UNIVERSIDAD DE EL SALVADOR
FACULTAD DE INGENIERIA Y ARQUITECTURA
ESCUELA DE INGENIERIA DE SISTEMAS INFORMATICOS
PROGRAMACION PARA DISPOSITIVOS MOVILES
PDM115

Ciclo I-2018

Guía de Laboratorio N°02

(para desarrollarse en dos semanas)

"Introducción a la Interfaz de usuario de Android"

Objetivos:

Que el estudiante:

- Aprenda la declaración e implementación de los controles básicos en la programación Android haciendo uso de **layouts** para realizar el diseño de la interfaz y el uso de **java** para definir el manejo de estos.
- Aprenda a Incorporar nuevos controles de forma intuitiva siguiendo la lógica de controles de similar comportamiento.

Descripción:

Esta práctica consistirá en crear un programa que contendrá una serie de Activities que mostrarán el uso de cada uno de los controles básicos de Android (Button, TextView, EditText, CheckBox, etc) con su respectiva interfaz de usuario (usando XML). Cada Activity será llamada por medio de otra Activity adicional que usaremos para mostrar un "Menú" por medio del componente ListView.

UNIVERSIDAD DE EL SALVADOR
FACULTAD DE INGENIERIA Y ARQUITECTURA
ESCUELA DE INGENIERIA DE SISTEMAS INFORMATICOS
PROGRAMACION PARA DISPOSITIVOS MOVILES
PDM115

Ciclo I-2018

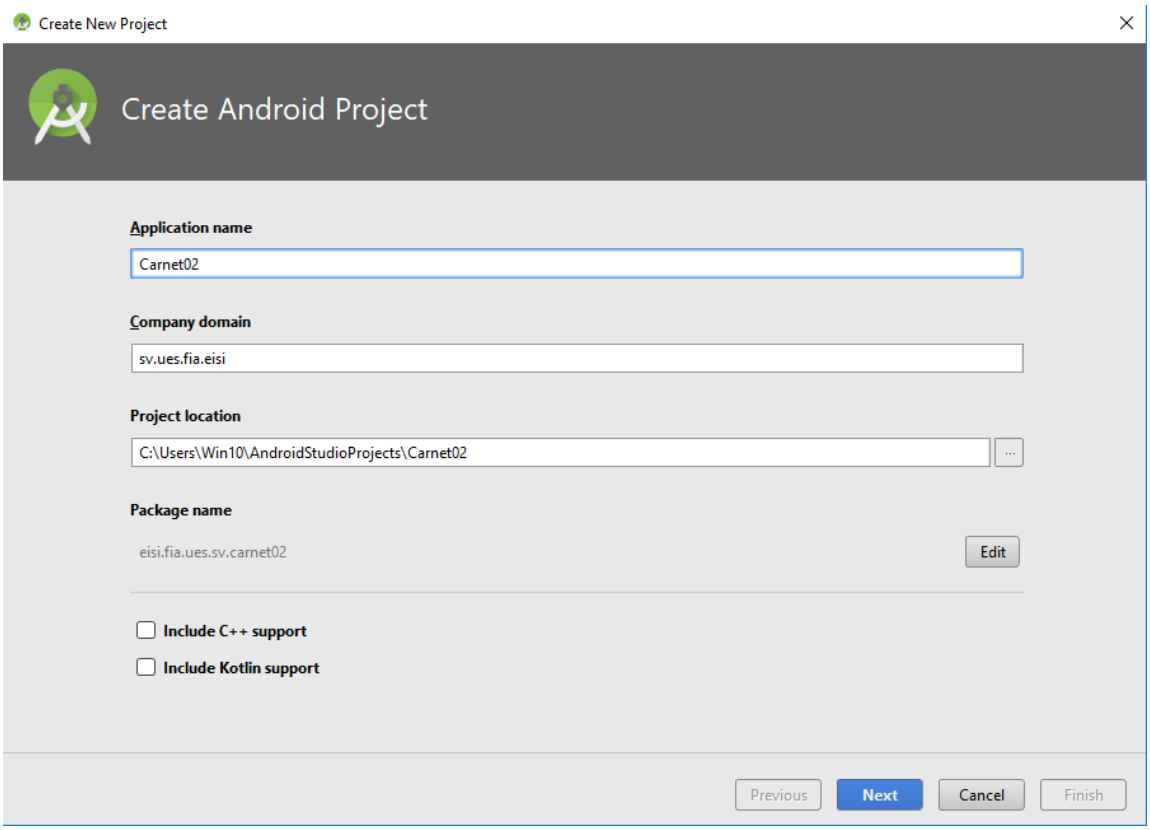
Índice

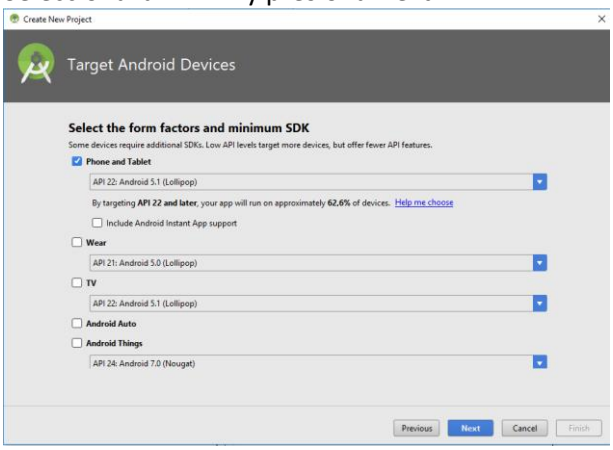
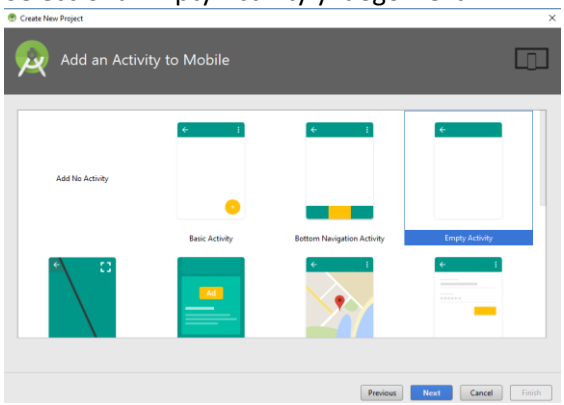
Creación de Proyecto

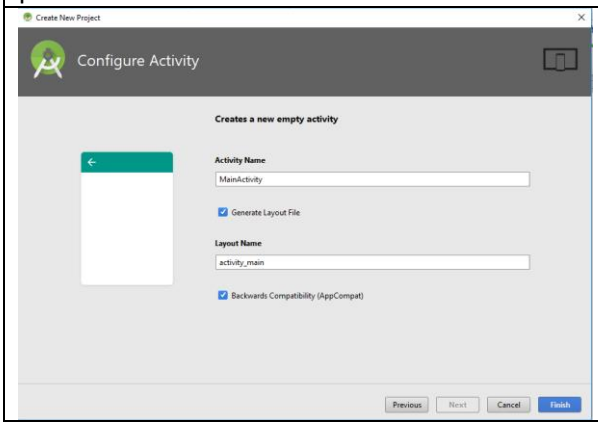
Proyecto con menú(base un activity llamado listview que invoca otros activities)

Crea un nuevo proyecto, desde el menú de eclipse en **File->New->Other, luego Android Application Project** ubicado dentro de la subcarpeta **Android** y presione el botón **Next>.**

Para este ejercicio escribe como nombre de la aplicación será ***Carnet02*,** El proyecto será: **sv.ues.fia.eisi.carnet02 .** selecciona **next** y busca las carpeta donde almacenas tus proyectos(carnet) .



Selecciona la API 22 y presiona Next.	Selecciona Empty Activity y luego Next
	

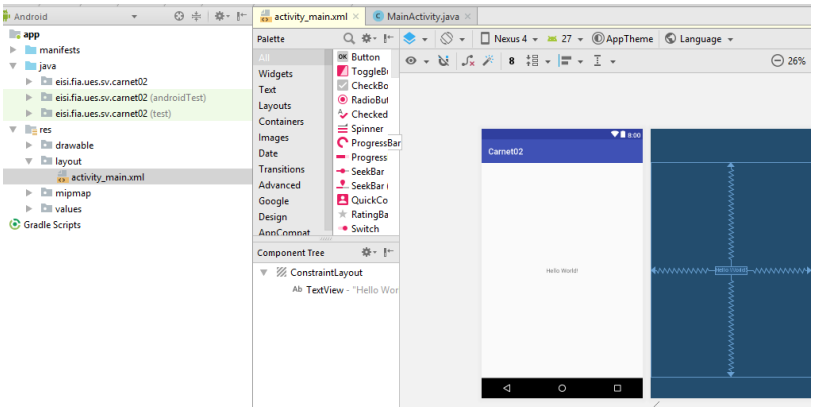
presiona Finish.	
	

.

Menú (Activity principal)

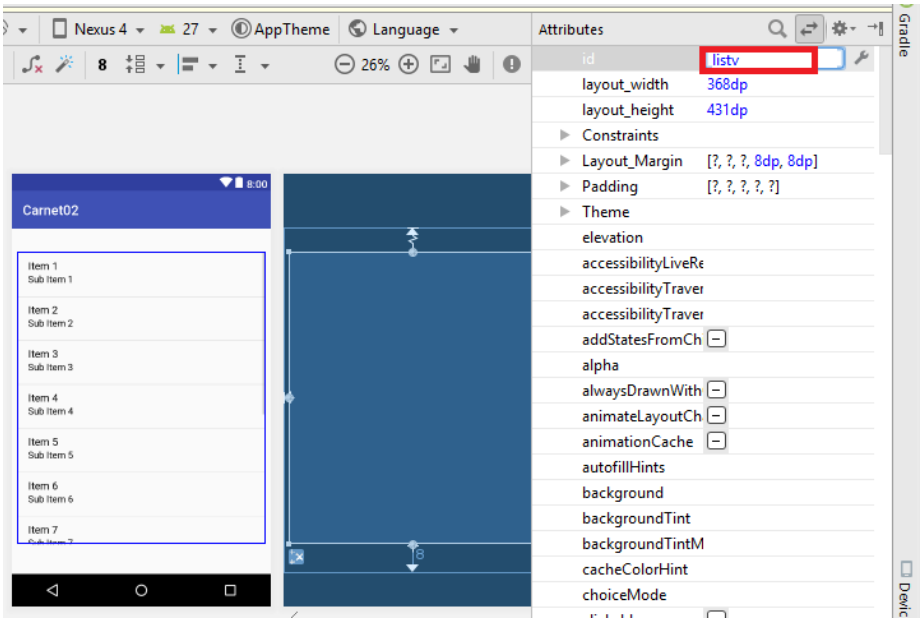
ListView(capa de presentación o vista)

Dentro del **Proyect Explorer** navega por el directorio **Carnet02->res->** y dentro de la carpeta **layout, Presiona clic en activity_main y clic en la pestaña de text**



Sustituye el código por el que se muestra a continuación:

Elimina el control EditView(HelloWord) e incorpora una lista, ponle el atributo id como se muestra(listv)



Nota: No olvide poner los puntos de fijación a los extremos (anclas).

Con lo anterior en formato xml el layout quedara asi:

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.constraint.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context="eisi.fia.ues.sv.carnet02.MainActivity">

    <ListView
        android:id="@+id/listv"
        android:layout_width="368dp"
        android:layout_height="431dp"
        android:layout_marginBottom="8dp"
        android:layout_marginEnd="8dp"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="1.0"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />
</android.support.constraint.ConstraintLayout>
```

Esta lista tiene un identificador llamado listv, este nombre lo utilizaremos en el controlador.

En la lista se tienen como atributos más relevantes los siguientes:

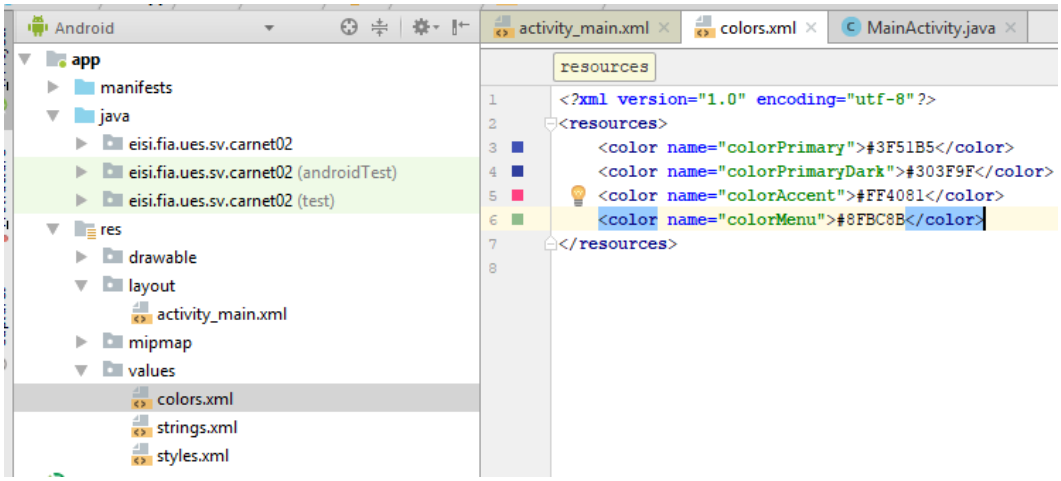
ATRIBUTOS RELEVANTES	DESCRIPCIÓN
android:background=””	Permite colocar el color de fondo a la lista
android:entries=""	Permite agregar un Array que contenga los componentes de la lista
android:clickable=""	Permite agregar un evento que realice un conjunto de Acciones y esto se hace colocando un nombre

Color del layout (opcional)

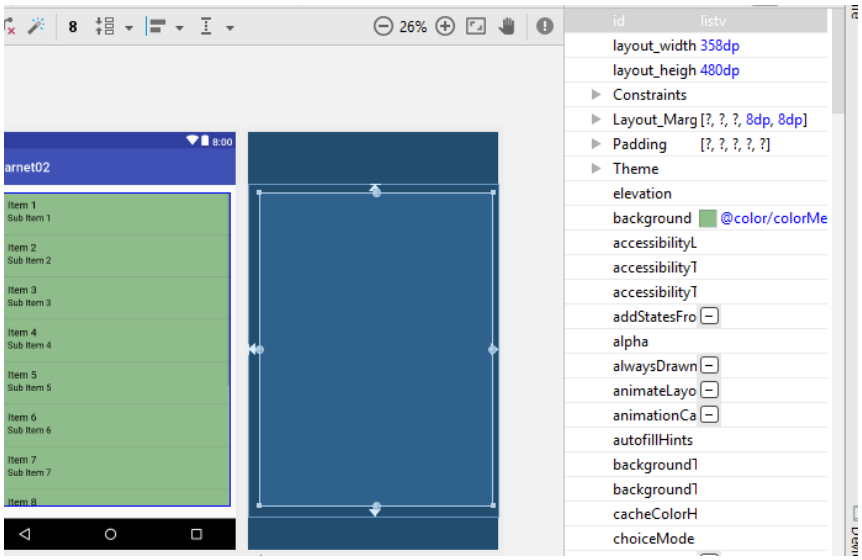
Aunque el color de fondo del layout está bien, puedes probar cambiar la propiedad background

De la siguiente forma.

a) agrega a string.xml el color que tendrá el layout(lista)



b) cambia la propiedad background de la lista en la ventana de diseño



En la vista Text en xml se vera asi:

activity_main.xml | strings.xml | colors.xml | MainActivity.java

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.constraint.ConstraintLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context="eisi.fia.ues.sv.carnet02.MainActivity">

    <ListView
        android:id="@+id/listv"
        android:layout_width="358dp"
        android:layout_height="480dp"
        android:layout_marginBottom="8dp"
        android:layout_marginEnd="8dp"
        android:background="@color/colorMenu"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="1.0"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />
</android.support.constraint.ConstraintLayout>
```

Si miras oscuro el color, puedes cambiarlo por otro como:FFFFFF o 8FBC8B u otro

(Consulta sitios como este si quieres ver más opciones http://roble.pntic.mec.es/apuente/nombre.html)

ListView (Aplicación o controlador)

Ahora abra la Activity *MainActivity.java,* ubicada dentro la carpeta **src->sv.ues.fia.carnet02**.

Usaremos dentro de ella, la clase predeterminada ArrayAdapter como adaptador para nuestro ListView y un diseño predefinido para mostrar la lista que ya viene incluido en el SDK de Android.

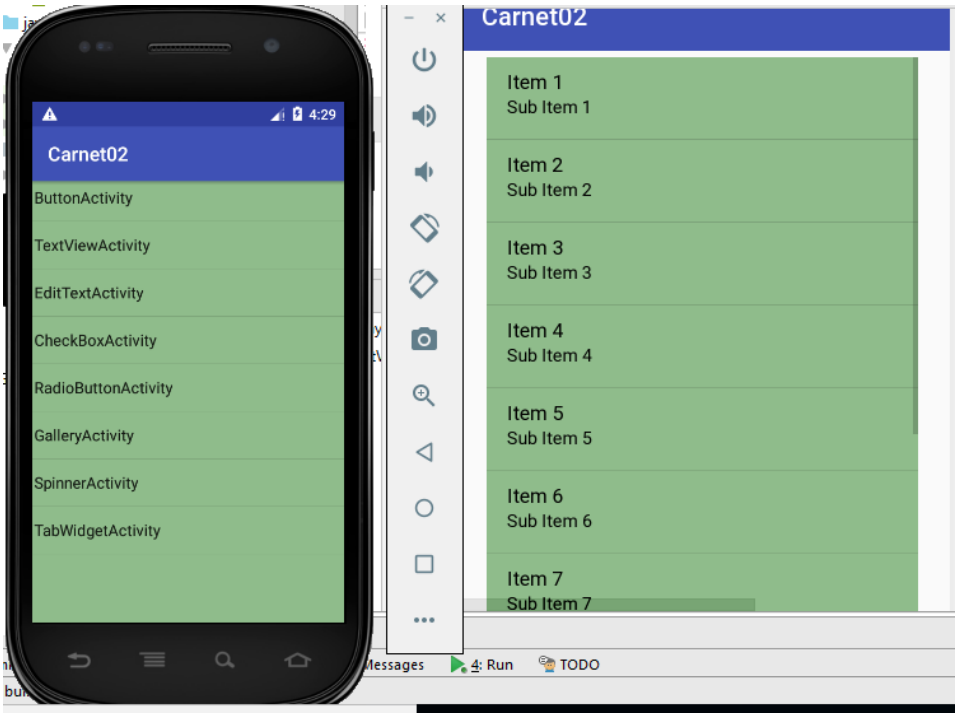
El código que deberá contener es el siguiente:

```
package eisi.fia.ues.sv.carnet02;
import android.content.Intent;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.AdapterView;
import android.widget.ArrayAdapter;
import android.widget.ListView;
public class MainActivity extends AppCompatActivity implements
AdapterView.OnItemClickListener {
    String values[]={"ButtonActivity","TextViewActivity",
            "EditTextActivity","CheckBoxActivity",
            "RadioButtonActivity","GalleryActivity",
            "SpinnerActivity","TabWidgetActivity"};
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        ArrayAdapter<String> adaptador =new ArrayAdapter<String>(this,
                android.R.layout.simple_list_item_1,values);
        ListView listVi =(ListView)findViewById(R.id.listv);
        listVi.setAdapter(adaptador);
        listVi.setOnItemClickListener(this);
    }

    @Override
    public void onItemClick(AdapterView<?> parent, View view, int
position, long id) {
        String nombreValue=values[position];

        try{

            Class<?> clase=Class.forName("eisi.fia.ues.sv.carnet02
."+nombreValue);
            Intent inte = new Intent(this,clase);
            this.startActivity(inte);
        }catch(ClassNotFoundException e){
            e.printStackTrace();
        }
    }
}
```

En el anterior código, hicimos un **implements** para otorgarle a nuestra Activity la capacidad de realizar una acción cuando se genere un evento Click sobre uno de los elementos de la lista.

El ListView obtiene los datos que se mostraran (en este caso los nombres que declaramos en el arreglo de String) a través de un adapter. Un adapter es responsable de proporcionar el modelo de datos para el ListView y para convertir los datos en los campos de la lista.

Ejecutamos nuestra aplicación y deberá mostrar lo siguiente:

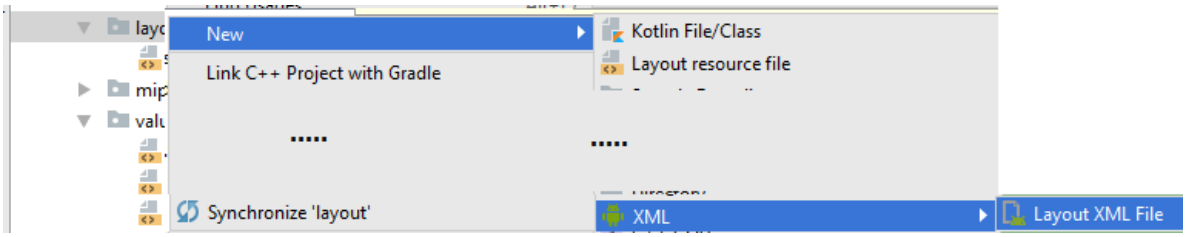


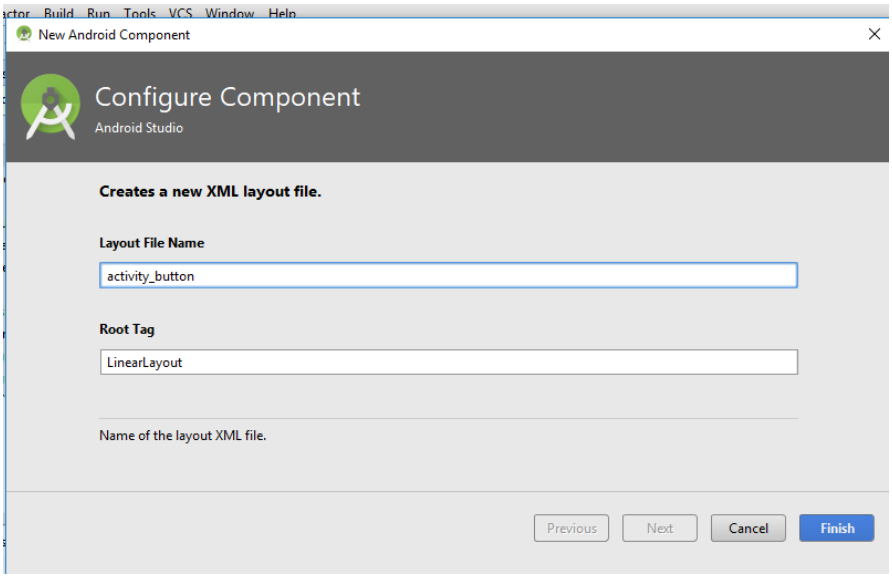
Con lo anterior tenemos creado el menú, pero sin que se pueda entrar a las opciones

Primera opción del menú(activity del botón)

Button (presentación o vista)

Cree un archivo xml llamado activity_*button.xml* en la carpeta ***Layout*** de nuestro proyecto con el siguiente código(clic derecho en layout, clic en new, clic XML, Layout File)





Sustituir el código del archivo activity_button.xml por el siguiente:

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

    <TextView
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:layout_marginTop="10dip"
        android:text="@string/title_button"
        />

    <Button
        android:id="@+id/button1"
```

```
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:onClick="actionButton"
        android:padding="10dp"
        android:text="@string/button1" />

</LinearLayout>
```

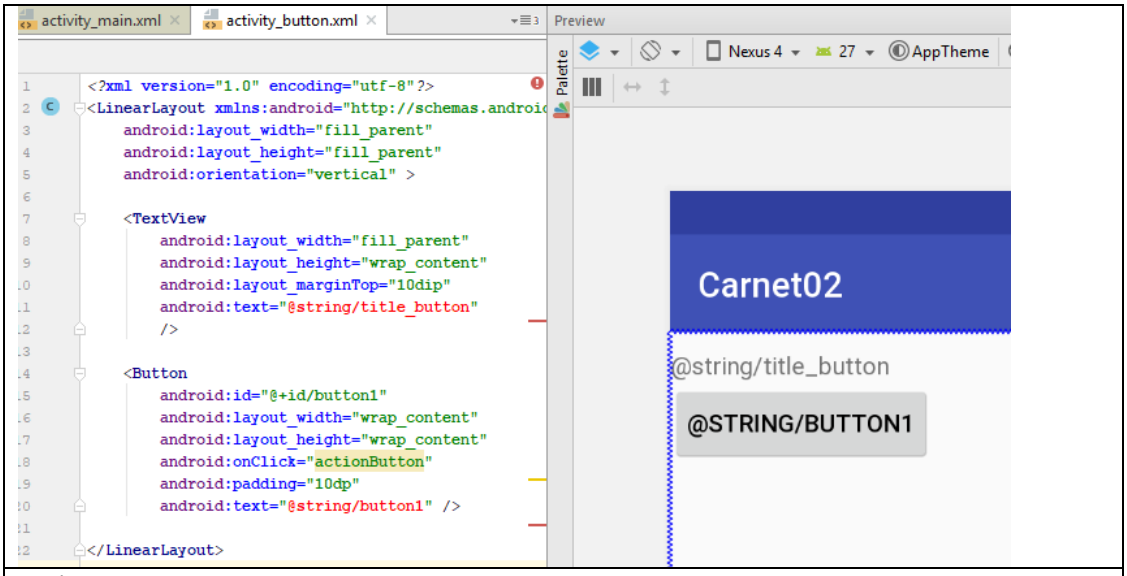
El control de boton tiene las siguientes propiedades:

ATRIBUTOS RELEVANTES	DESCRIPCIÓN
android:text=""	Permite colocar el nombre que aparece en el botón
android:textColor=""	Permite agregar color al texto que aparece en el botón
android:textSize=""	Permite colocar el tamaño de letra
android:shadowColor=""	Permite agregar sombra al texto colocado en el botón

Este elemento de la interfaz de usuario permite realizar una acción específica al usuario cuando es pulsado.

Aquí podemos observar el uso del atributo `android:onClick` que especifica el método a ejecutar cuando el *Button* sea pulsado(actionButton).

Nota: si digitaste o copiaste el código anterior, te saldrán dos errores como los que se muestra:

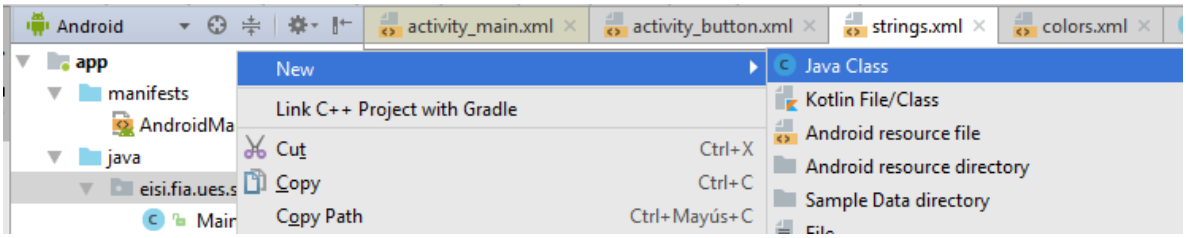


En vista text

Haz dos variables String(title_button=" Boton" y button1= "Presionar aquí") y continua, **si no te acuerdas como corregirlo, ve al anexo1**

Button (Aplicación o controlador)

Ahora el siguiente paso es crear dentro de nuestro paquete una nueva clase java(activity) llamada ***ButtonActivity.java,***



para esto da click derecho sobre el paquete ***eisi.ues.fia.carnet02*** ubicado dentro en la carpeta **src** desde el **Package Explorer**, y selecciona **New->Class**, y agrega el nombre en campo **Name**, y verifica que todo este tal y como la siguiente imagen, y posteriormente presiona **OK**:

Ahora abre ButtonActivity.java y agregar el siguiente código:

```
package eisi.fia.ues.sv.carnet02;
import android.app.Activity;
import android.os.Bundle;
import android.view.View;
import android.widget.Toast;

/**
 * Created by ing. Cesar on 11/3/2018.
 */
public class ButtonActivity extends Activity {
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_button);

    }
    public void actionButton(View v){
        Toast.makeText(ButtonActivity.this,
                "Boton Presionado", Toast.LENGTH_SHORT).show();

    }
}
```

Observaciones:

En el código anterior se define el método *actionButton* que es llamado cada vez se pulsa sobre el *Button,* este mostrará un mensaje indicándonos que el botón ha sido pulsado.

Y para terminar agregamos lo siguiente en el archivo **AndroidManifest.xml,** dentro del cuerpo de la etiqueta `<application>`

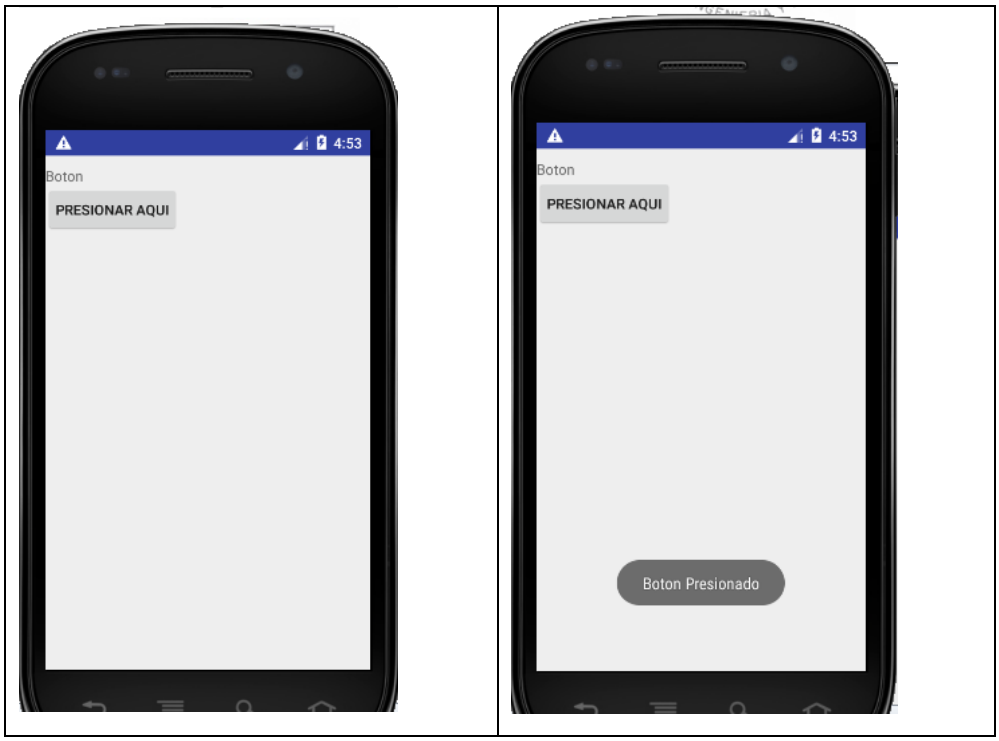
```
<activity
      android:name=".ButtonActivity"
      android:label="Button" >
</activity>
```

Si tienes dudas al respecto, consulta el anexo2

activity_main.xml | activity_button.xml | strings.xml | ButtonActivity.java | AndroidManifest.xml

```
1  <?xml version="1.0" encoding="utf-8"?>
2  <manifest xmlns:android="http://schemas.android.com/apk/res/android"
3      package="eisi.fia.ues.sv.carnet02">
4
5      <application
6          android:allowBackup="true"
7          android:icon="@mipmap/ic_launcher"
8          android:label="Carnet02"
9          android:roundIcon="@mipmap/ic_launcher_round"
10         android:supportsRtl="true"
11         android:theme="@style/AppTheme">
12         <activity android:name=".MainActivity">
13             <intent-filter>
14                 <action android:name="android.intent.action.MAIN" />
15
16                 <category android:name="android.intent.category.LAUNCHER" />
17             </intent-filter>
18         </activity>
19         <activity
20             android:name=".ButtonActivity"
21             android:label="Button" >
22         </activity>
23
24     </application>
25
26 </manifest>
```

Al correr nuestra aplicación y seleccionar la opción “Button” del menú se observará de la siguiente manera:



Nota:

Si no te corre la opción botón ve a modificar el código del programa MainActivity.java

Y sustituye la línea con nombre del paquete con error por la siguiente:

```
@Override
public void onItemClick(AdapterView<?> parent, View view, int position, long id) {
    String nombreValue=values[position];

    try{
        Class<?> clase=Class.forName("carnet02.fia.ues.sv.carnet02."+nombreValue);
        Intent inte = new Intent(this,clase);
        this.startActivity(inte);
    }catch(ClassNotFoundException e){
        e.printStackTrace();
    }
```

Forma Alternativa de creación de Vista(xml) y controlador(java)

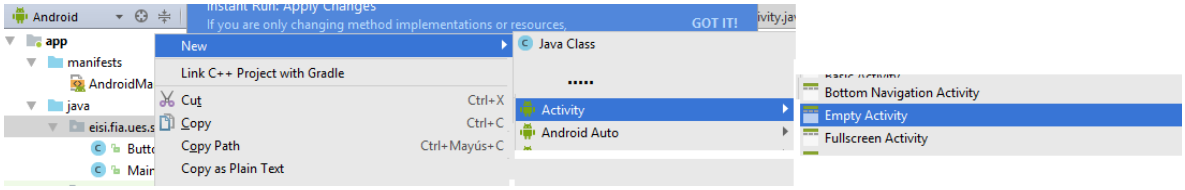
A continuación crearemos las demás opciones del menú de una forma más ágil.

Segunda opción del menú(activity del TextView)

Primeramente definiremos las opciones(activities) que faltan:
`"TextViewActivity","EditTextActivity","CheckBoxActivity","RadioButtonActivity","GalleryActivity","SpinnerActivity"`

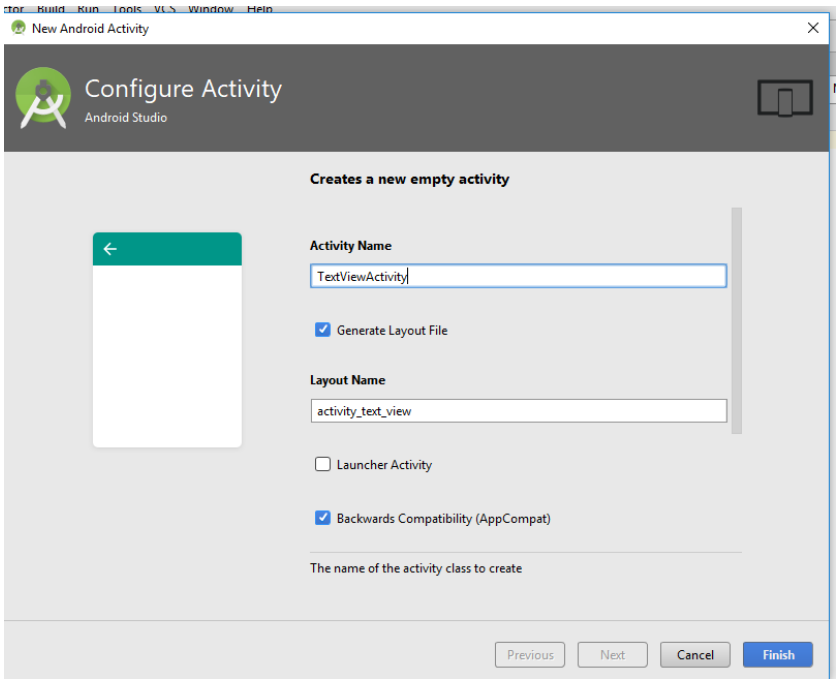
A continuación crearemos todos los programas con su interfaz para luego ir a sustituir los códigos en cada opción (de la segunda a la octava).

`Presionamos clic derecho en el paquete eisi.fia.ues.sv.carnet02`



Luego clic en New, Activity, Empty Activity

Se digita el nombre del Activity(en este caso TextViewActivity)

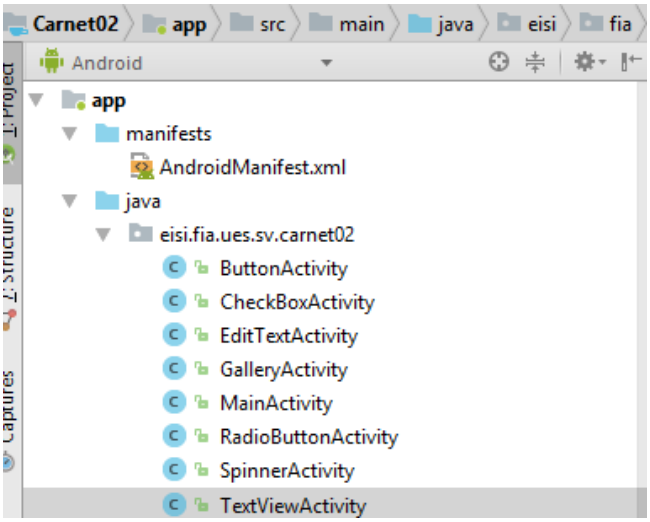
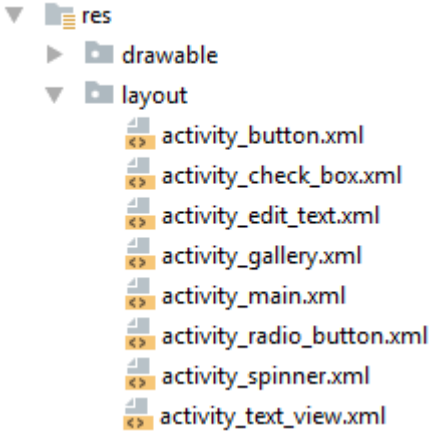


Automáticamente se creara el activity_text_view.xml(interfaz gráfica) y se registrara el activity en el AndroidManifest.

Presionaremos clic en Finish

Luego repetiremos lo anterior hasta hacer los otros seis activities(con interfaz).

Al finalizar veremos una estructura de árbol como la siguiente:

Programas(activities):	Y los layouts:
	

TextView(presentación o vista)

El TextView se utiliza para mostrar texto al usuario. Este es el punto de vista más básico que sin duda se encontrará cuando se desarrollan aplicaciones para Android. Si necesita que los usuarios puedan editar el texto que se muestra, se debe utilizar la subclase de TextView -> EditText.

Para demostrar el uso de un *TextView* modificaremos el archivo llamado activity*_text_view.xml* en la carpeta ***layout*** del proyecto y modificaremos su código de la siguiente manera:

```
<?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:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="@string/text" />
```

```
</LinearLayout>
```

El control de textview tiene las siguientes propiedades:

ATRIBUTOS RELEVANTES	DESCRIPCIÓN
android:enabled=""	Permite que se visualice o no el texto
android:textColor=""	Permite agregar color al texto que aparece en el botón
android:textSize=""	Permite colocar el tamaño de letra
android:shadowColor=""	Permite agregar sombra al texto colocado en el botón

TextView(Aplicación o controlador)

Ahora verificaremos (no se programara nada) que en la clase llamada *TextViewActivity.java* está el código de invocación de la interfaz gráfica:

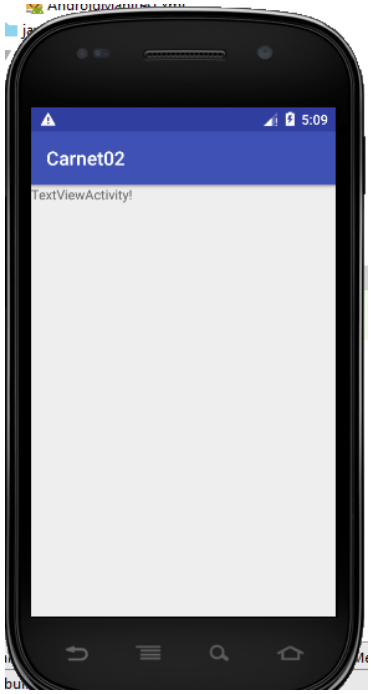
TextViewActivity.java

```
package eisi.fia.ues.sv.carnet02;

import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;

public class TextViewActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_text_view);

    }
}
```

También modificaremos el archivo *strings.xml* con el texto que nosotros deseamos. Para nuestro ejemplo debemos agregar las siguientes líneas de código dentro del cuerpo de la etiqueta <resource>:

```
    <string name="text">TextViewActivity!</string>
```

Al correr la aplicación y seleccionar “TextView “ en el menú deberías ver lo siguiente



Tercera opción del menú(activity del EditText)

EditText (presentación o vista)

Es una caja de Texto con capacidad de ser editable. Modifica el archivo activity_*edit_text.xml* y modifique su contenido con el siguiente código.

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical" >
    <EditText
        android:id="@+id/edit"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="@string/mensajeayuda"/>

    <ToggleButton
        android:id="@+id/toggle"
        android:checked="true"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:onClick="myMetodo"
        android:layout_centerHorizontal="true"
```

```
        android:layout_below="@id/edit" />

</RelativeLayout>
```

Observa que se utiliza adicionalmente, un **ToggleButton** al cual le asignaremos ciertos aspectos mediante código Java para que controle el estado del **EditText**.

> Nota: Debes de crear la variable String llamada mensajeayuda para que aparezca el hint "Texto inicial de prueba". Si no recuerdas como hacerlo consulta el anexo 3

El control de edittext tiene las siguientes propiedades:

ATRIBUTOS RELEVANTES	DESCRIPCIÓN
android:maxLength	Permite ingresar un filtro que limita el número máximo de caracteres que pueden ser escritos en el EditText
android:inputType	El tipo de dato que debera ingresarse al EditText
android:editable	Si el EditText permitirá ser editado o no.

EditText (Aplicación o controlador)

Ahora modifica el archivo *EditTextActivity.java*.

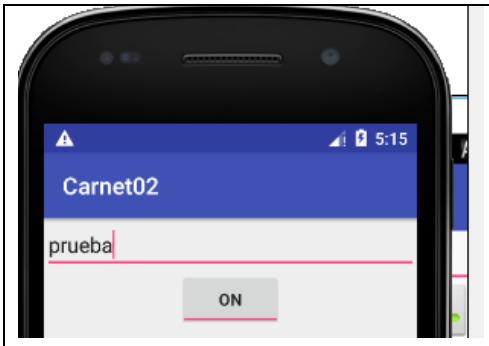
```
package eisi.fia.ues.sv.carnet02;
import android.support.v7.app.AppCompatActivity;

import android.os.Bundle;

import android.view.View;

import android.widget.EditText;

import android.widget.Toast;

import android.widget.ToggleButton;

public class EditTextActivity extends AppCompatActivity {

   @Override

   protected void onCreate(Bundle savedInstanceState) {

      super.onCreate(savedInstanceState);

      setContentView(R.layout.activity_edit_text);
```

```
  }

  public void myMetodo(View v){

     ToggleButton bt=(ToggleButton)findViewById(R.id.toggle);

     EditText edt= (EditText)findViewById(R.id.edit);


     if(bt.isChecked()){

        edt.setEnabled(true);

     }else{

        edt.setEnabled(false);

     }

     Toast.makeText(this, edt.getText().toString(), Toast.LENGTH_SHORT).show();

  }

}
```

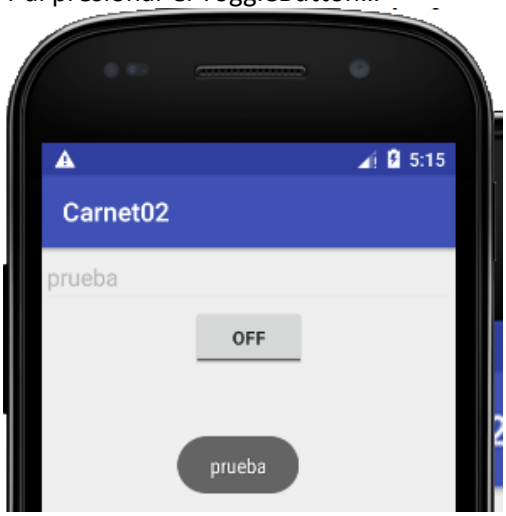
Agregue en el archivo strings.xml la siguiente linea

```
  <string name="mensajeayuda">Texto inicial de prueba</string>
```

Guarde y ejecute el programa. Al seleccionar la opción "EditText" del menú, debería observar algo como esto:



Y al presionar el ToggleButton…



Cuarta opción del menú(activity del CheckBox)

CheckBox (presentación o vista)

El **CheckBox** es un widget de Android, el cual consiste en un botón de dos estados que puede estar seleccionado o no. Pueden colocarse tantos como se necesiten y al mismo tiempo pueden estar seleccionados según se necesite.

Modifica el archivo activity_*check_box.xml* y modifique su contenido con el siguiente código.

```
<?xml version="1.0" encoding="utf-8"?>
<TableLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:stretchColumns="1"
    >
    <TableRow>
        <TextView android:text="@string/titulochequeos" />
    </TableRow>
    <TableRow>
        <TextView android:text="@string/opcion1"/>
        <CheckBox android:id="@+id/checkbox1"
            android:text="@string/text1"
            android:onClick="miMetodo1"
            />
    </TableRow>
```

```
    <TableRow>
        <TextView android:text="@string/opcion2" />
        <CheckBox android:id="@+id/checkbox2"
            android:text="@string/text2"
            android:onClick="miMetodo2"
            />
    </TableRow>

    <TableRow>
        <TextView android:text="@string/opcion3" />
        <CheckBox android:id="@+id/checkbox3"
            android:text="@string/text3"
            android:onClick="miMetodo3"
            />
    </TableRow>
</TableLayout>
```

Haga las variables string que se requieren para que su interfaz quede de la siguiente manera(si tiene dudas de como hacerlo consulte el anexo 3)

En el código pudimos observar el uso del atributo `android:onClick` que especifica el método a ejecutar cuando el *CheckBox* es seleccionado.

ATRIBUTOS RELEVANTES	DESCRIPCIÓN
android:checked=""	Indica cuando el checkbox esta activado o no
android:text=""	Texto que será colocado a la par del checkbox, pero de preferencia se recomienda el uso de un TextView

CheckBox(Aplicación o controlador)

Ahora el siguiente paso es modificar el archivo *CheckboxActivity.java*. Dicha Clase lo que permitirá es que cada vez que se seleccione un checkbox aparezca un mensaje que especifique que se ha seleccionado y que al mismo tiempo se sumen las cantidades que tienen los checkbox a un lado, dependiendo de los checkbox seleccionados así será la cantidad de la suma. Para ello agregar el siguiente código:

Recordar que es necesario realizar todos los imports necesarios para que nuestra aplicación funcione correctamente.

```
package eisi.fia.ues.sv.carnet02;
import android.support.v7.app.AppCompatActivity;

import android.os.Bundle;

import android.view.View;

import android.widget.CheckBox;

import android.widget.Toast;



public class CheckBoxActivity extends AppCompatActivity {

    /** Called when the activity is first created. */

    int sumador=0;

    int acum1=0;

    int acum2=0;

    int acum3=0;

    @Override

    public void onCreate(Bundle savedInstanceState) {

        super.onCreate(savedInstanceState);

        setContentView(R.layout.activity_check_box);

    }
```

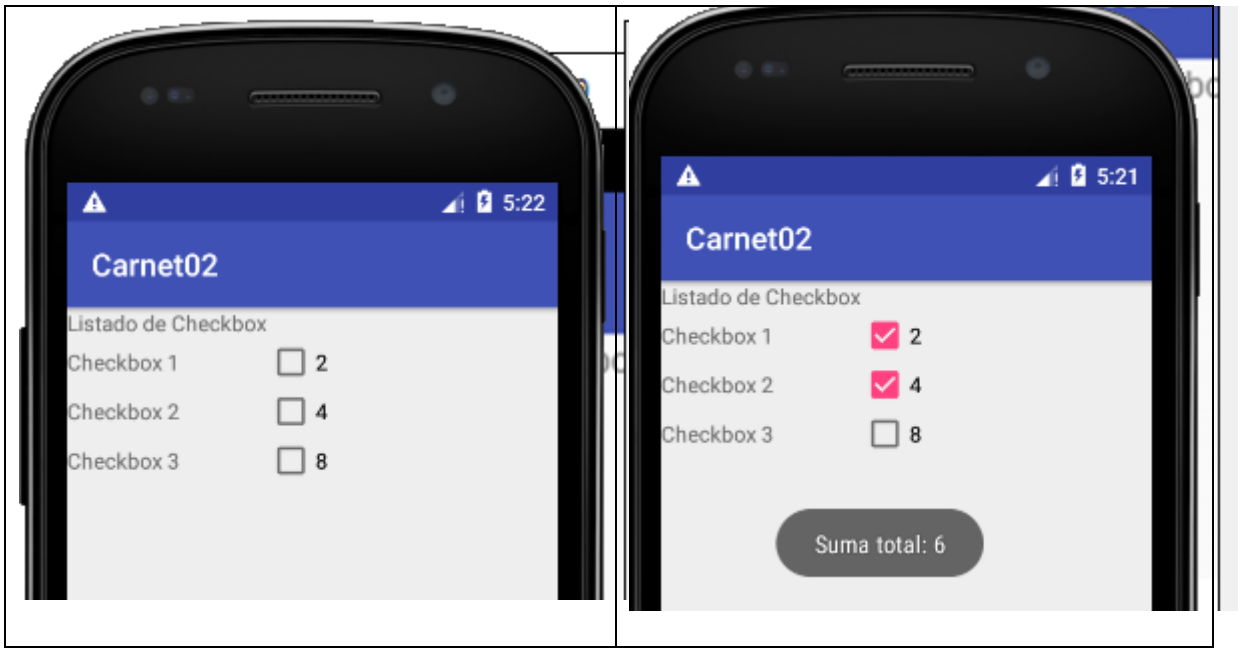
```
    public void miMetodo1(View v){

        CheckBox save1 = (CheckBox)findViewById(R.id.checkbox1);

        if (save1.isChecked()) {

            Toast.makeText(CheckBoxActivity.this,    "Seleccionado    Checkbox1",
Toast.LENGTH_SHORT).show();

            acum1=2;

        } else {

            acum1=0; }

        acumular();}

    public void miMetodo2(View v){

        CheckBox save2 = (CheckBox)findViewById(R.id.checkbox2);

        if (save2.isChecked()) {

            Toast.makeText(CheckBoxActivity.this,    "Seleccionado    Checkbox2",
Toast.LENGTH_SHORT).show();

            acum2=4;

        } else {

            acum2=0; }

        acumular();}

    public void miMetodo3(View v){

        CheckBox save3 = (CheckBox)findViewById(R.id.checkbox3);

        if (save3.isChecked()) {

            Toast.makeText(CheckBoxActivity.this,    "Seleccionado    Checkbox3",
Toast.LENGTH_SHORT).show();

            acum3=8;

        }  else {

            acum3=0;}

        acumular();}
```

```
    public void acumular(){

        sumador=acum1+acum2+acum3;

        Toast.makeText(CheckBoxActivity.this,    "Suma    total:    "+    sumador,
Toast.LENGTH_LONG).show();

    }

}
```

Observaciones:

Los métodos miMetodo1, miMetodo2 y miMetodo3 son los que nos permiten reconocer cuando un checkbox es seleccionado o no. Al mismo tiempo que los acumuladores y el sumador se actualizan dependiendo de los checkbox seleccionados. Al mismo tiempo que se muestran los mensajes a través de Toast.

Al correr nuestra aplicación y seleccionar la opción “CheckBox” de la lista se observará de la siguiente manera:



Quinta opción del menú(activity del RadioButton)

RadioButton(presentación o vista)

Es un botón de dos estados <activado o desactivado>, que a diferencia del **CheckBox** una vez que ha sido activado por el usuario, no puede ser desactivado al presionarlo o *clickearlo* nuevamente.

Para la implementación de los **RadioButton** utilizaremos adicionalmente un **RadioGroup** que se encargara de almacenar una serie de **RadioButton,** para evitar que más de uno de ellos este <activado> a la vez.

Para esto modifique el archivo activity_*radio_button.xml* y posteriormente agregue el siguiente código en él.

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

    <RadioGroup
        android:id="@+id/grupo"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_centerInParent="true"
        android:layout_centerHorizontal="true">

        <RadioButton
            android:id="@+id/radiobutton1"
            android:layout_width="match_parent"
             android:layout_height="wrap_content"
             android:text="@string/RadioButton1"
             android:onClick="myMetodo" />

        <RadioButton
            android:id="@+id/radiobutton2"
            android:layout_width="match_parent"
             android:layout_height="wrap_content"
             android:text="@string/RadioButton2"
             android:onClick="myMetodo" />

        <RadioButton
            android:id="@+id/radiobutton3"
            android:layout_width="match_parent"
             android:layout_height="wrap_content"
             android:text="@string/RadioButton3"
             android:onClick="myMetodo" />
```

```
    </RadioGroup>

    <TextView
            android:id="@+id/texto"
            android:text="@string/mensaje_error_radiobutton"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_centerHorizontal="true"
            android:layout_below="@id/grupo"/>

</RelativeLayout>
```

Como puedes observar, para la implementación de un **RadioGroup** solo basta con definir todos los **RadioButton** dentro del cuerpo de la etiqueta <RadioGroup>. Se utiliza un **TextView** para indicar cuál RadioButton se encuentra <activado> a la hora de ejecutar el programa, para ello cada uno de los RadioButton responderá a un evento **onClick** que definiremos en el método **myMetodo()** de *RadioButtonActivity*.

Haga las variables string que se requieren para no tener advertencias(warnings)

ATRIBUTOS RELEVANTES	DESCRIPCIÓN
android:checked	Si estado “chequeado” o “no chequeado”

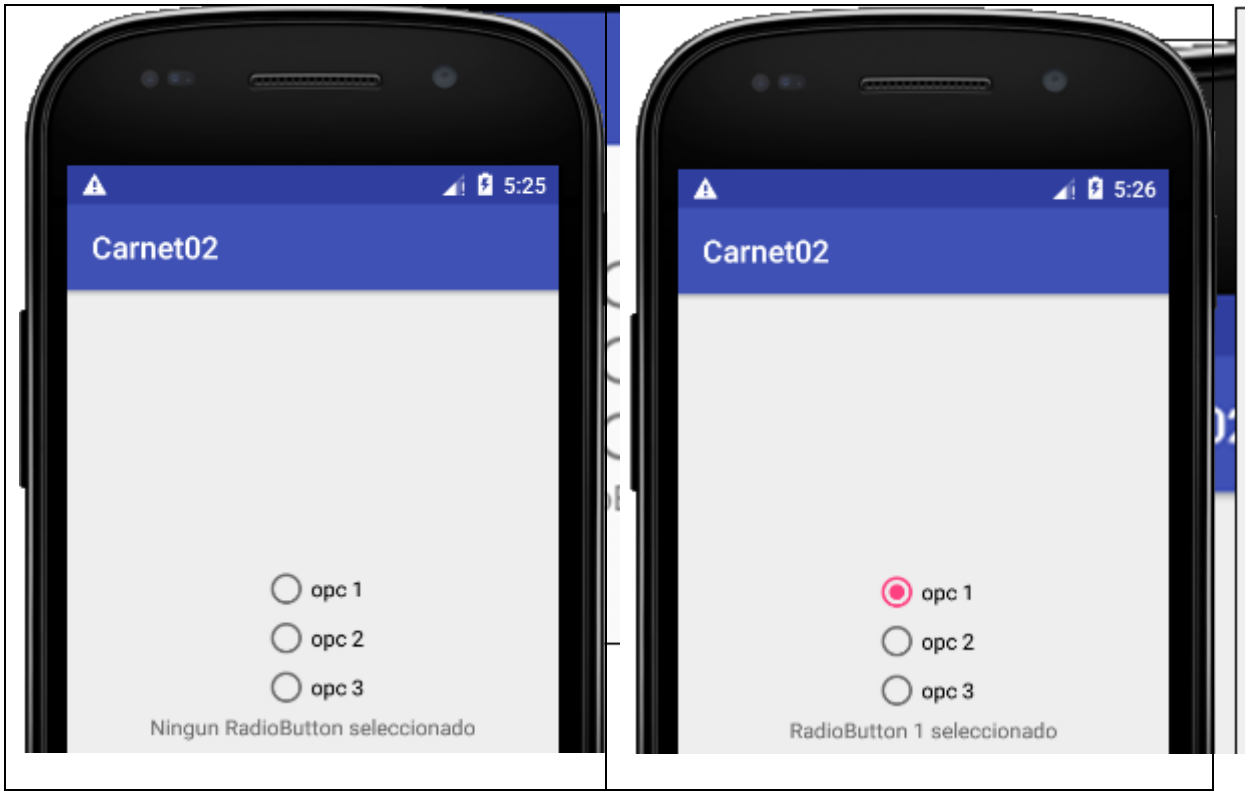
RadioButton(Aplicación o controlador)

Ahora modifique el archivo `sv.ues.fia.carnet02` y nombre *RadioButtonAcivity.java*. El código que deberá contener es el siguiente:

```
package eisi.fia.ues.sv.carnet02;
import android.support.v7.app.AppCompatActivity;

import android.os.Bundle;

import android.view.View;

import android.widget.RadioGroup;

import android.widget.TextView;


public class RadioButtonActivity extends AppCompatActivity {


    /** Called when the activity is first created. */
```

```
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_radio_button);
    }
    public void myMetodo(View v){
        RadioGroup rbg=(RadioGroup)findViewById(R.id.grupo);
        TextView text=(TextView)findViewById(R.id.texto);

        switch(rbg.getCheckedRadioButtonId()){
            case R.id.radiobutton1:
                text.setText("RadioButton 1 seleccionado"); break;
            case R.id.radiobutton2:
                text.setText("RadioButton 2 seleccionado"); break;
            case R.id.radiobutton3:
                text.setText("RadioButton 3 seleccionado"); break;
        }
    }

}
```

Guarde y ejecute el programa. Al seleccionar el opción “RadioButtonActivity” del menú, debería observar algo como esto:



Sexta opción del menú(activity de Galery)

Gallery(presentación o vista)

Gallery es un widget de Android que nos permite crear un layout para desplegar ítems de manera horizontal y al mismo tiempo colocarlos al centro del layout.

Para esto modifique el archivo activity_*gallery.xml* y posteriormente agregue el siguiente código en él.

```
<?xml version="1.0" encoding="utf-8"?>
<Gallery xmlns:android="http://schemas.android.com/apk/res/android"
   android:id="@+id/gallery"
   android:layout_width="fill_parent"
```

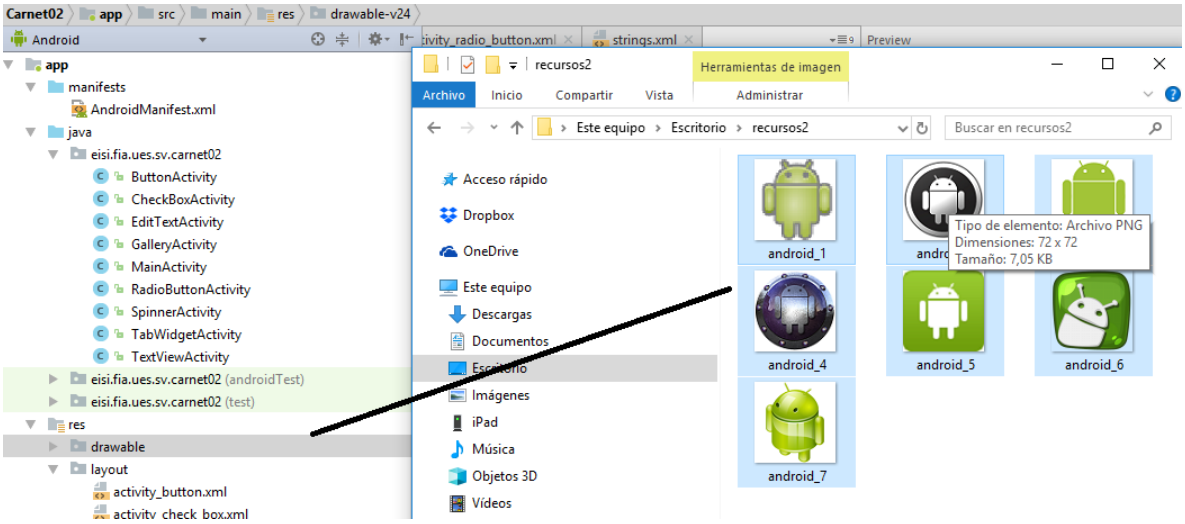
```
  android:layout_height="wrap_content" >
</Gallery>
```

ATRIBUTOS RELEVANTES	DESCRIPCIÓN
android:animationDuration=""	Permite colocar cuanto tiempo duraran las animaciones colocadas en el gallery
android:background=""	Permite colocar el color de fondo
android:soundEffectsEnabled=""	Permite habilitar los sonidos de una animación o con el simple hecho de pasar de imagen

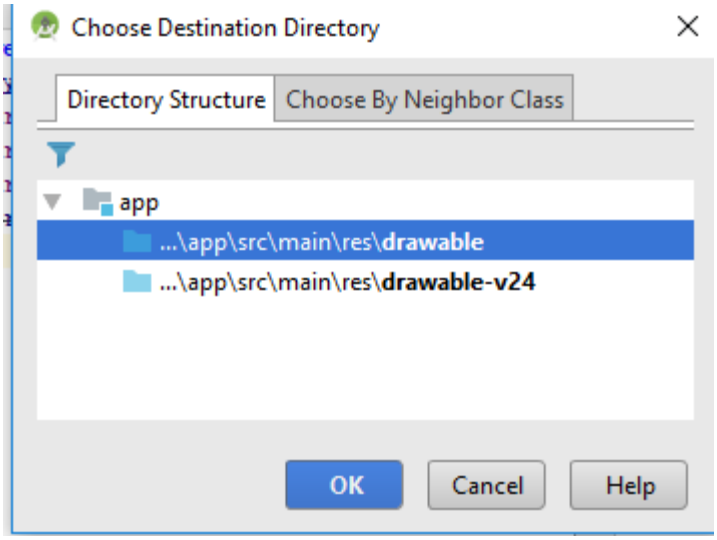
Gallery(Aplicación o controlador)

Descargue la carpeta recursos2 del aula virtual o del ftp de la asignatura y descomprimalo en una carpeta en el escritorio

Luego copielos archivos(clic derecho, copy) a la carpeta res/drawable de su proyecto(clic derecho, paste)



Presione ok para confirmar



Ahora modifique el archivo *GalleryActivity.java*. El código que deberá contener es el siguiente:

```
package carnet02.fia.ues.sv.carnet02;


import android.content.Context;

import android.content.res.TypedArray;

import android.support.v7.app.AppCompatActivity;

import android.os.Bundle;

import android.view.View;

import android.view.ViewGroup;

import android.widget.AdapterView;

import android.widget.BaseAdapter;

import android.widget.Gallery;

import android.widget.ImageView;

import android.widget.Toast;
```

```
public class GalleryActivity extends AppCompatActivity {


  /** Called when the activity is first created. */
  @Override
  public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_gallery);


    @SuppressWarnings({ "deprecation", "deprecation" })
    Gallery gallery = (Gallery) findViewById(R.id.gallery);
    gallery.setAdapter(new ImageAdapter(this));


    gallery.setOnItemClickListener(new AdapterView.OnItemClickListener() {
      public void onItemClick(@SuppressWarnings("rawtypes") AdapterView parent, View v, int position, long id) {
        Toast.makeText(GalleryActivity.this, "Usted Visualiza la imagen numero:" + position, Toast.LENGTH_SHORT).show();
      }
    });
  }



  public class ImageAdapter extends BaseAdapter {
    int mGalleryItemBackground;
    private Context mContext;
```

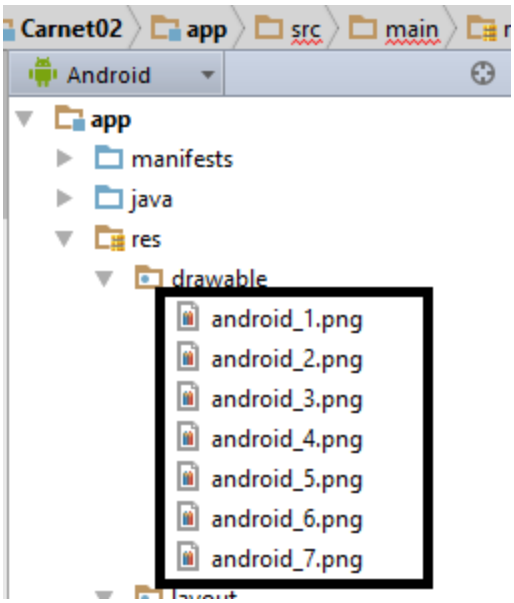
```
private Integer[] mImageIds = {
        R.drawable.android_1,
        R.drawable.android_2,
        R.drawable.android_3,
        R.drawable.android_4,
        R.drawable.android_5,
        R.drawable.android_6,
        R.drawable.android_7
};

public ImageAdapter(Context c) {
    mContext = c;
    TypedArray attr = mContext.obtainStyledAttributes(R.styleable.GalleryActivity);
    mGalleryItemBackground = attr.getResourceId(
            R.styleable.GalleryActivity_android_galleryItemBackground, 0);
    // attr.recycle();
}

public int getCount() {
    return mImageIds.length;
}

public Object getItem(int position) {
```

```
        return position;
    }

    public long getItemId(int position) {
        return position;
    }

    public View getView(int position, View convertView, ViewGroup parent) {
        ImageView imageView = new ImageView(mContext);

        imageView.setImageResource(mImageIds[position]);
        imageView.setLayoutParams(new Gallery.LayoutParams(150, 100));
        imageView.setScaleType(ImageView.ScaleType.FIT_XY);
        imageView.setBackgroundResource(mGalleryItemBackground);

        return imageView;
    }
  }
}
```

Observaciones:

Cabe recalcar que para lograr ejecutar una View de Gallery es necesario que cada componente funcione de acuerdo con el método setOnItemClickListener(**new** OnItemClickListener en cual permite que se mande a la clase **public class** ImageAdapter **extends** BaseAdapter, la posición de la selección y el ítem que se ha seleccionado. Dentro de la clase IMAGEADAPTER es donde se encuentra el diseño de gallery y los métodos que por defecto reconoce como lo es **public int**

getCount() . Dentro de esta misma clase se coloca un arreglo en donde se ponen las imágenes que deben agregarse en la carpeta **res-> drawable**



El método ImageAdapter es el que permite utilizar lo que es el background por default utilizado en el sistema Android, esto lo hace a través del xml denominado atrrs.xml el cual lo crearemos posteriormente. Y por último el método **public** View getView(**int** position, View convertView, ViewGroup parent) que es en el cual se reunen todos los elementos esenciales del diseño de Galley para crear lo que es su vista a través del XML GAlleryView.

Ahora es necesario irse a la carpeta **res->values** y crear un nuevo resource con el nombre de *atrrs.xml,* y colocarle el siguiente código:

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
  <declare-styleable name="GalleryActivity">
    <attr name="android:galleryItemBackground" />
  </declare-styleable>
</resources>
```

Al correr nuestra aplicación y seleccionar la opción "Gallery" de la lista se observará de la siguiente manera:

Septima opción del menú(activity de Spinner)

Spinner(presentación o vista)

ATRIBUTOS RELEVANTES	DESCRIPCIÓN
android:prompt=""	Permite colocar la lista de elementos que llenan el Spinner a través de un Array

Este componente le permite al usuario seleccionar un elemento de entra varios disponibles en la vista.

Modificaremos el archivo llamado activity_*spinner.xml* dentro de la carpeta *Layout* de nuestro proyecto y lo dejamos de la siguiente manera:

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

     <TextView
```

```
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:layout_marginTop="10dip"
        android:text="@string/colors_prompt"
    />

    <Spinner
        android:id="@+id/spinner"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:prompt="@string/colors_prompt"
    />

</LinearLayout>
```

Hay que resaltar que el atributo android:prompt especifica la lista de elementos que se mostraran en el *Spinner* para nuestro caso será una lista de colores de las cuales el usuario debe escoger uno.

Modificamos el archivo *strings.xml* agregando el siguiente código:

```
<string name="colors_prompt">Seleccione un Color</string>
  <string-array name="colors_array">
    <item>Blanco</item>
    <item>Negro</item>
    <item>Azul</item>
    <item>Rojo</item>
    <item>Verde</item>
    <item>Amarillo</item>
    <item>Gris</item>
    <item>Rosado</item>
  </string-array>
```

Podemos observar la definición de un <string-array> con las opciones para seleccionar dentro de nuestro *Spinner*.

Spinner(Aplicación o controlador)

A continuación creamos dentro de nuestro paquete la clase *SpinnerActivity.java* con el siguiente código:

```
package eisi.fia.ues.sv.carnet02;


import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.AdapterView;
import android.widget.ArrayAdapter;
import android.widget.Spinner;
import android.widget.Toast;

public class SpinnerActivity extends AppCompatActivity implements
```

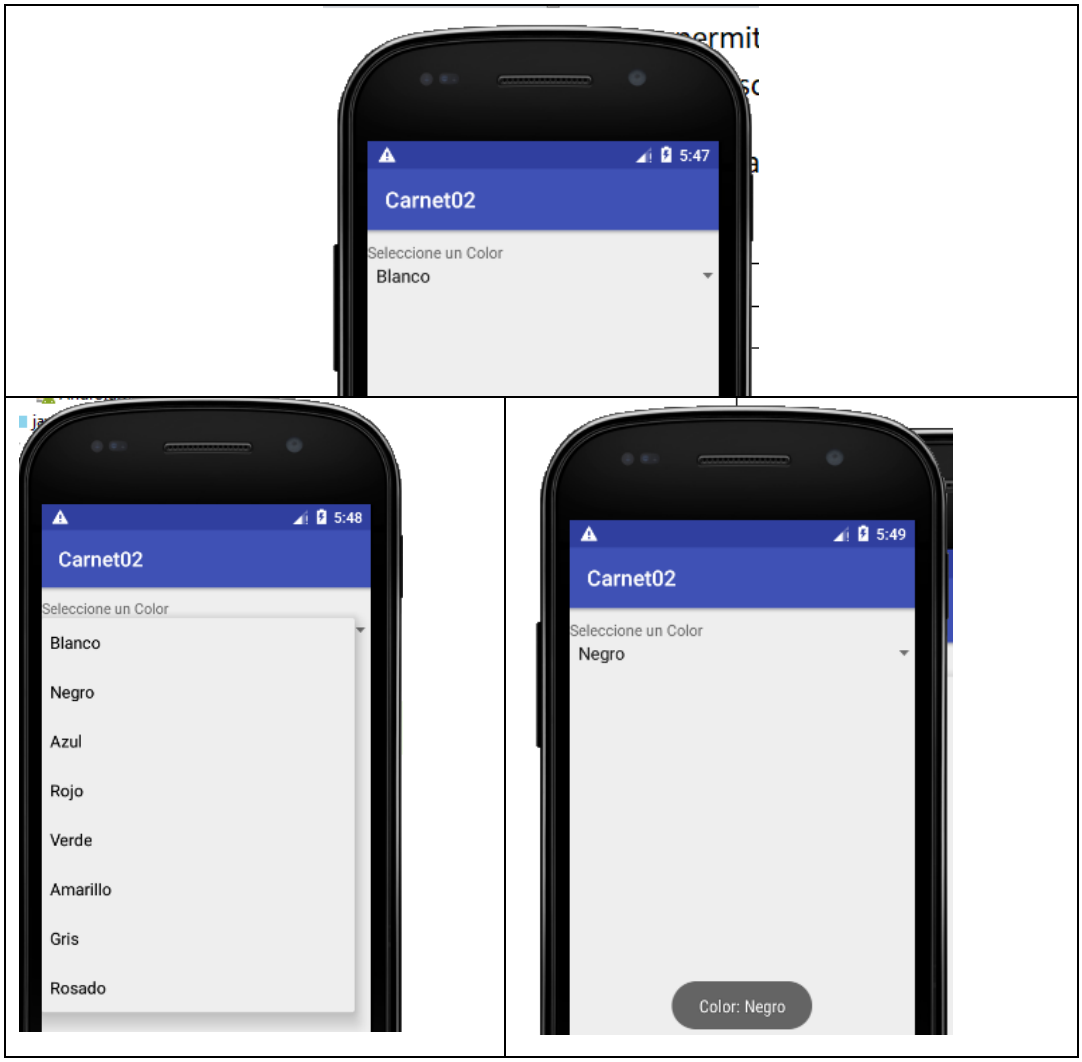
```
AdapterView.OnItemSelectedListener {

    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_spinner);
        Spinner spinner = (Spinner) findViewById(R.id.spinner);
        ArrayAdapter<CharSequence> adapter = ArrayAdapter.createFromResource(
                this, R.array.colors_array,
android.R.layout.simple_spinner_item);

adapter.setDropDownViewResource(android.R.layout.simple_spinner_dropdown_item);
        spinner.setAdapter(adapter);
        spinner.setOnItemSelectedListener(this);
    }

    public void onItemSelected(AdapterView<?> parent, View view, int pos,
                               long id) {
        Toast.makeText(parent.getContext(), "Color: " +
                parent.getItemAtPosition(pos).toString(),
Toast.LENGTH_LONG).show();

    }

    public void onNothingSelected(AdapterView<?> arg0) {
        // No se Utiliza

    }


}
```

Observaciones:

Para poder saber que elemento del *Spinner* fue seleccionado debemos auxiliarnos de la Clase *ArrayAdapter* que proporciona un modelo para manejar un arreglo arbitrario de objetos y así poder obtener los atributos o valores del elemento seleccionado.

Así mismo dentro de nuestra Activity es necesario implementar la interfaz *OnItemSelectedListener* que nos permite establecer las acciones a realizar una vez se seleccione un elemento del *Spinner,* en nuestro caso se desplegara un mensaje indicando el color seleccionado.

Y al ejecutar la aplicación y seleccionar la opción “Spinner” del menú, se mostrara de la siguiente manera:



Comprima su carpeta de proyecto y súbalo en el link respectivo (Guía de Laboratorio 2(Android))

Tarea opcional

Realice a través de un Spinner, cada una de las actividades que mandara a llamar a cada uno de los componentes vistos en la presente guía. Para cada uno de los cuales se tendrán las siguientes modificaciones:

- **EditText: que la información colocada en el edittext aparezca en el Toast y se dé cuando se desactive el edittext.**
- **Checkbox: que al seleccionar cada checkbox se forme una cadena de caracteres.**
- **RadioButton: que haga un contador de cuantas veces se selecciona el radiobutton de un conjunto de tres.**
- **TextView: utilizarlo en la mayoría de las opciones para dar más vista al diseño de la interfaz.**
- **Button: crear tres botones y con la ayuda de los textview realizar diversas operaciones la primera será suma, la segunda resta y la tercera multiplicación.**
- **Gallery: crear su propia galería de fotos y asignar mensajes respecto a la foto seleccionada.**

Anexos

Anexo1

En strings.xml (inicial)

```
        <string name="button1">Presionar aqui</string>
        <string name="title">Botón de Comando</string>
```

Anexo 2

AndroidManifest.xml(final)

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="carnet02.fia.ues.sv.carnet02">
    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:supportsRtl="true"
        android:theme="@style/AppTheme">
        <activity android:name=".MainActivity">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
        <activity
            android:name=".ButtonActivity"
            android:label="Button" />
        <activity android:name=".TextViewActivity" />
        <activity android:name=".EditTextActivity" />
        <activity android:name=".CheckBoxActivity" />
        <activity android:name=".RadioButtonActivity" />
        <activity android:name=".GalleryActivity" />
        <activity android:name=".SpinnerActivity" />
    </application>

</manifest>
```

Anexo 3

String.xml(final)

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string name="app_name">Carnet02</string>
    <string name="action_settings">Settings</string>
    <string name="hello_world">Hello world!</string>
    <color name="color_menu">#8FBC8B</color>
    <string name="button1">Presionar aqui</string>
    <string name="title_button">Boton</string>
    <string name="title_activity_button">ButtonActivity</string>
    <string name="title_activity_text_view">TextViewActivity</string>
    <string name="title_activity_edit_text">EditTextActivity</string>
    <string name="title_activity_check_box">CheckBoxActivity</string>
    <string name="title_activity_radio_button">RadioButtonActivity</string>
    <string name="title_activity_gallery">GalleryActivity</string>
    <string name="title_activity_spinner">SpinnerActivity</string>
    <string name="title_activity_tab_widget">TabWidgetActivity</string>
    <string name="text">TextViewActivity!</string>
    <string name="mensajeayuda">Texto inicial de prueba</string>
    <string name="opcion1">Checkbox 1</string>
    <string name="opcion2">Checkbox 2</string>
    <string name="opcion3">Checkbox 3</string>
    <string name="text1">2</string>
    <string name="text2">4</string>
    <string name="text3">8</string>
    <string name="titulochequeos">Listado de Checkbox</string>
    <string name="RadioButton1">opc 1</string>
    <string name="RadioButton2">opc 2</string>
    <string name="RadioButton3">opc 3</string>
    <string name="mensaje_error_radiobutton">Ningun RadioButton
seleccionado</string>
    <string name="colors_prompt">Seleccione un Color</string>
    <string-array name="colors_array">
        <item>Blanco</item>
        <item>Negro</item>
        <item>Azul</item>
        <item>Rojo</item>
        <item>Verde</item>
        <item>Amarillo</item>
        <item>Gris</item>
        <item>Rosado</item>
    </string-array>
    <string name="tabcontent">Contenido del Tab</string>

</resources>
```