

Seminario de Lenguajes Android

Práctica 1

- 1) Abrir Android Studio y crear un nuevo proyecto con una Actividad vacía (Empty Activity)
- 2) Abrir el Android Virtual Device Manager y crear como dispositivo virtual un Galaxy Nexus con el nombre Nexus Seminario Android
- 3) Probar la aplicación creada en el ejercicio 1 en el emulador
- 4) Describir qué representa una Activity
- 5) Abrir el archivo AndroidManifest.xml. ¿Por qué MainActivity tiene un intent-filter con action MAIN y category LAUNCHER?
- 6) Crear 2 actividades (NuevaActivity1, NuevaActivity2) y ver qué se modificó en el archivo AndroidManifest.xml
- 7) Pegar el siguiente código en res/layout/activity_main.xml

```
<RelativeLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <Button
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Click"
        android:onClick="onBtnClick"/>
</RelativeLayout>
```

8) Añadir el siguiente código en la clase MainActivity.java, probar en el emulador y analizar el resultado

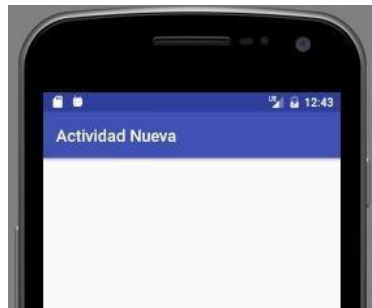
```
fun onBtnClick(view: View?) {  
    val i = Intent(this, NuevaActivity2::class.java)  
    startActivity(i)  
}
```

* Se deben importar estos paquetes: `android.view.View`, `android.content.Intent`

9) ¿Qué significa *this* en el código del ejercicio anterior?

10) ¿Lo que utilizamos fue un intent implícito o explícito? ¿Cual es la diferencia entre ambos?

11) A través del AndroidManifest modificar el nombre que se muestra al usuario para la actividad 2 para que al hacer click se muestre con el texto *Actividad Nueva*



12) Modificar el comportamiento de onBtnClick para que a través de un intent abra una página web

13) En el método onCreate de la actividad nueva añadir la siguiente línea.

```
Log.d("APP_DE_PRUEBA", "Este es mi mensaje de debug");
```

Correr la aplicación en modo debug y revisar la consola de Debug luego de abrir la actividad 2.
¿Qué ocurrió? ¿Cuál es su utilidad?

14) Al hacer click en el botón se debe pasar a la actividad 2 un texto como parámetro. Cuando la actividad 2 se muestra se debe imprimir por consola el texto recibido como parámetro.

15) Describa cuales son los estados por los que puede pasar una actividad

16) Describa cuales son los eventos generados a partir de un cambio de estado de una actividad

17) Crear una nueva aplicación en la que se imprima por la consola los cambios de estados de una actividad utilizando lo investigado en el punto 15 y 16

18) Genere una nueva aplicación con una actividad vacía en Android Studio. Edite el archivo AndroidManifest.xml y elimine las siguientes líneas:

```

<intent-filter>
    <action android:name="android.intent.action.MAIN" />
    <category android:name="android.intent.category.LAUNCHER" />
</intent-filter>

```

- ¿Qué error se produce en el entorno de desarrollo?. ¿Cuál es el motivo del error?
- Vuelva a colocar el código eliminado para que la aplicación funcione correctamente
- Agregue un TextView a la Activity generada con el valor "Español" y un botón con el valor 'Cambiar idioma. Al hacer click en el botón el textview debe cambiar su texto a "Inglés". Si se presiona nuevamente sobre el botón, el textview debe contener el valor "Español" nuevamente.

19) Crear una nueva aplicación con 2 actividades como se ven en la figura.



Al hacer click en "Realizar operación" se debe abrir la segunda actividad. Al hacer click en los botones "Incrementar" o "Decrementar" la actividad debe cerrarse y aplicar la operación correspondiente sobre el TextView. Si se presiona "Cancelar" solo debe cerrarse la segunda actividad sin realizar cambios sobre el TextView

A continuación se detallan los layouts de las actividades para simplificar el ejercicio (Notar el uso de LinearLayout):

Activity1:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical">

    <TextView android:id="@+id/txtContador"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="0"
        android:textAlignment="center"
        android:textSize="34sp"
    />

    <Button android:id="@+id/btnRealizarOperacion"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Realizar operación"
    />
</LinearLayout>
```

Activity2:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="horizontal">

    <Button android:id="@+id/btnIncrementar"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Incrementar"
    />

    <Button android:id="@+id/btnDecrementar"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Decrementar"
    />

    <Button android:id="@+id/btnCancelar"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Cancelar"
    />

</LinearLayout>
```

- 20) Agregar un control más al ejercicio anterior para que no pueda decrementarse si el valor es 0. Para ello al retornar a la actividad 1 verificar si el valor es 0 y desplegar un mensaje Toast informando el error.

21) Modifique el ejercicio anterior para que la segunda actividad (operaciones) se abra con un intent implícito.

22) ¿Qué sucede en el ejercicio anterior si se modifica la orientación del dispositivo (horizontal/vertical)?

a) Solucione el problema mediante `saveInstanceState` / `restoreInstanceState`.

23) Generar una actividad con nombre `LifeCycleActivity` y pruebe el siguiente código:

```
override fun onDestroy() {  
    super.onDestroy()  
    val i = Intent(this, LifeCycleActivity::class.java)  
    this.startActivity(i)  
}
```

Ejecute la aplicación:

a) Intente destruir la actividad mediante el botón "Atrás" del dispositivo.

b) Intente destruir la actividad mediante el botón de intercambio de tareas. (Botón central del Nexus S)

24) En el ejercicio anterior, agregue a la actividad un `TextView` con id "texto".

Agregue al método `onDestroy` el siguiente código:

```
override fun onDestroy() {  
    super.onDestroy()  
    (findViewById<View>(R.id.texto) as TextView).text = "HOLA MUNDO!"  
    val i = Intent(this, LifeCycleActivity::class.java)  
    this.startActivity(i)  
}
```

Intente destruir la actividad mediante el botón "Atrás" del dispositivo.

a) ¿Se ve el mensaje "HOLA MUNDO" en la componente `TextView`? ¿Por qué?

b) ¿Se puede resolver mediante `saveInstanceState` / `restoreInstanceState`?

c) ¿Qué sucede con la instancia de `LifeCycleActivity`?

- 25) Crear una nueva aplicación que tenga 2 actividades. En la actividad 1 reemplazar el código del archivo *activity_main.xml* por el presentado a continuación.

```
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical">
    <!-- Aquí van los botones -->
</LinearLayout>
```

La primera actividad debe tener 2 botones. El primero debe tener el texto ¿Qué hora es? y al hacer click debe imprimir en la consola de Debug la hora actual.

El segundo debe contener el texto ¿Qué día es? y al hacer click pasar como parámetro el día de hoy a la 2da actividad. Al abrirse la segunda actividad debe imprimir el valor recibido.

* *Investigar la clase SimpleDateFormat para convertir la fecha a String en un formato específico*

Seminario de Lenguajes Android

Práctica 2

1. Crear un nuevo proyecto en Android Studio con una Actividad vacía y modificar el contenido del archivo xml de la actividad creada (en res/layouts), colocar el siguiente código en él y probar en el dispositivo o emulador:

<RelativeLayout

xmlns:android="http://schemas.android.com/apk/res/android"

android:layout_width="match_parent"

android:layout_height="match_parent">

<TextView

android:layout_width="200dp"

android:layout_height="200dp"

android:text="Hola Mundo" />

</RelativeLayout>

2. Investigue la utilidad de los atributos "android:layout_width" y "android:layout_height".
 - a. ¿Qué sucede si se omite alguno de los dos en el elemento RelativeLayout?
 - b. ¿Qué sucede si en lugar de "match_parent" se establece como valor "wrap_content"?
3. Añadir un color de fondo al TextView y modificar el color de las letras.
4. Modificar el ancho del TextView para que siempre se adapte al ancho del contenedor. Luego centrar el texto horizontalmente.
5. Cambiar el tamaño de la tipografía (utilizar la unidad de medida sp)
6. Crear un TextView debajo del *Hola Mundo* con el texto *Aquí Estoy*. ¿Qué ocurrió? ¿Por qué?

7. Reemplazar en el código creado *RelativeLayout* por un *LinearLayout* de la siguiente manera y ver el resultado. ¿Para qué sirve especificar la orientación del *LinearLayout*?

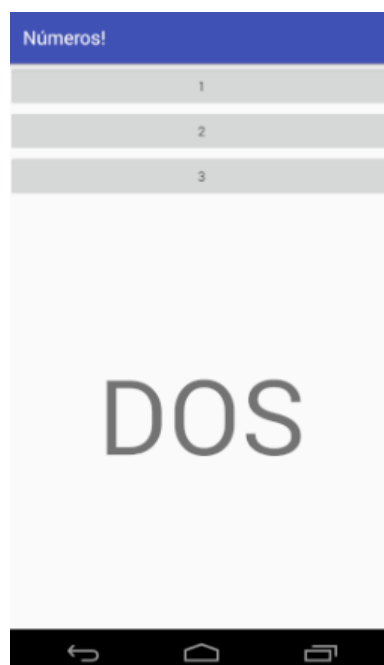
<LinearLayout

```
xmlns:android="http://schemas.android.com/apk/res/android"  
android:layout_width="match_parent"  
android:layout_height="match_parent"  
android:orientation="vertical">
```

8. Resuelva el problema visual que surgió en el punto 6 sin utilizar *LinearLayout*.
(Nota: Utilice la propiedad *layout_below*).
9. Genere una disposición de componentes visuales como la siguiente:



10. A partir de la experiencia obtenida con el uso de Layouts.
- ¿Por qué cree que son importantes los Layouts?
 - ¿Cuál hubiera sido la problemática de definir la posición/tamaño de las componentes visuales a través de coordenadas absolutas?
 - ¿Esta problemática solo existe en dispositivos móviles?
11. Implemente una aplicación con un layout como el siguiente:



Cuando se haga click sobre uno de los botones, la aplicación deberá mostrar en pantalla de forma centrada y con una tipografía más grande la representación en letras mayúsculas del número sobre el cual se hizo click.

El ancho de los botones deberá adaptarse al ancho de la pantalla.

El espacio en donde se visualiza el texto deberá adaptarse al espacio disponible y el texto se mostrará centrado horizontal y verticalmente.

- a. Implemente utilizando un manejador de click para cada botón.
- b. Implemente utilizando un único manejador de click para todos los botones.

12. Implemente una aplicación con un layout como el siguiente:



Cada vez que se hace click en el botón "Tirar Dado", la aplicación deberá generar aleatoriamente un valor entre 1 y 6 mostrándolo en pantalla.

Para la generación de números aleatorios investigue el uso de la clase `java.util.Random`

13. Usando un layout similar al desarrollado en el ejercicio 12, agregar una imagen usando el componente **<ImageView>**.

La imagen deberá incluirse en el directorio `/proyecto/app/src/main/res/drawable/`

Nota: puede usar el siguiente código:

```
<ImageView
    android:src="@drawable/imagen"
    android:layout_width="match_parent"
    android:layout_height="200dp"
    android:scaleType="center" />
```

Práctica 2

Facultad de Informática



OCULTAR

MOSTRAR

- ¿Qué sucede al cambiar el valor del atributo `scaleType` por `"centerCrop"`? ¿Y `"fitXY"`?
- Al hacer click sobre los botones , deberá ocultarse o mostrarse la imagen usando el método `setVisibility(View.GONE)` y `setVisibility(View.VISIBLE)` sobre la imagen.



¿Qué cambia si en vez de ocultar la imagen usando `View.GONE`, usamos `View.INVISIBLE`?

- Agregar el atributo `android:animateLayoutChanges="true"` en el `LinearLayout` que contiene a los elementos. ¿Qué diferencia hay al cambiar la visibilidad de la imagen?

Seminario de Lenguajes Android

Práctica 3

1. Modificar el ejercicio 12 de la práctica 2 para que, al girar el dispositivo, se mantenga el número mostrado en pantalla. Utilice los métodos `onSaveInstanceState/onRestoreInstanceState`.
2. Implemente la siguiente Activity con cuatro imágenes y un título, usando un `ScrollView`



3. Modificar el ejercicio anterior para que muestre una única imagen. Cada vez que se clickea el botón "Siguiente", se deberá reemplazar la imagen. Investigar el uso del mensaje `setImageResource` de la clase `ImageView`.

Al girar la pantalla, no debe cambiarse la imagen y el layout debe visualizarse correctamente.

Nota: Puede almacenar las referencias a las imágenes en un arreglo de la siguiente manera:



```
private int[] imagenes = {R.drawable.portada, R.drawable.interior, R.drawable.foto3};
```

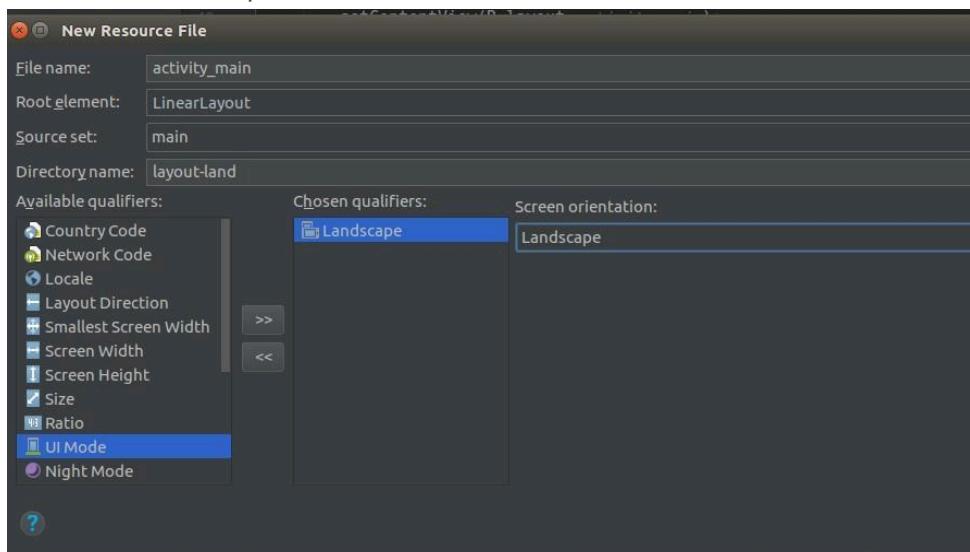
4. Modifique el ejercicio anterior de manera que al girar la pantalla, se modifique el layout de la siguiente forma.



Implementar la grilla de botones usando GridLayout y un XML diferenciado según la orientación.

Nota: para incluir un layout dependiente de la orientación, desde Android Studio, se debe hacer click derecho en el directorio *layout* → *New* → *Layout resource file*.

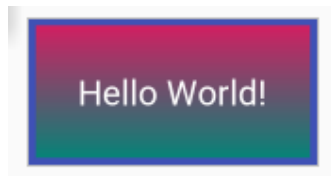
Deberá tener el mismo nombre de archivo que el layout que usamos para la versión vertical. Luego elegimos “*Orientation*” en el cuadro de “*Available qualifiers*”, y luego la orientación *Landscape*:



Por defecto, cuando la aplicación esté en orientación *Portrait*, usará el layout que

habíamos definido originalmente, y ahora la orientación *Landscape* usará este nuevo layout.

5. Investigue sobre los recursos de elementos de diseño de Android. (Recursos drawable)
 - a. ¿Para qué sirven?
 - b. ¿Qué tipos de elementos de diseño están disponibles?
 - c. ¿Qué tipo de elemento de diseño utilizaría para mostrar una imagen?
 - d. ¿Qué tipo de elemento de diseño utilizaría para mostrar un rectángulo con colores?
 - e. ¿Qué tipo de elemento de diseño utilizaría para mostrar los estados “presionado” y “suelto” de un botón?
6. Utilice un recurso drawable para definir el fondo de un TextView que se visualice de la siguiente manera:

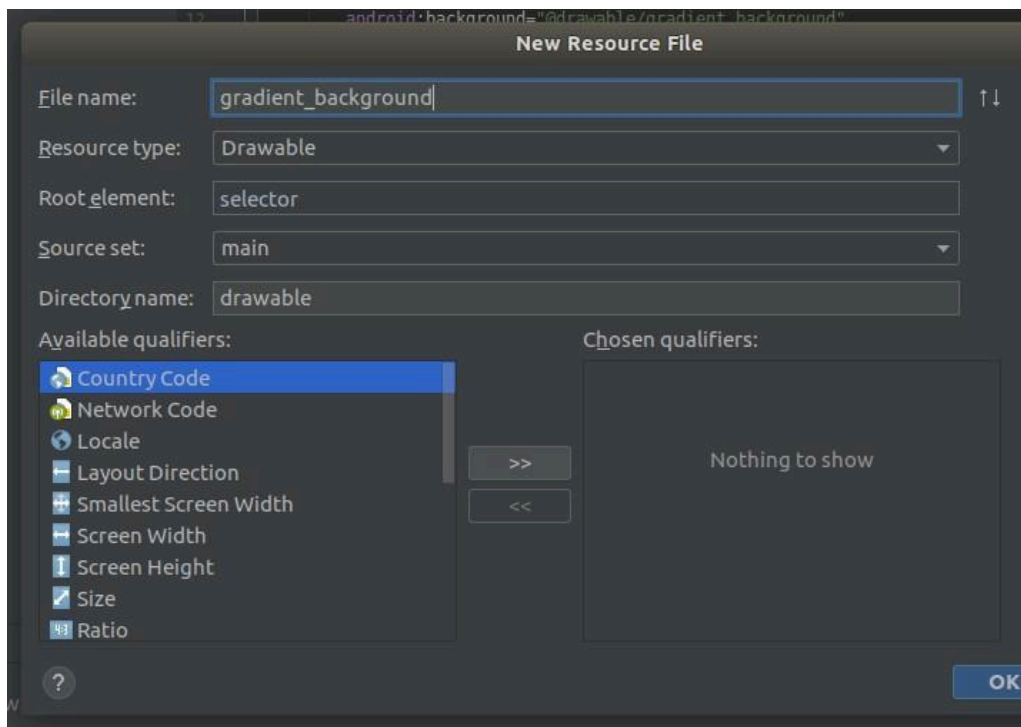


Para ello deberá:

- a. Generar un recurso drawable denominado a modo de ejemplo: *gradient_background*

Click derecho -> New > Android Resource File

Notar que Resource type debe ser: Drawable



- b. En el archivo xml generado deberá definir un diseño gradiente vertical lineal entre dos colores. Al mismo tiempo debe definir un borde de 3dp de ancho con un color diferente al del gradiente. Deberá utilizar un Shape. Puede obtener información sobre la sintaxis a utilizar en:

<https://developer.android.com/guide/topics/resources/drawable-resource?hl=es-419#Shape>

- c. Una vez definido el recurso drawable, puede asignárselo a un TextView simplemente estableciendo el atributo android:background de la siguiente manera:

```
<TextView android:background="@drawable/gradient_background"
```

- 7. Genere un Botón que utilice como background el gradiente definido en el punto anterior. Cuando el usuario presiona el botón, deberá cambiar el gradiente a un color diferente. Para ello, genere un nuevo recurso similar al del punto anterior pero con colores diferentes para el gradiente con nombre: gradient_background_pressed
A continuación defina un nuevo recurso drawable que defina los dos estados del botón y que haga referencia a los drawables previamente definidos. Deberá utilizar un StateList. Puede obtener información de la sintaxis a utilizar en:
<https://developer.android.com/guide/topics/resources/drawable-resource?hl=es-419#StateList>
- 8. Intente utilizar el drawable StateList definido en el ejercicio anterior en otra componente visual como un ImageView o un TextView. ¿Funciona el cambio de estado al presionar sobre la componente?.

Establezca el atributo “clickable” de la componente en true y vuelva a intentarlo.

Seminario de Lenguajes Android

Práctica 4

1. Crear una aplicación con un TextView centrado en pantalla que contenga un contador que vaya incrementándose automáticamente cada un segundo desde el momento que se abre la aplicación.
 - a. ¿Puede implementar el temporizador utilizando un for / while?
 - b. Utilice la clase Handler para resolver el temporizador. Investigue el método *postDelayed*.
2. Implemente en la aplicación anterior un botón que permita detener/reanudar el contador.
3. Implemente un estado intermedio en el contador en donde el número cambie a color rojo 500 milisegundos antes de incrementarse. Una vez que incrementa, debe volver a su color inicial.
 - a. Implemente utilizando dos Handlers
 - b. Implemente utilizando un solo Handler
4. Crear una aplicación con dos TextView centrados en la pantalla, "Texto Uno" y "Texto Dos".

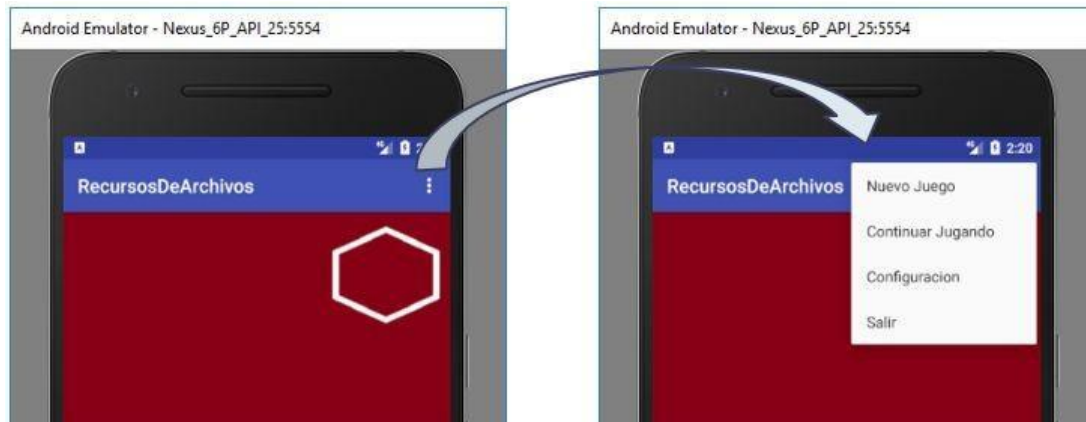
El primer TextView deberá tener como tamaño del texto (textSize) "18dp", y el segundo, "18sp".

Ejecutar en un dispositivo o emulador. Luego, desde la configuración del sistema Android, cambiar el tamaño de la fuente, y verificar cómo impacta esto en la aplicación desarrollada.
5. En base a la aplicación desarrollada en el punto 1, agregar soporte para idioma inglés y portugués, usando archivos XML de strings con calificadores de idioma.

Verificar que los textos cambien de idioma, al cambiar el idioma del sistema operativo desde la configuración.
6. Agregar a la aplicación anterior una imagen como recurso drawable llamado "bandera" que se muestre debajo de los textos. Usando calificadores de idiomas para la carpeta drawable, esta bandera deberá ser la de Argentina, Brasil o Reino Unido.
7. Cambiar la apariencia de la galería de imágenes del ejercicio 3 de la práctica 3, para que use un fondo oscuro y letras blancas. Aplicar un estilo con estas características, usando el atributo "theme" en la activity dentro del Manifest.
8. Investigar cómo aplicar un tema a la activity programáticamente desde el código kotlin. Declarar dos estilos en XML, uno llamado "día" con colores claros y otro "noche", con colores oscuros. Los colores deberán estar declarados en el archivo de recursos de colores XML. Según la hora actual al iniciarse la aplicación, deberá mostrar uno u otro estilo.

Nota: el método `setTheme()` debe ejecutarse antes del método `setContentview()` en el `onCreate()` de la Activity. De este modo, se aplican los cambios de estilo antes de cargar la vista.

9. Agregar un menú principal, como el visto la teoría, a la aplicación de galería de imágenes. Deberá contar con dos opciones, una “Siguiente”, que adelanta una imagen y la otra “Anterior”, que vuelve a la imagen anterior.

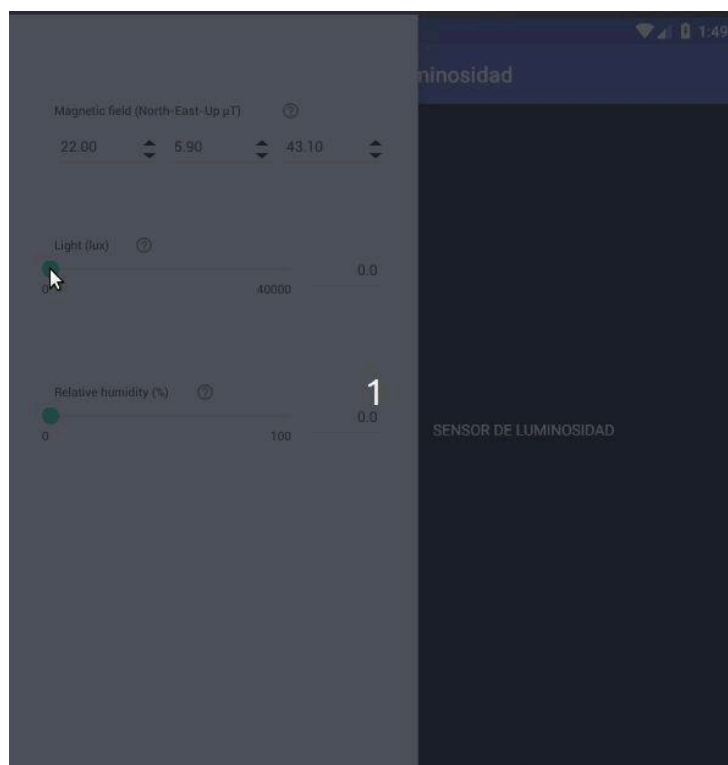


Seminario de Lenguajes

Android Práctica 5

NOTA: En los ejercicios que utilizan sensores, es posible realizar pruebas en el emulador del dispositivo sin necesidad de probar con un dispositivo físico. Los sensores pueden alterarse a través de las opciones del emulador (icono de puntos suspensivos / Virtual Sensors / Additional sensors)

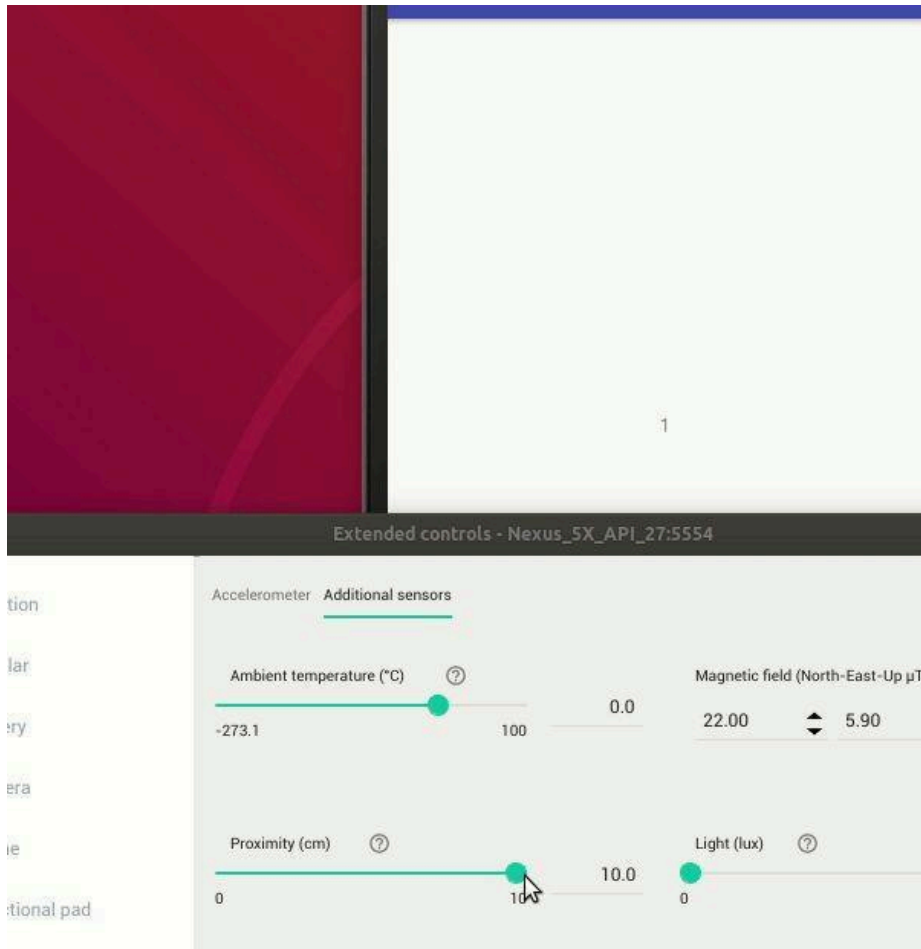
- 1) Investigue el uso de los diferentes sensores del dispositivo en la documentación oficial de Android: https://developer.android.com/guide/topics/sensors/sensors_environment (Sección: *Use the light, pressure, and temperature sensors*)
 - a) ¿Cuál es la función de la clase `SensorManager`?
 - b) ¿Cómo se genera una instancia de un `Sensor`?
 - c) ¿Para qué sirven los métodos `registerListener` y `unregisterListener` de la clase `Sensor`?
 - d) ¿Por qué se utilizan las transiciones de estado `onResume` y `onPause` para registrar/eliminar el listener del `Sensor`?
- 2) Sensor de Luminosidad
 - a) Implemente una aplicación que muestre en un `TextView` el valor del `Sensor` en tiempo real.
 - b) Implemente una aplicación con un `TextView` que ocupe toda la pantalla y que contenga un texto fijo. El color de fondo y el texto del `TextView` deberá reaccionar al nivel de luminosidad de manera que se vea el fondo blanco y letras negras en condiciones de mucha luz, mientras que en condiciones de poca luz se visualizará el fondo negro con letras blancas.



3) Sensor de proximidad

- a) Implemente una aplicación que muestre un TextView con un contador numérico. El contador debe incrementarse cada vez que el usuario pase su mano por encima del dispositivo.

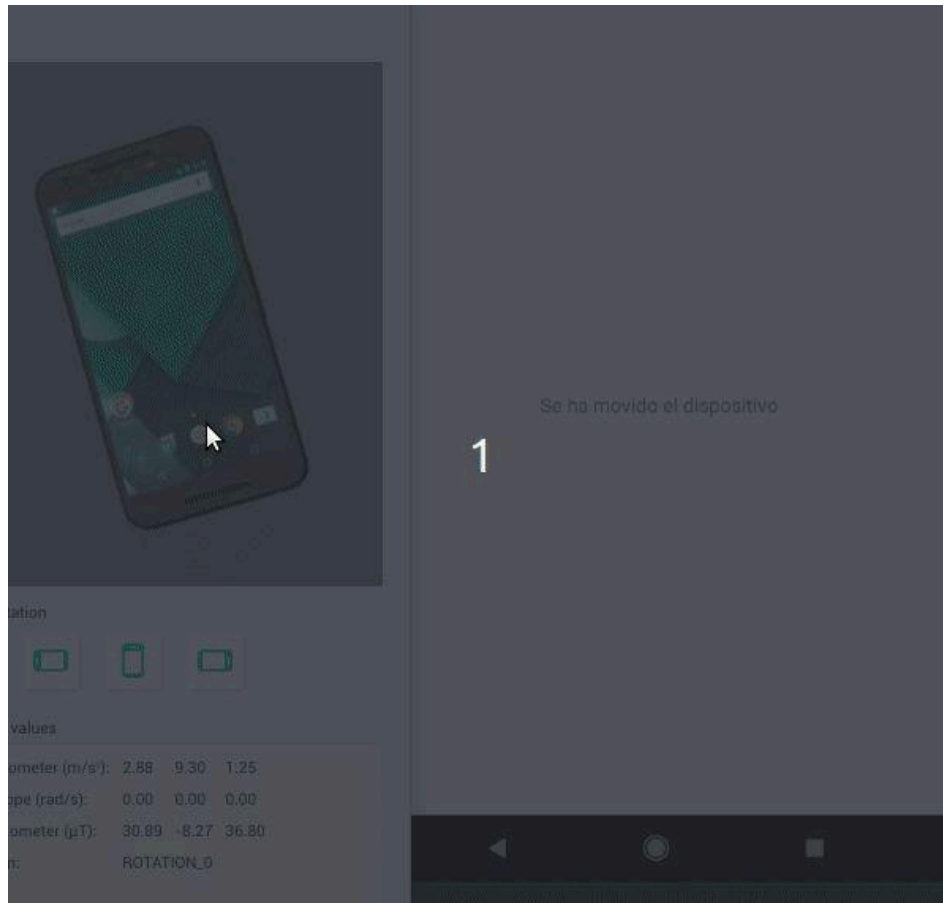
NOTA: Para determinar que el usuario realizó el gesto de pasar su mano por encima del dispositivo, deberá detectar un estado de “no proximidad”, seguido de un estado de “proximidad”.



4) Acelerómetro

NOTA: Para estos ejercicios deberá utilizar el sensor de Aceleración LINEAL, dado que el mismo no tiene en cuenta la fuerza de gravedad.

- Generar una aplicación que muestre en un TextView los valores de aceleración en los tres ejes X, Y, Z.
- Cuando el dispositivo está en reposo. ¿Los valores son estrictamente cero?
- Genere una aplicación que sea capaz de detectar el estado de reposo (completamente quieto) del dispositivo y que genere un aviso en un TextView cuando el mismo pierda su estado de reposo (alguien mueva el dispositivo).



5) Permisos

- a) ¿Por qué algunos sensores requieren solicitar permisos en tiempo de ejecución y otros pueden usarse directamente? Investigar los niveles de los permisos en <https://developer.android.com/guide/topics/permissions/overview#normal-dangerous>
- b) Implementar una aplicación que solicite permisos en tiempo de ejecución para usar la ubicación del usuario.
Puede utilizar los métodos para consultar y pedir los permisos definidos a continuación.
Recuerde declarar el permiso tanto en el Manifest como en la Activity.

```
override fun onStart() {
    super.onStart()
    if (!chequearPermisosLocalizacion()) {
        pedirPermisosLocalizacion()
    } else {
        obtenerLocalizacion()
    }
}

override fun onRequestPermissionsResult(
    requestCode: Int,
    permissions: Array<String?>,
    grantResults: IntArray
) {
    super.onRequestPermissionsResult(requestCode, permissions,
    grantResults)
    if (requestCode == PERMISSION_GRANTED) {
        if (grantResults.size > 0
            && grantResults[0] ==
PackageManager.PERMISSION_GRANTED
        ) {
            // El usuario concedió el permiso, ya podemos usar
            la ubicación
            obtenerLocalizacion()
        }
    }
}

private fun chequearPermisosLocalizacion(): Boolean {
    val permissionState =
        ActivityCompat.checkSelfPermission(this,
Manifest.permission.ACCESS_COARSE_LOCATION)
    return permissionState == PackageManager.PERMISSION_GRANTED
}

private fun pedirPermisosLocalizacion() {
    ActivityCompat.requestPermissions(
        this@MainActivity,
        arrayOf(Manifest.permission.ACCESS_COARSE_LOCATION),
        PERMISSION_GRANTED
    )
}

fun obtenerLocalizacion() {
```

```
//Acá usamos el método de geolocalización que hayamos  
elegido  
}
```

6) Cámara

- Implementar una aplicación que tome una fotografía usando un intent implícito y la muestre en un ImageView, tal como lo visto en la clase teórica.
- Al usar la cámara con un intent implícito, no necesitamos solicitar permiso al usuario en tiempo de ejecución. ¿Qué diferencia hay si queremos usar la cámara de manera directa dentro de nuestra app?
Investigar en <https://developer.android.com/guide/topics/media/camera#manifest>
- Si queremos guardar la imagen en la memoria de almacenamiento, ¿alcanza con haber solicitado el permiso de uso de la cámara?

7) Posicionamiento

- Investigar las diferentes estrategias de obtener el posicionamiento del usuario (GPS, WiFi, Celular), sus ventajas y desventajas.
Ver <https://developer.android.com/guide/topics/location/strategies>
- Implementar una aplicación que muestre la latitud y longitud del usuario en un TextView, usando las APIs de ubicación de Google Play Services.
Para esto, se debe agregar Google Play Services al archivo Gradle del proyecto, ubicado en directorio principal del proyecto:

```
classpath 'com.google.gms:google-services:4.3.15'
```

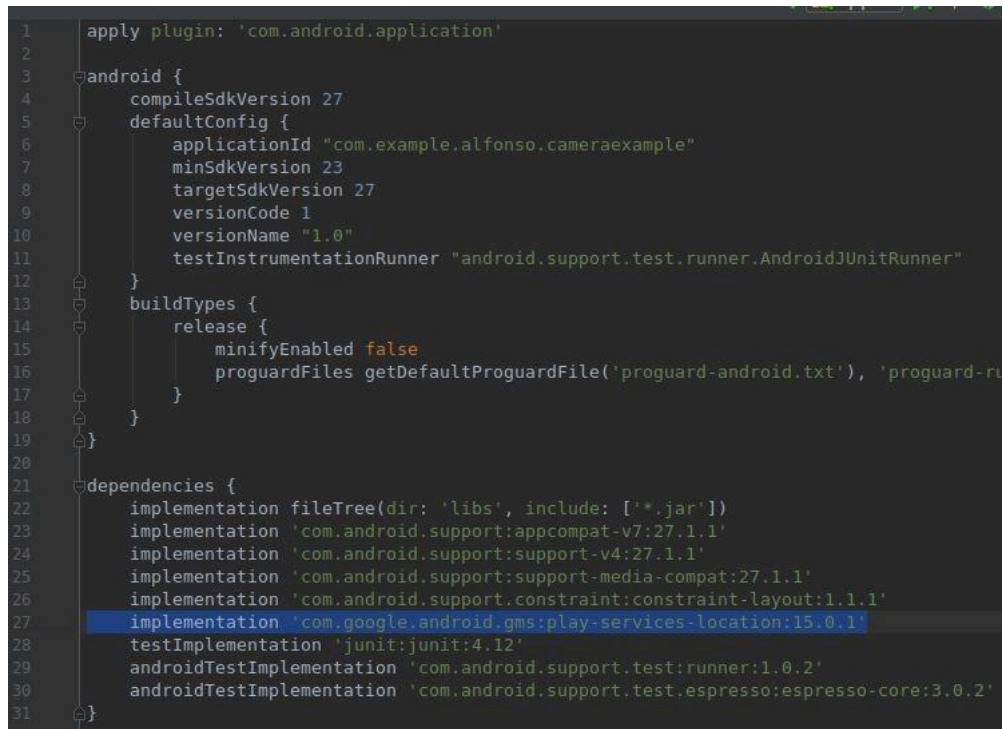
Debe quedar de la siguiente forma:

```
// Top-level build file where you can add configuration options common to all  
// subprojects (like repositories) for all modules (main, test, androidTest)  
//  
// NOTE: Do not place your application dependencies here; they belong  
// in the individual module build.gradle files  
  
buildscript {  
    repositories {  
        google()  
        jcenter()  
    }  
    dependencies {  
        classpath 'com.android.tools.build:gradle:3.1.3'  
        classpath 'com.google.gms:google-services:3.2.0'  
    }  
}  
  
allprojects {  
    repositories {  
        google()  
        jcenter()  
    }  
}  
  
task clean(type: Delete) {  
    delete rootProject.buildDir  
}
```

Luego agregamos la dependencia de las APIs de ubicación en el archivo gradle de

/app/build.gradle:

```
implementation 'com.google.android.gms:play-services-location:15.0.1'
```



```
1  apply plugin: 'com.android.application'
2
3  android {
4      compileSdkVersion 27
5      defaultConfig {
6          applicationId "com.example.alfonso.cameraexample"
7          minSdkVersion 23
8          targetSdkVersion 27
9          versionCode 1
10         versionName "1.0"
11         testInstrumentationRunner "android.support.test.runner.AndroidJUnitRunner"
12     }
13     buildTypes {
14         release {
15             minifyEnabled false
16             proguardFiles getDefaultProguardFile('proguard-android.txt'), 'proguard-rules.pro'
17         }
18     }
19 }
20
21 dependencies {
22     implementation fileTree(dir: 'libs', include: ['*.jar'])
23     implementation 'com.android.support:appcompat-v7:27.1.1'
24     implementation 'com.android.support:support-v4:27.1.1'
25     implementation 'com.android.support:support-media-compat:27.1.1'
26     implementation 'com.android.support.constraint:constraint-layout:1.1.1'
27     implementation 'com.google.android.gms:play-services-location:15.0.1'
28     testImplementation 'junit:junit:4.12'
29     androidTestImplementation 'com.android.support.test:runner:1.0.2'
30     androidTestImplementation 'com.android.support.test.espresso:espresso-core:3.0.2'
31 }
```

Una vez hecho esto, ya se incluye la librería de ubicación de Google Play Services en nuestra aplicación. Por lo tanto, podemos usar las clases que nos provee la librería para solicitar la ubicación:

```
private var client: FusedLocationProviderClient? = null
private var locationCallback: LocationCallback? = null
```

En el método onCreate, inicializamos estas variables:

```
client = LocationServices.getFusedLocationProviderClient(this)
locationCallback = object :
    LocationCallback() {
        override fun onLocationResult(locationResult: LocationResult) {
            super.onLocationResult(locationResult)
            //usamos locationResult.getLastLocation() para tener
            //la ubicación más reciente
        }
    }
}
```

Y una vez que tengamos los permisos, le solicitamos al cliente que envíe la información de localización a nuestro callback:

```
client?.requestLocationUpdates(LocationRequest.create(),
    locationCallback as LocationCallback, null
)
```

- c) Implementar una aplicación que, al iniciar, muestre a qué distancia se encuentra el usuario de la Facultad de Informática.

Investigar los métodos que provee la clase Location

La ubicación de la Facultad es -34.9037905, -57.9378442