Fabián Zamora Ramírez
Jeff Schmidt Peralta

Día 1

1. Agregar dependencias al proyecto

a. En el archivo **build.gradle** del módulo app se debe agregar la dependencia hacia la biblioteca de soporte:
   i. *compile 'com.android.support:appcompat-v7:24.2.1'*

2. Crear LandingActivity

a. Crear la clase **LandingActivity** que hereda de [android.support.v7.app.AppCompatActivity].
b. **LandingActivity** debe estar declarada dentro del **AndroidManifest.xml** y contener el intent-filter para que sea esta Activity la que se muestra al inicio de la aplicación.
c. Crear un **layout (xml)** que se muestre en el *onCreate()* de LandingActivity. Este layout contiene un view contenedor (FrameLayout) que se utiliza para agregar/remover Fragments. Importante agregar atributo ***id*** al view.

3. Crear LoginFragment

a. Crear la clase **LoginFragment** que hereda de [android.support.v4.app.Fragment].
b. Crear un **layout (xml)** que se muestre en el *onCreateView()* de LoginFragment. Este layout contiene los EditText necesarios para obtener la información del usuario. Esto se logra mediante el método *findViewById(int).*

4. Crear RegisterFragment

a. Crear la clase **RegisterFragment** que hereda de [android.support.v4.app.Fragment].
b. Crear un **layout (xml)** que se muestre en el *onCreateView()* de RegisterFragment. Este layout contiene los EditText necesarios para obtener la información del usuario. Esto se logra mediante el método *findViewById(int).*

5. Mostrar LoginFragment desde LandingActivity

a. El fragment se debe mostrar **después** de haber establecido el view de la Activity utilizando *setContentView(int)* dentro del método *onCreate().*

Fabián Zamora Ramírez
Jeff Schmidt Peralta

6. Reemplazar LoginFragment con RegisterFragment

a. Para esto se utiliza el método *FragmentTransaction#replace()* en vez de *FragmentTransaction#add().*

Fabián Zamora Ramírez
Jeff Schmidt Peralta

Día 2

1. Agregar dependencias al proyecto

a. En el archivo **build.gradle** del módulo app se debe agregar la dependencia hacia la biblioteca de soporte de diseño:
   i. *compile 'com.android.support:design:24.2.1'*

2. Crear MainActivity

a. Crear la clase **MainActivity** que hereda de [android.support.v7.app.AppCompatActivity].
b. **MainActivity** debe estar declarada dentro del **AndroidManifest.xml.**
c. Crear un **layout (xml)** que se muestre en el *onCreate()* de MainActivity. Este layout contiene la siguiente jerarquía de views:
   i. DrawerLayout
      1. CoordinatorLayout
         a. AppBarLayout
            i. Toolbar
         b. FrameLayout [Contenedor para Fragments]
      2. NavigationView

3. Crear ActionBarDrawerToggle

a. Se debe configurar el **toolbar** del layout del **MainActivity** como el default de la Activity.
b. Se debe **habilitar** el icono de **home** en el toolbar.
c. Se debe crear un **ActionBarDrawerToggle** y agregar como listener del **DrawerLayout**.
d. Se deben sobreescribir los métodos *MainActivity#onConfigurationChanged()* y *MainActivity#onOptionsItemSelected()* y llamar los métodos correspondientes del ActionBarDrawerToggle.
e. Se debe sobreescribir el método *MainActivity#onPostCreate()* y llamar el método *ActionBarDrawerToggle#syncState()*

4. Crear nuevo estilo para eliminar ActionBar del sistema

a. Se debe crear un estilo en el archivo **res/values/styles.xml** que oculte el action bar del sistema. Este estilo se le aplica al **MainActivity** dentro del AndroidManifest con la propiedad **android:theme**.

Fabián Zamora Ramírez
Jeff Schmidt Peralta

5. Agregar los elementos al NavigationView

a. Crear el folder res/menu
b. Crear un xml dentro de res/menu que contenga los ítems a mostrar. Importante cada ítem debe tener un ***id***.
c. Dentro del layout (xml) del MainActivity al elemento **NavigationView** se le debe configurar el atributo **app:menu**.

6. Reaccionar a clicks en el menu

a. Esto se logra llamando el método *NavigationView#setNavigationItemSelectedListener()* en el método *onCreate()* de **MainActivity** después de haber llamado *setContentView(int).*
b. Dentro del método callback se puede agregar/reemplazar fragments en el view contenedor que se agregó al layout.

7. Crear TasksFragment

a. Crear la clase **TasksFragment** que hereda de [android.support.v4.app.Fragment].
b. Crear un **layout (xml)** que se muestre en el *onCreateView()* de TasksFragment. Este layout contiene contenido Dummy solo para verificar que el Fragment se muestre en pantalla.

8. Ejercicios adicionales

a. Agregar un **header** al NavigationView. Como referencia pueden crear un nuevo proyecto en Android Studio y seleccionar Navigation Drawer Activity como Activity inicial.

Fabián Zamora Ramírez
Jeff Schmidt Peralta

Dia 3

1. Agregar dependencias al proyecto

a. En el archivo build.gradle del módulo app se debe agregar la dependencia hacia la biblioteca de soporte de diseño:
   i. *compile 'com.android.support:cardview-v7:24.2.1'*
   ii. *compile 'com.android.support:recyclerview-v7:24.2.1'*

2. Introducción a listas

a. Agregar RecyclerView al layout del TasksFragment.

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

b. 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.

c. 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.

d. 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.
e. Relacionar el adapter con el RecyclerView
    i. Se realiza dentro *TasksFragment#onCreateView()* mediante el método *RecyclerView#setAdapter()*.
f. Agregar un LayoutManager al RecyclerView
    i. Se realiza dentro *TasksFragment#onCreateView()* mediante el método *RecyclerView#setLayoutManager()*.

3. 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 TasksFragment
    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()*.