

UT.7.1 INTRODUCCIÓN AL DESARROLLO EN ANDROID

Programación multimedia y dispositivos móviles

ÍNDICE

1. Arquitectura
2. Fragmentación
3. Componentes Básicos Apps Android
4. Activación de componentes
5. Modelo estados aplicación Android
6. Distribución

I.ARQUITECTURA

APP

APP no es un acrónimo, es la contracción de *application*

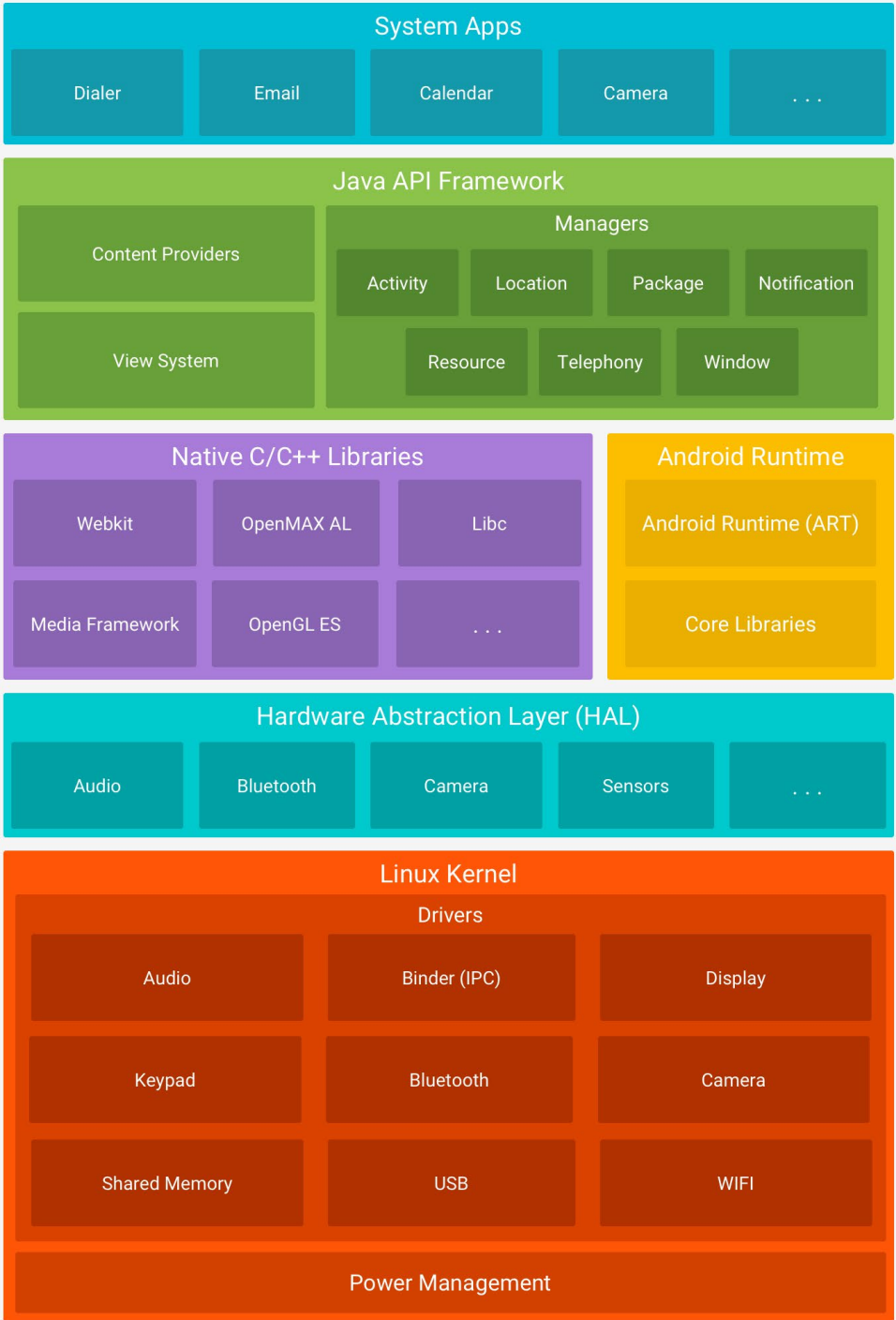
NO SE DICE A PE PE

APPS

Escritas en



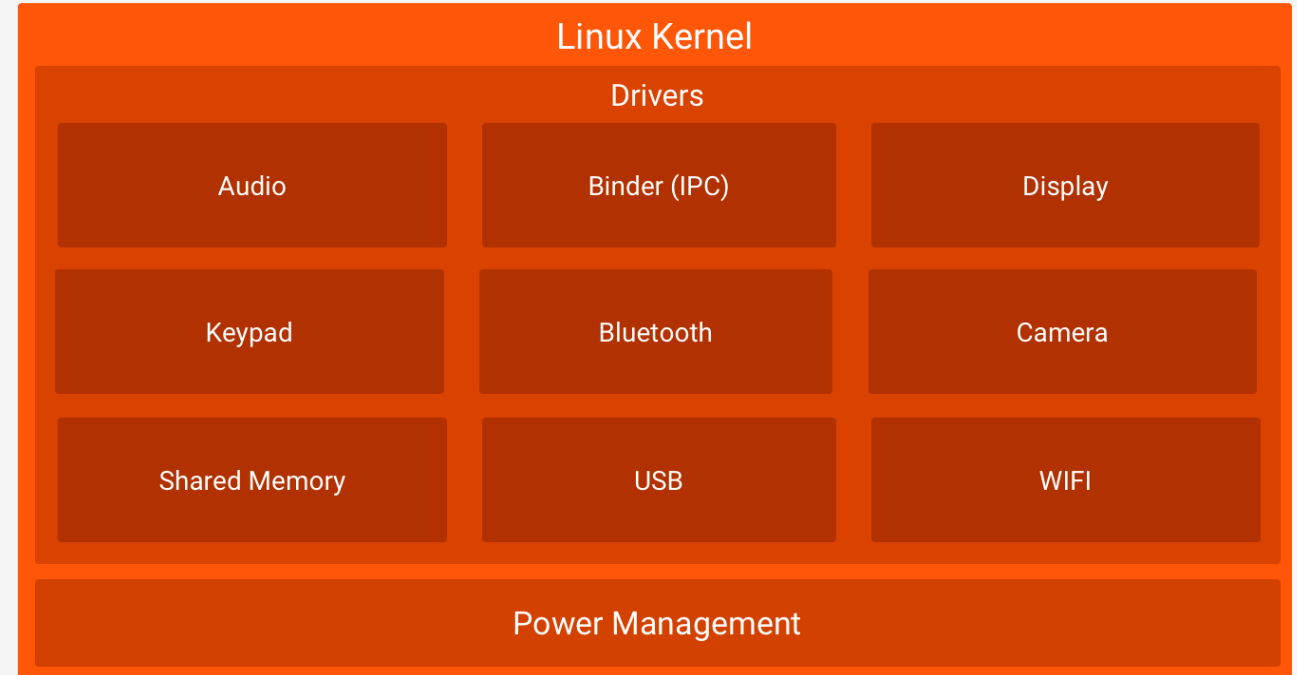
 **Kotlin**



ARQUITECTURA ANDROID

LINUX KERNEL

- Utilizado por ART para la gestión de memoria y creación de subprocesos
- Características de seguridad: APP=Usuario
- Fabricantes de HW pueden desarrollar drivers contra un Kernel conocido

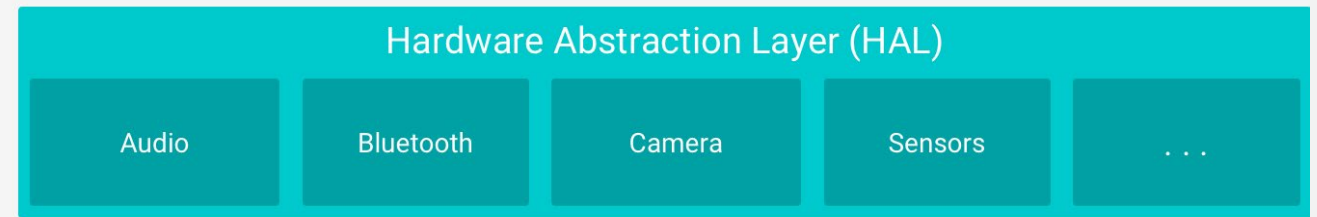


SISTEMA OPERATIVO LINUX

- Android es un Sistema Operativo Linux multiusuario.
- Cada app es un usuario del sistema:
 - Con su propio ID (desconocido por la propia app).
 - Con sus propios permisos para los ficheros: solo la app puede acceder a los ficheros.
- Cada app se ejecuta en su propio proceso de Linux, dentro de su propia máquina virtual de java
 - El código corre aislado del resto de apps.
 - Los procesos los abre el sistema Android cuando se tiene que interactuar con la app y los cierra cuando se cierra la app o necesita recursos para otras apps.

CAPA DE ABSTRACCIÓN DE HARDWARE (HAL)

- Ofrece las capacidades del hardware del dispositivo a través de una interfaz estandarizada al marco de trabajo de Java



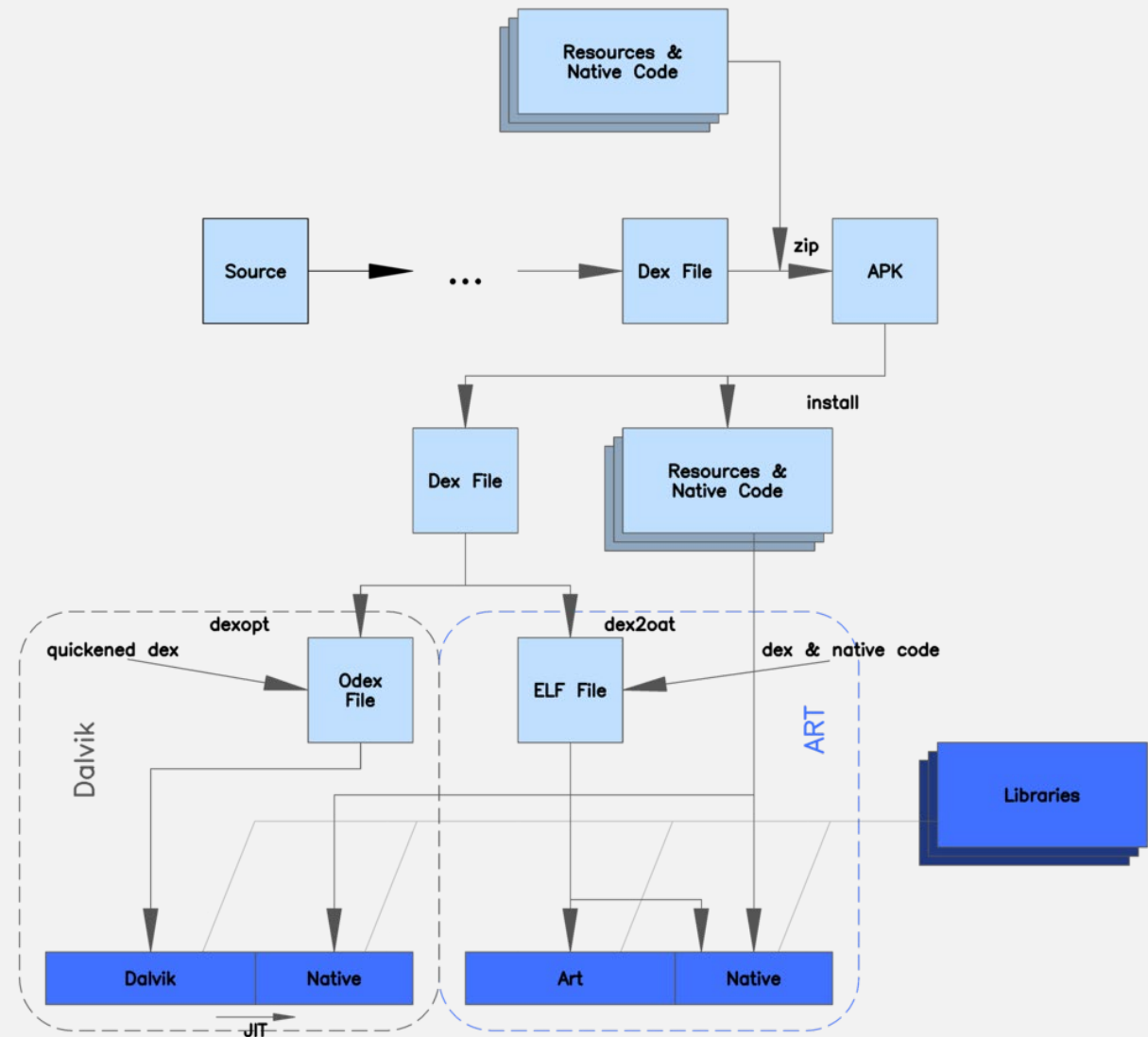


CAPA DE ABSTRACCIÓN DE HARDWARE (HAL)

- Native: HAL y ART están escritos en código nativo y requieren de las bibliotecas nativas escritas en C/C++. Además se ofrece el acceso a algunas de estas bibliotecas a través de la API de Java, como por ejemplo a OpenGL para dibujar y manipular objetos 2D y 3D.

ART: ANDROID RUNTIME

- Máquina virtual de Java de Android.
- Sustituye a Dalvik a partir de Android 5.0 “Lollipop”
 - **Dalvik compila** Just In Time **al ejecutar** app.
- Compila bytecode en formato dex a nativo
 - **ART compila** AOT (Ahead Of Time) **al instalar la app**.

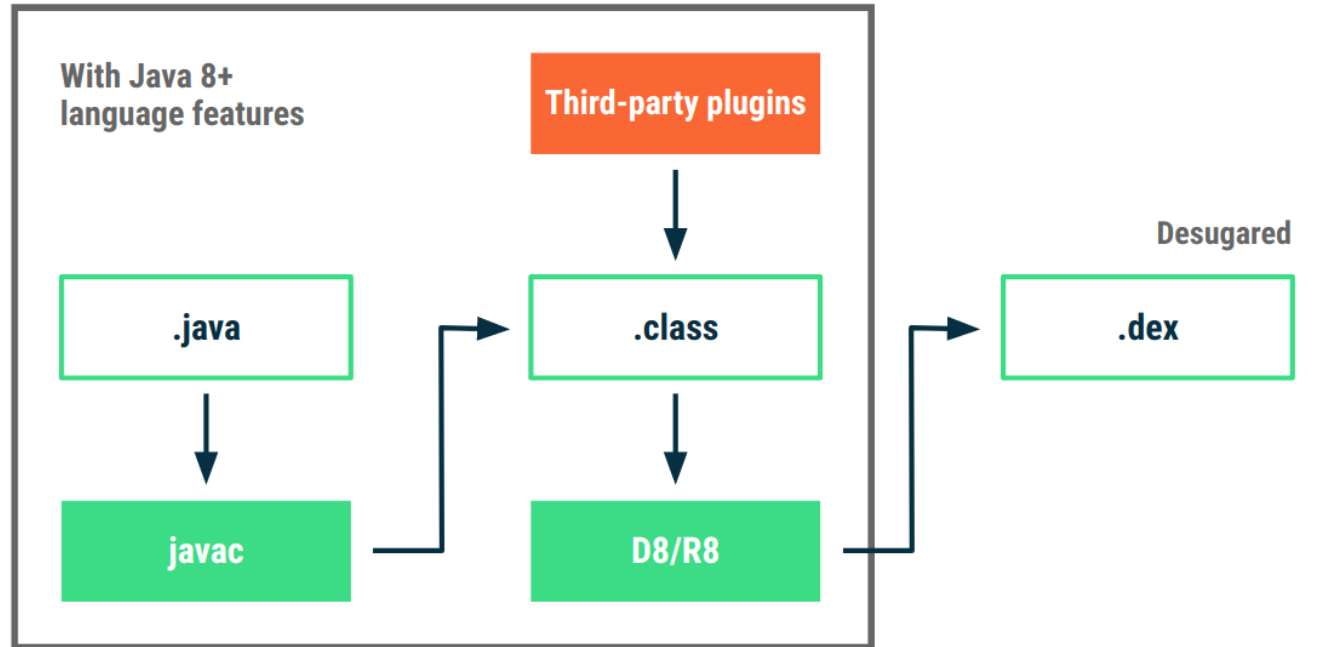


VERSIÓN DE JAVA

- Por defecto Java 7, hasta 11
- Pero muchas funcionalidades posteriores [del lenguaje](#) pueden usarse
- Requisitos: Gradle Android Plugin > 7.0.0 (actual 7.4.0)
- Para la [versión 11](#) de Java, configurar en build.gradle:

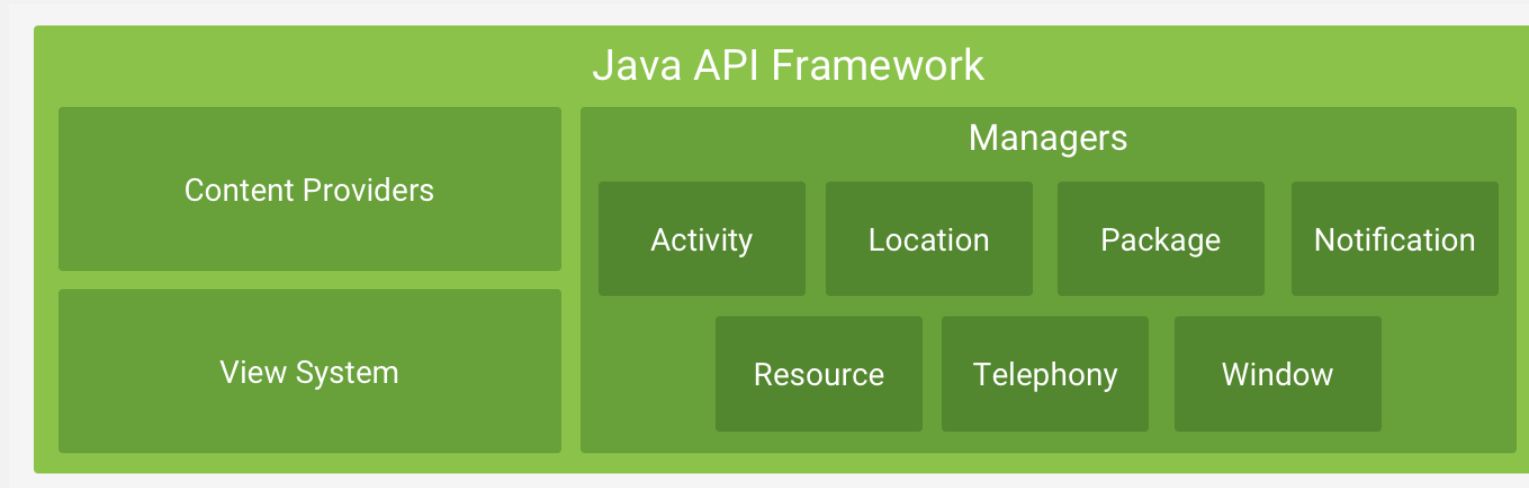
```
compileOptions {  
    sourceCompatibility  
        JavaVersion.VERSION_11  
    targetCompatibility  
        JavaVersion.VERSION_11  
}
```

DESUGARING



PRACTICAMOS

Configurar un proyecto de Android Studio para poder utilizar inferencia automática de tipos (var)



MARCO DE TRABAJO DE LA API DE JAVA

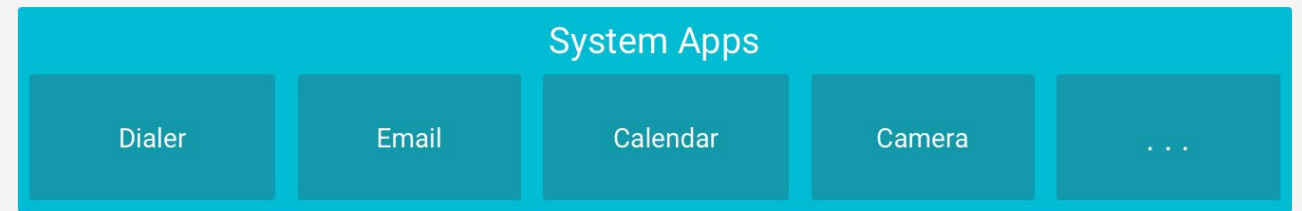
- Toda la funcionalidad de Android se ofrece a través de una API Java

MARCO DE TRABAJO DE LA API DE JAVA

- Sistema de **vistas** que permite crear la interfaz de usuario
- Gestor de **recursos**, proporcionando acceso a elementos que no son código, como cadenas localizadas, gráficos y ficheros de diseño
- Gestor de **notificaciones** que permiten a las aplicaciones mostrar notificaciones en diversas partes del sistema, como la barra de estado, las insignias y el cajón de notificaciones
- Gestor de **actividades** que gestiona el ciclo de vida de las aplicaciones y la navegación hacia atrás
- **Proveedores de contenido** que permiten obtener datos de otras aplicaciones

APLICACIONES DEL SISTEMA

- Android tiene una serie de aplicaciones esenciales para el correo, la mensajería SMS, el teléfono, el calendario o la cámara
- Estas aplicaciones se pueden sustituir por otras de terceros
- Proporcionan funcionalidad al usuario, pero también a las apps que no tienen que implementar esa funcionalidad ellas mismas.



2.FRAGMENTACIÓN

VERSIÓN DE ANDROID

- Determina la versión más antigua de Android en el que funciona la aplicación
- Balance entre
 - Número de dispositivos que quiero alcanzar
 - Funcionalidad proporcionada por la plataforma

API LEVEL DISTRIBUTION CHART

ANDROID PLATFORM VERSION		API LEVEL	CUMULATIVE DISTRIBUTION
4.4	KitKat	19	
5.0	Lollipop	21	99,3%
5.1	Lollipop	22	99,0%
6.0	Marshmallow	23	97,2%
7.0	Nougat	24	94,4%
7.1	Nougat	25	92,5%
8.0	Oreo	26	90,7%
8.1	Oreo	27	88,1%
			81,2%
9.0	Pie	28	
			68,0%
10.	Q	29	
			48,5%
11.	R	30	
			24,1%
12.	S	31	
13.	T	33	5,2%

API LEVEL

- **minSdkVersion:** la mínima versión de Android para la que **funciona** la app
- **compileSdkVersion:** el SDK de Android con el que has **compilado** la app
- **targetSdkVersion:** la versión más alta con la que has **probado** la app. Facilita la degradación elegante en tiempo de ejecución de funciones de Sistema

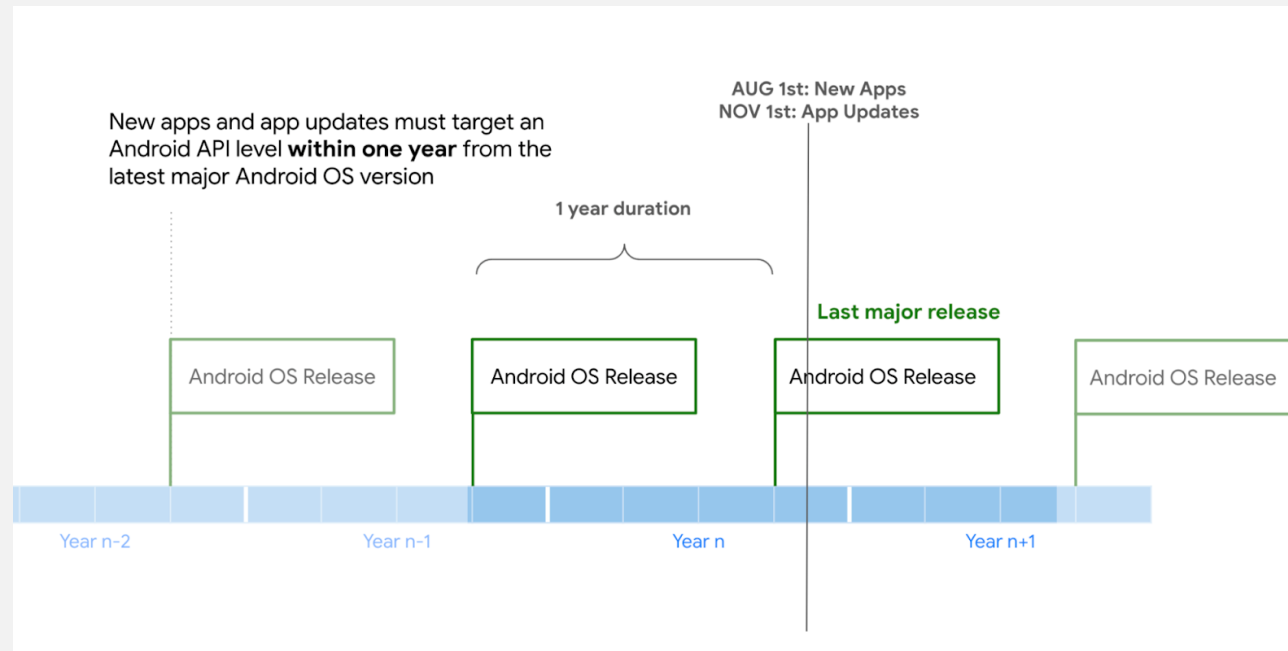
- Obligatoriamente

```
minSdkVersion <= targetSdkVersion <= compileSdkVersion
```

- Idealmente

```
minSdkVersion (lowest possible) <=
targetSdkVersion == compileSdkVersion
(latest SDK)
```

TARGET API LEVEL REQUIREMENTS



COMPATIBILIDAD HACIA ADELANTE

- Android es compatible hacia adelante por defecto
- Una app con minSDKVersion 3.0 se puede instalar en cualquier teléfono con SDK Version ≥ 3.0



COMPATIBILIDAD HACIA ATRÁS

- Android no es compatible hacia atrás por defecto
- Una app con minSDKVersion 5.0 no se puede instalar en teléfonos con SDK Version < 5.0



PRACTICAMOS API LEVEL

- Las Splash Screen se añadieron en Android 12 (API Level 31)
- En Android 13 (API Level 33) se añade `windowSplashScreenBehavior`
- Para poder utilizarla debemos
 - Instalar el SDK 33
 - Actualizar `sdkCompileVersion` a 33
 - Actualizar `minSdkVersion` a 33
 - Disponer de un dispositivo con API Level 33

```
1 <resources xmlns:tools="http://schemas.android.com/tools">
2     <!-- Base application theme. -->
3     <style name="Theme.Sdkversions" parent="Theme.MaterialComponents" >
4         <!-- Primary brand color. -->
5         <item name="colorPrimary">@color/purple_500</item>
6         <item name="colorPrimaryVariant">@color/purple_700</item>
7         <item name="colorOnPrimary">@color/white</item>
8         <!-- Secondary brand color. -->
9         <item name="colorSecondary">@color/teal_200</item>
10        <item name="colorSecondaryVariant">@color/teal_700</item>
11        <item name="colorOnSecondary">@color/black</item>
12        <!-- Status bar color. -->
13        <item name="android:statusBarColor" tools:targetApi="33">@color/black</item>
14        <!-- Customize your theme here. -->
15        <item name="android:windowSplashScreenBackground">@color/purple_500</item>
16        <item name="android:windowSplashScreenIconBackground">@color/white</item>
17        <item name="android:windowSplashScreenAnimatedIcon">@color/purple_500</item>
18        <item name="android:windowSplashScreenAnimationDuration">1000</item>
19        <item name="android:windowSplashScreenBehavior">icon</item>
20    </style>
21 </resources>
```

3. COMPONENTES BÁSICOS APPS ANDROID

COMPONENTES BÁSICOS DE UNA APLICACIÓN

1. Actividades (activities)
2. Servicios (services)
3. Receptores de emisiones (broadcast receivers)
4. Proveedores de contenidos (content providers)

ACTIVIDADES

- Escritorio: único punto de entrada.
- Móvil: diversos puntos de entrada requieren diferentes experiencias.
- Cada una de estas interacciones se modela con una subclase de `Activity`
 - Representa una pantalla en una aplicación
 - `Main activity`, pero puede haber más
 - Registrar en el `manifest`
 - Controlar el ciclo de vida
- Ejemplo email:
 - Actividad principal: lista de correos
 - Actividad de escribir email, invocada desde otras apps

LAS ACTIVIDADES PERMITEN

- Saber qué está haciendo el usuario
 - Y mantener ese proceso vivo.
- Saber qué procesos han sido previamente utilizados por el usuario a los que podría volver
 - Para priorizar mantenerlos accesibles.
- Ayudar a las apps a gestionar la destrucción del proceso
 - Para que éstas puedan guardar su estado.
- Proporcionar medios para implementar flujos de uso entre aplicaciones, coordinados por el sistema.
 - Caso típico: compartir.

SERVICIOS

- Permiten mantener una aplicación trabajando en segundo plano
 - Reproducir música
 - Sincronizar datos
- Iniciados por la propia aplicación o por otras aplicaciones.
- Se ejecutan en el mismo hilo de ejecución que la app.
- Aunque no tiene IU, pueden mostrar notificaciones:
 - De sistema, p.ej., reproduciendo audio
 - En barra de estado, p.ej., final de descarga
- Se implementan como subclases de `Service`

RECEPTORES DE EMISIONES

- Permiten a la app reaccionar a emisiones de sistema, aún cuando no se están ejecutando, como por ejemplo batería baja, pantalla capturada, etc.
- La app puede programar sus propias alarmas y recibirlas como emisiones.
- También puede emitir para otras aplicaciones.
- No tienen IU pero pueden mostrar notificaciones en la barra de estado.
- Se implementan como subclases de `BroadcastReceiver`

PROVEEDORES DE CONTENIDO

- Habilitan la compartición de datos entre diferentes aplicaciones del sistema
 - Consulta
 - Modificación si lo permite la app
- Se implementan como subclases de `ContentProvider`
- El acceso se realiza mediante URIs

4.ACTIVACIÓN DE COMPONENTES

INTENT

- Actividades
- Receptores de emisiones
- Servicios
- Se activan de forma asíncrona mediante `Intents`

TIPOS DE INTENTS

EXPLÍCITOS

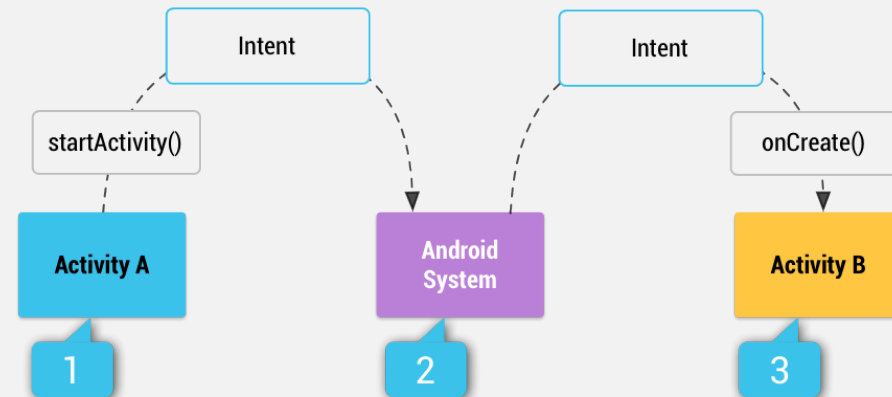
- Especifican la app que las va a administrar
- Generalmente de la propia app a si misma, p.ej.:
 - Iniciar otra actividad en resultado a una acción de usuario
 - Para iniciar un servicio para descargar un archivo en segundo plano

IMPLÍCITOS

- No indican un receptor
- Indican una acción general a realizar, p.ej. mostrar ubicación del usuario, de forma que otra app se encargue de mostrarla.
- Especificar
 - Acción (ACTION_VIEW, ACTION_SEND, etc.)
 - Datos (URI y MIME/Type)
 - Categoría (no necesaria, clarifica aún más el intent)
- Además
 - Extras

LANZAR ACTIVIDADES

1. Crear un `Intent` y llamar a `startActivity`
2. El sistema Android busca una aplicación con un filtro de *intents* que incluya el tipo de *intent* enviado.
3. El sistema Android inicia la actividad registrada en el filtro de *intents*, invoca `onCreate()` y le pasa el `intent`.
 1. Si encuentra varias, muestra un cuadro de diálogo para elegir qué aplicación se hará cargo.



INTENT FILTER

```
<activity android:name=".MainActivity">  
  <intent-filter>  
    <action android:name="android.intent.action.MAIN" />  
    <category  
android:name="android.intent.category.LAUNCHER" />  
    <data android:mimeType="text/plain"/>  
  </intent-filter>  
</activity>
```

INTENTS COMUNES

- Alarma
- Calendario
- Cámara
- Contactos
- Correo electrónico
- Almacenamiento de archivo
- Acciones locales
- Mapas
- Música o vídeo
- Nota nueva
- Teléfono
- Buscar
- Configuración
- Mensajería de texto
- Navegador Web

PRACTICAMOS

Crear una aplicación que realiza la suma de dos números y la muestra en pantalla mediante **Intents explícitos**

PRACTICAMOS

Crear una aplicación que permita mostrar una dirección en [Google Maps](#) mediante **Intents implícitos**.

RECEPTOR DE EMISIONES

- Intents implícitos
 - A partir de Android 8 solo se permiten registrar dinámicamente cuando la aplicación está en ejecución

- Intents explícitos

- Se registran en manifest

```
<receiver android:name=".MyBroadcastReceiver" android:exported="true">  
  <intent-filter>  
    <action android:name="com.juanagui.intents.bcast.custom"/>  
  </intent-filter>  
</receiver>
```

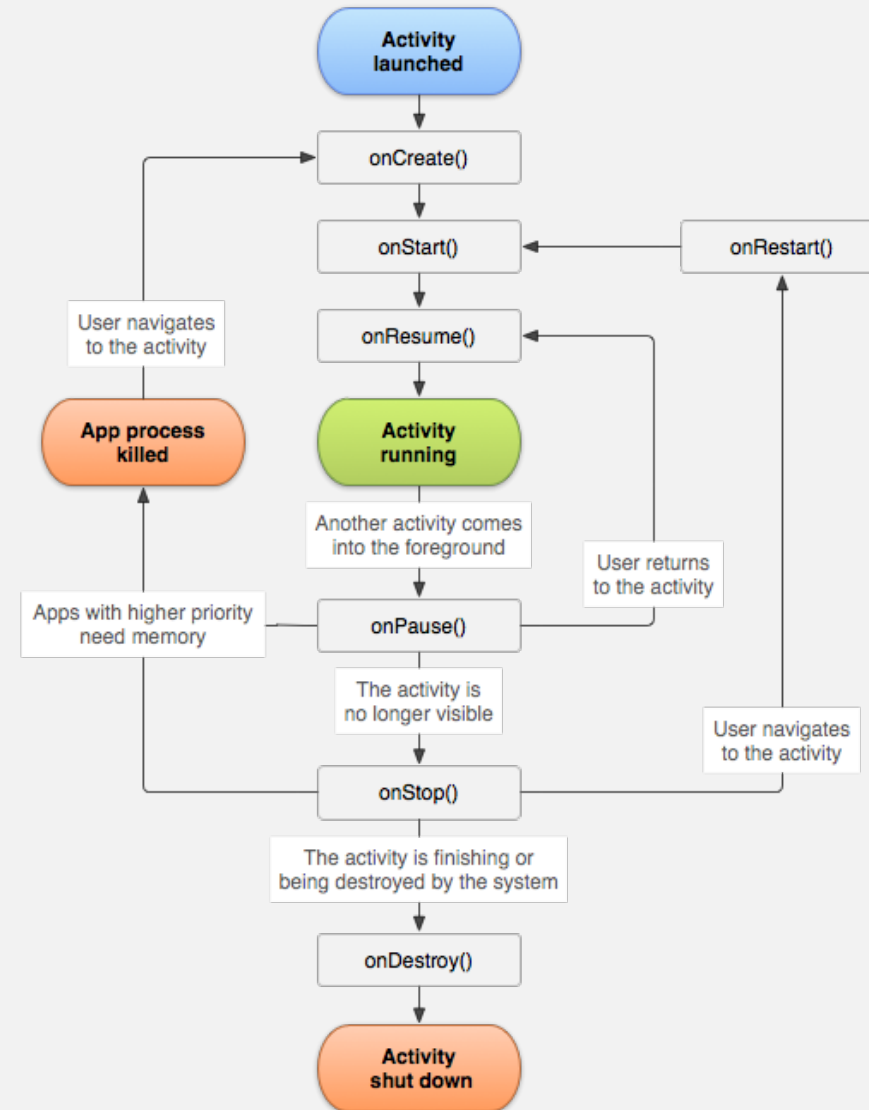
PRACTICAMOS

Crear una muestra la hora en hh:mm y la cambie cada vez que reciba la emisión de ACTION_TIME_TICK

5. MODELO ESTADOS APLICACIÓN ANDROID

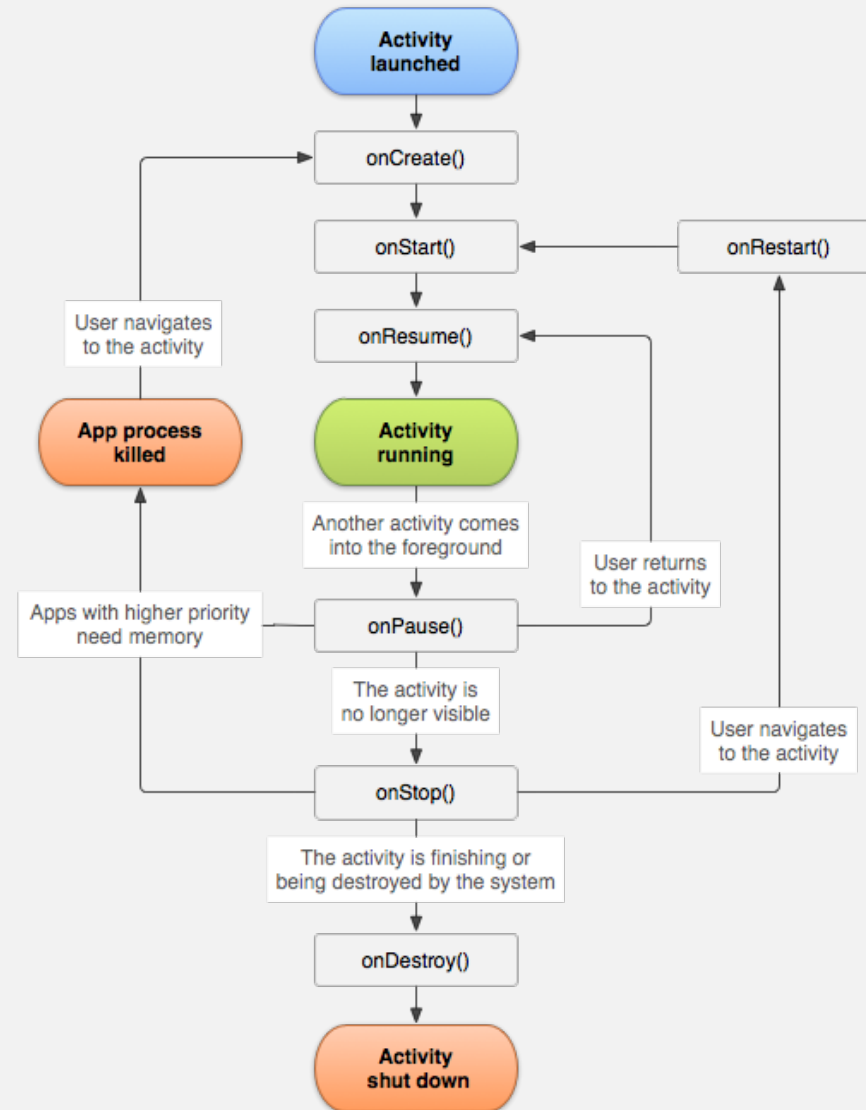
ESTADOS ACTIVIDADES ANDROID

- **Ejecutando** (*running*): la actividad está visible e interactuando con el usuario
- **Pausada** (*paused*): la actividad continua visible pero está parcialmente oculta. Sigue ejecutándose pero podría ser finalizada por el sistema.
- **Detenida** (*stopped*): la actividad no está visible pero sigue ejecutándose aunque podría ser finalizada por el sistema.
- **Finalizada** (*killed*): la actividad ha sido finalizada por el sistema o por una llamada a `finished`



MÉTODOS ACTIVIDADES ANDROID

- **onCreate:** solo se llama cuando se crea la actividad.
- **onStart:** cuando se retorna a la actividad desde un estado en el que no está visible.
- **onResume:** cuando se vuelve a interactuar con la actividad.
- **onPause:** cuando la actividad deja de estar en primer plano.
- **onStop:** cuando la actividad va a dejar de estar visible.



LOGCAT

- Log._(tag,msg)
- Log.v(String, String)
- Log.d(String, String)
- Log.i(String, String)
- Log.w(String, String)
- Log.e(String, String)
- Log.wtf(String, String)
- Depurar utilizando [logcat](#)
- Orden de verbosidad decreciente, cada nivel contiene al siguiente
 - Verbose
 - Debug
 - Information
 - Warning
 - Error
 - Assert
 - Debug: se eliminan en tiempo de ejecución
 - Error, Warning, Information se mantienen

```
Emulator Pixel_4_API_31 Android  com.juanagui.activity.lifecycle (8162  Info  Q- ACTIVITY_LIFECYCLE
2021-10-24 08:47:00.076 8162-8162/com.juanagui.activity.lifecycle I/ACTIVITY_LIFECYCLE: onCreate()
2021-10-24 08:47:00.090 8162-8162/com.juanagui.activity.lifecycle I/ACTIVITY_LIFECYCLE: onStart()
2021-10-24 08:47:00.095 8162-8162/com.juanagui.activity.lifecycle I/ACTIVITY_LIFECYCLE: onResume()
```

PRACTICAMOS

- Crea una aplicación en la que para los métodos onCreate, onStart, onResume, onPause, onRestart, onStop y onDestroy de la actividad principal escriba por Logcat el nombre del método llamado.

ARRANQUE EN FRÍO

```
com.juanagui.activity.lifecycle I/ACTIVITY_LIFECYCLE: onCreate()  
com.juanagui.activity.lifecycle I/ACTIVITY_LIFECYCLE: onStart()  
com.juanagui.activity.lifecycle I/ACTIVITY_LIFECYCLE: onResume()
```


PASAR A SEGUNDO PLANO

```
/com.juanagui.activity.lifecycle I/ACTIVITY_LIFECYCLE: onPause()  
/com.juanagui.activity.lifecycle I/ACTIVITY_LIFECYCLE: onStop()
```

VOLVER DESDE SEGUNDO PLANO

```
/com.juanagui.activity.lifecycle I/ACTIVITY_LIFECYCLE: onRestart()  
/com.juanagui.activity.lifecycle I/ACTIVITY_LIFECYCLE: onStart()  
/com.juanagui.activity.lifecycle I/ACTIVITY_LIFECYCLE: onResume()
```

DESTRUCCIÓN EN SEGUNDO PLANO

```
/com.juanagui.activity.lifecycle I/ACTIVITY_LIFECYCLE: onDestroy()
```

CAMBIOS DE CONFIGURACIÓN: ORIENTACIÓN

- La actividad se destruye y crea con los cambios de orientación
- Los valores de los diseños en los que se ha proporcionado un *id* se restauran.
- Otros valores deben guardarse en *onSaveInstanceState*

PRACTICAMOS

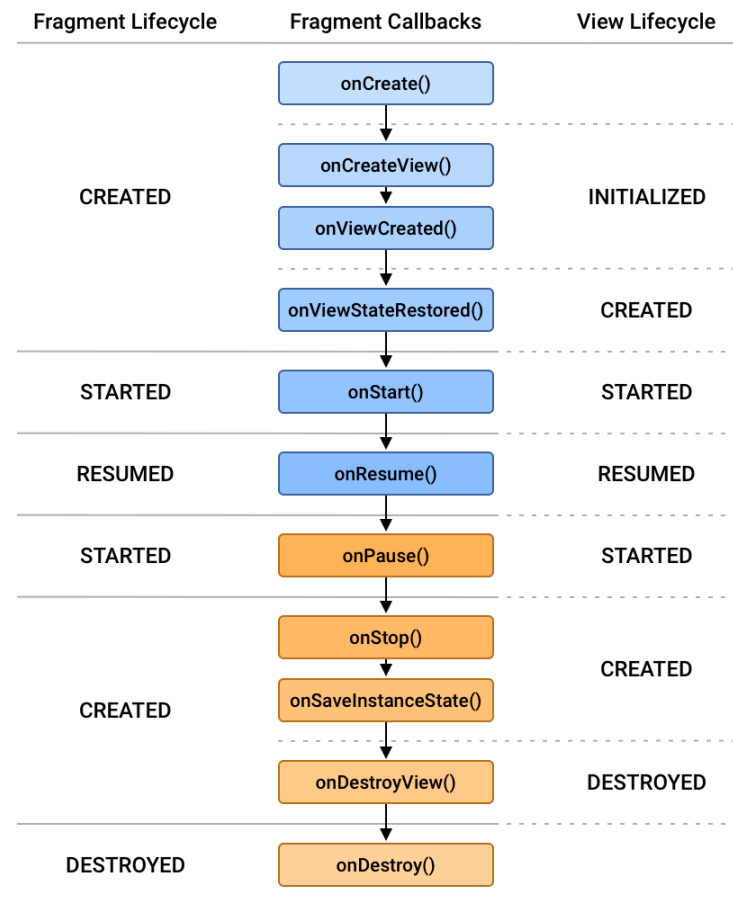
Crear una aplicación con dos diseños y
comprobar como se restauran los
valores de los textos

FRAGMENTOS

FRAGMENTOS

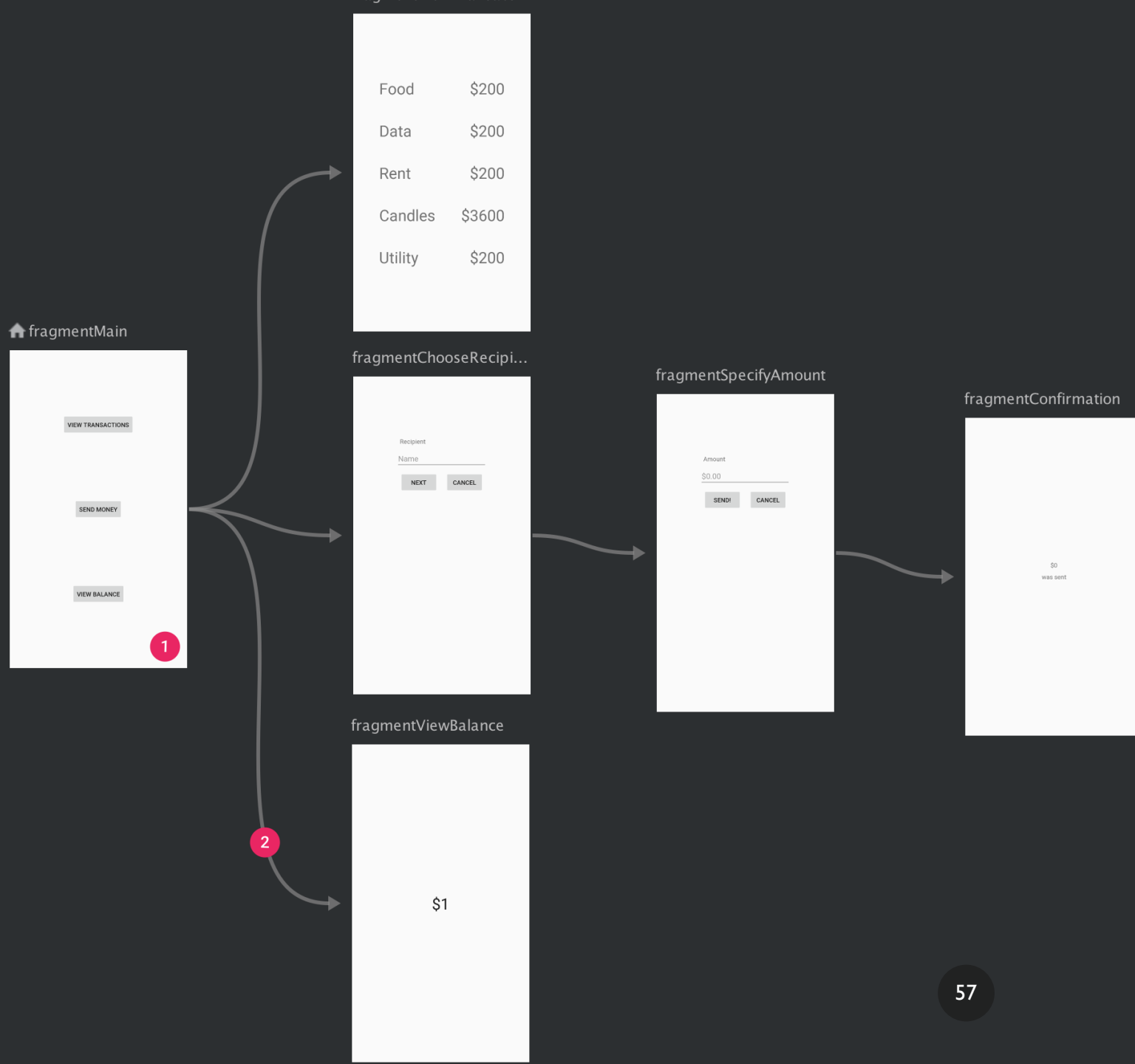
- Representan porciones de la interfaz de usuario que deseamos reutilizar entre diferentes actividades
- Tienen su propio ciclo de vida, pero está relacionado con el de la actividad a la que están adjuntas
- Meter en `FragmentManager`
- Poner el `name` del Fragment

CICLO DE VIDA



USOS DE FRAGMENTS

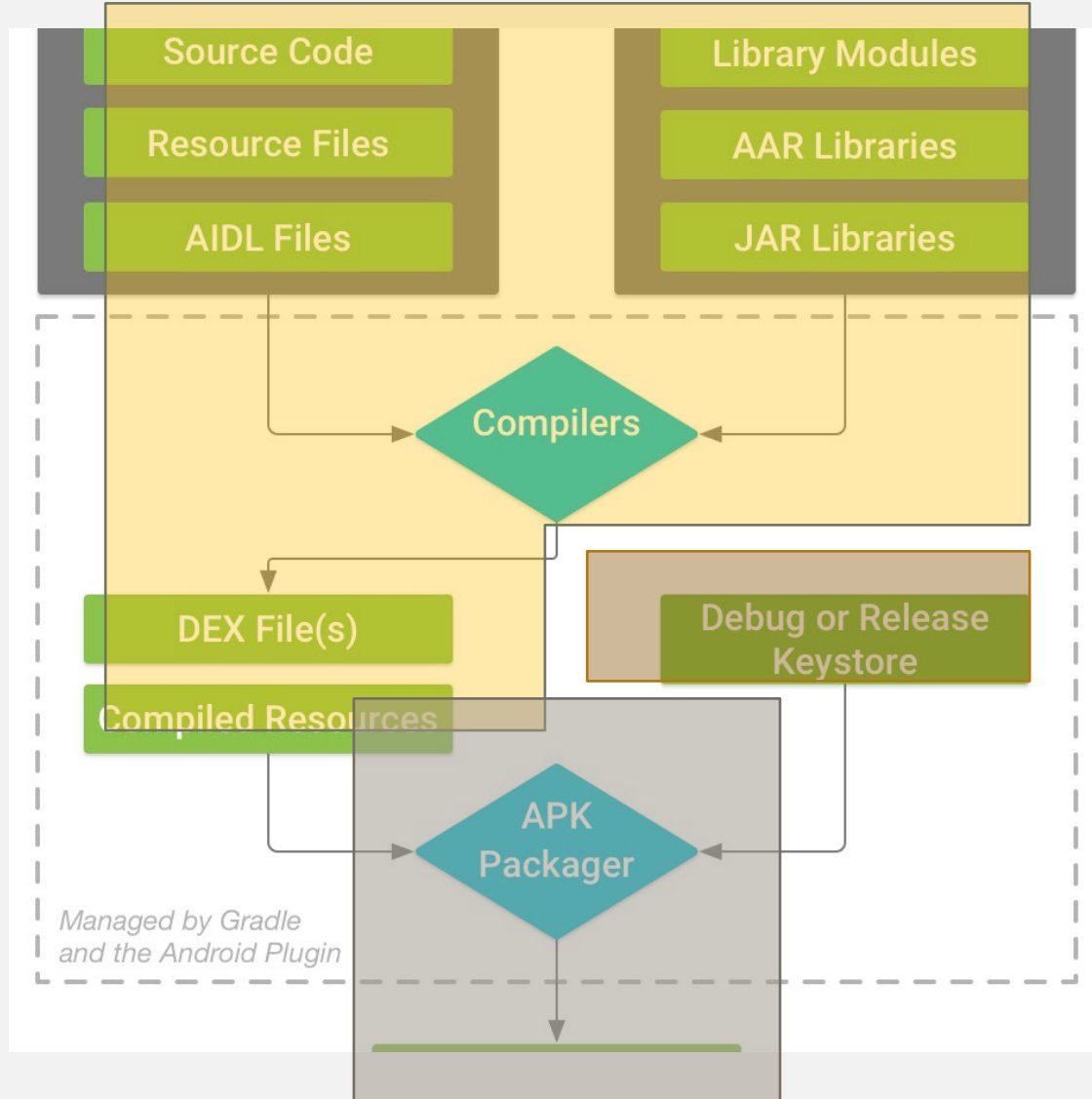
Navegación



DISTRIBUCIÓN

COMPILACIÓN

1. Compilar a DEX
2. Firma
3. Empaquetado

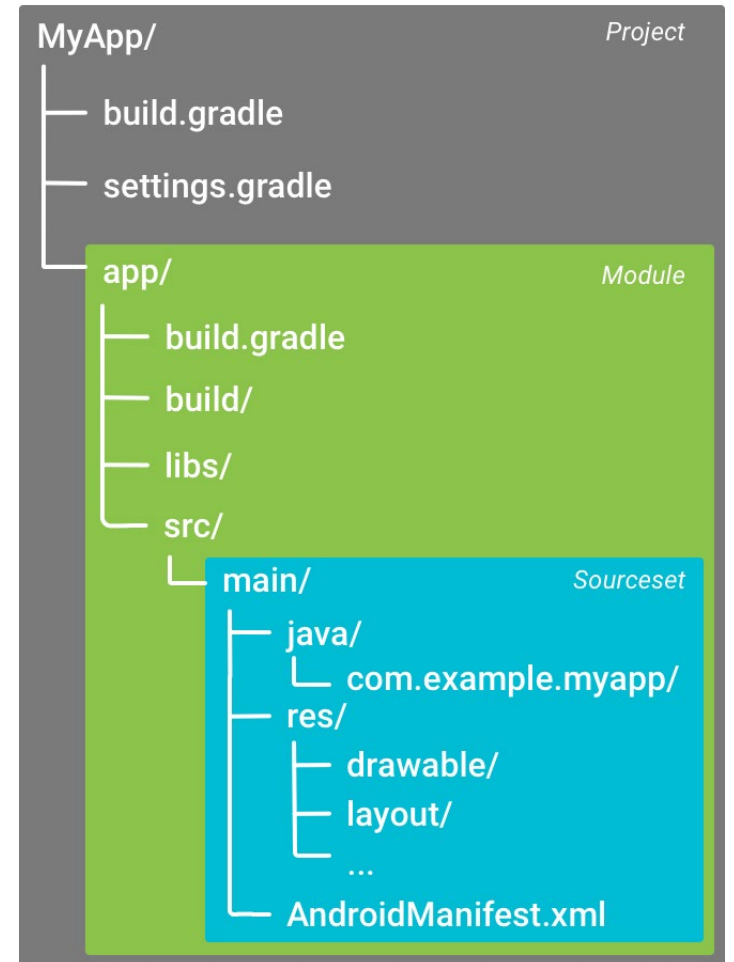


CONFIGURACIÓN DE LA COMPILACIÓN

settings.gradle

build.gradle (top)

build.gradle (módulo)



SETTINGS.GRADLE

Que módulos se incluyen
en la app

```
dependencyResolutionManagement { DependencyResolutionManagement it ->
    repositoriesMode.set(RepositoriesMode.FAIL_ON_PROJECT_REPOS)
    repositories { RepositoryHandler it ->
        google()
        mavenCentral()
    }
}

rootProject.name = "ConfigurationChanges"
include ':app'
```

BUILD.GRADLE (TOP)

Repositorios y
dependencias comunes
a todos los módulos

```
// Top-level build file where you can add configuration options common to all sub-projects/modules.  
buildscript {  
    repositories {  
        google()  
        mavenCentral()  
    }  
    dependencies {  
        classpath "com.android.tools.build:gradle:7.0.3"  
  
        // NOTE: Do not place your application dependencies here; they belong  
        // in the individual module build.gradle files  
    }  
}  
  
task clean(type: Delete) {  
    delete rootProject.buildDir  
}
```

BUILD.GRADLE (TOP)

- **ext:** parámetros extras comunes para varios módulos

- top

```
ext {  
    appCompatVersion = "1.3.1"  
}
```

- módulo

```
implementation "androidx.appcompat:appcompat:${rootProject.ext.appCompatVersion}"
```

BUILD.GRADLE (MÓDULO)

```
android {  
    compileSdk 31  
  
    defaultConfig {  
        applicationId "com.juanagui.activity.configurationchanges"  
        minSdk 22  
        targetSdk 31x  
        versionCode 1  
        versionName "1.0"  
        testInstrumentationRunner "androidx.test.runner.AndroidJUnitRunner"  
    }  
  
    buildTypes {  
        release {  
            minifyEnabled false  
            shrinkResources true  
            proguardFiles getDefaultProguardFile('proguard-android-optimize.txt'), 'proguard-rules.pro'  
        }  
    }  
    compileOptions {  
        sourceCompatibility JavaVersion.VERSION_1_8  
        targetCompatibility JavaVersion.VERSION_1_8  
    }  
}
```


BUILD.GRADLE (MÓDULO)

```
dependencies {  
    implementation "androidx.appcompat:appcompat:${rootProject.ext.appCompatVersion}"  
    implementation 'com.google.android.material:material:1.4.0'  
    implementation 'androidx.constraintlayout:constraintlayout:2.1.1'  
    testImplementation 'junit:junit:4.13.2'  
    androidTestImplementation 'androidx.test.ext:junit:1.1.3'  
    androidTestImplementation 'androidx.test.espresso:espresso-core:3.4.0'  
}
```

REDUCIR Y OFUSCAR

- R8: herramienta que reduce y ofusca el código
- minifyEnabled
 - Reducción de código: elimina código no llamado
 - Ofuscación: acorta nombres
 - Optimización: busca caminos
- shrinkResources
 - Elimina recursos no utilizados
- proguardfiles
 - Configura la ofuscación

FORMATO APK

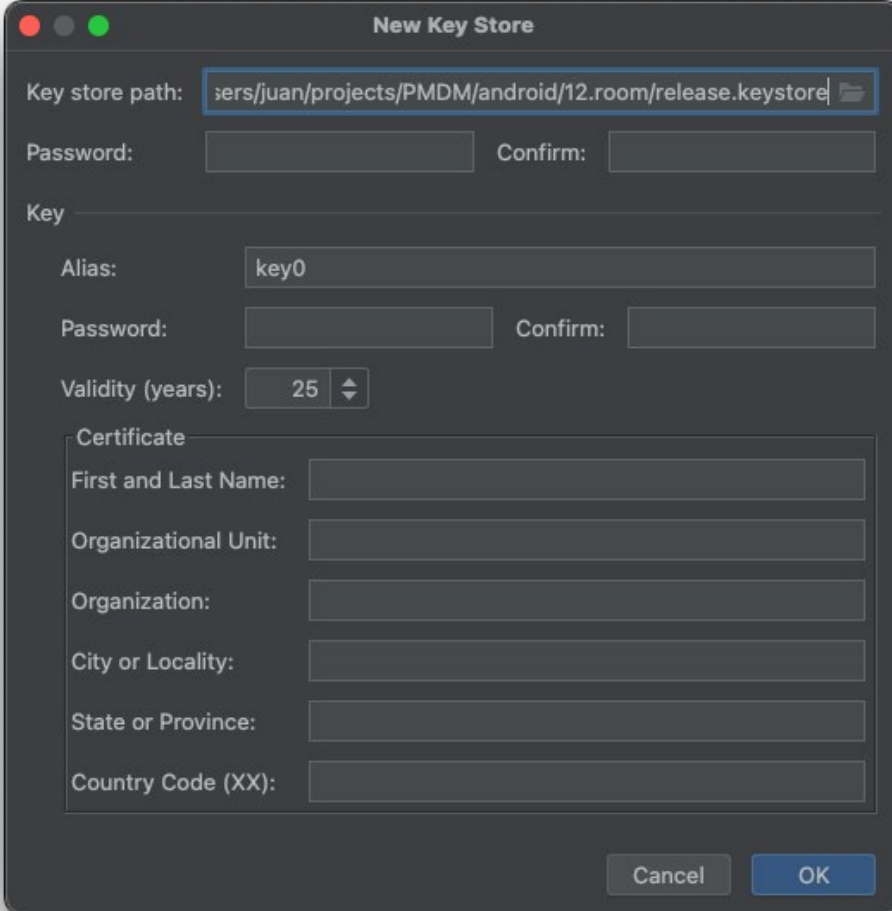
- *Android Package*
- Formato antiguo
- Generalmente se usaba el APK Universal:
 - Todos los ABI (32/64 ARM/Intel)
 - Todos las densidades de pantalla (MDPI, HDPI, etc.)
 - Tamaño muy grande
- Firmado por el desarrollador

GENERAR ALMACÉN DE CLAVES

```
keytool -genkeypair -alias example -keyalg RSA -keysize 4096 -sigalg  
SHA256withRSA -dname "cn=example.com,ou=exampleou,dc=example,dc=com"  
-keypass changeit2 -validity 365 -storetype JKS -storepass changeit1 -keystore  
release.keystore
```

- Autofirmado.
- Alias de la clave example
- Password de la clave y del almacén: changeit
- Firmar con Build > Generate Signed Bundle/APK
 - Seleccionar APK
 - Poner ruta a keystore, alias de la clave y contraseñas

GENERAR ALMACÉN DE CLAVES



New Key Store

Key store path:

Password: Confirm:

Key

Alias:

Password: Confirm:

Validity (years):

Certificate

First and Last Name:

Organizational Unit:

Organization:

City or Locality:

State or Province:

Country Code (XX):

Cancel OK

```
keytool -printcert -jarfile app/release/app-release.apk
```

Signer #1:

Certificate #1:

Owner: C=ES, ST=Madrid, L=Madrid, O=juanagui, OU=juanagui.com, CN=Juan Agüí

Issuer: C=ES, ST=Madrid, L=Madrid, O=juanagui, OU=juanagui.com, CN=Juan Agüí

Serial number: 1

Valid from: Sun Mar 05 18:25:23 CET 2023 until: Thu Feb 27 18:25:23 CET 2048

Certificate fingerprints:

SHA1: 57:8C:60:0F:BA:45:92:EB:64:4A:C7:D4:49:46:37:85:9E:8F:64:39

SHA256:

3E:C5:B3:B6:FC:6C:C6:92:74:93:DA:65:36:D7:61:30:A5:42:5B:2C:D0:92:39:30:84:38:59:23:EF:AE:F6:25

Signature algorithm name: SHA256withRSA

Subject Public Key Algorithm: 2048-bit RSA key

Version: 1

VERIFICAR LA FIRMA

VERIFICAR LA FIRMA

```
~/Library/Android/sdk/build-tools/33.0.2/apksigner verify --print-certs app/release/app-release.apk
```

Signer #1 certificate DN: C=ES, ST=Madrid, L=Madrid, O=juanagui, OU=juanagui.com, CN=Juan Agüí

Signer #1 certificate SHA-256 digest:

3ec5b3b6fc6cc6927493da6536d76130a5425b2cd092393084385923efaef625

Signer #1 certificate SHA-1 digest: 578c600fba4592eb644ac7d4494637859e8f6439

Signer #1 certificate MD5 digest: a7f412db90cdf7a1cc459d27fef16984

FIRMAR CON MI CERTIFICADO FNMT

1. Exportar el certificado a formato p12
2. Convertir p12 a jks

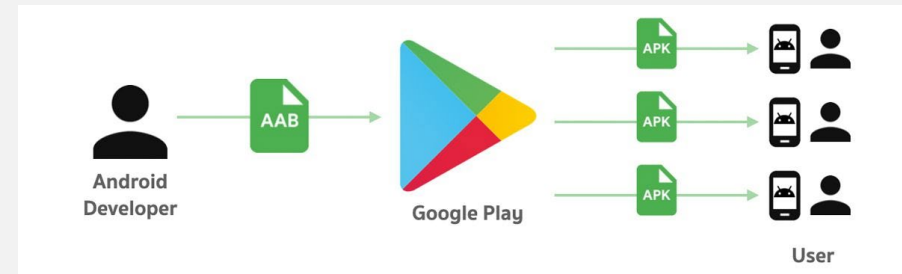
```
keytool -importkeystore -srckeystore Certificados.p12 -  
destkeystore key.jks -deststoretype jks
```

1. Firmar con el certificado

```
~/Library/Android/sdk/build-tools/33.0.2/apksigner sign -ks  
key.jks --in app/release/app-release.apk --out  
app/release/app-release-signed.apk
```

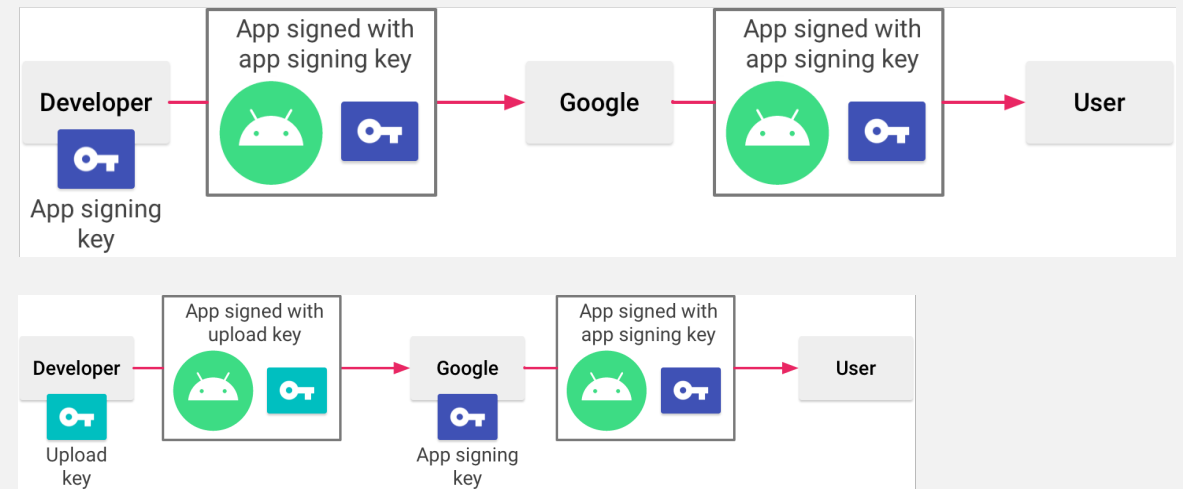

FORMATO AAB

- *Android Application Bundle*
- Obligatorio para subir al Play Store desde verano de 2021
- Play Store genera las diferentes combinaciones de ABI/densidad



FIRMA AAB

- Firma Play Store con una clave privada:
 - Generada por Google
 - Subida a Play Console por el desarrollador
 - Reutilizada de una app de la misma cuenta



PRACTICAMOS

Crear un APK y un AAB para una de las aplicaciones de ejemplo