Kotlin and Android

Projecte Integrat de Software (PIS)

Universitat de Barcelona

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

11-13 de Febrer de 2020

Overview

Calendari

Calendari Pràctiques Projecte Integrat de Software

Setmana	Pràctiques	Entregues
10-Feb	Introducció Kotlin + Android	
17-Feb	Layouts	Proposta individual app
24-Feb	Navegació	**E1 - Proposta Grup App**
2-Mar	Eines Treball en equip - Github/Trello	
9-Mar	Reunió setmanal equip	**E2 - User Interface**
16-Mar	Reunió setmanal equip	
23-Mar	Reunió setmanal equip	
30-Mar	**Setmana Examens**	
6-Abr	**Setmana Santa**	
13-Abr	Revisió UI - Reunió setmanal grup	**E3 - Disseny Software**
20-Abr	Revisió setmanal grup	
27-Abr	Reunió revisió DS	**E4 - DEMO**
4-May	Reunió setmanal equip	
11-May	Reunió setmanal equip	**Inter-group feedback**
18-May	Reunió revisió DEMO	
25-May	Reunió revisió DEMO	**Entrega FINAL**

Grups - Reunions

- Grups de màxim **4 persones**.
- Semanalment 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.
- La mini-reunió ha d'estar preparada amb antelació, ja que forma part dels elements avaluables.
- Durant la reunió podem fer preguntes individuals per discernir la contribució de cada integrant al projecte.

Avaluació

MOLT IMPORTANT L'avaluació no només consta del propi projecte. Altres elements importants per a la evaluació són:

- Organització i documentació.
- Implicació de tots els integrants.
- Presentació dels progressos, dificultats i solucions.

És responsabilitat de cada grup assegurar-se que tots els integrants responen i participen al projecte.

Eines per organitzar projectes

En aquesta assignatura utilitzarem eines especifiques pel desenvolupament de software Android (més endavant). També utilitzarem eines que ajuden al desenvolupament y organització de projectes de software, com GitHub o Trello.

Breve historia de Kotlin

- Creat en 2011 per l'equip de JetBrains (IntelliJ IDEA, PyCharm, etc)
- Està disenyat per a operar integrament amb Java.
- Google I/O 2017 va anomenar Kotlin llenguatge de programació oficial per a Android juntament amb Java.
- És un llenguatge orientat a objecte.
- La sintaxi respecte a Java te diferencies. Els creadors de Kotlin buscaven un llenguatge més simplificat.

Herramientas - IDLE

Pel desenvolupament de Kotlin s'utilitzen diferents IDLE. Podeu utilitzsr el més afí a les vostres característiques. A les pràctiques utilitzarem Android Studio o IntelliJ IDEA.

Figure: Software for PIS

Link for Kotlin sintaxis

https://kotlinlang.org/docs/reference/basic-syntax.html

First steps with Android

Treballarem en Android Studio i el codi serà en Kotlin. 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

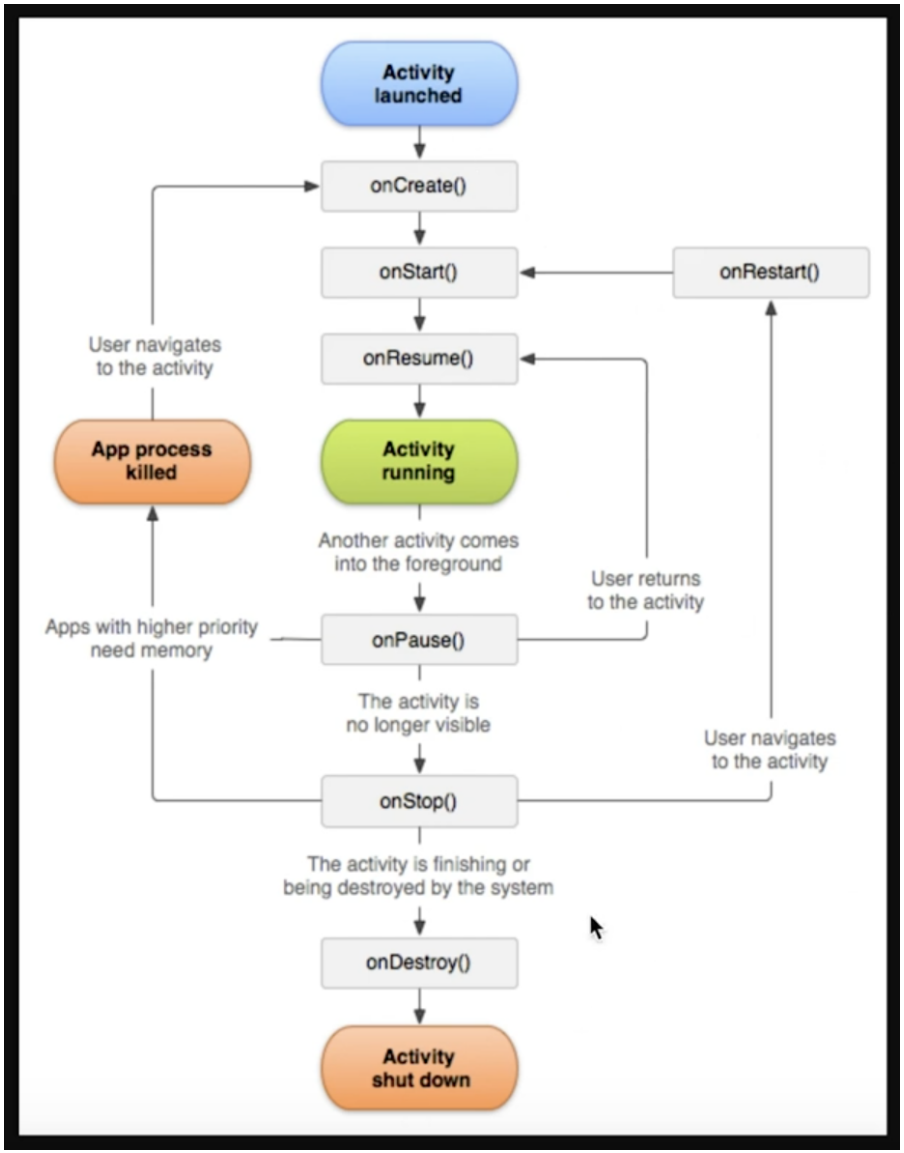
Component: Punt d'entrada pel qual s'accedeix a l'aplicació. 4 tipus:

- **Activitats:** Una activitat es el punt d'entrada de l'interacció amb el usuari. Representa una pantalla individual amb una interficie d'usuari. Les activitats s'implementen com subclasse de la class activity.
- **Serveis** Es un punt d'entrada general que permet mantenir l'execució d'una aplicació en segon pla per diversos motius. Es un component que s'executa en segon pla per realitzar activitats d'execució prolongada o per realitzar tasques de procesos remots: Exemple: Spotify.

Components d'una aplicació

- **Receptors d'emissió** Es 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. Aquest tipus de components no tenen interficie propia, simplement actuen com portes d'enllaç.
- **Proveidors de contingut** 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 a la que tingui accés la teva aplicació.

Cicle de vida d'una aplicació



Les apps tenen diferents processos. Es important definir un mock-up inicial i ordenar tots els processos amb els seus respectius layouts.

- Es pot dividir en 3 procesos cíclics (onResume, onStart and onCreate) depenent de la interficie i els procesos en marxa.

Cicle de vida d'una aplicació

En les funcions del codi, cal definir les següents funcions. És altament recomanable utilitzar la nomenclatura presentada aqui per definir les activitats.

- **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 y requereix la interacció de l'usuari.
- **onPause()** Altra activitat ocupa la pantalla principal. L'activitat iniciada queda en Standby.
- **onStop()** Activitat a segon pla.
- **onDestroy()** L'activitat es tanca (break). S'alliberen recursos.

Executar l'aplicació

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

- **Dispositu real**
  1. Seleccionar l'app d'execució
  2. Seleccionar en la barra d'eines el dispositiu on voleu realitzar l'execució de l'aplicació.
  3. Una vegada seleccionats els dos paràmetres anteriors, prémer RUN i el propi programa instalarà l'aplicació.
- **Emulador**
  1. Selecciona **Tools**>**AVD Manager**, selecciona el tipus de virtual device.
  2. Selecciona el tipus de Hardware en el que desenvoluparàs la teva aplicació.
  3. Selecciona la **System Image** (Android 9.0, Android, 7.1, etc). Recorda que has de seleccionar un OS que sigui compatible amb el seleccionat previament quan vàres crear el projecte Android.

Recursos de l'aplicació

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

- Cal mantenir-los externalitzats
- Organitzats 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
```

Tota 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 circumstancies, es requereix de recursos molt especifics segons el terminal on s'executi l'aplicació. Els diferents idiomes, les diferents qualitats d'imatge o diferents conexions 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 podreu 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 telefon. Tots els permisos queden registrats al Manifest una vegada declarats, i es sobreescriuen automàticament.
4. Requisits de software i hardware. El Manifest es l'arxiu de lectura que permét establir els paràmetres de compatibilitat de la app amb el SO/terminal.

Importancia del manifest

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.

Kotlin Koans Online Tutorial

Durant la resta de la pràctica:
`https://play.kotlinlang.org/koans/overview`

References

Android documentation: developer.android.com/guide

End