UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escola Tècnica Superior d'Enginyeria Informàtica

3. Ingeniería inversa de aplicaciones móviles Android

Ciberseguridad en Dispositivos móviles
DISCA – ETS de Ingeniería informática (UPV)

Indice

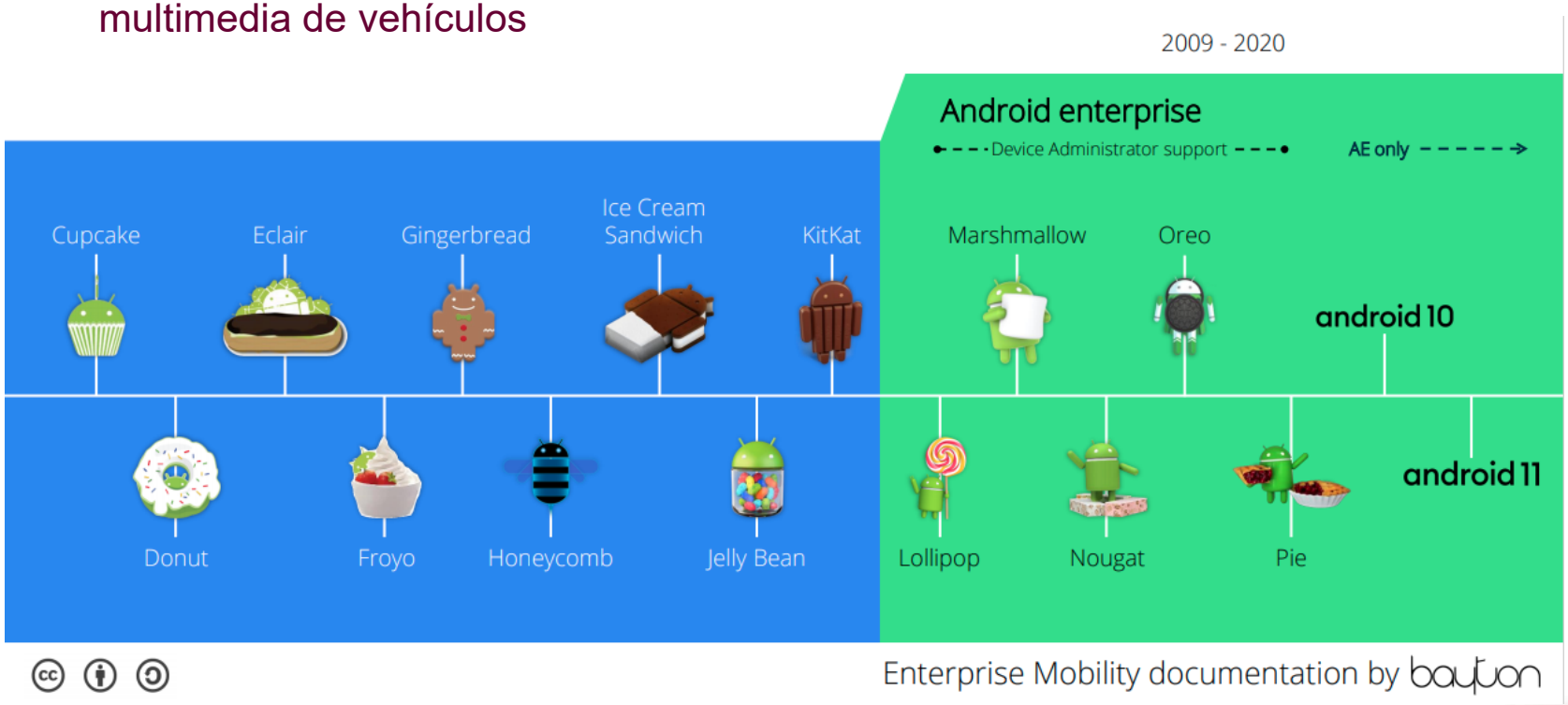
- ¿Qué es Android? Conceptos básicos
- Anatomía de una app Android sencilla
- Herramientas para el desarrollo y emulación de apps Android
- Reversing de apks
- Smali y parcheado de apks

Indice

- **¿Qué es Android? Conceptos básicos**
- Anatomía de una app Android sencilla
- Herramientas para el desarrollo y emulación de apps Android
- Reversing de apks
- Smali y parcheado de apks

¿Qué es Android?

- Sistema operativo multiusuario basado en el kernel de Linux que está diseñado para dispositivos móviles con pantalla táctil
  - Inicialmente surgió de un proyecto liderado por Google, pero promovido por la *Open Handset Alliance que* actualmente cuenta ya con 84 miembros
  - Actualmente usado en teléfonos y relojes inteligentes, tablets, televisores y sistemas multimedia de vehículos



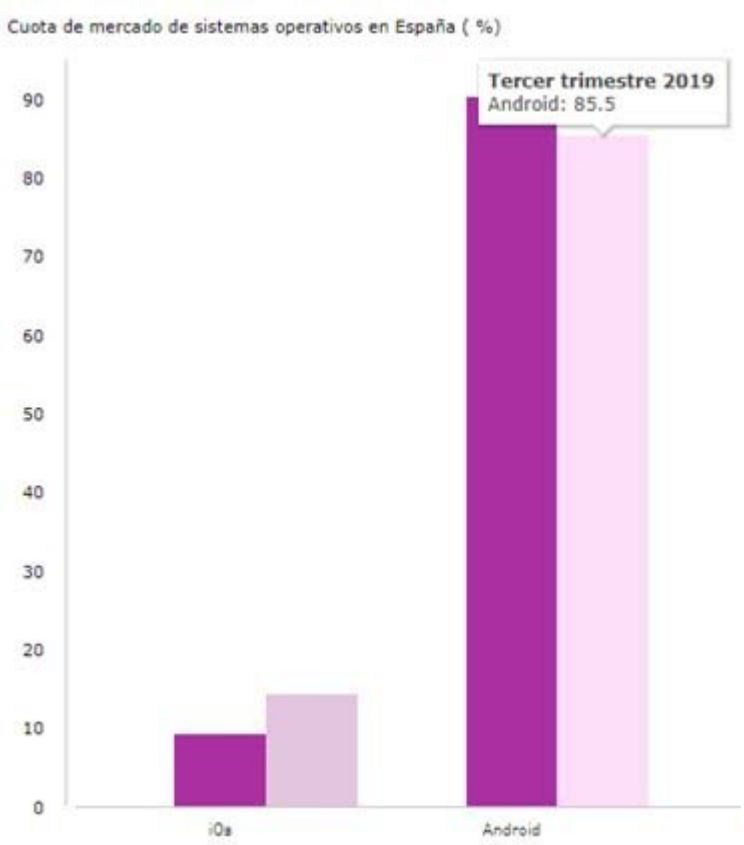
Fragmentación: Plataforma y mercado

Android Platform/API Version Distribution

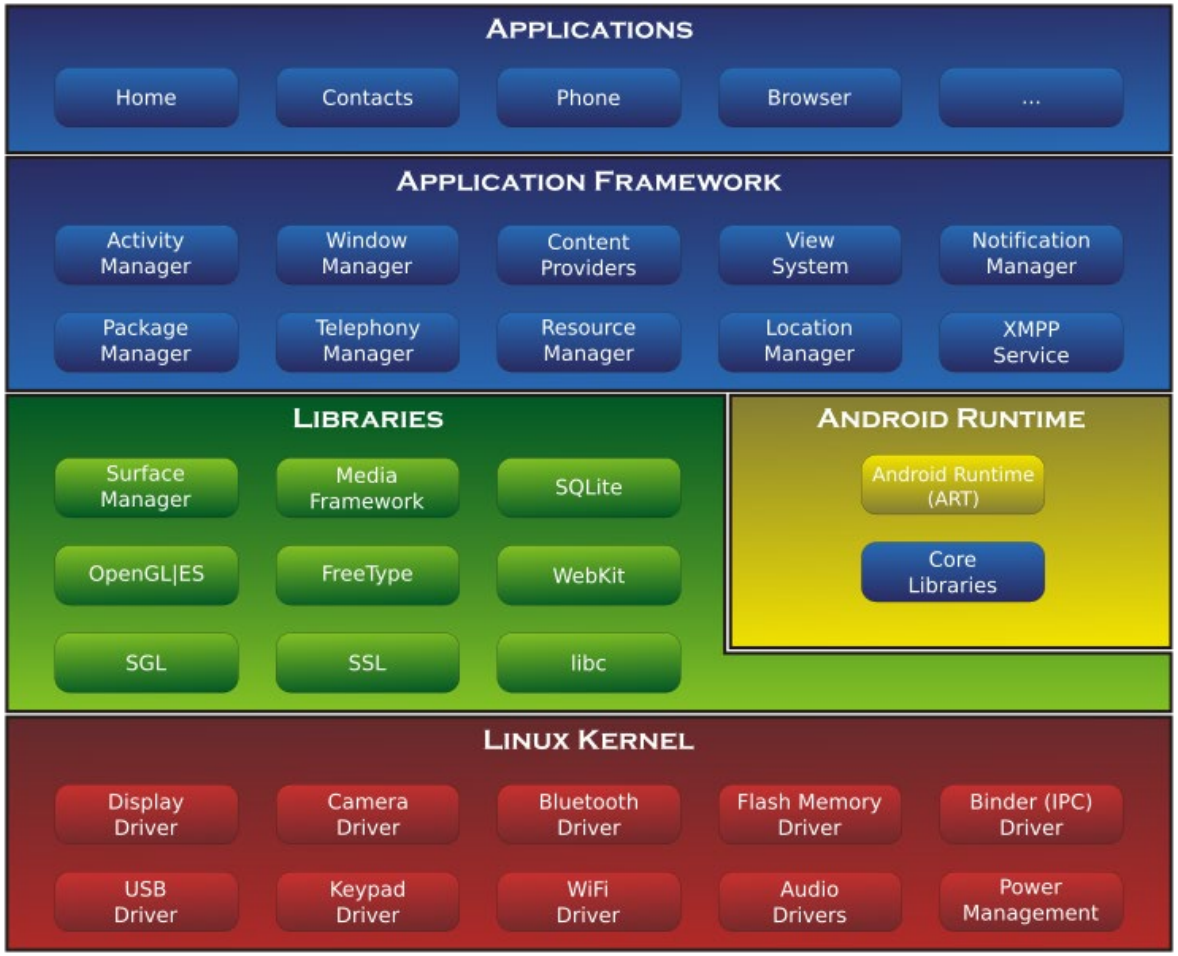
ANDROID PLATFORM VERSION	API LEVEL	CUMULATIVE DISTRIBUTION
4.0 Ice Cream Sandwich	15	
4.1 Jelly Bean	16	99,8%
4.2 Jelly Bean	17	99,2%
4.3 Jelly Bean	18	98,4%
4.4 KitKat	19	98,1%
5.0 Lollipop	21	94,1%
5.1 Lollipop	22	92,3%
6.0 Marshmallow	23	84,9%
7.0 Nougat	24	73,7%
7.1 Nougat	25	66,2%
8.0 Oreo	26	60,8%
8.1 Oreo	27	53,5%
9.0 Pie	28	39,5%
10. Android 10	29	8,2%

Minimum SDK: API 30: Android 11.0 (R)

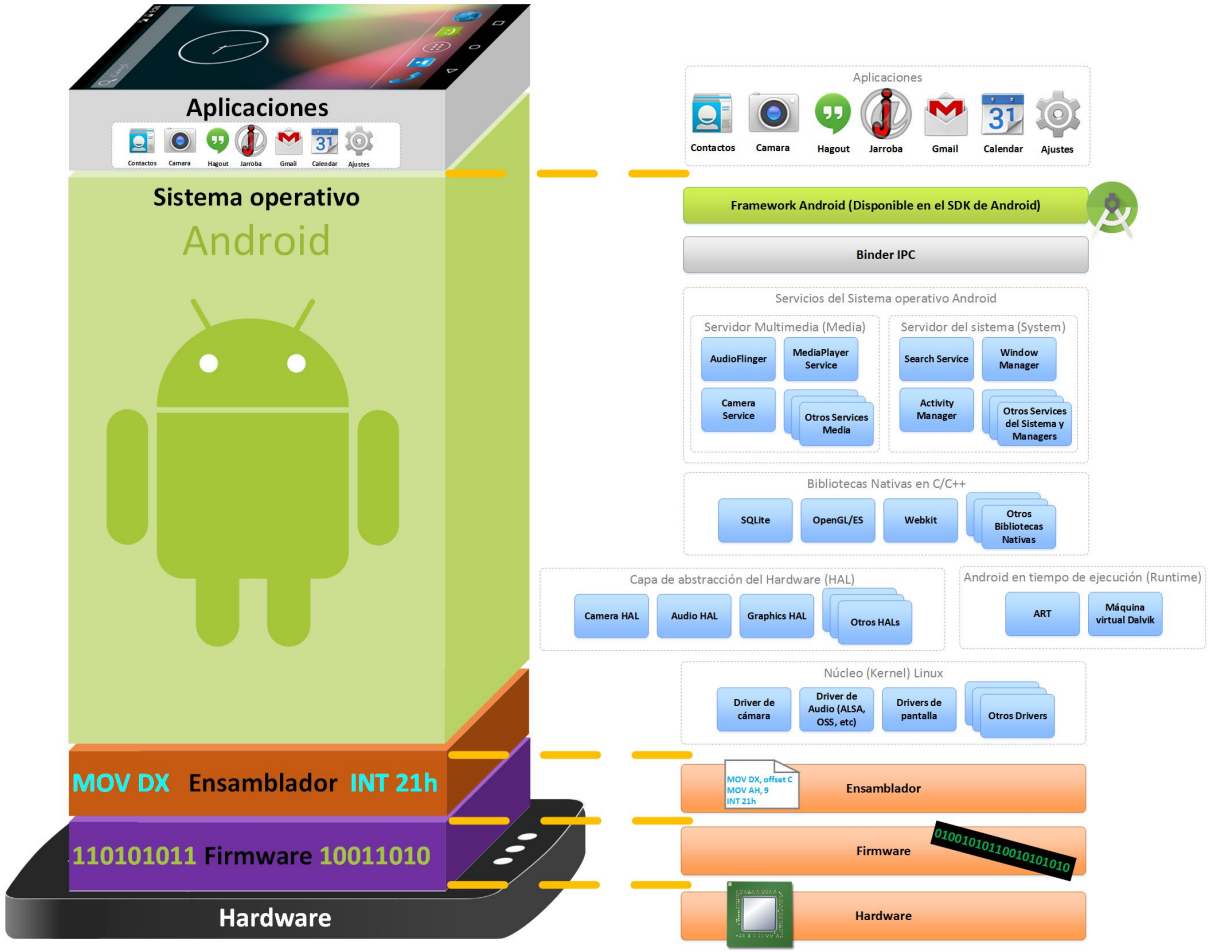
Your app will run on < **1%** of devices.



Componentes del sistema



Detalle


Aplicaciones
Contactos
Camara
Hagout
Jarroba
Gmail
Calendar
Ajustes
Sistema operativo
Android
MOV DX Ensamblador INT 21h
110101011 Firmware 10011010
Hardware
Aplicaciones
Framework Android (Disponible en el SDK de Android)
Binder IPC
Servicios del Sistema operativo Android
Servidor Multimedia (Media)
AudioFlinger
MediaPlayer Service
Camera Service
Otros Services Media
Servidor del sistema (System)
Search Service
Window Manager
Activity Manager
Otros Services del Sistema y Managers
Bibliotecas Nativas en C/C++
SQLite
OpenGL/ES
Webkit
Otros Bibliotecas Nativas
Capa de abstracción del Hardware (HAL)
Camera HAL
Audio HAL
Graphics HAL
Otros HALs
Android en tiempo de ejecución (Runtime)
ART
Máquina virtual Dalvik
Núcleo (Kernel) Linux
Driver de cámara
Driver de Audio (ALSA, OSS, etc)
Drivers de pantalla
Otros Drivers
MOV DX, offset C
MOV AH, 9
INT 21h
Ensamblador
Firmware
0100101011001010101
Hardware

Principio de mínimo privilegio

- Cada aplicación sólo tiene acceso solo a los componentes que necesita para llevar a cabo su trabajo
  - El sistema le asigna a cada aplicación un ID de usuario de Linux único
  - El sistema establece permisos para todos los archivos en una aplicación de modo que solo el ID de usuario asignado a esa aplicación pueda acceder a ellos.
  - Cada proceso tiene su propia máquina virtual (VM), por lo que el código de una aplicación se ejecuta de forma independiente de otras aplicaciones.
  - Cada aplicación ejecuta su propio proceso de Linux. El sistema Android inicia el proceso cuando se requiere la ejecución de alguno de los componentes de la aplicación y, luego, lo cierra cuando el proceso ya no es necesario o cuando el sistema debe recuperar memoria para otras aplicaciones.

Componentes de una app

- Cada componente es un punto de entrada por el que el sistema o un usuario ingresan a una aplicación
- Las apps Android tienen 4 tipos de componentes:
  - Actividades
  - Servicios
  - Receptores de emisiones (Broadcast Receivers)
  - Proveedores de contenidos (Content Providers)
- Cada componente tiene un fin específico y un ciclo de vida característico.

Actividad

- Una actividad es el punto de entrada de interacción con el usuario
- Representa una pantalla individual con una interfaz de usuario
  - Realizar un seguimiento de lo interesa al usuario (lo que está en pantalla) para garantizar que el sistema siga ejecutando el proceso que aloja la actividad
  - Saber que los procesos usados con anterioridad contienen elementos a los que el usuario puede regresar (actividades detenidas) y, en consecuencia, priorizar más esos procesos que otros
  - Ayudar a la aplicación a controlar la finalización de su proceso para que el usuario pueda regresar a las actividades con el estado anterior restaurado
  - Permitir que las aplicaciones implementen flujos de usuarios entre sí y que el sistema los coordine (el ejemplo más común es compartir)
- Las actividades se implementan como subclases de la clase ***Activity***
- Más información
  https://developer.android.com/reference/android/app/Activity

Servicios

- Un servicio es un componente que se ejecuta en segundo plano
  - realizar operaciones de ejecución prolongada. Ej.: reproducción de música (el usuario es consciente)
  - realizar tareas de procesos remotos
    Ej. Sincronización de datos (el usuario no lo percibe)
- Un servicio no proporciona una interfaz de usuario
- Un servicio se implementa como una subclase de ***Service***
- Más información
  https://developer.android.com/reference/android/app/Service

Broadcast Receivers

- Los *receptores de emisión* permiten que el sistema entregue eventos a la aplicación fuera de un flujo de usuarios habitual → las apps pueden responder a anuncios de emisión del sistema u otras apps
  - Ejemplo: programación de una alarma futura
- Los receptores de emisión no exhiben una interfaz de usuario, aunque pueden crear notificaciones en la barra de estado que alerten al usuario cuando se produzca un evento de emisión
- Un receptor de emisión se implementa como una subclase de ***BroadcastReceiver*** y cada receptor de emisión se entrega por medio de un objeto de tipo *Intent*
- Más información
  https://developer.android.com/reference/android/content/BroadcastReceiver

Content Provider

- Un proveedor de contenido administra un conjunto compartido de datos de la aplicación que se pueden almacenar en
  - el sistema de archivos, la web
  - una base de datos SQLite
  - cualquier otra ubicación de almacenamiento persistente a la que tenga acceso una app
- A través del proveedor de contenido, otras aplicaciones pueden consultar o modificar los datos si el proveedor de contenido lo permite
- Un proveedor de contenido se implementa como una subclase de ***ContentProvider*** y debe implementar un conjunto estándar de API que permitan a otras aplicaciones realizar transacciones
- Más información
  https://developer.android.com/reference/android/content/ContentProvider

Interacción entre aplicaciones

- Si dos aplicaciones está firmadas con el mismo certificado
  - Pueden compartir el mismo ID de usuario de Linux para que cada una pueda acceder a los archivos de la otra
  - Pueden coordinar su ejecución dentro del mismo proceso de Linux, compartiendo así la misma VM y consumiendo menos recursos
- Una aplicación puede solicitar permiso para acceder a datos del dispositivo (contactos, mensajes, etc.), la tarjeta SD, la cámara y la conexión Bluetooth.
  - El usuario debe conceder de manera explícita estos permisos.

Activación de componentes

- Un aspecto exclusivo del diseño del sistema Android es que cualquier aplicación puede iniciar un componente de otra aplicación
- Cuando el sistema inicia un componente, inicia el proceso para esa aplicación (si es que no se está ejecutando) y crea una instancia de las clases necesarias para el componente
  - A diferencia de lo que sucede en otros sistemas, las aplicaciones de Android no tienen un solo punto de entrada (no existe la función *main()*)
  - Como cada app se ejecuta en un proceso independiente con permisos limitan el acceso a otras apps, una app no puede activar directamente componentes de otras → Lo hace el sistema Android
- Para activar un componente en otra app, e envía un mensaje al sistema que especifique tu intención (*Intent*) de iniciar un componente específico → luego, el sistema activa ese componente

Envío de Intents

- De los cuatro tipos de componentes, tres (actividades, servicios y receptores de emisión) se activan mediante mensaje asíncronos denominado intent
  - Las intents vinculan componentes entre sí durante el tiempo de ejecución
  - Son mensajeros que solicitan acciones de otros componentes
- Una intent se crea con un objeto *Intent*
  - Se inicia una actividad o asignarle una tarea nueva al pasar un Intent a **startActivity()** o **startActivityForResult()**
  - Se establece un enlace con el servicio al pasar un Intent a **bindService()**
  - Se comienza una emisión al pasar un Intent a métodos como **sendBroadcast()**, **sendOrderedBroadcast()** o **sendStickyBroadcast()**
  - Se realiza una consulta a un proveedor de contenido si llamas a **query()** en un ContentResolver
- Más información
  https://developer.android.com/reference/android/content/Intent

Intents implícitos vs explícitos

- Los Intents pueden ser **implícitos** (se elige el componente más adecuado durante la ejecución de la app) o **explícitos** (se indica el componente específico al que va dirigido el Intent)

```
//Ejemplo Intent explícito
Intent intent = new Intent(ACTION_VIEW,Uri.parse("http://www.google.com"));

//Ejemplo intent implícito
Intent intent = new Intent(this,Target.class);
//…
//Podemos enviar información a través de intent implícitos utilizando su método putExtra(…), el que recibe el intent puede obtener dicha información utilizando getIntent().getExtras()
```

El archivo de Manifiesto

- El sistema Android conoce los componentes de la app para poder gestionarlos adecuadamente a través de la información que suministra su (AndroidManifest.xml)
  - Identifica los permisos de usuario que necesita la app
  - Declara el nivel de API mínimo que requiere la app en función de las API que usa
  - Define las características HW y SW de la app (cámara, servicios de Bluetooth, pantalla multitáctil …)
  - Declara las APIS que necesita (además de las propias de Android) para ejecutarse, como por ejemplo las APIs de Google Maps
- Más información
  https://developer.android.com/guide/topics/manifest/manifest-intro

Manifest.xml

(ejemplo)

```
<?xml version="1.0" encoding="utf-8"?>
<manifest
  xmlns:android= "http://schemas.android.com/apk/res/android"
  package="com.cdm.example.MiPrimeraAppAndroid"
  android:versionCode="1"
  android:versionName="1.0" >
  <uses-sdk
    android:minSdkVersion="8"
    android:targetSdkVersion="16" />
  <application
    android:allowBackup="true"
    android:icon="@drawable/ic_launcher"
    android:label="@string/app_name"
    android:theme="@style/AppTheme" >
    <activity
      android:name=
        "com.example.myfirstandroidapp.ActividadPrincipal"
      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>
```

Esquema xml del fichero

Paquete de la app

Versión SDK

Recursos (icono, nombre y estilo)

Actividad principal
(será el componente que el sistema activará cuando hagamos click sobre el icono de la app que tendremos en la lanzadera de aplicaciones del terminal)

Recursos de una app

- Son los archivos adicionales y el contenido estático que usan las apps para su ejecución: su código, imágenes, diseños de interfaz, cadenas de caracteres, animaciones, etc.
- Organización típica:

```
MyProject/
    src/
        MyActivity.java
    res/
        drawable/
            graphic.png
        layout/
            main.xml
            info.xml
        mipmap/
            icon.png
        values/
            strings.xml
```

- Más información https://developer.android.com/guide/topics/resources/providing-resources

Indice

- ¿Qué es Android? Conceptos básicos
- **Anatomía de una app Android sencilla**
- Herramientas para el desarrollo y emulación de apps Android
- Reversing de apks
- Smali y parcheado de apks

App Android

- Capa de presentación
  - Conjunto de recursos definidos en XML: Layouts, Strings, Imágenes
  - Se generan para ellos unas estructuras de datos para nacerlos accesibles desde la lógica de negocio
- Lógica de negocio
  - Definida programáticamente utilizando código Java/Kotlin
  - Existe código nativo (de más bajo nivel para control y acceso al HW o a sus controladores) que puede desarrollarse con C/C++ → Uso de la NDK

Interfaz de una Actividad

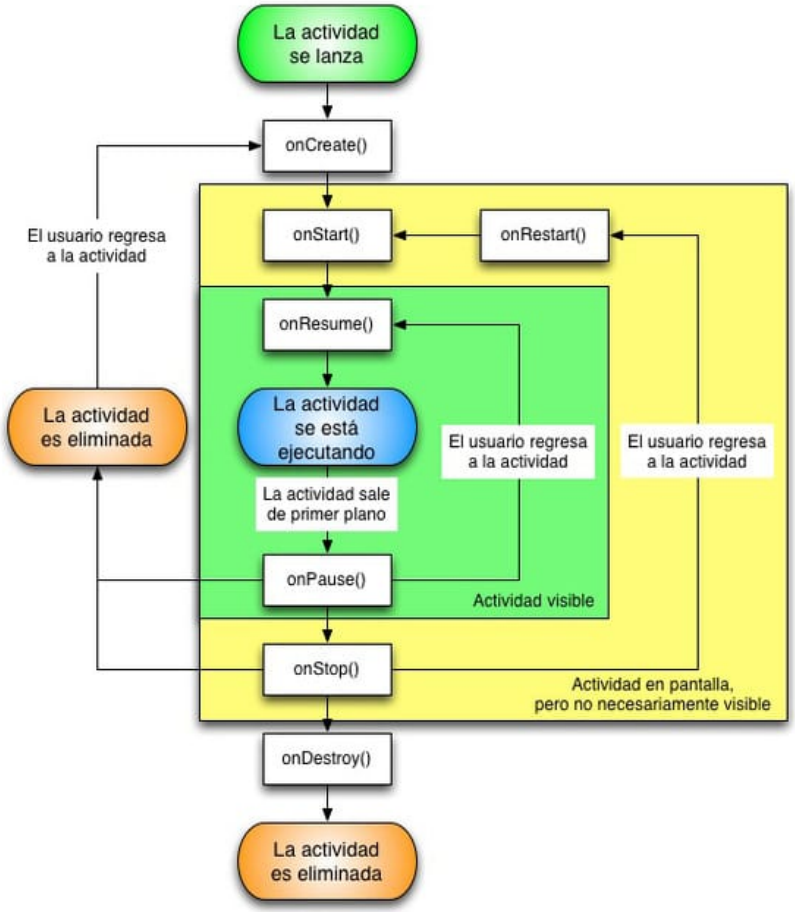
- Se define como un Layout especificado en XML
- Existen distintos tipos de layout:
  - LinearLayout, RelativeLayout, AbsoluteLayout, TableLayout, FrameLayout, ConstraintLayout

```xml
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
   android:layout_width="fill_parent"
   android:layout_height="fill_parent"
   android:orientation="vertical" >

   <TextView android:id="@+id/text"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:text="This is a TextView" />

   <Button android:id="@+id/button"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:text="This is a Button" />

   <!-- More GUI components go here  -->

</LinearLayout>
```

  - El xml se liga a una actividad en tiempo de ejecución

```java
public void onCreate(Bundle savedInstanceState) {
   super.onCreate(savedInstanceState);
   setContentView(R.layout.activity_main);
}
```

Ciclo de vida de una app Android



Fuentes de información

- Guía de desarrollo: https://developer.android.com/guide
- Laboratorios de código: https://codelabs.developers.google.com/?cat=Android
- Cursos de capacitación (tanto kotlin como java): https://developer.android.com/courses
- Guía para crear y ejecutar un primer proyecto con Android Studio (“Hello, World!” app) https://developer.android.com/training/basics/firstapp
  - Básicamente todo el código lo genera el entorno
  - Necesitamos entender varios conceptos básicos de Android para ser capaces de leer dicho código

Nuestra primera app

HolaMundo 〉 app 〉 src 〉 main 〉 res 〉 values 〉 strings.xml

Android

- app
  - manifests
    - AndroidManifest.xml
  - java
    - com.cdm.ejemplo.miprimerapp
      - HolaMundo
    - com.cdm.ejemplo.miprimerapp (androidTest)
    - com.cdm.ejemplo.miprimerapp (test)
  - java (generated)
  - res
    - drawable
      - ic_launcher_background.xml
      - ic_launcher_foreground.xml (v24)
    - layout
      - activity_hola_mundo.xml
    - mipmap
    - values
      - colors.xml
      - dimens.xml
      - strings.xml
      - themes (2)
  - res (generated)

AndroidManifest.xml | activity_hola_mundo.xml | strings.xml | HolaMundo.java

```xml
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.cdm.ejemplo.miprimerapp">

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/Theme.HolaMundo">
        <activity android:name=".HolaMundo">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

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

</manifest>
```

AndroidManifest.xml | activity_hola_mundo.xml

```java
package com.cdm.ejemplo.miprimerapp;

import ...

public class HolaMundo extends AppCompatActivity {

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

droidManifest.xml | activity_hola_mundo.xml | strings.xml | Hola

nslations for all locales in the translations editor.

```xml
<resources>
    <string name="app_name">HolaMundo</string>
    <string name="title_activity_hola_mundo">HolaMundo</string>
    <!-- Strings used for fragments for navigation -->
    <string name="saludo">¡¡¡ Hola a todos !!!</string>
</resources>
```

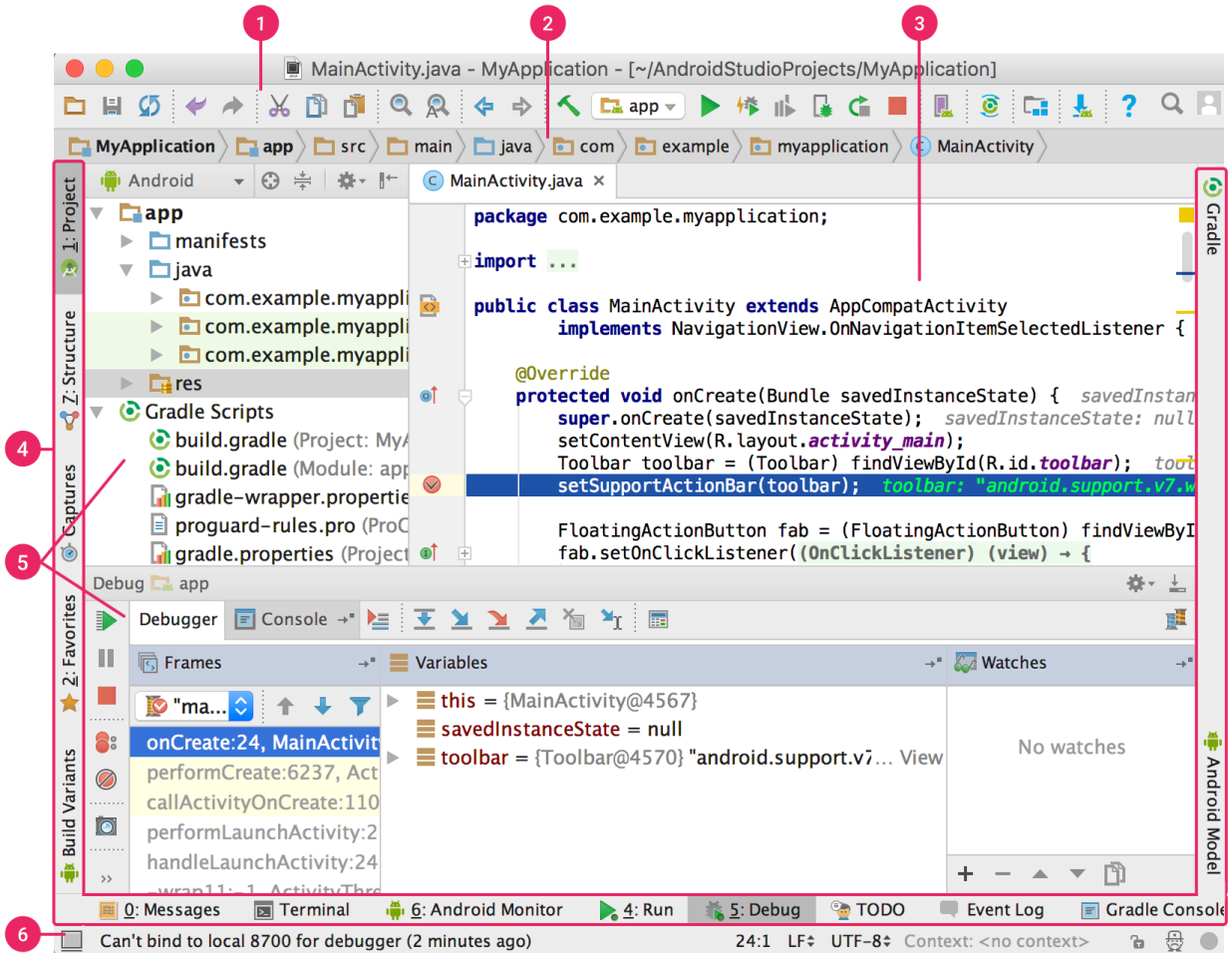
Indice

- ¿Qué es Android? Conceptos básicos
- **Herramientas para el desarrollo y emulación de aplicaciones Android**
- Depuración con ADB
- Reversing de apks
- Smali y parcheado de apks

Android Studio

- Android Studio es el entorno de desarrollo integrado (IDE) oficial para el desarrollo de apps para Android
  - Sistema de compilación basado en Gradle
  - Compatibilidad con C++ y NDK
  - Capacidades de emulación rápida para todos los dispositivos Android
  - Requiere de 2GB de HDD y de 8GB de RAM para su instalación y ejecución
  - Disponible para Windows, Mac, Linux y Chrome OS
- Descarga: https://developer.android.com/studio
- Instalación: https://developer.android.com/studio/install
- Guía de uso: https://developer.android.com/studio/intro

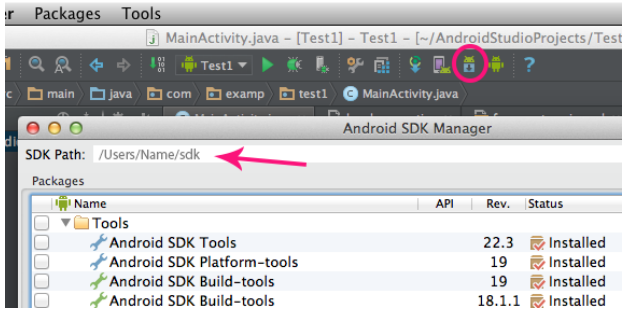
Interfaz de usuario



1. Barra de herramientas
   - Ejecución de las aplicaciones
   - Iniciar herramientas android
2. Barra de navegación
3. Ventana del editor
4. Barra de la ventana de herramientas
5. Ventanas para tareas específicas
   - Estructura y administración del proyecto
   - Control de versiones
   - Depurador
   - etc.
6. Barra de estado

Combinaciones de teclas admitidas por el IDE: https://developer.android.com/studio/intro/keyboard-shortcuts

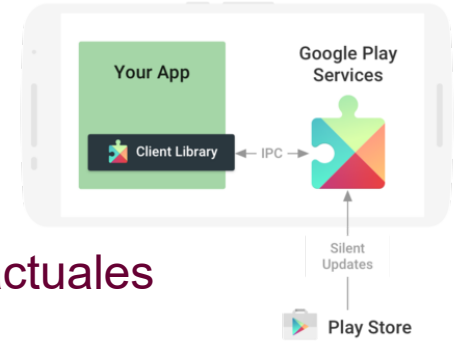
Android SDK tools



- No es necesario descargarlas por separado ya que instalan como parte de Android Studio
- … pero se pueden descargar aparte https://developer.android.com/studio (Command line tools only)
  - SDK Manager accesible desde línea de órdenes ejecutando la orden: **sdkmanager**
  - Órdenes disponibles y su uso: https://developer.android.com/studio/command-line
- Algunas de las órdenes incluidas en la SDK son esenciales a la hora de acceder a un dispositivo/emulador Android y analizar sus aplicaciones
  - Ejemplos: adb, apksigner, apkanalyzer, logcat, dumpsys, etc.
- Directorio de instalación
  - Windows: `%LOCALAPPDATA%\Android\sdk`
  - Mac: `~/Library/Android/sdk`
  - Linux: `~/Android/Sdk`

Google Play Services

- Es una librería creada por Google para que los desarrolladores facilitar el uso de las API y servicios de Google
  - Muchas veces se confunden con el propio Android al venir preinstalados en todos los móviles lanzados con Play Store
  - Definen el nexo de unión entre Android y Google y evolucionan independientemente de si lo hace o no la plataforma
  - Ofrecen a los desarrolladores un modo estandarizado y abstracto de acceder a funciones avanzadas permitiendo que móviles con versiones de Android antiguas sigan pudiendo usar funciones y aplicaciones actuales
  - Estos servicios se actualizan a través de Google Play

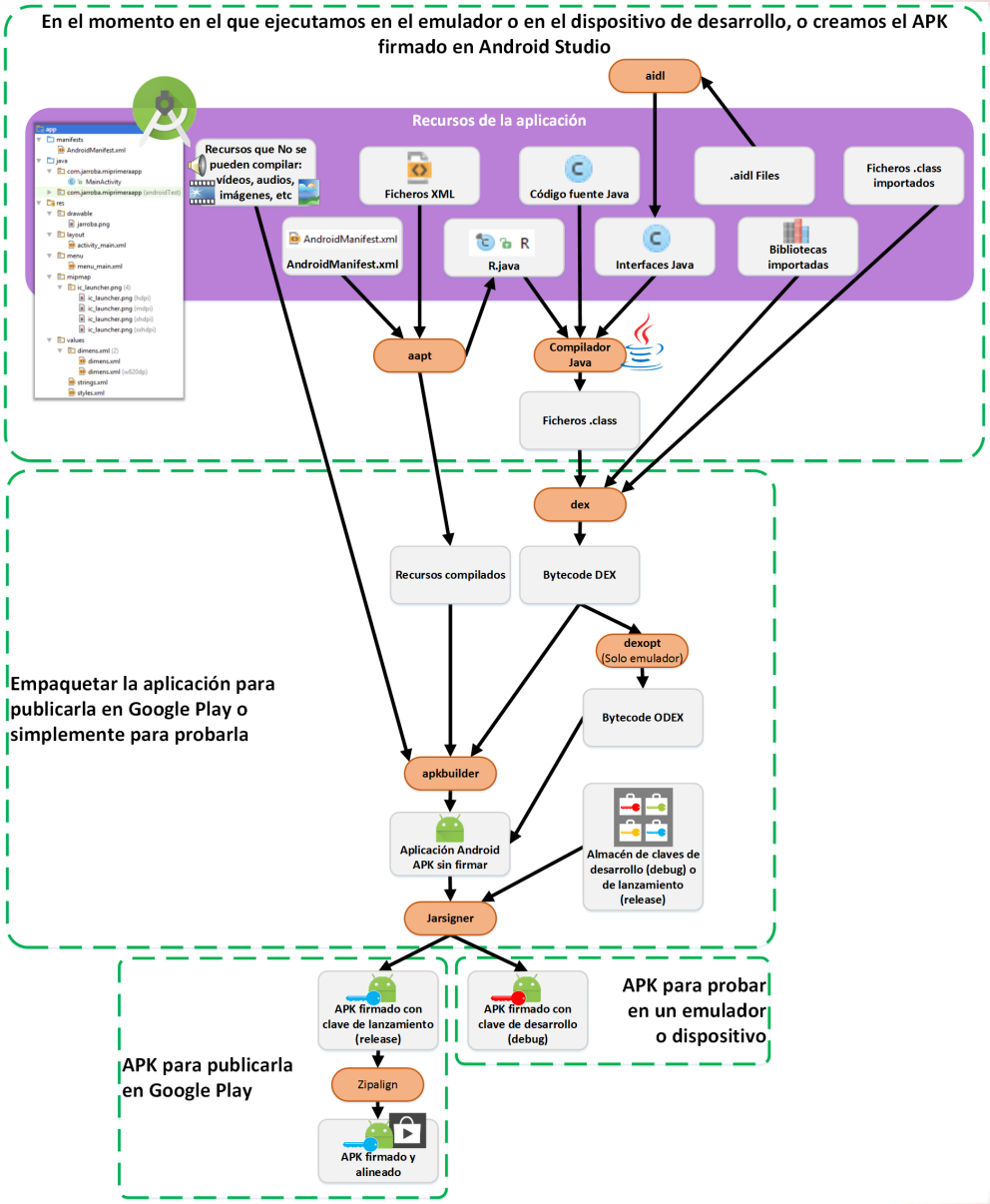


- ¿Y cuáles son estos servicios?
  - App analytics, clound messagind, authentication, storage, maps, places, sig in with Google, notifications, adwords, admob, etc.
  - Más info en: https://developers.google.com/android

Proceso de compilación

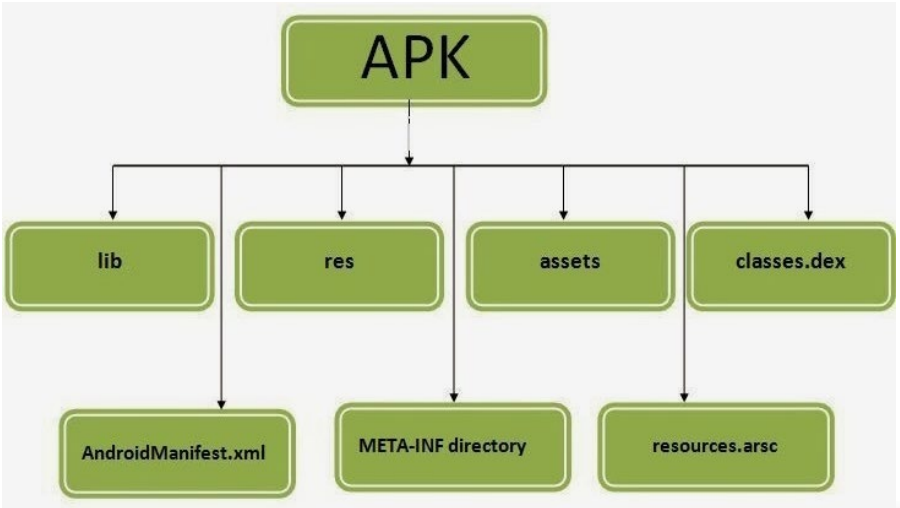
DEX → Dalvik Executable Format

ODEX → Versión optimizada de DEX



Estructura de un apk

- Las apps Android se distribuyen e instalan a través de archivos apk (Android Application Package)
- Variante de los archivos JAR, los .apk son en esencia ficheros comprimidos de tipo “zip” que contienen
  - El manifiesto de la app que describe su funcionalidad y características principales (versión de Android requerida, icono, permisos, etc.)
  - Las librerías, recursos y código compilado necesarios para ejecutar la app
  - Certificado de la aplicación, lista de recursos y resumen SHA-1 de los mismos de acuerdo a la información contenida en el manifiesto
- Se pueden intercambiar con facilidad



Fuentes de APKs

- **Repositorio oficial** de apps Android: Google Play
- Apkpure
  - Repositorio no oficial de apps Android
  - Permite encontrar apps que no están en Google Play porque
    - No cumplen algunas de las condiciones que impone ese repositorio
    - Su desarrollo ha sido abandonado
    - Son versiones antiguas de las apps disponibles
  - La instalación de las APKs requiere de la activación en el terminal del **permiso para ejecutar programas de fuentes desconocidas**, que por defecto está deshabilitado
  - https://apkpure.com
- Otras alternativas:
  - Aptoide (https://es.aptoide.com)
  - ApkMirror (https://www.apkmirror.com)
  - F-Droid: Repositorio FOSS (https://f-droid.org)

Instalación de un apk

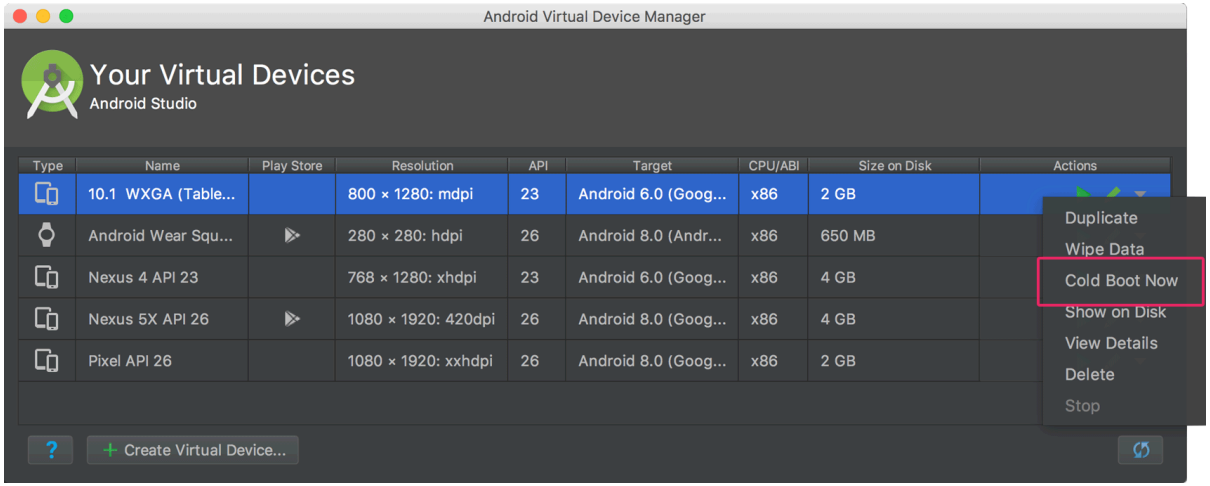
- Para ejecutar un archivo .APK bastará con abrirlo en un dispositivo Android
- Los apks sirven para instalar aplicaciones sin necesidad de pasar por Google Play

Apreciaciones de seguridad

- Los archivos APK pueden ser la puerta de entrada de amenazas al sistema (troyanos, virus, malware, ransomware, etc.)
- Aunque se publicitan libres de dichas amenazas, no se aplican los filtros de seguridad existentes en Google Play
  - https://developer.android.com/google/play/filters
- Este problema puede mitigarse utilizando herramientas de análisis específicas. Algunas pueden usarse online
  - Antivirus
    - Virus Total (https://www.virustotal.com)
    - Metadefender Cloud (https://metadefender.opswat.com)
  - Análisis híbrido: Ativirus + análisis estático
    - https://www.hybrid-analysis.com

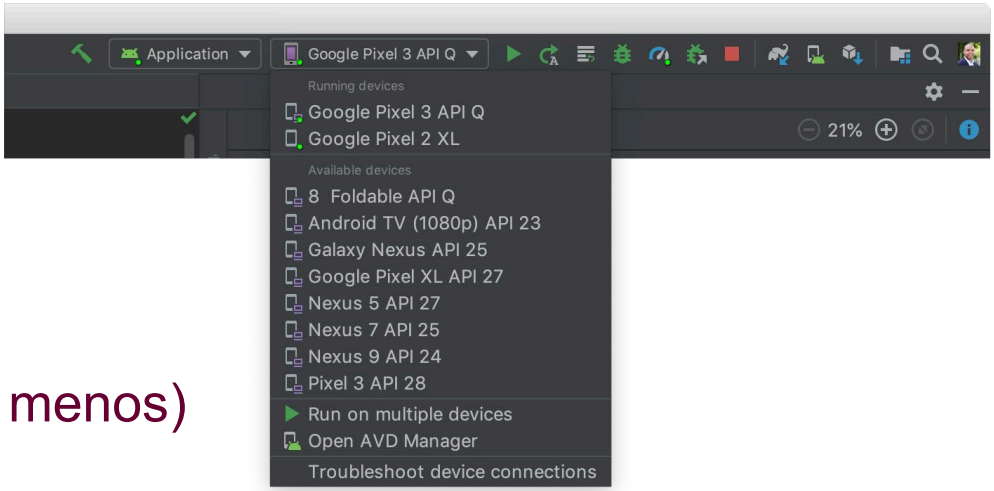
Emulación de apks (1/3)

- El emulador se instaló con las SDK tools de Android que vienen con el Android Studio https://developer.android.com/studio/run/emulator
- La gestión de los emuladores se realiza a través del AVD (Android Virtual Device) Manager

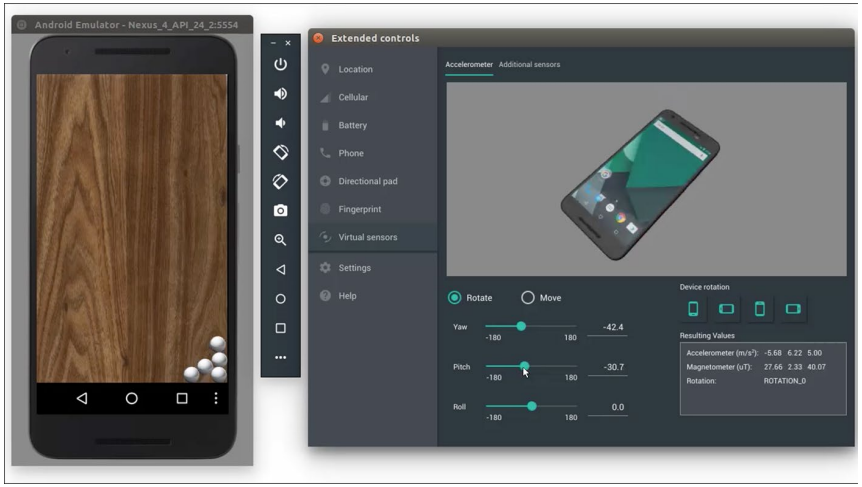


Emulación de apks (2/3)

- Lanzamiento a ejecución del emulador
  - Le cuesta un poco
  - Requiere 8GB de RAM (al menos)



- Si el apk de la app se ha descargado de algún sitio podemos instalar la app simplemente dejando caer el archivo sobre el emulador



Emulación de apks (3/3)

- Podemos trabajar con el emulador directamente desde la línea de comandos
  - Para saber los emuladores disponibles

```
emulator -list-avds
```

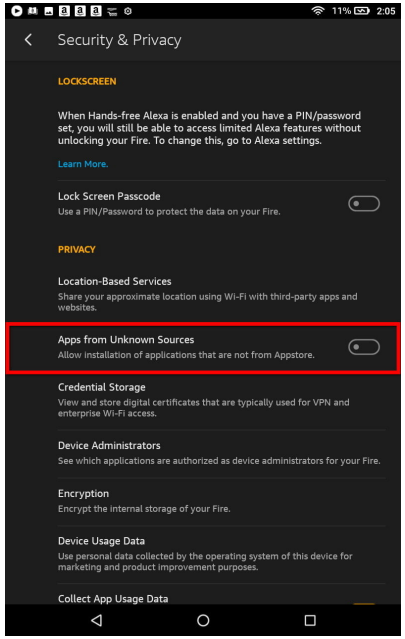
  - Para lanzar a ejecución un emulador

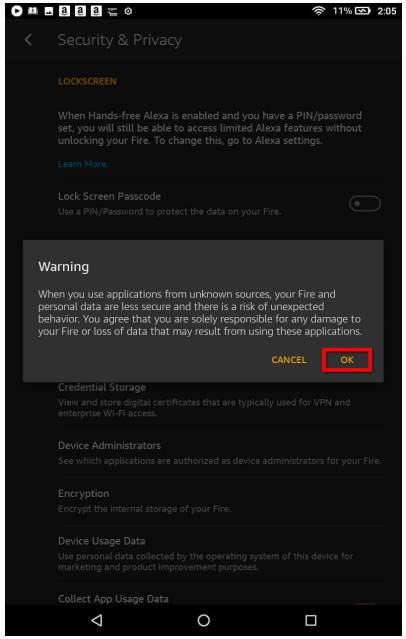
```
emulator -avd avd_name
```

NOTA: El emulador es una herramienta más de las incluidas en la SDK de Android, con lo que, obviamente, la orden sólo se reconocerá si se lanza desde el directorio correcto (o se incluye dicho directorio en el PATH del sistema)

Ejecución en dispositivos reales

- El teléfono (si se usa) debe tener las opciones de desarrollador (developer options) habilitadas y dentro de estas permitir la depuración por USB (USB debugging) https://developer.android.com/studio/run/device
- Además, hay que permitir también la instalación de apps desde fuentes desconocidas
  - Ej. Dispositivo Samsung →
- Mucho ojito con los cables que utilizamos



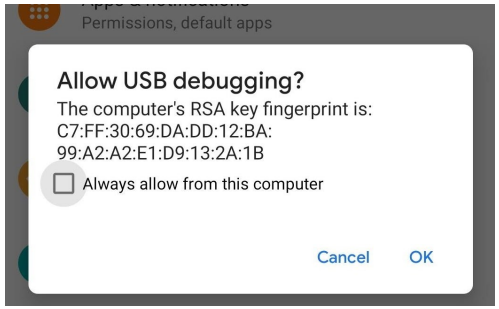


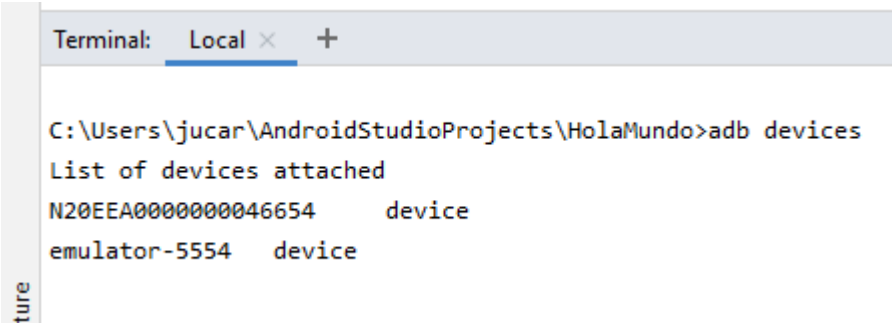
Emuladores alternativos

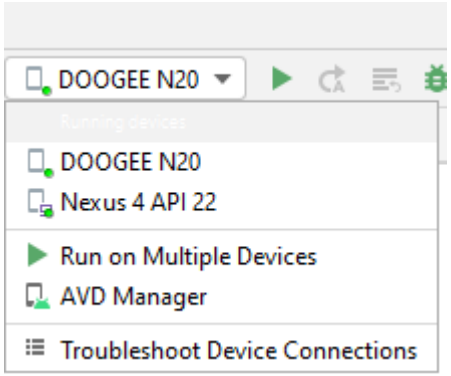
- Genymotion: Emulador optimizado
  - Producto comercial orientado a ofrecer Android “as a Service”
  - Ofrece dos alternativas de trabajo: escritorio y nube → la versión de escritorio (Mac, Windows y Linux) es gratuita si se utiliza sin una finalidad comercial
  - Máquina Virtual muy optimizada: https://www.genymotion.com
- Androl4b: Emulador en máquina virtual
  - Máquina virtual basada en Ubuntu mate orientada a la ingeniería inversa de apks y al análisis de malware
  - Incorpora preinstaladas gran cantidad de herramientas y una versión (algo antigua) de Android Studio
  - https://github.com/sh4hin/Androl4b
- Todas vienen sin Google Apps, pero éstas pueden obtenerse desde: https://github.com/opengapps/opengapps

Reconocimiento del dispositivo

- Android Studio manejará el dispositivo conectado por USB como si fuera un emulador más instalado en el entorno
  - ADB (Android Debug Bridge) es la herramienta que permite esto y se instala como parte de las SDK tools de Android
  - En el dispositivo se debe aceptar la depuración desde el entorno de desarrollo
  - https://developer.android.com/studio/run/oem-usb







Indice

- Herramientas para el desarrollo y emulación de aplicaciones Android
- **Depuración de apks**
- Reversing de apks
- Smali y parcheado de apks