Fabián Zamora Ramírez
Jeff Schmidt Peralta

Dia 3 (Listas y acceso a cámara)

1. Introducción a listas

a. Agregar dependencias al proyecto

i. En el archivo build.gradle del módulo app se debe agregar la dependencia hacia la biblioteca de soporte de diseño:

1. *compile 'com.android.support:cardview-v7:24.2.1'*
2. *compile 'com.android.support:recyclerview-v7:24.2.1'*

b. Agregar RecyclerView al layout del TasksFragment.

i. Cambiar el layout (xml) de TasksFragment para que contenga los siguientes elementos:

1. FrameLayout
    a. RecyclerView

c. Crear datos Dummy

i. Crear una nueva clase **Task**. No debe heredar de nada en específico.

1. *Task*
    a. *mTitle:String*
    b. *mDetail:String*

ii. Agregar un método estático que devuelve una lista de Tasks alambrada para pruebas.

d. Crear un Layout y un ViewHolder para cada fila del RecyclerView

i. Se debe crear un nuevo layout (xml) que **se va a utilizar en cada fila** del RecyclerView.

ii. Este layout debe tener como root view un CardView y campos para mostrar la información de cada Task.

1. CardView
    a. LinearLayout
        i. TextView [Muestra Título]
        ii. TextView [Muestra Detalle]

iii. Ahora se debe crear una clase **TasksHolder** que herede de **RecyclerView.ViewHolder**. Esta debe guardar referencia a los views que contienen información específica de de cada fila.

Fabián Zamora Ramírez
Jeff Schmidt Peralta

e. Crear un adapter para manejar los datos del RecyclerView
    i. Se debe crear un adapter (una nueva clase **TasksAdapter**) que va a manejar la creación de los ViewHolder para las filas del RecyclerView. El adapter debe heredar de **RecyclerView.Adapter<TasksHolder>**.
    ii. Se deben sobreescribir los métodos necesarios y utilizar la lista Dummy creada anteriormente.
f. Relacionar el adapter con el RecyclerView
    i. Se realiza dentro *TasksFragment#onCreateView()* mediante el método *RecyclerView#setAdapter()*.
g. Agregar un LayoutManager al RecyclerView
    i. Se realiza dentro *TasksFragment#onCreateView()* mediante el método *RecyclerView#setLayoutManager()*.

2. Introducción al manejo de la cámara

a. Agregar FAB al layout de TasksFragment.
    i. Cambiar el layout (xml) de TasksFragment para que contenga los siguientes elementos:
        1. FrameLayout
            a. RecyclerView
            b. FloatingActionButton
b. Crear CreateTaskFragment
    i. Crear la clase **CreateTaskFragment** que hereda de [android.support.v4.app.Fragment].
    ii. Crear un **layout (xml)** que se muestre en el *onCreateView()* de **CreateTaskFragment**. Este layout contiene un ImageView y un botón.
        1. LinearLayout
            a. ImageView
            b. Button
c. Mostrar TasksFragment y controlar el stack de Fragments
    i. Mostrar el Fragment cuando se hace click en el FAB.
    ii. Agregar el fragment al stack de fragments con el método *FragmentTransaction#addToBackStack()* y agregar un listener que

escuche cuando se agregan Fragments al stack en *MainActivity#onCreate()* para que cambie el comportamiento del ActionBarDrawerToggle.

iii. Agregar un switch/case en el método *MainActivity#onOptionsItemSelected()* para reaccionar al item **android.R.id.home** y devolverse en el stack de Fragments.

d. Agregar permisos para utilizar la cámara

i. Dentro del **AndroidManifest.xml** se deben agregar los permisos para utilizar la cámara y escribir en la memoria del teléfono.

ii. Desde Android 6.0 en adelante también se necesita **pedir los permisos en código**.

e. Agregar un FileProvider

i. Se debe agregar un archivo para especificar los directorios disponibles para el FileProvider. Se crea el directorio **res/xml** y se crea el archivo **file_paths.xml**.

ii. Se debe agregar el **FileProvider** al **AndroidManifest.xml**.

f. Iniciar la aplicación Cámara del dispositivo

i. Esto se hace por medio de un Intent y los métodos *Activity#startActivityForResult(Intent)* y *Activity#onActivityResult()*.

ii. En el Intent se debe especificar la dirección completo donde se va a guardar la imagen. Por lo que se debe crear el archivo primero y enviar el **Uri** como valor para el key *MediaStore.ACTION_IMAGE_CAPTURE*.

iii. La imagen guardada en el archivo se puede abrir con el método *BitmapFactory#decodeFile()*.