



## CFGS: DESENVOLUPAMENT D'APLICACIONS MULTIPLATAFORMA

Mòdul: Programació multimèdia i dispositius mòbils

## TEMA 2: ARQUITECTURA I ESTRUCTURA D'UNA APLICACIÓ ANDROID

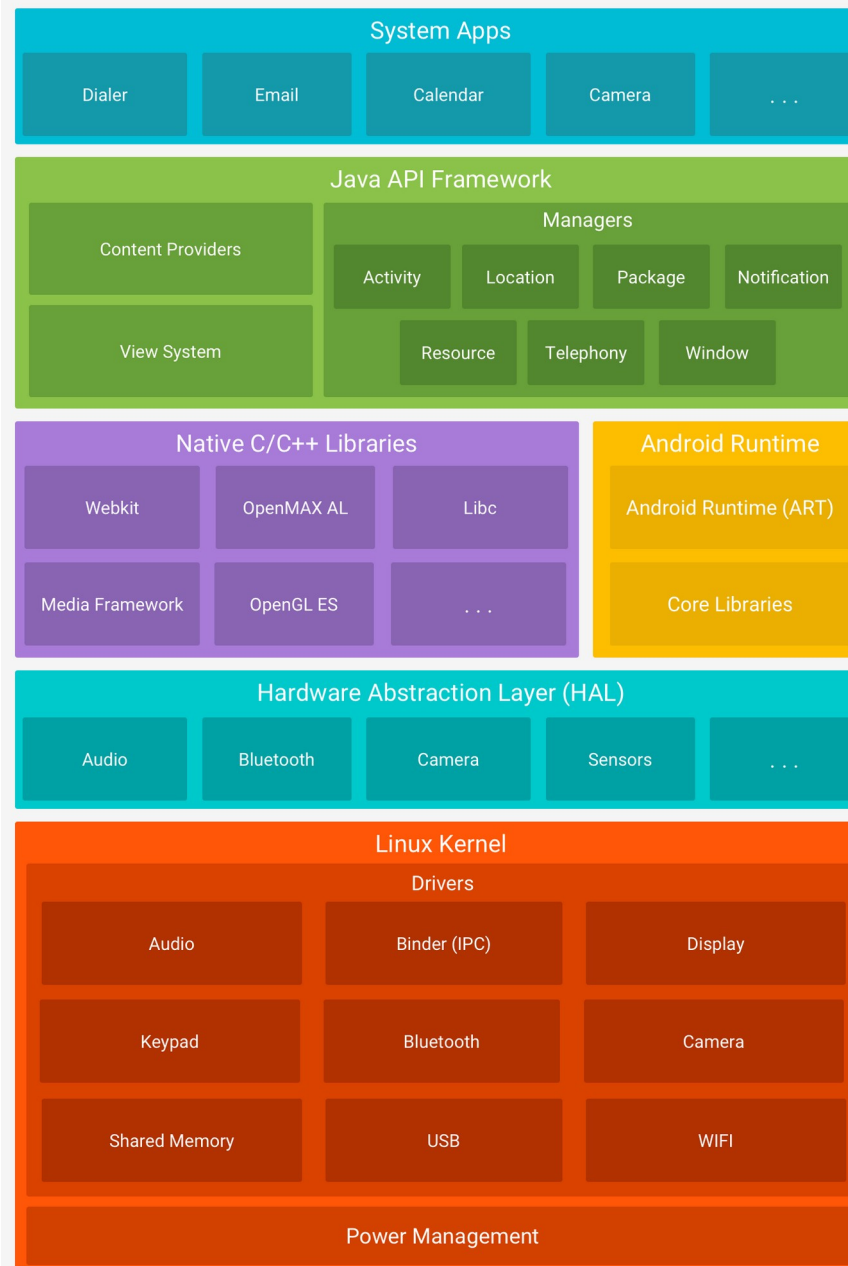
Germán Gascón Grau  
ggascon@gmail.com

110011100101100110101011001101101001100110101111001110001101100101100111001011001101010110011011010011001101011110



## Introducció

- L'arquitectura d'Android està formada per 6 capes:
  - Un **nucli de Linux** com a base que s'encarrega de la gestió del maquinari i programari.
  - La capa d'**abstracció del maquinari** (HAL) proporciona interfícies estàndard que exposen les capacitats del maquinari del dispositiu al framework de la Java API.
  - L'**Android Runtime** (ART), desde la versió 5.0 o superior en versions anteriors s'utilitzava Dalvik, proporciona a cada app la seua pròpia instància de la màquina virtual. Algunes de les principals funcions d'ART són:
    - La compilació anticipada (AOT) transforma el bytecode a codi natiu
    - La compilació en temps d'execució (JIT) transforma el bytecode en temps d'execució.
    - Recol·lector d'escombraries (GC)
    - Millores amb la compatibilitat per a la depuració de codi.
  - Una sèrie de **biblioteques** escrites en C/C++ utilitzades per Android que proporcionen la major parts de les seues capacitats.
  - El **framework de la Java API** per a les aplicacions on es troben tots els components accessibles per aquestes. Totes les funcions del SO Android estan disponibles mitjançant API escrites amb el llenguatge Java.
  - Les **apps del sistema** com el marcadore telefònic, els contactes, el navegador, etc.. Les apps del sistema funcionen com les apps del usuari.





## Fonaments de l'aplicacions (I)

- Les aplicacions natives d'Android estan escrites en el llenguatge de programació Java.
- El SDK d'Android compila tot el codi amb totes les dades i arxius de recursos en un paquet Android, un arxiu d'extensió **.apk**.
- Tot el codi dins d'un arxiu .apk es considera una aplicació i és el que usen els dispositius per a instal·lar aquesta aplicació.
- Una vegada instal·lada en el dispositiu, cada aplicació corre sota la seua **pròpia àrea de seguretat**.



## Fonaments de l'aplicacions (II)

- El sistema operatiu Android és un sistema Linux multi-usuari en el qual **cada aplicació** es considera un **usuari diferent**.
- Per defecte, el sistema assigna a cada aplicació un **únic ID** d'usuari Linux (L'ID l'usa només el sistema i l'aplicació no el coneix). El sistema assigna **permisos** per a tots els arxius de l'aplicació perquè **només l'ID d'usuari assignat a aqueixa aplicació** tinga accés a ells.
- **Cada procés té el seu pròpi equip virtual (EV)**, per la qual cosa el codi d'una aplicació s'executa de forma aïllada d'altres aplicacions.
- Per defecte, cada aplicació s'executa en **el seu propi procés Linux**. Android inicia el procés quan qualsevol dels components de l'aplicació necessita ser executat, a continuació **finalitza el procés quan ja no siga necessari** que segueixca en execució **o quan el sistema necessita recuperar la memòria** per a altres aplicacions.





## Compartir dades entre aplicacions

- És possible que dues aplicacions compartisquen el mateix ID d'usuari Linux, en tal cas podran accedir als arxius de l'altra aplicació.
- Per a conservar els recursos del sistema, les aplicacions amb el mateix ID poden executar-se en el mateix procés i compartir el mateix EV, per a això han d'estar signades amb el mateix certificat.
- Una aplicació pot sol·licitar permisos per a accedir a dades del dispositiu com els contactes, SMS, la targeta SD, càmera, bluetooth i més. Els permisos de l'aplicació depenen de la versió d'Android es poden concedir en la instal·lació d'aquesta, o bé en el mateix moment que són usats. L'usuari decideix si els concedeix o els denega.



## Components d'una aplicació

- Una aplicació Android està formada per un o diversos components de diferents tipus.
- Alguns depenen els uns dels altres, però cadascun existeix com a **entitat pròpia** i exerceix un paper específic en el qual cadascun és un element únic que ajuda a definir el comportament global de l'aplicació.
- Alguns components poden actuar com a **punts d'entrada** a l'aplicació.
- Hi ha **quatre tipus** principals de components. Cada tipus és per a un propòsit en concret i té un cicle de vida diferent, que defineix com es crea i destrueix el component.



## Tipus de components

- **Activities:** Una Activitat representa una **pantalla amb una interfície d'usuari**. Encara que les activitats treballen juntes de forma coherent, cadascuna és independent de l'altra. Per tant, una aplicació diferent pot iniciar qualsevol d'aquestes activitats.
- **Services:** Un servei és un component que s'executa en **segon pla** per a dur a terme **operacions de llarga durada** o per a realitzar treballs de processos remots. Un servei no proporciona una interfície d'usuari.
- **Content providers:** gestionen un conjunt de **dades compartides** d'una aplicació. Mitjançant ells altres aplicacions poden consultar o fins i tot modificar la informació (Si el proveïdor de contingut ho permet).
- **Broadcast receivers:** **responen a l'emissió d'un missatge** en el sistema, per exemple, una emissió que anuncia que la pantalla s'ha apagat, la bateria està baixa o que s'ha tirat una foto.





## Activació de components

- Qualsevol aplicació pot iniciar un component d'una altra aplicació. Per exemple, si volem que l'usuari prengui una foto amb la càmera del dispositiu, probablement hi ha una altra aplicació per a això i la teua aplicació pugui usar-la, en lloc de desenvolupar una activitat per a prendre una foto.
- Els components s'activen mitjançant missatges asíncrons anomenats **Intent** (excepte els content providers) que poden ser implícits o explícits.
- Per a activities i services l'Intent defineix l'acció a realitzar.
  - `startActivity()`, `startActivityForResult()` passant-li un Intent.
  - `startService()`, `bindService()` passant-li un Intent.
- Per als broadcast receivers l'Intent defineix l'anunci del broadcast.
  - Podem enviar un broadcast passant un Intent a `sendBroadcast()`, `sendOrderedBroadcast()`.



## Crear una aplicació (I)

- Des de la pantalla de benvinguda premerem l'opció “***Start a new Android Studio project***” per a iniciar l'assistent de creació d'un nou projecte.
  - Si ja havíem obert anteriorment Android Studio és possible que s'òbriga directament l'aplicació principal en lloc de la pantalla de benvinguda. En aqueix cas accedirem al menú “***File / Newproject ...***” per a crear el nou projecte.
- L'assistent de creació del projecte ens guiarà per les diferents opcions de creació i configuració d'un nou projecte Android.



## Crear una aplicació (II)

- En la primera pantalla indicarem, per aquest ordre:
  - **Application name:** el nom que li volem posar a l'aplicació, per tal d'evitar problemes sols hem d'utilitzar caracters alfanumèrics i números, evitant utilitzar accents.
  - **Company domain:** el domini de l'empresa. Si anem a publicar a Google Play és molt important que aquest paràmetre siga únic ja que de lo contrari tindrem problemes. Si no tenim nom de domini propi, podem posar un que pensem que puga ser únic, per exemple el nostre «nom\_y\_cognoms.com».
  - **Project location:** l'ubicació on volem que Android Studio cree el projecte.
  - **Package name:** aquest paràmetre es calcula automàticament a partir del nom de l'aplicació i el package name (nomDomini.nomAplicació tot en minúscules). Google enmagatzenarà els projectes que publiquem al Google Play en una estructura de carpetes en funció del nom de paquet. Per aixó és important que siga únic.
  - **Include C++ support:** si partim d'un projecte que té codi en C++ podem marcar aquesta opció per a que el nostre projecte suporti C++.
  - **Include Kotlin support:** si volem utilitzar Kotlin al nostre projecte marcarem aquesta opció.



## Crear una aplicació (III)

Create New Project

Create Android Project

**Application name**  
My Application

**Company domain**  
ggascon.example.com

**Project location**  
/home/ggascon/AndroidStudioProjects/MyApplication ...

**Package name**  
com.example.ggascon.myapplication Edit

☐ Include C++ support

☐ Include Kotlin support

Previous Next Cancel Finish



## Crear una aplicació (IV)


- La següent pantalla de configuració ens demana que indiquem a quins tipus de dispositius volem donar suport i a partir de quina versió d'Android (API).
- Podem donar suport als següents tipus de dispositius:
  - **Phone a Tablet:** telèfons i tablets.
  - **Wear:** els dispositius wereables, principalment rellotges i polseres d'activitat.
  - **TV:** les smartTV que utilitzen Android com a SO.
  - **Android Auto:** si volem que la nostra aplicació pugui ser controlada desde cotxe.
  - **Android Things:** la nova plataforma de Google per a incloure aquells dispositius que es poden connectar a Internet (Internet de les Coses) però no són ordinadors a l'ús. Per exemple: les bombetes de llum que podem controlar la intensitat i el color desde Internet, o una càmera de fotos que tingui connexió a Internet, etc.





## Crear una aplicació (V)

Create New Project

Target Android Devices

### Select the form factors and minimum SDK

Some devices require additional SDKs. Low API levels target more devices, but offer fewer API features.

☒ **Phone and Tablet**

API 15: Android 4.0.3 (IceCreamSandwich)

By targeting **API 15 and later**, your app will run on approximately **100%** of devices. [Help me choose](#)

☐ Include Android Instant App support

☐ **Wear**

API 21: Android 5.0 (Lollipop)

☐ **TV**

API 21: Android 5.0 (Lollipop)

☐ **Android Auto**

☐ **Android Things**

API 24: Android 7.0 (Nougat)

Previous

Next

Cancel

Finish

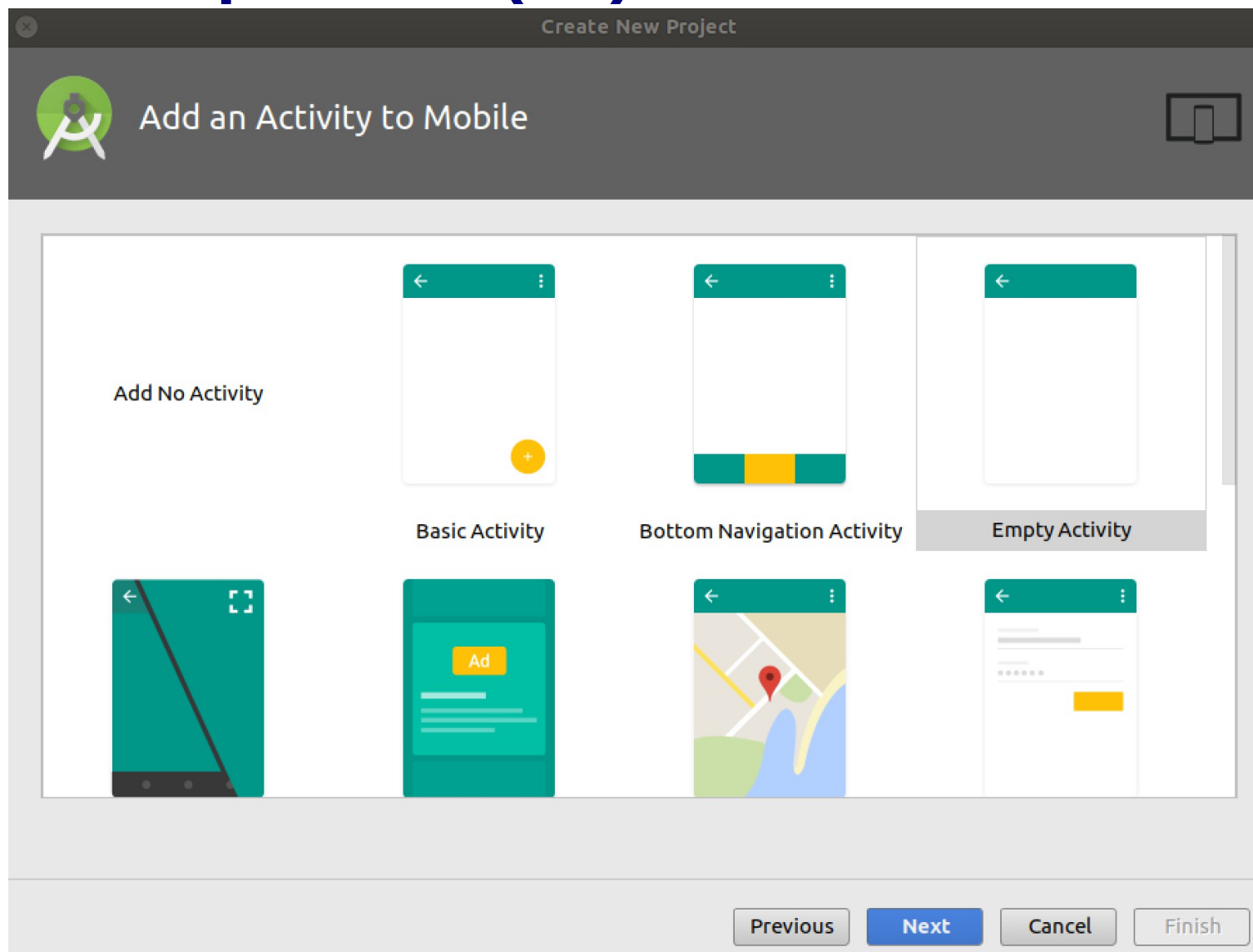


## Crear una aplicació (VI)

- En la següent pantalla podem escollir l'esquelet inicial de l'aplicació, és a dir, si volem que ens cree una pantalla inicial o no. Hi ha moltes opcions, algunes d'elles són:
  - **Add No Activity:** sense pantalla inicial.
  - **Basic Activity:** una pantalla inicial amb un botó.
  - **Empty Activity:** crea una pantalla inicial però sense cap contingut.
  - **Login Activity:** una pantalla per introduir usuari i contrasenya.



## Crear una aplicació (VII)





## Crear una aplicació (VIII)

- En el següent pas de l'assistent indicarem:
  - **Activity Name:** el nom que li volem posar a la pantalla inicial
  - **Layout Name:** el nom que li volem posar al fitxer que conté el disseny de la pantalla. En Android cadascuna de les pantalles té associat com a mínim dos fitxers, una classe Java i un fitxer xml on es defineixen els components que conté i la seua disposició.
- Si marquem l'opció **Backwards Compatibility (AppCompat)** estarem donant support a alguns components definits en API's posteriors incloent les llibreries als nostre projecte. Convé tenir activada aquesta opció.



## Crear una aplicació (IX)

Create New Project

Configure Activity

Creates a new empty activity

Activity Name: MainActivity

☒ Generate Layout File

Layout Name: activity\_main

☒ Backwards Compatibility (AppCompat)

If true, a layout file will be generated

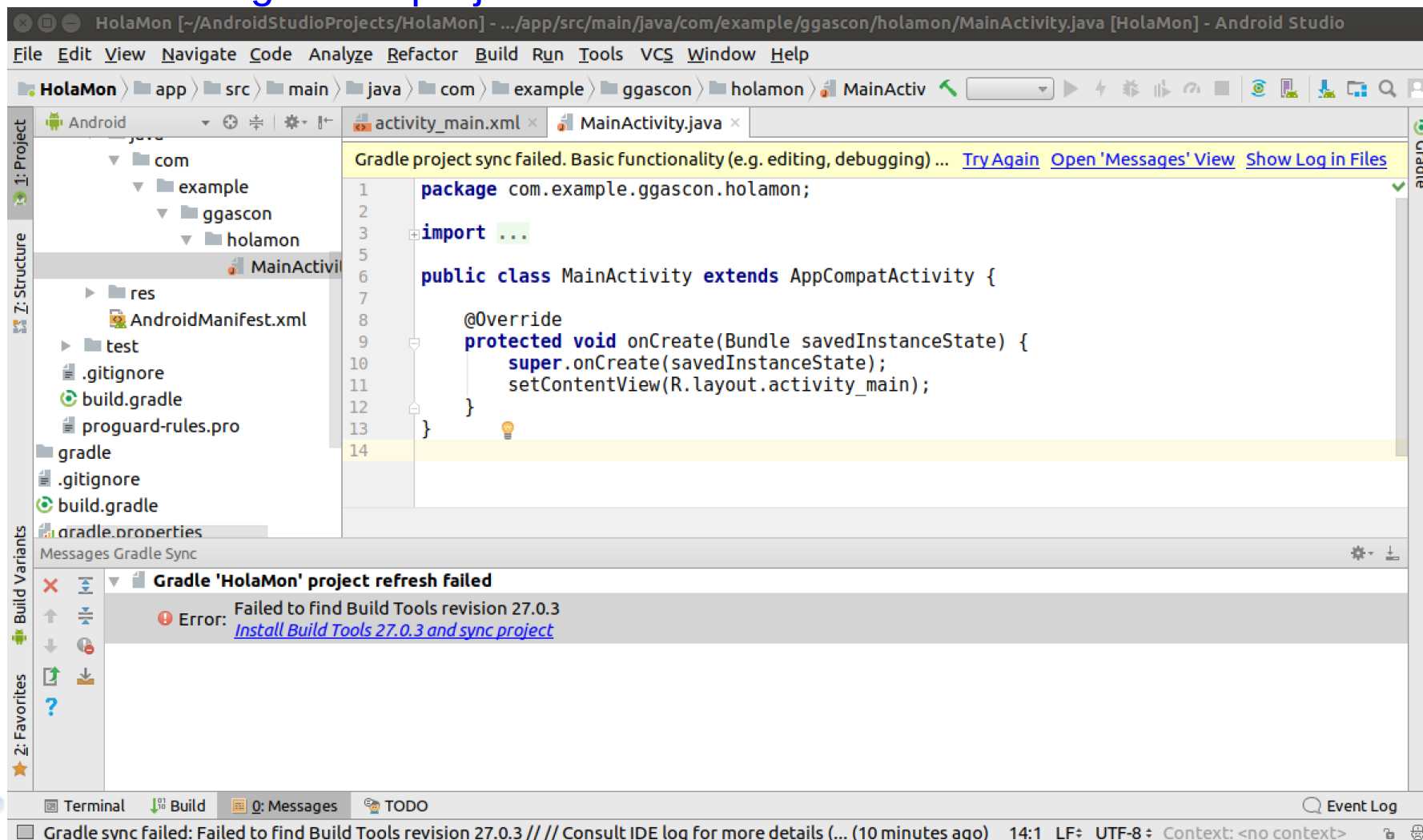
Previous Next Cancel Finish





## Crear una aplicació (X)

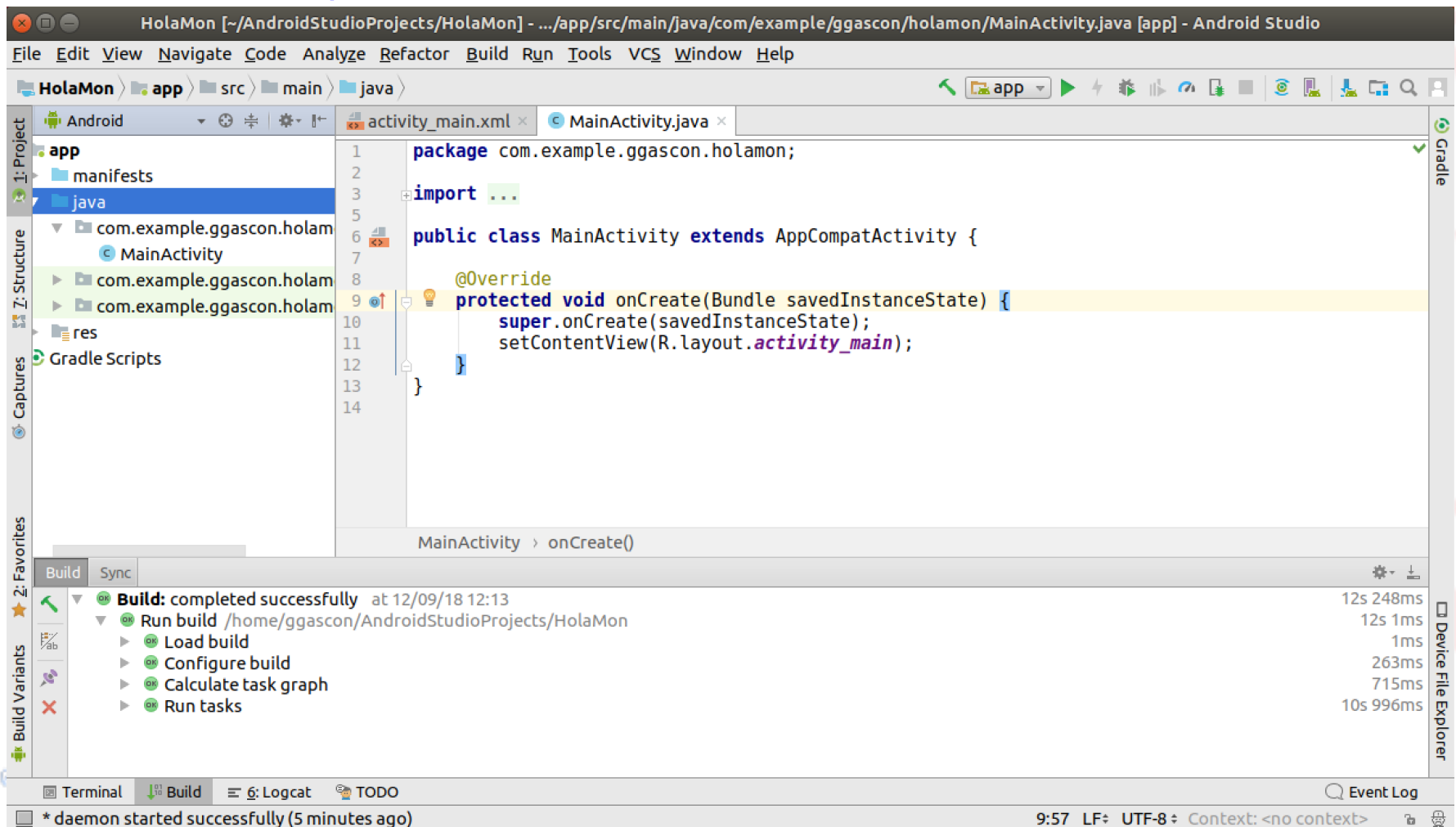
- Finalment, s'obrirà en Android Studio la classe principal de la pantalla que em creat al configurar el projecte.





## Crear una aplicació (XI)

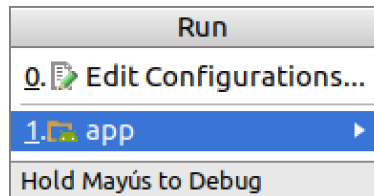
- Si apareix el missatge Error: Failed to find Build Tools revision ..., es perquè no tenim instal·lat el paquet necessari, només em de polsar a l'enllaç que apareix baix per a instal·lar-lo i desapareixerà l'error.





## Executar i provar la nostra aplicació

- Una vegada creat el nostre projecte, anem a executar la nostra aplicació. Per a fer-ho anem al menú Run → Run ...
- S'obrirà una pantalla on podem seleccionar que és el que volem executar. Seleccionen app.

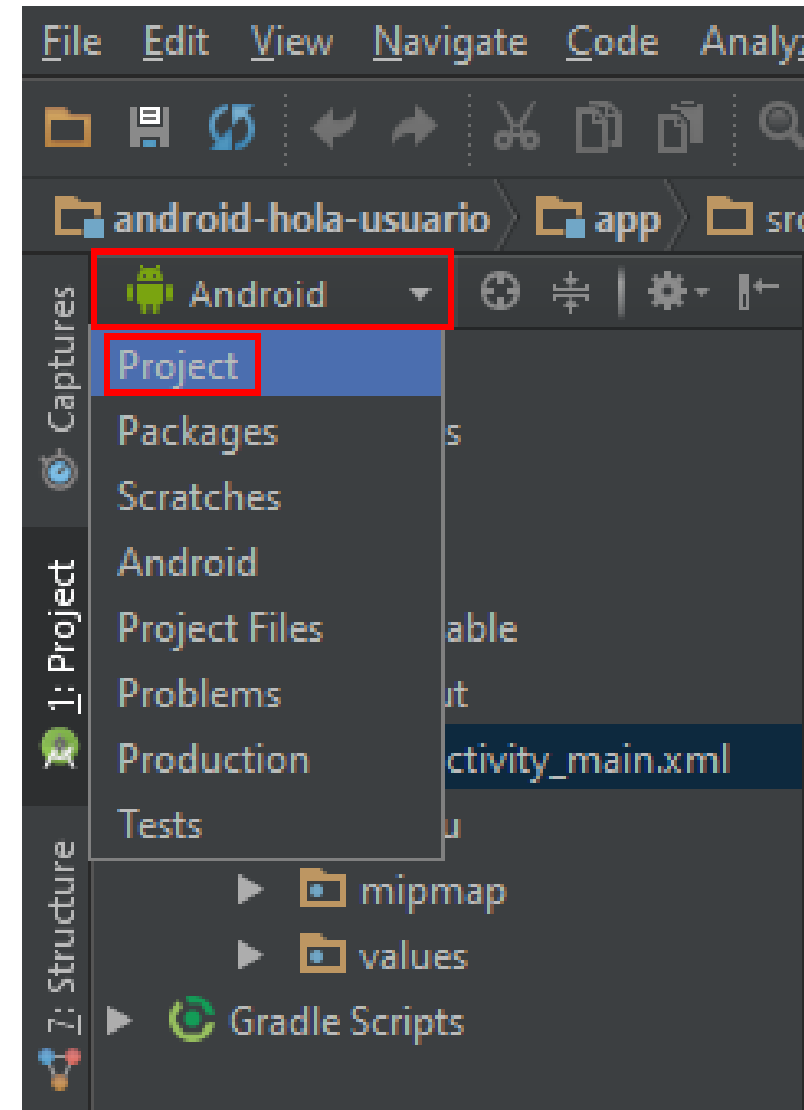


- Per a executar una aplicació en Android tenim principalment 2 opcions:
  1. Executar l'aplicació en un dispositiu real fent ús d'un cable USB (requereix que el sistema operatiu tinga els drivers)
  2. Executar l'aplicació en un emulador.



## Estructura d'un projecte (I)

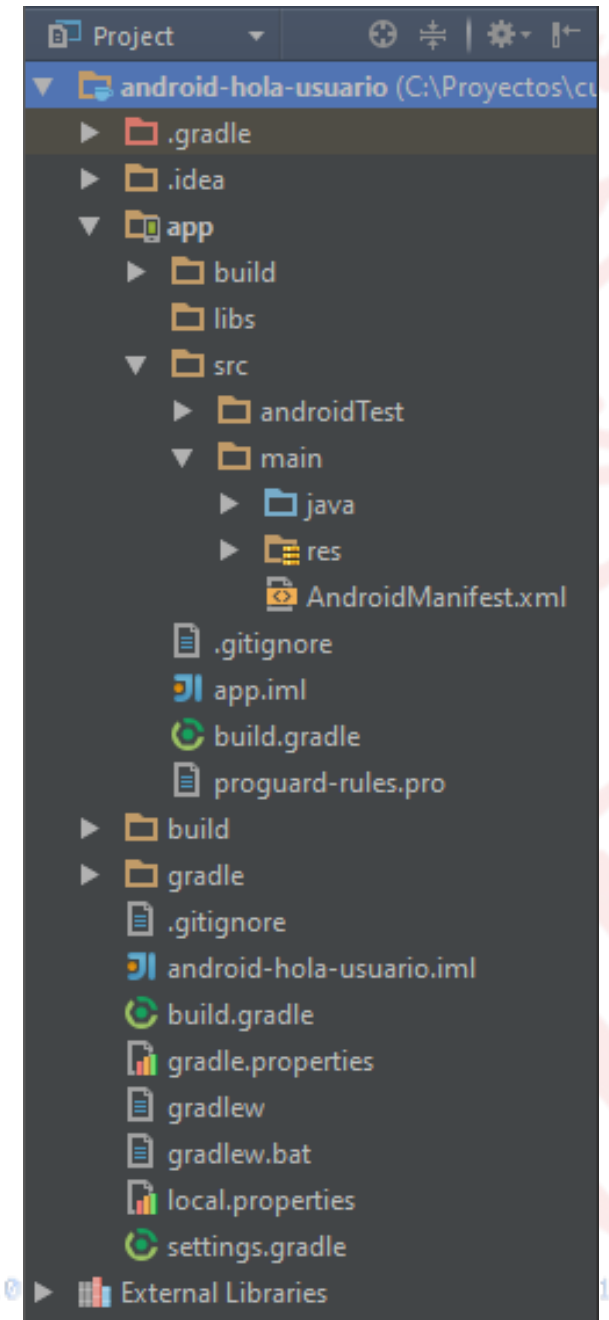
- Per a entendre millor l'estructura **del projecte** canviarem momentàniament la forma en la qual Android Studio ens la mostra. Per a això, premerem sobre la llista desplegable situada en la part superior esquerra, i canviarem la vista de projecte a la manera **“Project”**.





## Estructura d'un projecte (II)

- Després de fer això, l'estructura del projecte canvia una mica d'aspecte i passa a ser com s'observa en la imatge.
- L'entitat projecte és **única**, i engloba a tots els altres elements.
- Dins d'un projecte podem incloure diversos **mòduls**, que poden representar aplicacions diferents, versions diferents d'una mateixa aplicació, o diferents components d'un sistema (aplicació mòbil, aplicació servidor, llibreries, ...).
- En la majoria dels casos, treballarem amb un projecte que contindrà només un mòdul corresponent a la nostra aplicació principal, en aquest cas el **mòdul app**.







## Estructura d'un projecte (III)

- Carpeta **/app/src/main/java** conté tot el **codi font** de l'aplicació, classes auxiliars, etc. Inicialment, Android Studio crearà per nosaltres el codi bàsic de la pantalla (activitat o activity) principal de l'aplicació, que recordem que en el nostre cas era MainActivity, i sempre sota l'estructura del paquet java definit durant la creació del projecte.



## Estructura d'un projecte (IV)

- Carpeta **/app/src/main/res/** conté tots els fitxers de recursos necessaris per al projecte: imatges, layouts, cadenes de text, etc.
- Els diferents tipus de recursos es poden distribuir entre les següents subcarpetas:
  - **res/drawable/** imatges i altres **elements gràfics** usats per l'aplicació per a **diferents densitats**.
  - **res/mipmap/** **icones** de llançament de l'aplicació (la icona que apareixerà en el menú d'aplicacions del dispositiu) per a les **diferents densitats** de pantalla existents
  - **res/layout/** fitxers de definició XML de les diferents **pantalles** de la interfície gràfica depenent de la seua orientació (/layout vertical, /layout-land horitzontal).
  - **res/anim/** definició de les **animacions** utilitzades per l'aplicació.
  - **res/color/** arxius XML de definició de **llistes de colors**.
  - **res/menu/** arxius XML dels **menús** de l'aplicació
  - **res/xml/** arxius XML de **dades**.
  - **res/values/** arxius XML de **recursos** de l'aplicació (strings.xml, styles.xml, ...).
  - **res/raw/** **recursos addicionals**, en format diferent de XML que no s'incloguen en la resta de carpetes.



## Estructura d'un projecte (IV)

- Fitxer **/app/src/main/AndroidManifest.xml** conté la definició en XML de molts dels **aspectes principals de l'aplicació**, com per exemple la seua identificació (nom, icona, ...), els seus components (pantalles, serveis, ...), o els permisos necessaris per la seua execució. En següents diapositives veurem més detalls d'aquest fitxer.
- Fitxer **/app/build.gradle** informació necessària per a la **compilació del projecte**, per exemple la versió del SDK d'Android utilitzada per a compilar, la mínima versió d'Android que suportarà l'aplicació, referències a les llibreries externes utilitzades, etc.
  - En un projecte poden existir diversos fitxers build.gradle, per a definir determinats paràmetres a diferents nivells. Per exemple, en el nostre projecte podem veure que existeix un fitxer build.gradle a nivell de projecte, i un altre a nivell de mòdul dins de la carpeta /app. El primer d'ells definirà paràmetres globals a tots els mòduls del projecte, i el segon només tindrà efecte per a cada mòdul en particular.



## Estructura d'un projecte (V)

- Carpeta **/app/libs/** pot contindre les **llibreries java externes** (fitxers .jar) que utilitze la nostra aplicació.
  - Normalment no inclourem directament ací cap llibreria, sinó que farem referència a elles en el fitxer build.gradle descrit en el punt anterior, de manera que entren en el procés de compilació de la nostra aplicació.
- Carpeta **/app/build/** conté una sèrie d'elements de codi **generats automàticament** en compilar el projecte, entre ells un arxiu anomenat **R.java** on es defineix la classe R que contindrà en tot moment una sèrie de constants amb **els identificadors (ID) de tots els recursos de l'aplicació inclosos en la carpeta /app/src/main/res/**, de manera que puguem accedir fàcilment a aquests recursos des del nostre codi java a través d'aquesta dada. **No s'han de modificar manualment.**



## El fitxer: AndroidManifest.xml (I)

- Perquè Android pugui iniciar un **component** ha de saber prèviament que existeix llegint l'arxiu AndroidManifest.xml.
- Ha de residir en **el directori arrel** de l'aplicació.
- Indica els **permisos** que requereix l'aplicació per a funcionar.
- Pot declarar l'API **mínim** que l'aplicació requereix, encara que en les últimes versions és més recomanable fer-ho mitjançant l'arxiu build.gradle.
- També es poden indicar els **requisits programari i maquinari** de l'aplicació. Ej: NFC, càmera, etc.
- **API de llibreries** que necessiten ser incloses en l'aplicació. Ej: Google Maps Library.





## El fitxer: AndroidManifest.xml (II)

- `<application>` l'element `android:icon` fa referència a la icona principal de l'aplicació.
- Els **components** s'han de declarar de la següent forma:
  - `<activity>` per a les activitats. L'element `android:name` especifica la classe java que serà una subclasse d'Activity. L'element `android:label` especifica nom visible per a l'usuari.
  - `<service>` per als services.
  - `<receiver>` per als broadcast receivers.
  - `<provider>` per als content providers.



## El fitxer: AndroidManifest.xml (III)

- Les activities, services i content providers que formen part del codi font però no s'indiquen en el manifest, no seran visibles per al sistema i per tant, no podran ser instanciades.
- Els broadcast receivers poden declarar-se en el manifest o bé poden crear-se dinàmicament des de codi Java, utilitzant objectes de tipus `BroadcastReceiver`, i registrar-los mitjançant el mètode *registerReceiver()*.



## El fitxer: AndroidManifest.xml (IV)

- `<uses-permission>` mitjançant aquesta etiqueta especifiquem els permisos que necessitarà la nostra aplicació per a poder executar-se, a més són els que haurà d'acceptar l'usuari abans d'instal·lar-la. Per exemple, si es desitja utilitzar funcionalitats amb Internet o el vibrador del telèfon, cal indicar que la nostra aplicació requereix aqueixos permisos.
- Podem trobar una llista de tots els permisos aplicables en <http://developer.android.com/reference/android/manifest.permission.html>
- Exemple:
  - `<uses-permission  
android:name="android.permission.WRITE_EXTERNAL_STORAGE" />`



## El fitxer: AndroidManifest.xml (IV)

- **<intent-filter>** especifica els tipus d'intents als quals una activity, service o broadcast receiver pot respondre.
- Ha de contindre almenys un element **<action>** on s'especifique l'acció que l'element que conté l'intent-filter ha de realitzar.
- A més pot contindre els elements:
  - **<data>** on s'especifiquen les dades amb les quals ha de treballar mitjançant una URI.
  - **<category>** especifica informació addicional sobre el tipus de component que ha de manejar l'intent.



## Exemple: AndroidManifest.xml

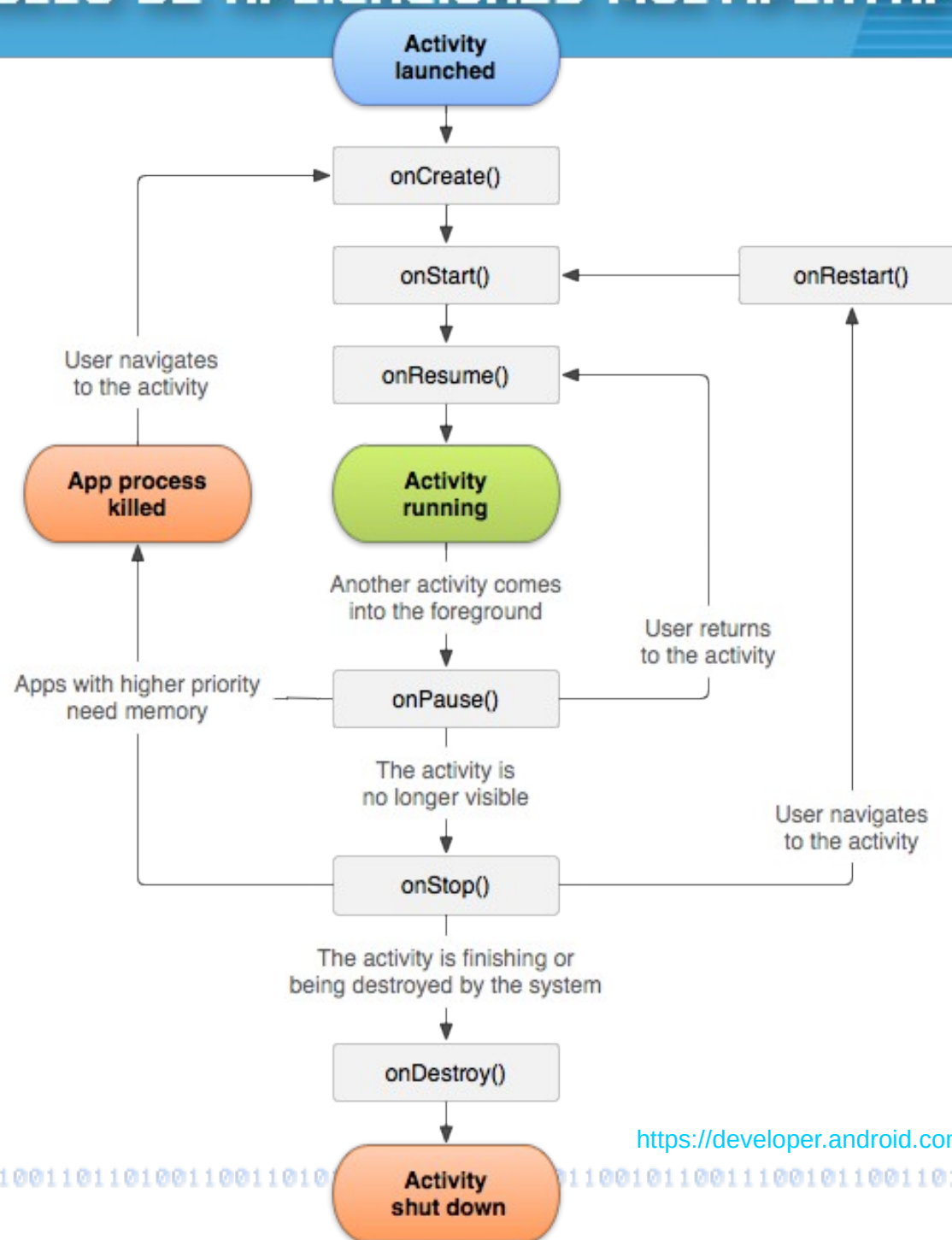
```
1  <?xml version="1.0" encoding="utf-8"?>
2  <manifest package="com.germangascon.holamon"
3          xmlns:android="http://schemas.android.com/apk/res/android">
4
5      <application
6          android:allowBackup="true"
7          android:icon="@mipmap/ic_launcher"
8          android:label="HolaMon"
9          android:roundIcon="@mipmap/ic_launcher_round"
10         android:supportsRtl="true"
11         android:theme="@style/AppTheme">
12         <activity android:name=".MainActivity">
13             <intent-filter>
14                 <action android:name="android.intent.action.MAIN"/>
15
16                 <category android:name="android.intent.category.LAUNCHER"/>
17             </intent-filter>
18         </activity>
19     </application>
20
21 </manifest>
```



## Cicle de vida d'una Activity (I)

- El **sistema operatiu** és l'encarregat de **pausar, parar o destruir la nostra aplicació** segons les necessitats de recursos del dispositiu, encara així, nosaltres com a desenvolupadors hem d'aprendre a controlar tots aquests esdeveniments per a fer les nostres aplicacions robustes i millorar el rendiment dels telèfons.
- Per això, és necessari entendre el cicle de vida d'una activity i d'aquest manera no tindre problemes de pèrdua de dades.







## Cicle de vida d'una Activity (II)

- **onCreate():** Es dispara quan l'Activity és cridada per primera vegada. Ací és on hem de crear la inicialització normal de l'aplicació, crear vistes, fer els bind de les dades, etc. A este mètode se li passa un objecte agrupat que conté l'estat anterior de l'Activity que es va tancar, si es que existeix. Després d'aquest mètode sempre s'invoca a l'onStart().
- **onRestart():** S'executa quan l'Activity ha sigut parada, i volem tornar a utilitzar-la. En el diagrama es pot veure que després d'un onStop() s'executa l'onRestart() i immediatament s'invoca a un onStart().



## Cicle de vida d'una Activity (III)

- **onStart():** S'executa **just abans que l'Activity es faça visible** per a l'usuari. Seguit per onResume() si l'Activity passa a primer pla, o per onStop() si s'oculta.
- **onResume():** S'executa quan l'usuari pot començar a interactuar amb l'Activity. En aquest moment l'Activity està en la part més alta de la pila d'Activities. Sempre seguit pel mètode onPause().
- **onPause():** S'executa quan **el sistema està a punt de rependre altra Activity**. Cal procurar que **la invocació** a aquest mètode **sigi ràpida** ja que fins que no s'acabe la seua execució no es podrà arrancar la nova Activity. Després d'aquesta invocació pot venir un onResume() si l'Activity torna a aparèixer en primer pla o un onStop() si es fa invisible per a l'usuari.



## Cicle de vida d'una Activity (IV)

- **onStop():** S'executa quan l'Activity **ja no és visible per a l'usuari** perquè una altra Activity ha passat a primer pla. Si veiem el diagrama, després que s'ha executat aquest mètode ens queden tres opcions: executar l'onRestart() perquè l'Activity torne a aparèixer en primer pla, que el sistema elimine aquest procés perquè altres processos requereixen memòria o executar l'onDestroy() per a apagar l'aplicació.
- **onDestroy():** Aquest és l'últim mètode que executarà l'Activity, després d'aquesta, és totalment destruïda. Això pot passar perquè els requeriments de memòria que tinga el sistema o perquè de manera explícita s'ha invocat el mètode finish() d'aquesta Activity. Si vullguerem tornar a executar l'Activity s'arrancaria un nou cicle de vida.
  - **COMPTE!:** si hi ha poca memòria, és possible que l'activitat es destrueixca sense cridar a aquest mètode.



## Exercici

- En aquest exercici implementarem tots els mètodes del cicle de vida de l'activitat MainActivity i afegirem un **Toast** per a mostrar quan s'executa. D'aquesta forma comprendrem millor quan s'invoca a cada mètode.
  - Obri l'Activity principal del projecte.
  - Afig en cadascun dels mètodes del cicle de vida: `Toast.makeText(this, "onCreate", Toast.LENGTH_SHORT).show();`  
*Canviant el text "OnCreate" pel qual corresponga en cada mètode.*
  - Executa l'aplicació i observa la seqüència Toast.
  - Prem el botó Home, inicia qualsevol altra aplicació i observa la seqüència Toast.
  - Torna a executar la nostra aplicació i observa la seqüència Toast.
  - Ix de l'aplicació prement el botó arrere i observa la seqüència Toast.