



HLC

2° DAW

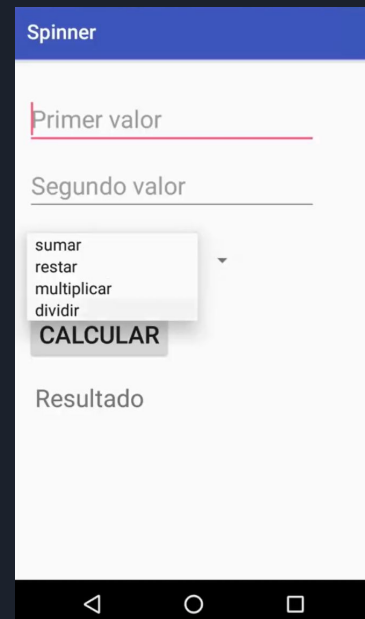
FUNCIONALIDADES ALGO MÁS AVANZADAS.

CALCULADORA CON CONTROL SPINNER

Una vez realizado el último proyecto en el tema anterior: Una calculadora que permitía sumar o restar con Radio Group, que se mostraba bien en pantalla gracias a los ajustes blueprint y que contenía el texto en un fichero xml a parte, añadiremos una funcionalidad más:

Los controles de número o Control Spinner


<https://developer.android.com/develop/ui/views/components/spinner?hl=es-419>





FUNCIONALIDADES ALGO MÁS AVANZADAS. CALCULADORA CON CONTROL SPINNER

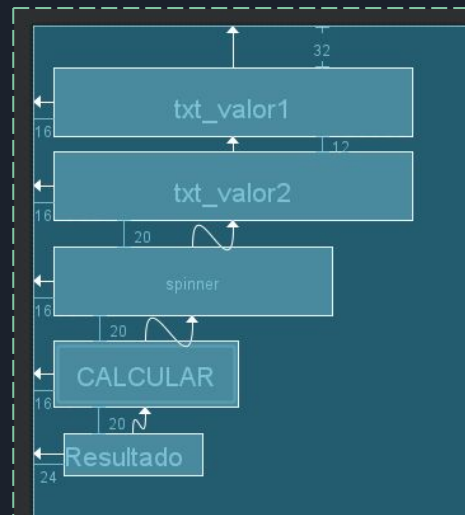
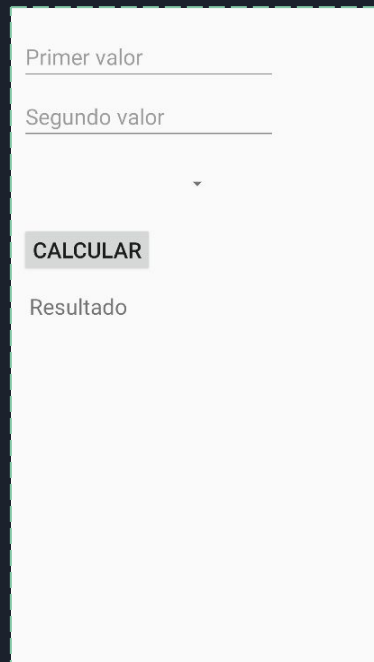
Los **controles spinner** ofrecen una manera rápida de seleccionar un valor de un conjunto. En el estado predeterminado, un spinner muestra su valor actualmente seleccionado. Al tocar el control, se muestra un menú desplegable con todos los demás valores disponibles, de los cuales el usuario puede seleccionar uno.

Planet: 

CALCULADORA CON CONTROL SPINNER

PARTE GRÁFICA

1. Crearemos un nuevo proyecto llamado CalculadoraSpinner, por ejemplo. Seleccionamos una activity vacía.
2. Borramos el Hola Mundo que aparece.
3. Añadimos 5 elementos (o controles):
 - a. 2 Edit Text de tipo Number
 - b. Un Spinner
 - c. Un button
 - d. Un TextView
4. En blueprint establecemos las distancias
5. Colocamos los id
6. Añadimos texto en strings.xml
7. Enlazamos los @strings
8. Podemos aumentar el textSize





CALCULADORA CON CONTROL SPINNER

PARTE LÓGICA


1. Creamos los objetos private en `MainActivity`
2. Dentro de `onCreate()`, agregamos la relación parte gráfica/parte lógica
3. Asignamos un texto o varios textos a nuestro spinner mediante un array
4. Asignamos los valores del array al spinner de la parte gráfica con `ArrayAdapter`
5. Finalmente añadimos esto a nuestro objeto spinner
6. Creamos el método para nuestro botón
7. Dentro de este tenemos que llamar a la selección de nuestro spinner con `getSelectedItem()`
8. Añadimos las condicionales para cada una de las operaciones
9. Validamos para que no se pueda dividir entre 0



CALCULADORA CON CONTROL SPINNER

Una posible implementación con ArrayAdapter

```
protected void onCreate(Bundle savedInstanceState) {  
  
    ...  
  
    spinner1 = findViewById(R.id.spinner);  
  
    String [] opciones = {"sumar","restar","multiplicar","dividir"};  
  
    ArrayAdapter<String> adapter = new ArrayAdapter<String>(this,  
        androidx.appcompat.R.layout.support_simple_spinner_dropdown_item, opciones);  
    spinner1.setAdapter(adapter);  
  
}
```



CALCULADORA CON CONTROL SPINNER

El método para el botón

```
public void calcular(View view){
    String valor1_String = et1.getText().toString();
    String valor2_String = et2.getText().toString();

    int valor1_int = Integer.parseInt(valor1_String);
    int valor2_int = Integer.parseInt(valor2_String);

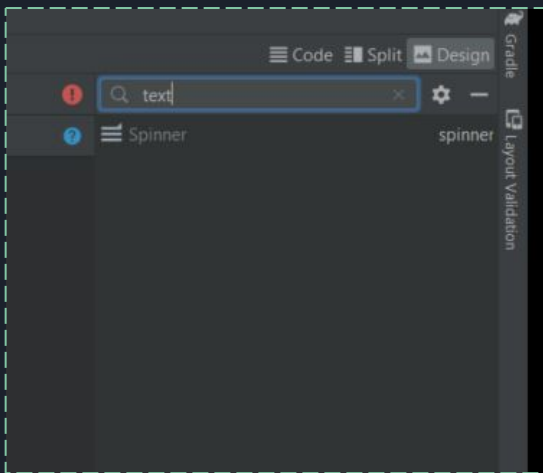
    String seleccion = spinner1.getSelectedItem().toString();

    if(seleccion.equals("sumar")){
        int suma = valor1_int + valor2_int;
        String resultado = String.valueOf(suma);
        tv1.setText(resultado);
    } else if(seleccion.equals("restar")){
        int resta = valor1_int - valor2_int;
        String resultado = String.valueOf(resta);
        tv1.setText(resultado);
    } else if(seleccion.equals("multiplicar")) {
        int multi = valor1_int * valor2_int;
        String resultado = String.valueOf(multi);
        tv1.setText(resultado);
    } else if(seleccion.equals("dividir")){
        if(valor2_int != 0){
            int div = valor1_int / valor2_int;
            String resultado = String.valueOf(div);
            tv1.setText(resultado);
        } else {
            Toast.makeText(this, "No se puede dividir entre cero",
                Toast.LENGTH_LONG).show();
        }
    }
}
```

CALCULADORA CON CONTROL SPINNER

¿Cómo añadir estilo a nuestro spinner?

Si, por ejemplo, necesitásemos modificar el `textSize` de nuestro spinner, nos sería imposible desde el menú lateral de la vista de diseño de Android Studio.



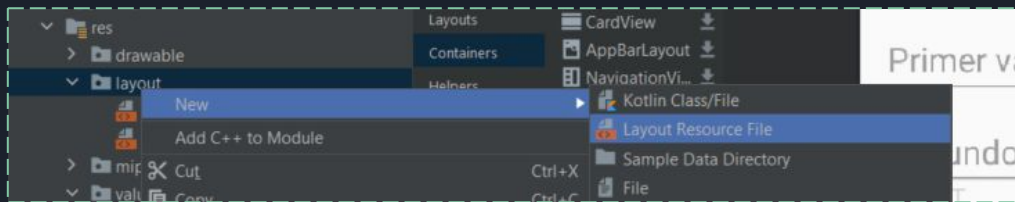
Para modificar un control a nuestro antojo, podemos crear un `Layout Resource File`, es decir, un recurso de diseño.

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

CALCULADORA CON CONTROL SPINNER

¿Cómo añadir estilo a nuestro spinner?

Desde la carpeta **res>layout** de nuestro proyecto, con click derecho, añadiremos un **New >**



Le pondremos un nombre característico, definitorio, para poder llamarlo luego desde nuestra MainActivity: *spinner_estilo*
Vemos cómo se nos ha creado un archivo xml.



CALCULADORA CON CONTROL SPINNER

¿Cómo añadir estilo a nuestro spinner?

Eliminamos el código restante generado para que el contenido de dicho archivo sea:

```
<?xml version="1.0" encoding="utf-8"?>
<TextView
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:background="#022b54"
    android:textSize="24sp"
    android:padding="10sp"
    android:textColor="#ffffff"
    xmlns:android="http://schemas.android.com/apk/res/android" />
```

CALCULADORA CON CONTROL SPINNER

¿Cómo añadir estilo a nuestro spinner?

El contenido de dicho archivo podrá ser modificado a nuestro gusto:

Cómo evitar usar tamaños de diseño hard-coded

Para asegurarte de que tu diseño sea flexible y se adapte a diferentes tamaños de pantalla, debes usar

`"wrap_content"` y `"match_parent"` para el ancho y la altura de la mayoría de los componentes de la vista, en lugar de los tamaños hard-coded.

`"wrap_content"` le indica a la vista que establezca el tamaño necesario para ajustar el contenido dentro de esa vista.

`"match_parent"` hace que la vista se expanda lo más posible dentro de la vista principal.

Por ejemplo:

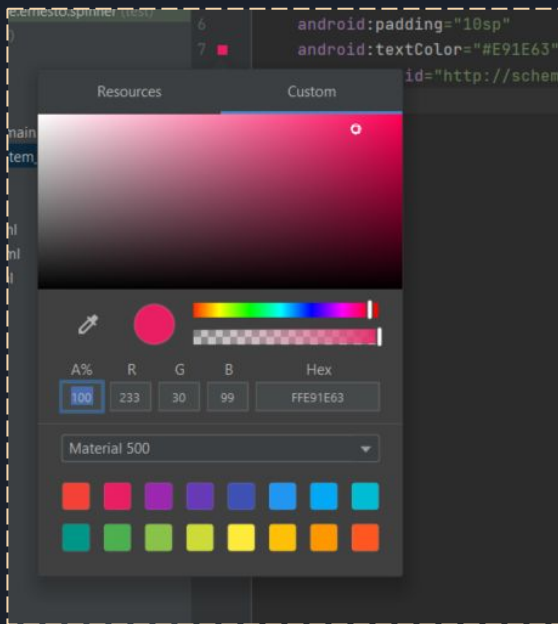
```
<TextView
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:text="@string/lorem_ipsum" />
```



CALCULADORA CON CONTROL SPINNER

¿Cómo añadir estilo a nuestro spinner?

Tip para elegir el color:





CALCULADORA CON CONTROL SPINNER

¿Cómo añadir estilo a nuestro spinner?

Modificamos nuestro MainActivity.java :

```
ArrayAdapter<String> adapter = new ArrayAdapter<String>(this, R.layout.spinner_estilo, opciones);
```

Nombre del recurso de diseño creado



LISTA DE OPCIONES CON CONTROL LISTVIEW

NUEVA PRÁCTICA

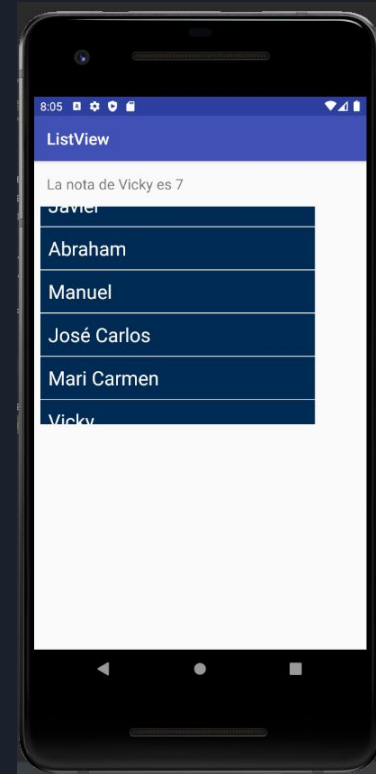
Lista de alumnado

LISTA DE OPCIONES CON CONTROL LISTVIEW

Crearemos una App en la que aparezca una lista desplazable con los nombres de los alumnos y alumnas de clase. Al pulsar en cada uno de ellos aparecerá la nota que tiene en el módulo.

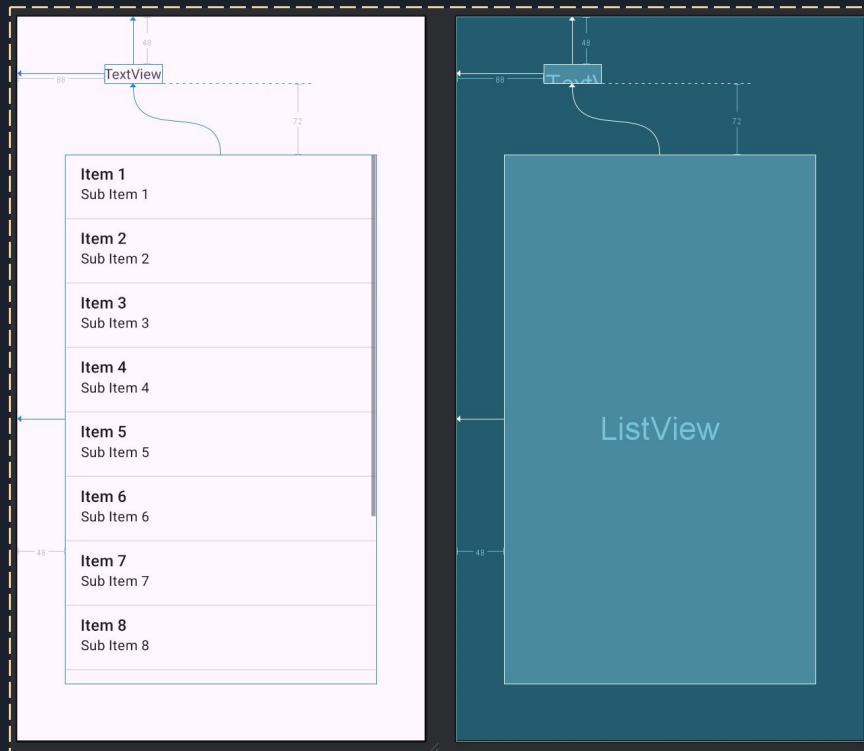
Para ello haremos uso de `ListView` y de `ArrayAdapter` al igual que en la práctica anterior.

También añadiremos un recurso de diseño a la lista para cambiar el color de fondo y texto de la misma.



LISTA DE OPCIONES CON CONTROL LISTVIEW

Dicha app tendrá una vista
de diseño:



LISTA DE OPCIONES CON CONTROL LISTVIEW

Dicha app tendrá una parte lógica:

<https://developer.android.com/reference/android/widget/AdapterView.OnItemSelectedListener>

```
public class MainActivity extends AppCompatActivity {

    private TextView tv1;
    private ListView lv1;

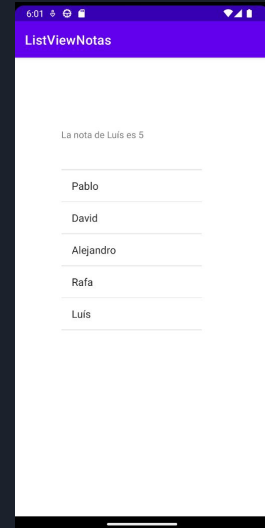
    private String nombres[] = {"Nacho", "Alba", "Manuel", "Pablo", "David", "Alejandro",
        "Rafa", "Luis", "Miriam", "Julio"};
    private String notas[] = {"8", "5", "2", "7", "4", "0", "7", "5", "9", "3"};

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

        tv1 = findViewById(R.id.tv1);
        lv1 = findViewById(R.id.lv1);

        ArrayAdapter<String> adapter = new ArrayAdapter<String>(this, androidx.appcompat.R.layout.support_simple_spinner_dropdown_item,
nombres);
        lv1.setAdapter(adapter);

        lv1.setOnItemClickListener(new AdapterView.OnItemClickListener() {
            @Override
            public void onItemClick(AdapterView<?> adapterView, View view, int i, long l) {
                tv1.setText("La nota de " + lv1.getItemAtPosition(i) + " es " + notas[i]);
            }
        });
    }
}
```





LISTA DE OPCIONES CON CONTROL LISTVIEW

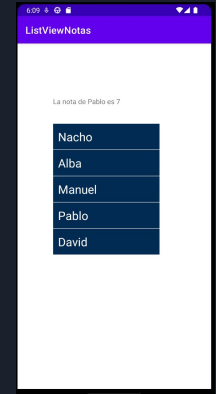
Al igual que en la práctica anterior, creamos un recurso de diseño para añadir estilo a nuestro ListView.

```
<?xml version="1.0" encoding="utf-8"?>
<TextView
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:background="#022b54"
    android:textSize="24sp"
    android:padding="10sp"
    android:textColor="#ffffff"
    xmlns:android="http://schemas.android.com/apk/res/android" />
```

LISTA DE OPCIONES CON CONTROL LISTVIEW

Modificamos MainActivity.java para aplicarle dicho recurso de diseño, incluyendo en la modificación el R.layout:

```
...  
// ArrayAdapter<String> adapter = new ArrayAdapter<String>(this, android.R.layout.simple_list_item_1, nombres);  
ArrayAdapter<String> adapter = new ArrayAdapter<String>(this, R.layout.estilo_lista, nombres);  
...
```





LISTA DE OPCIONES CON CONTROL LISTVIEW

También vamos a modificar el strings.xml

```
<resources>
  <string name="app_name">ListView</string>
  <string name="TextView">Selecciona un nombre</string>
</resources>
```

IMAGE BUTTON

Ahora trabajaremos con un nuevo control.

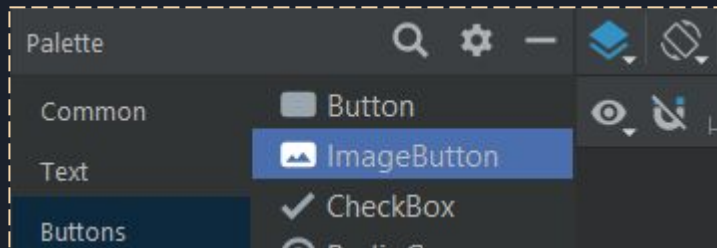
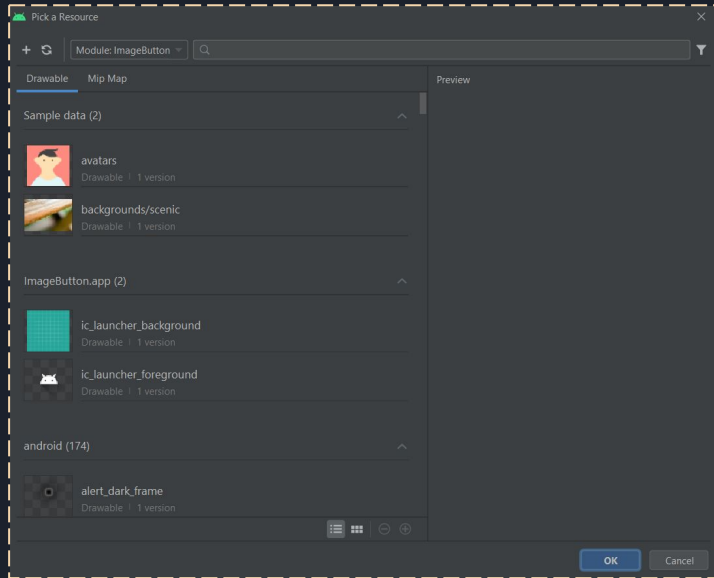


IMAGE BUTTON

Cuando arrastramos ImageButton a nuestra Activity vacía, nos sale esto:



De esta manera podremos elegir entre un amplio abanico de imágenes logos, iconos, etc. que nos facilita Android Studio.

Pero, además, veremos cómo poder añadir imágenes propias y personalizadas.

Recomendamos que el tamaño de dichas imágenes sea de 50x50px . **El nombre de las mismas debe estar en minúsculas.**

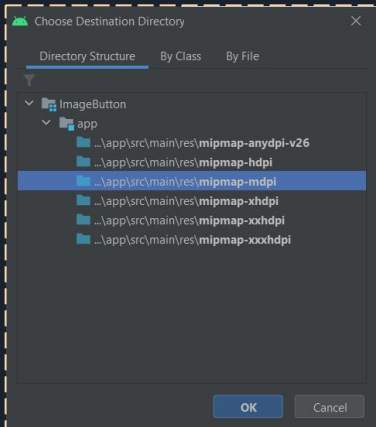
Para nuestra práctica, aún no hacemos nada: le damos a Cancel

IMAGE BUTTON

No agregamos aún ningún control de los que nos propone nuestro IDE, vamos a agregar nosotros un par de imágenes.

Copiamos nuestra imagen (*junta50px.png*)

Nos dirigimos en nuestra vista de Proyectos a **app>res>mipmap**. En esta última haremos click derecho en “Pegar”.



Haremos click en OK en ambas ventanas.

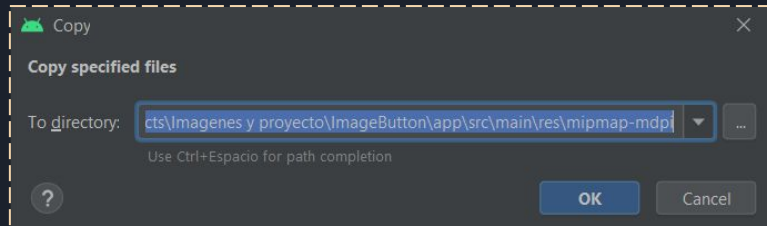
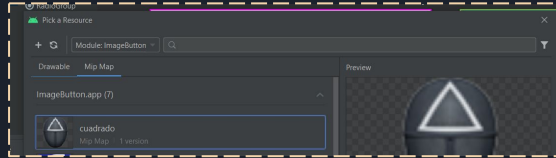


IMAGE BUTTON

Una vez que nos haya agregado nuestras imágenes, arrastramos nuestro Image Button.



Seleccionamos una (en Mip Map) y le damos a ok, y a continuación, la otra, y tendremos ambos controles en nuestra vista de diseño. Establecemos las distancias desde la vista blueprint.

Nos aparecerán un par de warnings, donde nos pide que añadamos un texto descriptivo desde el atributo: `contentDescription`. Como ya sabemos, lo haremos desde nuestro archivo strings.



IMAGE BUTTON

Recordamos que a los botones, como tal, no hay que establecer una relación gráfica/diseño, sino que únicamente le asignamos eventos.

```
public class MainActivity extends AppCompatActivity {  
  
    ...  
  
    public void mensajeBoton (View view){  
        Toast.makeText(this, "Icono", Toast.LENGTH_LONG).show();  
    }  
  
}
```