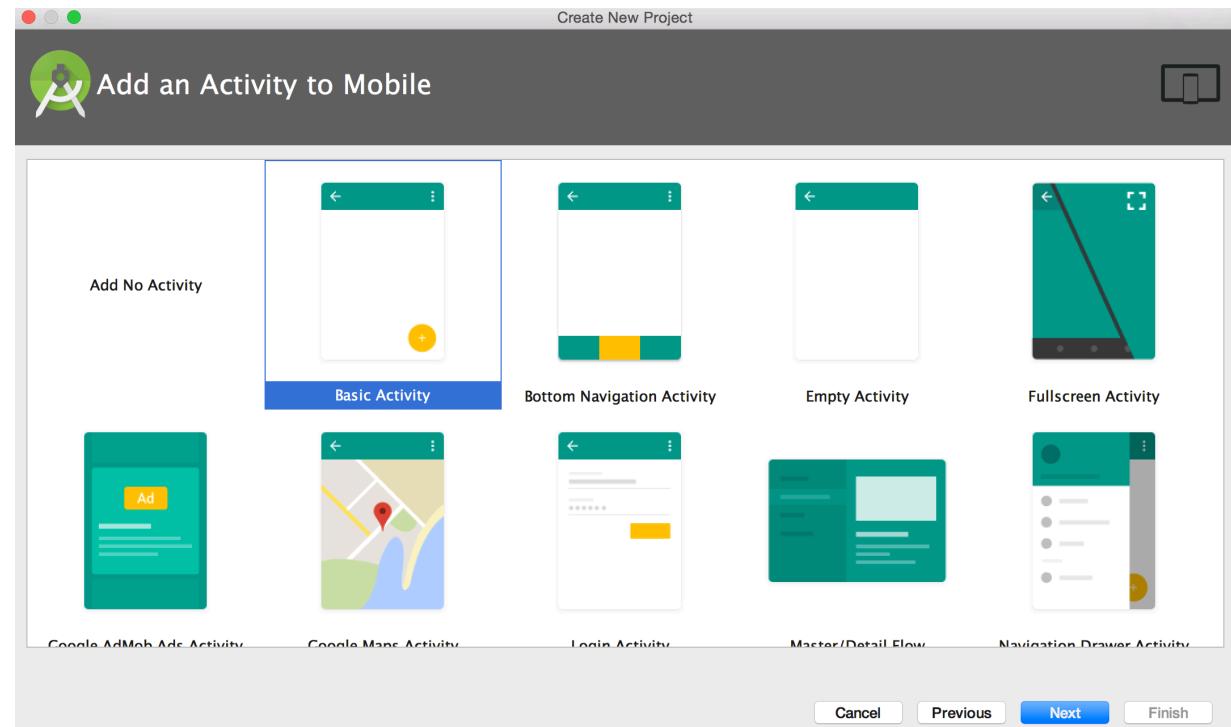


API de la aplicación

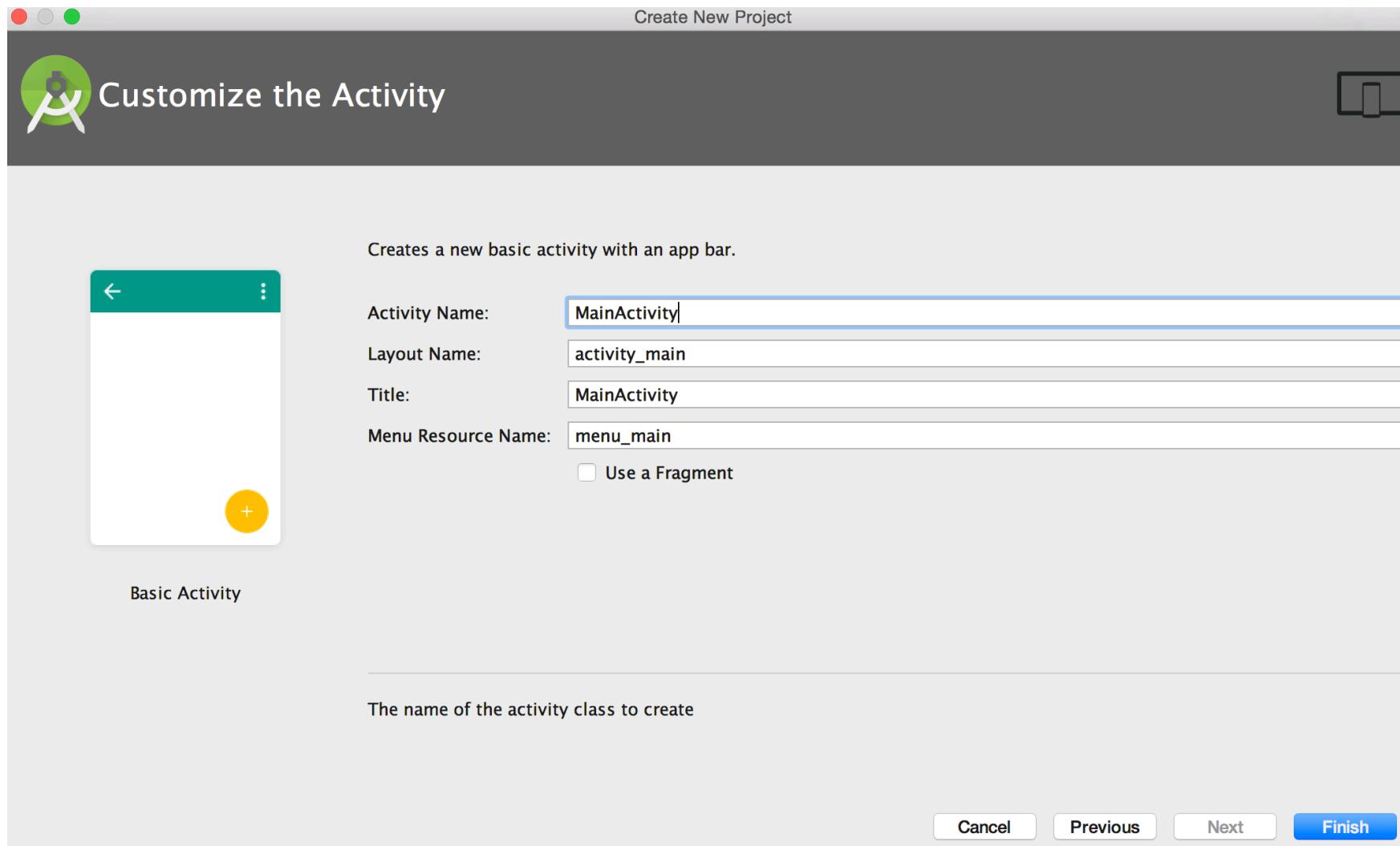


Añadimos la activity

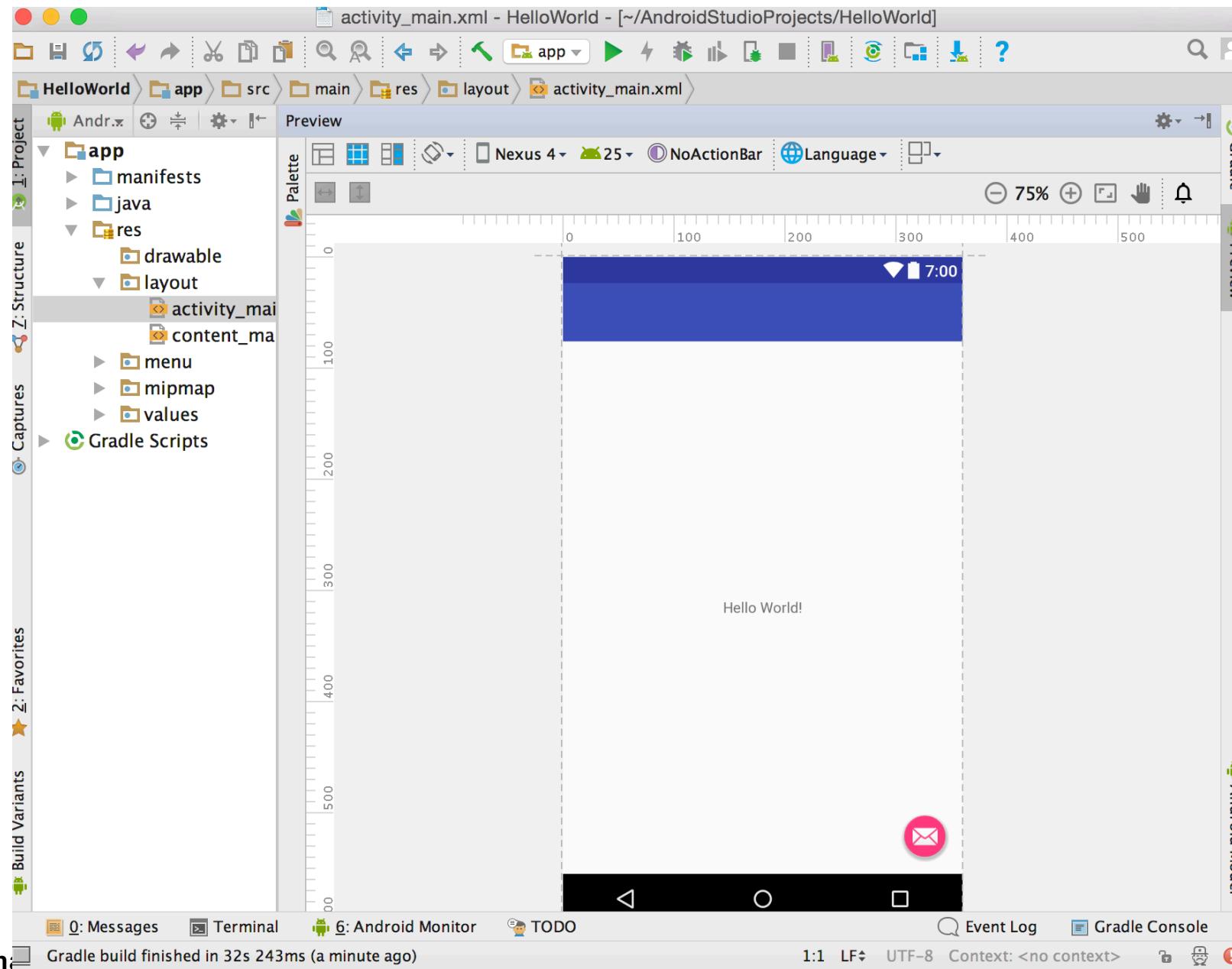
- En esta ventana elegiremos el primer layout de nuestra aplicación
- Elegimos la primera, que es la estándar para crear un layout básico fácilmente editable (podría aparecer como blank activity)



Main Activity



Primer layout (.xml)



Main en java

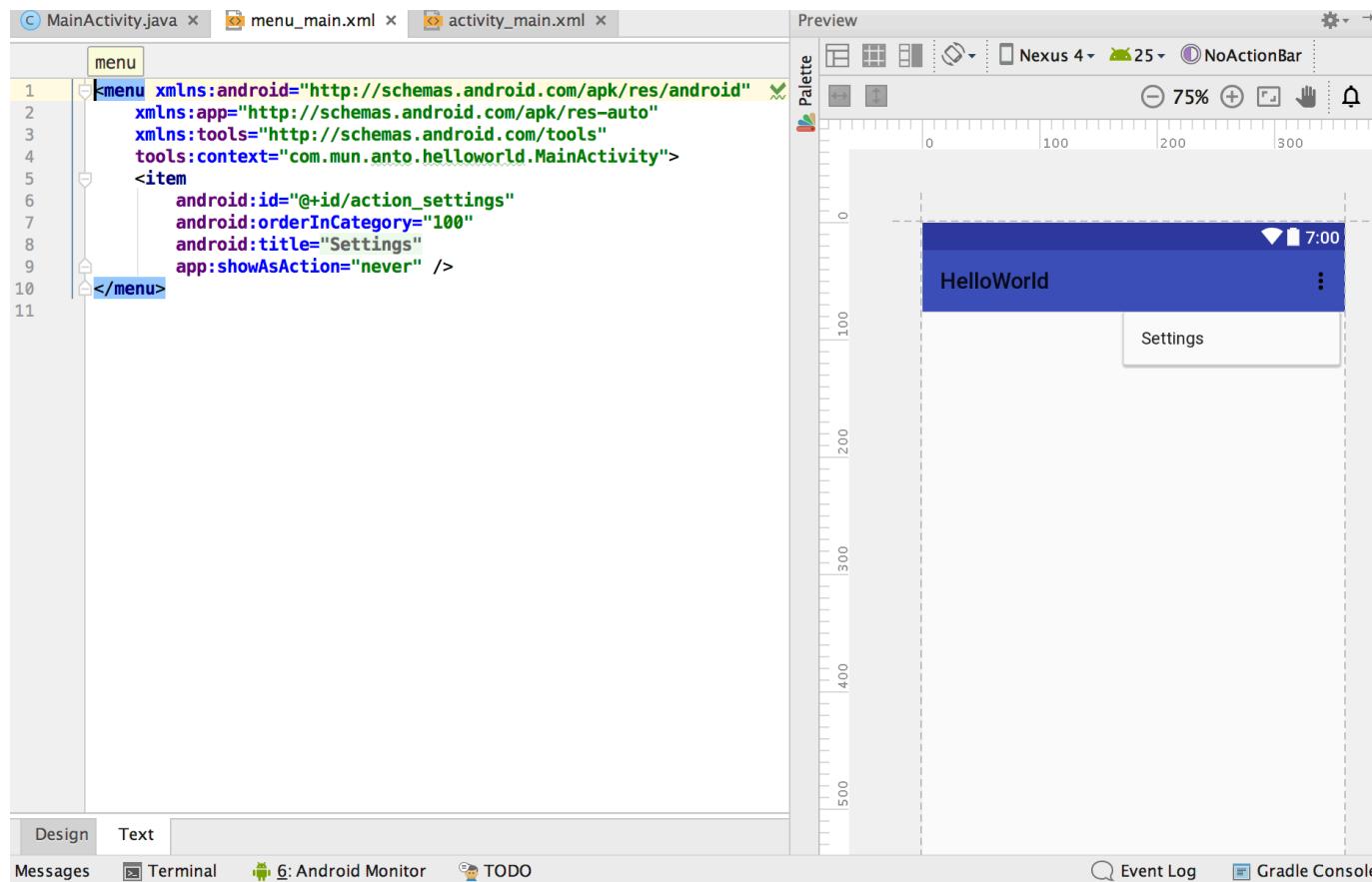
El primer código en Java de la aplicación es este **MainActivity**, que definirá el comportamiento de nuestra aplicación.

Se crean por defecto:

- **onCreate**: cargamos el layout anterior, además de otras funciones
- **onCreateOptionsMenu**: cargamos el menú de nuestra aplicación (action bar)
- **onOptionsItemSelected**: definimos el comportamiento de los botones del menú en el action bar

```
1 package com.mun.anto.helloworld;
2
3 import ...
4
5 public class MainActivity extends AppCompatActivity {
6
7     @Override
8     protected void onCreate(Bundle savedInstanceState) {
9         super.onCreate(savedInstanceState);
10        setContentView(R.layout.activity_main);
11        Toolbar toolbar = (Toolbar) findViewById(R.id.toolbar);
12        setSupportActionBar(toolbar);
13
14        FloatingActionButton fab = (FloatingActionButton) findViewById(R.id.fab);
15        fab.setOnClickListener((view) -> {
16            Snackbar.make(view, "Replace with your own action", Snackbar.LENGTH_LONG)
17                .setAction("Action", null).show();
18        });
19
20    }
21
22    @Override
23    public boolean onCreateOptionsMenu(Menu menu) {
24        // Inflate the menu; this adds items to the action bar if it is present.
25        getMenuInflater().inflate(R.menu.menu_main, menu);
26        return true;
27    }
28
29    @Override
30    public boolean onOptionsItemSelected(MenuItem item) {
31        // Handle action bar item clicks here. The action bar will
32        // automatically handle clicks on the Home/Up button, so long
33        // as you specify a parent activity in AndroidManifest.xml.
34        int id = item.getItemId();
35
36        //noinspection SimplifiableIfStatement
37        if (id == R.id.action_settings) {
38            return true;
39        }
40
41        return super.onOptionsItemSelected(item);
42    }
43
44}
```

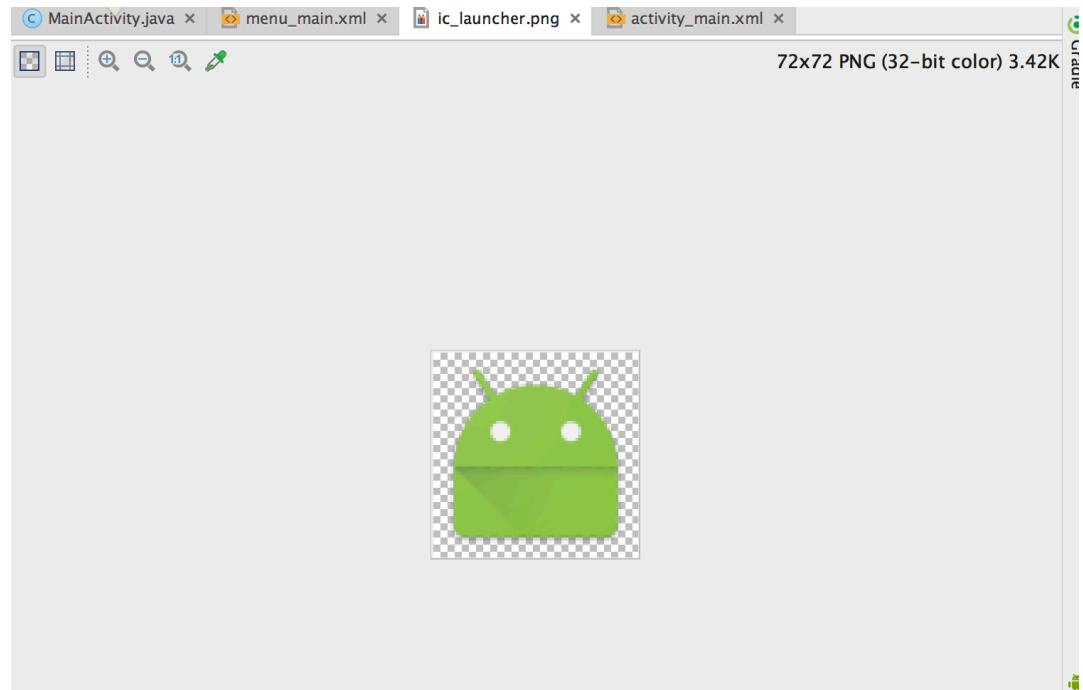
Menú por defecto



Icono de la aplicación

ic_launcher.png por defecto el android verde. Totalmente customizable

Es importante que sea un archivo .png y con un **nombre sin caracteres extraños o mayúsculas**, sino nos saltará un error



Run 'app'

- Tras crear la aplicación, ya podemos probarla en nuestro terminal Android
- O bien podemos probarla en un emulador del propio Android Studio
- Para ejecutar la aplicación, tendremos que hacer clic en “Run”, el icono verde de Play
- Si todo ha ido bien se abrirá en nuestro terminal (que debemos conectar por USB para que reconozca)
- Además de **activar la depuración USB y permitir el reconocimiento de huella digital** para que ambos dispositivos se vinculen
- Aparecerá en la siguiente lista para elegir, sino podemos optar por el emulador (clic en **Launch emulator** y crear un nuevo device)