Introducció a Android

Projecte Integrat de Software (PIS)

Universitat de Barcelona

victor.campello@ub.edu
carlos.martinisla@ub.edu

15-17 de Febrer de 2022

Contingut

Projecte Integrat de Software

En què consisteix?

- Desenvolupament d'un projecte de software de dimensions mitjanes durant el curs.
- Consisteix en una aplicació Android de temàtica lliure.
- Es realitzarà en grups de 3 o 4 alumnes.
- Implica l'estudi del projecte, la implementació, la verificació i l'avaluació del seu rendiment.

Sessions pràctiques

A les sessions de pràctiques hi haurà dues fases:

1. Fase inicial: introducció i explicació de conceptes pràctics – aspectes bàsics d'Android, disseny de la interfície d'usuari, navegació, patrons de disseny, bases de dades, etc.
2. Fase grupal: treball en el projecte i seguiment per part del professor. Inclou resolució de dubtes i errors de desenvolupament.

Calendari de pràctiques

- 1a setmana: Introducció a Android
- 2a setmana: Layouts
- 3a setmana: Patrons de disseny
- 4a setmana: Navegació i base de dades
- A partir de la 5a setmana: treball en grup i seguiment

Fase grupal

A partir de la cinquena setmana passareu a treballar completament en el projecte.

- Farem una **reunió personalitzada per a cada grup**. A la reunió discutirem la pròpia organització de tots els integrants, les tasques assignades, el progrés d'aquestes i les responsabilitats de cadascú.
- Teniu en compte que la nota serà grupal, però hi haurà un **factor de rendiment individual** avaluat pels companys de grup i el professor de pràctiques.

Eines per organitzar el projecte

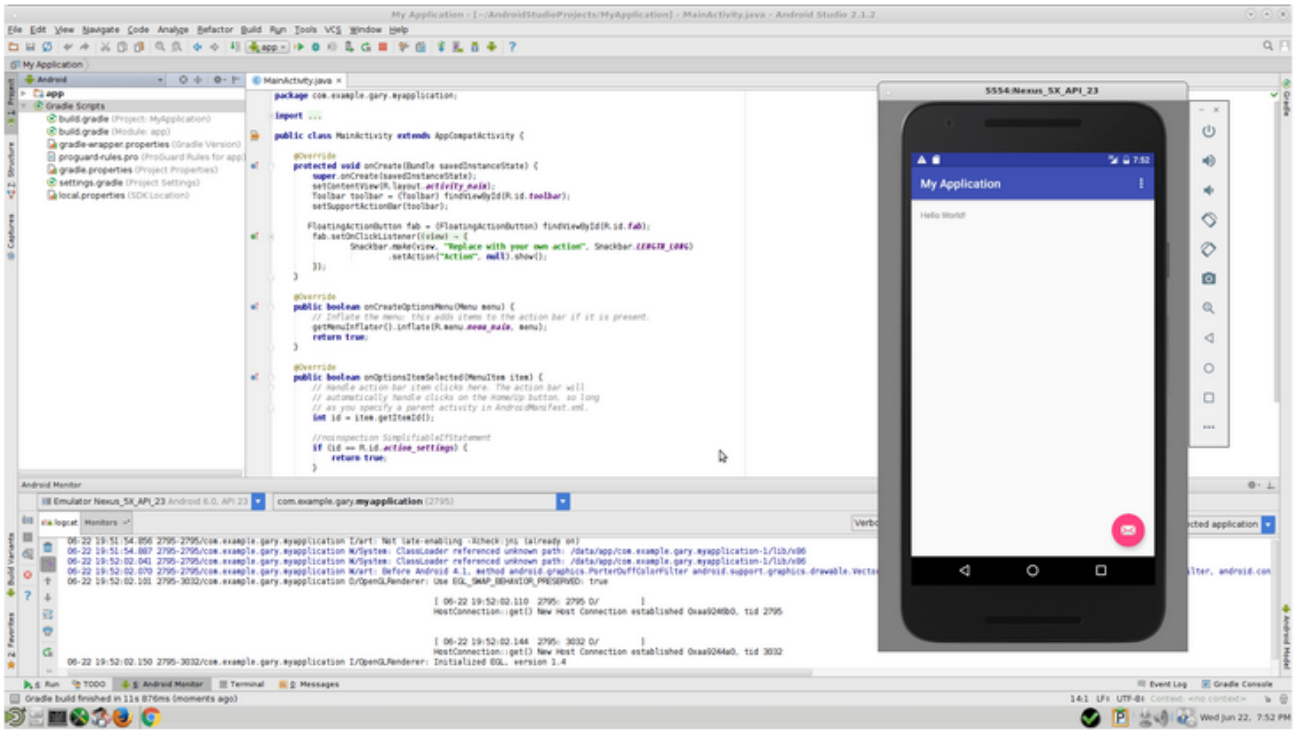
En aquesta assignatura utilitzarem eines específiques pel desenvolupament de software Android (que veurem a continuació). També utilitzarem eines que ajuden al desenvolupament i organització de projectes de software: eines per al control de versions (git) i eines per a l'organització (Trello).

Per què Android?

- Android és un sistema operatiu de **codi obert** per a dispositius mòbils, amb un nucli basat en Linux.
- Te una **quota de mercat majoritària** (vora el 70%, segons statista) – important per arribar als potencials usuaris de les aplicacions.
- És **senzill per desenvolupar** i portar el producte des del banc de treball fins al dispositiu final (el vostre o la Play Store).
- **L'ús del mòbil és cada vegada major** en comparació amb el del PC. És el dispositiu més important per a moltes empreses.

Android Studio

Pel desenvolupament del codi utilitzarem Android Studio. Un IDLE (*Integrated Development and Learning Environment*) basat en IntelliJ IDEA.



El podeu obrir als vostres ordinadors... ⌨

Primeres passes amb Android

Treballareu en Android Studio i programareu amb Java. Pel disseny d'una app, cal tenir en compte els següents aspectes fonamentals:

1. Components d'una aplicació
2. Cicle de vida d'una aplicació
3. Compatibilitat de dispositius
4. Execució de l'aplicació
5. Recursos de l'aplicació
6. Manifest

Components d'una aplicació

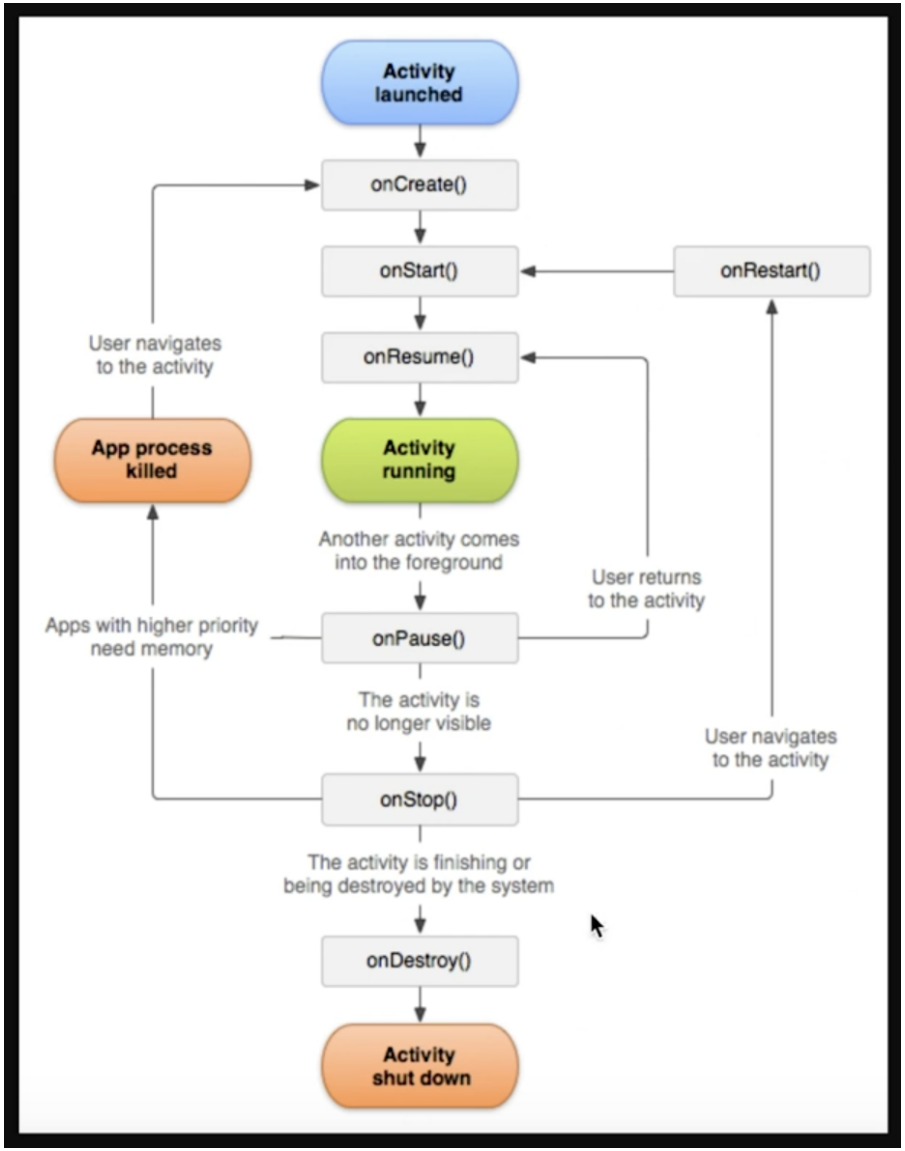
Una aplicació d'Android consisteix en un conjunt de components lligats entre si per un fitxer, anomenat *Manifest*, que descriu cada component i com interactuen entre ells.
Hi ha quatre components principals o punts d'entrada a l'aplicació:

1. **Activitats:** Component visual de l'aplicació. Rep l'input de l'usuari. Les interfícies d'usuari estaran formades per classes que hereden de *Activity* i contindran *Views* i *Fragments*, que veurem més endavant.
2. **Serveis:** Component que s'executa sense interfície d'usuari. S'utilitza per a execucions llargues com actualitzar fonts de dades, (des)carregar un fitxer o escoltar música.

Components d'una aplicació

3. **Receptors d'emissió** (*Broadcast Receivers*): És un component de possibilita que el sistema entregui events a l'aplicació fora d'un flux d'usuaris habituals. Això permet que l'aplicació respongui a anuncis d'emissió de tot el sistema.
4. **Proveïdors de contingut** (*Content Providers*): Administra un conjunt compartit de dades de l'aplicació que es poden emmagatzemar en un sistema d'arxius, en una base de dades SQlite, a la web o a qualsevol recurs d'emmagatzematge local al que tingui accés l'aplicació.

Cicle de vida d'una activitat



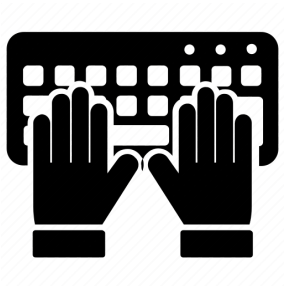
Cicle de vida d'una activitat

Una activitat implementa per defecte les següents funcions:

- *onCreate()* – Es crea l'activitat. Generada per l'assistent d'Android. Defineix tot el que necessita l'activitat.
- *onStart()* – Una vegada generada, l'activitat passa a pantalla.
- *onResume()* – La activitat està en pantalla i requereix la interacció de l'usuari.
- *onPause()* – Una altra activitat ocupa la pantalla principal. L'activitat iniciada queda en *stand by*.
- *onStop()* – L'activitat passa a segon pla.
- *onDestroy()* – L'activitat es tanca. S'alliberen recursos.

My first application

És el moment de crear la nostra primera aplicació d'Android.



Executar l'aplicació

Podem executar l'aplicació en un dispositiu real o en un emulador generat pel propi Android Studio.

- **Dispositu real**
  1. Seleccioneu l'app d'execució
  2. Seleccioneu a la barra d'eines el dispositiu on voleu realitzar l'execució de l'aplicació.
  3. Una vegada seleccionats els dos paràmetres anteriors, premeu RUN i el propi programa instal·larà l'aplicació.
- **Emulador**
  1. Seleccioneu **Tools** > **AVD Manager**.
  2. Seleccioneu el tipus de Hardware en el que desenvolupareu la vostra aplicació.
  3. Seleccioneu la **System Image** (Android 9.0, Android, 7.1, etc). Recordeu que heu de seleccionar una versió de sistema operatiu que sigui compatible amb el seleccionat previament al crear el projecte Android.

Recursos de l'aplicació

Els recursos son els arxius adicionals i el contingut estàtic que utilitza el vostre codi, com els mapes de bits, definicions de disseny, text d'interfície d'usuari, etc. Aspectes a tenir en compte:

- Cal mantenir-los externalitzats.
- S'organitzen a la classe R del teu codi.

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

Teniu la informació més detallada a: `https://developer.android.com/guide/topics/resources/providing-resources`

Tipus de recursos predeterminats - 1

Directorio	Tipo de recurso
`animator/`	Archivos XML que definen animaciones de propiedades.
`anim/`	Archivos XML que definen animaciones de interpolación de movimiento. En este directorio, también se pueden guardar animaciones de propiedades, pero se prefiere el directorio `animator/` para las animaciones de propiedades, a fin de distinguir entre los dos tipos.
`color/`	Archivos XML que definen una lista de estados de colores. Consulta la sección Recurso de lista de estado de colores.
`drawable/`	Archivos de mapas de bits (`.png`, `.9.png`, `.jpg` y `.gif`) o archivos XML que se han compilado en los siguientes subtipos de recursos de elemento de diseño: • Archivos de mapas de bits • Nueve parches (mapas de bits reajustables) • Listas de estados • Formas • Elementos de diseño de animaciones • Otros elementos de diseño Consulta la sección Recursos dibujables.
`mipmap/`	Archivos de elementos de diseño para diferentes densidades de los íconos de selectores. Para obtener más información sobre la administración de los íconos de selectores con carpetas `mipmap/`, consulta la sección Información general sobre la administración de proyectos.

Tipus de recursos predeterminats - 2

`layout/`	Archivos XML que definen el diseño de una interfaz de usuario. Consulta la sección Recurso de diseño.
`menu/`	Archivos XML que definen menús de aplicaciones, como un menú de opciones, un menú contextual o un submenú. Consulta la sección Recurso de menú.
`raw/`	Archivos arbitrarios para guardar sin procesar. Para abrir estos recursos con un objeto `InputStream` sin procesar, llama a `Resources.openRawResource()` con el ID del recurso, que es `R.raw.filename`. Sin embargo, si necesitas acceder a los nombres de los archivos originales y a la jerarquía de archivos, puedes considerar la posibilidad de guardar algunos recursos en el directorio `assets/` (en lugar de `res/raw/`). A los archivos de `assets/` no se les asigna un ID de recurso, por lo cual puedes leerlos solamente mediante `AssetManager`.
`values/`	Archivos XML que contienen valores simples, como strings, valores enteros y colores. Los archivos de recursos XML en otros subdirectorios `res/` definen un único recurso basado en el nombre del archivo XML, mientras que los archivos del directorio `values/` describen varios recursos. En el caso de un archivo de este directorio, cada campo secundario del elemento `<resources>` define un único recurso. Por ejemplo, un elemento `<string>` crea un recurso `R.string`, y un elemento `<color>` crea un recurso `R.color`. Dado que cada recurso se define con su propio elemento XML, puedes asignar el nombre que desees al archivo y colocar diferentes tipos de recursos en un archivo. Sin embargo, para mayor claridad, es recomendable que coloques tipos de recursos únicos en diferentes archivos. Por ejemplo, a continuación, se incluyen algunas convenciones de asignación de nombres de archivos para los recursos que puedes crear en este directorio: • arrays.xml para matrices de recursos (matrices escritas). • colors.xml para valores de color. • dimens.xml para valores de dimensión. • strings.xml para valores de strings. • styles.xml para estilos.

Tipus de recursos predeterminats - 3

`xml/`	Archivos XML arbitrarios que se pueden leer en tiempo de ejecución llamando a `Resources.getXML()`. Aquí se deben guardar diversos archivos de configuración XML, por ejemplo, una configuración que permite búsqueda.
`font/`	Archivos de fuentes, con extensiones como `.ttf`, `.otf` o `.ttc`, o archivos XML que incluyan un elemento `<font-family>`. Para obtener más información sobre las fuentes como recursos, ve a Fuentes en XML.

Recursos alternatius

En algunes circumstàncies, es requereix de recursos molt específics segons el terminal on s'executi l'aplicació. Els diferents idiomes, les diferents qualitats d'imatge o diferents connexions de xarxa poden ser exemples de recursos alternatius.

1. Crea en `res/` un directorio nuevo cuyo nombre tenga el formato `<resources_name>-<config_qualifier>`.
    - `<resources_name>` es el nombre del directorio de los recursos predeterminados correspondientes (definidos en la tabla 1).
    - `<qualifier>` es un nombre que especifica una configuración individual para la cual se deben usar estos recursos (que se definen en la tabla 2).

    Puedes agregar más de un `<qualifier>`. Separa cada uno con un guion.

En el següent link trobareu tots els exemples de recursos alternatius: `https://developer.android.com/guide/topics/resources/providing-resources`

Manifest de l'aplicació

L'arxiu Manifest descriu l'informació essencial de l'aplicació per les eïnes de creació de Android, el SO i Google Play. Tindrà el nom AndroidManifest.xml a l'arrel del projecte.

1. El nom del paquet de l'aplicació. Servirà posteriorment per identificar l'app a Google Play.
2. S'ha de declarar tots els components de l'aplicació i les seves característiques bàsiques.
3. Permisos per la interacció de la app amb altres components o apps del sistema. Per exemple, càmara del telèfon. Tots els permisos queden registrats al Manifest una vegada declarats, i es sobreescriuen automàticament.
4. Requisits de software i hardware. El Manifest és l'arxiu de lectura que permet establir els paràmetres de compatibilitat de la app amb el SO/terminal.

Importància del manifest

El manifest és probablement el document més important de la app. Antigament, es modificava manualment. Les noves actualitzacions d'Android permeten una actualització automàtica del document a mesura que es modifica o es fan canvis en el codi del projecte.

Referències

Documentació d'Android: developer.android.com/guide