Práctica 02

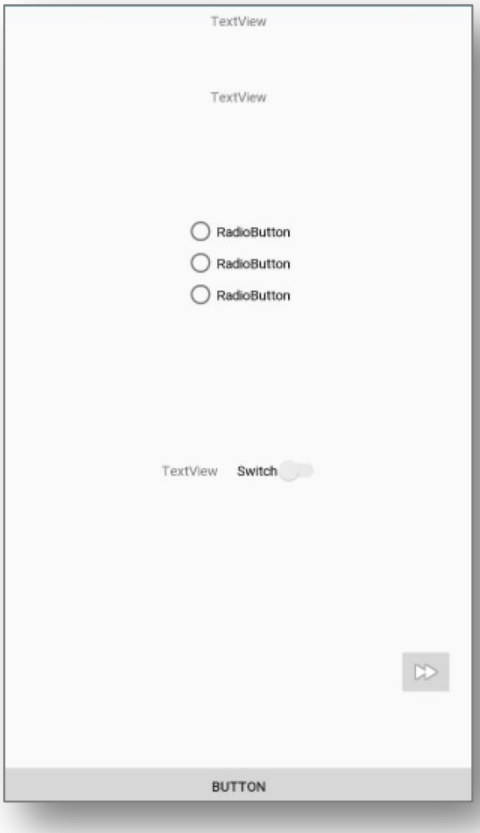
Pasando Información entre Activities

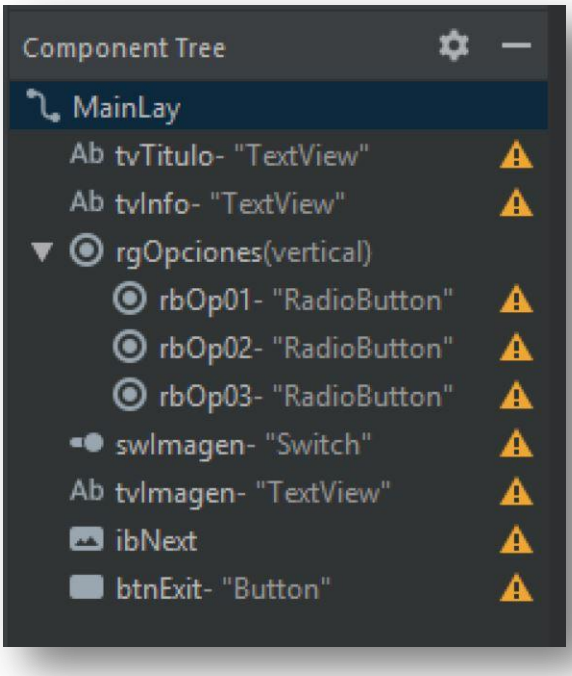
Bueno ya sabemos cómo crear un proyecto nuevo y como dar forma a nuestra primera Activity, pero puede ocurrir que la información que le pedimos al usuario en esta pantalla la usemos para ir a otras pantallas y para mostrar diferentes informaciones.

FASE 1

Nuestra aplicación va a constar de 2 activities en la primera que se va a llamar MainActivity vamos a tener la siguiente distribución:

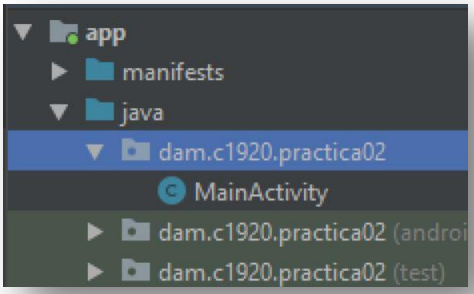
- TEXTVIEW → para el título.
- TEXTVIEW→ para darle información al usuario de lo que tiene que hacer
- RADIOGROUP → Con 3 RADIOBUTTON que son las 3 opciones mutuamente excluyentes que le damos al usuario.
- TEXTVIEW → Es la etiqueta para el switch
- SWITCH → El usuario lo activara o desactivará para indica si quiere las imágenes o no
- IMAGEBUTTON → le hemos puesto el dibujo de avance multimedia y al pulsarlo el usuario ira a la siguiente activity.
- BUTTON →Va a ser el botón de salir de la APP





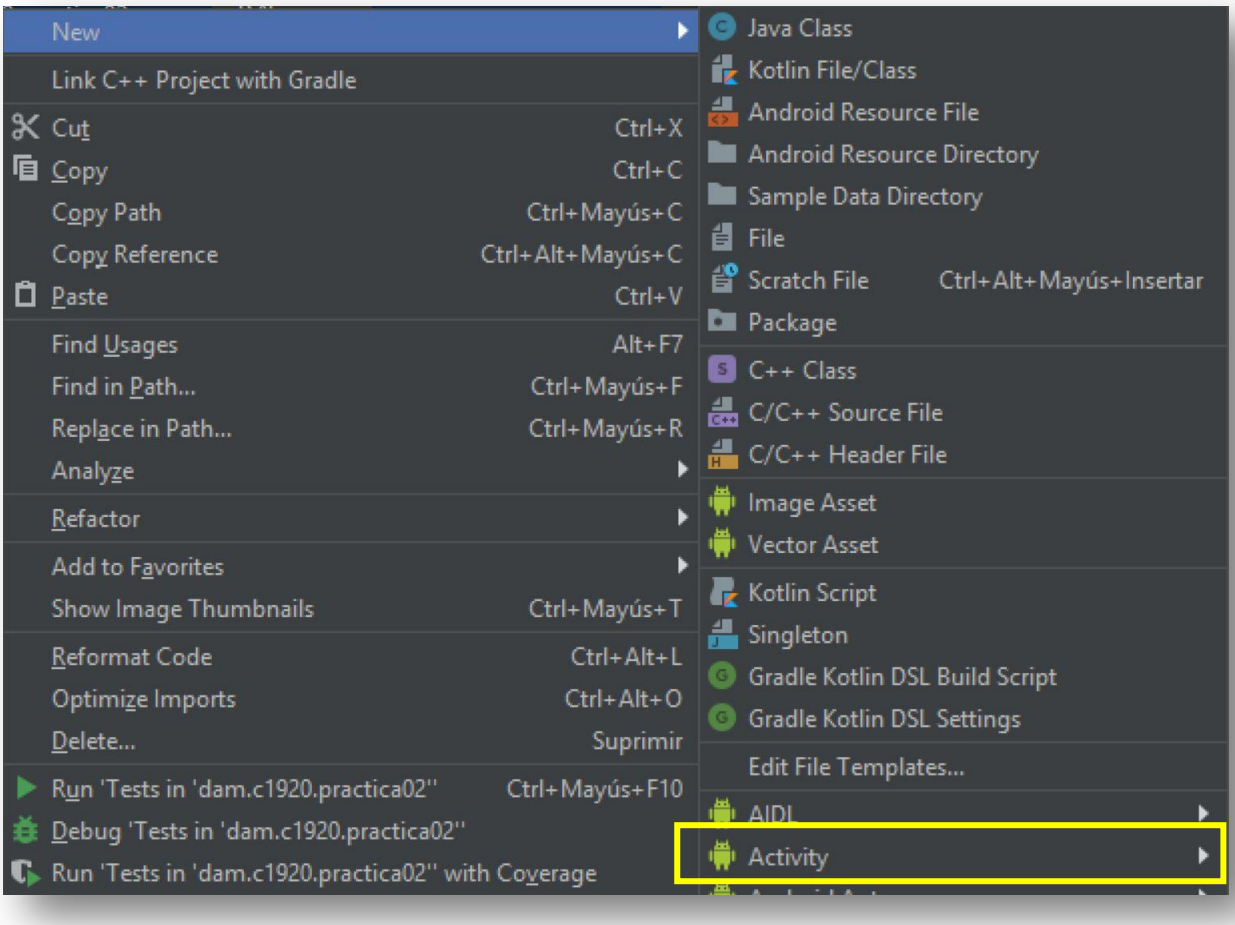
Cuando tenemos los elementos en el layout más o menos distribuidos, les vamos asignando los identificadores que se ven a la izquierda.

Ahora vamos a crear nuestra segunda activity, la vamos a llamar FinalActivity y su layout será activity_final.xml, para ello nos posicionamos sobre el nombre de nuestro proyecto en la carpeta java



tal y como aparece resaltado en la imagen de la izquierda.

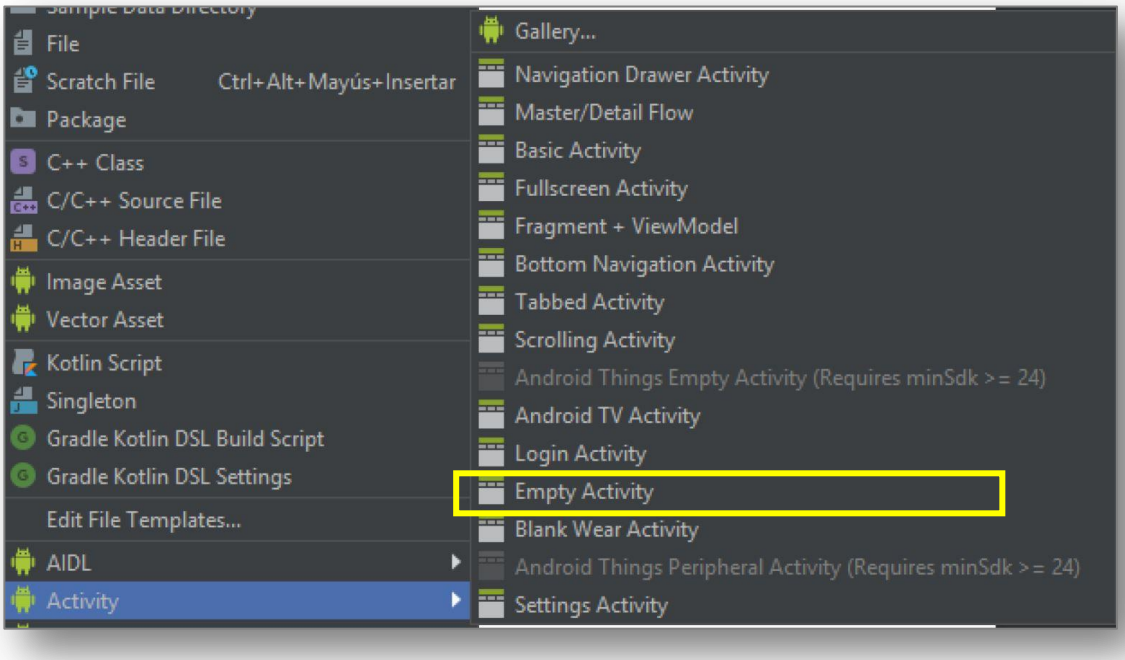
Haremos que aparezca el menú contextual que aparece en la siguiente imagen y desplegamos la opción **New** tal y como se ve a continuación



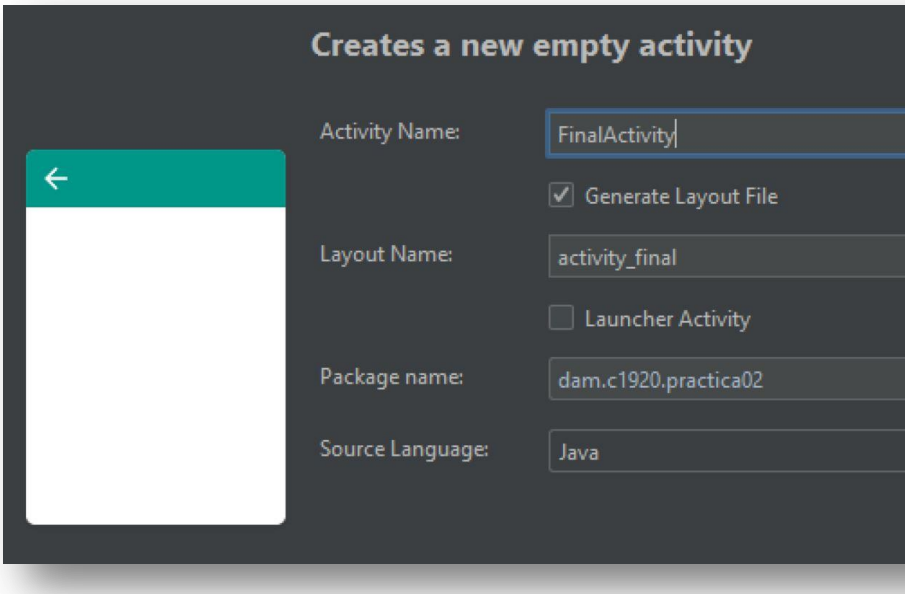
Desplegamos el menú de Activity y obtenemos los tipos de activity que podemos utilizar, tal y como se ve en la siguiente imagen

Práctica 02

Pasando Información entre Activities

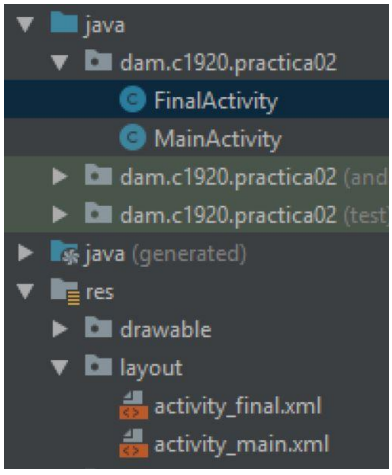


Vamos a utilizar otra vez la Empty Activity.

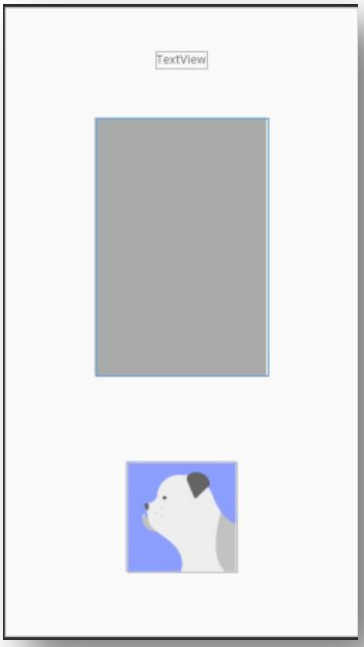


Le damos el nombre a la activity y al layout y pulsamos el botón Finish para que la cree.

A la derecha podemos verificar que se han creado tanto la Activity como el layout

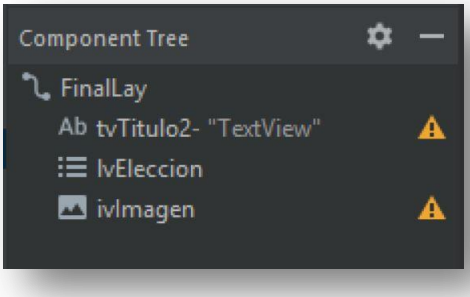


Ahora ya podemos organizar el layout de nuestra FinalActivity:



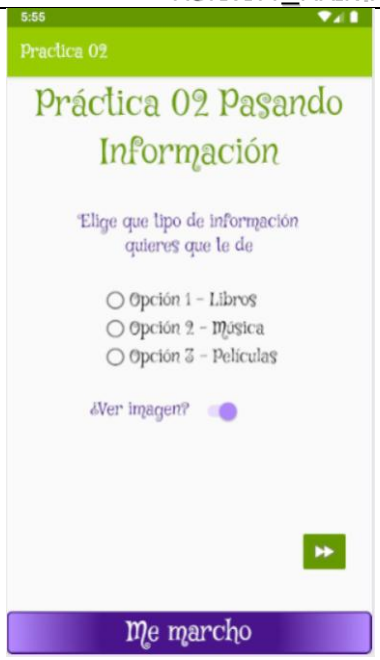
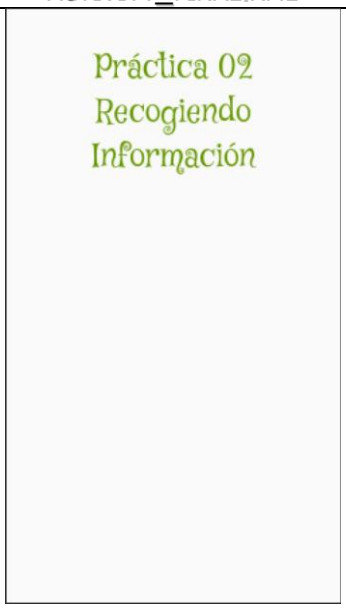
- TEXTVIEW → para el título.
- LISTVIEW→ rectángulo gris, mostraremos un contenido de lista según la elección del usuario.
- IMAGEVIEW → mostraremos una imagen o no según nos diga el usuario

Los identificadores de estos elementos son:



FASE 2

Vamos a maquetar el contenido de los dos layouts con los mismos estilos para que el diseño sea coherente. Quedarán más o menos:

ACTIVITY_MAIN.XML	ACTIVITY_FINAL.XML
 al pulsar se ve En grises 	 La lista se ve en ejecución y la imagen: visibility = gone

Pasando Información entre Activities

FASE 3

Vamos a ver en primer lugar que variables de clase vamos a necesitar:

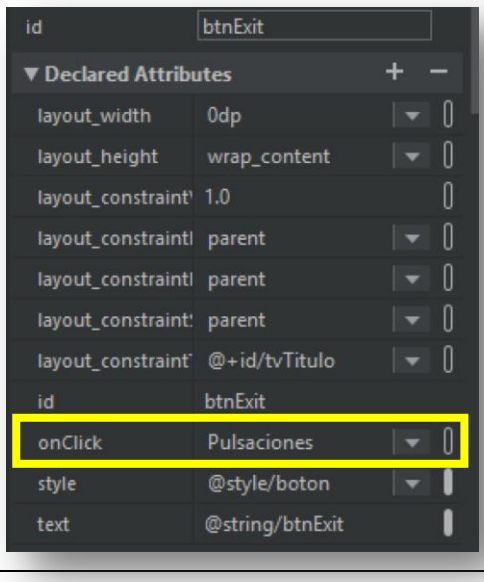
```
public class MainActivity extends AppCompatActivity
        implements RadioGroup.OnCheckedChangeListener{

    private int opcion = 0;
    private boolean imagen = false;
    private RadioGroup rg;
    private Switch sw;
```

Vemos que usamos un implements para establecer el listener que nos va a permitir controlar el RadioGroup, en este caso el método que necesitamos sobreescribir es:

```
@Override
public void onCheckedChanged(RadioGroup radioGroup, int opc) {
      switch(radioGroup.getCheckedRadioButtonId()){
          case R.id.rbOp01: opcion = 1; break;
          case R.id.rbOp02: opcion = 2; break;
          case R.id.rbOp03: opcion = 3; break;
          default: opcion = 0;
      }
}
```

Pero no hacemos ninguna importación más y tenemos que controlar el onClick de dos botones y de un switch, vamos a crearnos un método que controle estos eventos pero el listener se lo dejamos a Android.



En todos los elementos del tipo vista (View) tenemos un propiedad onClick, donde podemos indicarle que método de nuestro código Java (PULSACIONES())se va a ocupar de él cuando el usuario haga click . Vemos como ejemplo el botón btnExit:

Ahora tenemos que ver cómo hay que declarar este método y que parámetros necesita para que funcione correctamente:

```
public void Pulsaciones(View cosa)
{
    switch (cosa.getId())
    {
        case R.id.swImagen:
            break;
        case R.id.ibNext:
            break;
        case R.id.btnExit:
            break;
        default: break;
    }
}// fin Pulsaciones
```

Bueno pues si lo comparamos con el método onClick que reescribíamos en la práctica anterior no hay mucha diferencia no??

Por supuesto tendremos que haber redireccionado el onClick de esos 3 elementos a este método tal y como hemos visto que se hace para btnExit.

Ahora solo queda completar el código, para que haga lo siguiente:

- El Swich lo vamos a tratar como los CheckBox de la práctica anterior y según que el usuario lo active o no la variable booleana estará a true o false.
- El Boton de Exit de momento lo único que hace es que termine la activity.
- El Botón de Next tiene que recoger los valores de las variables opcion e imagen y mandarlas al lanzar la siguiente activity. Veamos este código que es nuevo:

```
case R.id.ibNext:
    Intent next = new Intent( packageContext: this, FinalActivity.class);
    next.putExtra( name: "opcion", opcion);
    next.putExtra( name: "imagen", imagen);
    startActivity(next);
    break;
```

Facil verdad??

Pasando Información entre Activities

Ahora tenemos que ver como recoger en FinalActivity esa información que nos llega desde MainActivity, recordemos que la manda mediante un Intent.

Bueno pues vamos a hacerlo de la forma más fácil, en primer lugar veamos que variables nos definimos en esta Activity:

```
public class FinalActivity extends AppCompatActivity {

    private Bundle datos;
    private String[] lista;
    private ListView lv;
    private ImageView img;
```

Vemos que tenemos un tipo nuevo de datos BUNDLE, nos permitirá recoger en esa variable datos toda la información que nos llega sin establecer previamente cuantos datos, ni de qué tipo son.

Después contamos con un Array de Strings donde recogeremos los valores que queremos mostrar en la lista.

Dos variables para conectar con el ListView y el ImageView que tenemos en el layout.

Vamos a ver qué hacemos con todo esto no??

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

    datos = getIntent().getExtras();

    lv = findViewById(R.id.lvEleccion);
    img = findViewById(R.id.ivImagen);
    mostrarLista(datos.getInt( key: "opcion",  defaultValue: 0));
    mostrarImagen(datos.getBoolean( key: "imagen"), datos.getInt( key: "opcion",  defaultValue: 0));
```

```
datos = getIntent().getExtras();
```

Así es como recogemos la información que nos llega y la guardamos.

Tenemos dos métodos `mostrarLista` y `mostrarImagen`, a los que pasamos como parámetros la información que nos ha llegado y se encargarán de seleccionar los datos que nos ha pedido el usuario y mostrarlos.

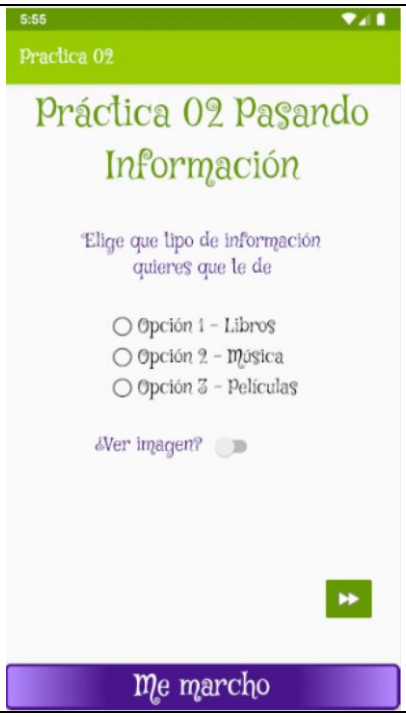
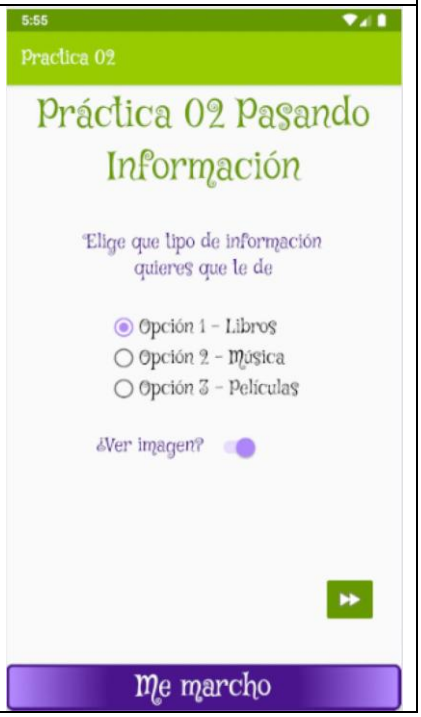
Veamos estos dos métodos:

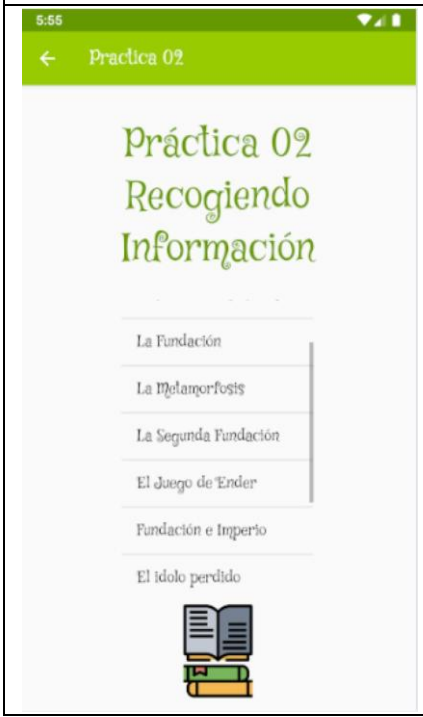
```
private void mostrarLista(int pos) {
    switch (pos) {
        case 1: lista = getResources().getStringArray(R.array.libros);  break;
        case 2: lista = getResources().getStringArray(R.array.musica);  break;
        case 3: lista = getResources().getStringArray(R.array.peliculas);  break;
        default: lista = new String[4];
                 lista[0] = "na de na";
                 lista[1] = "nothing of nothing";
                 lista[2] = "rien de rien";
                 lista[3] = "niente di nulla"; break;
    }
    ArrayAdapter<String> cosas = new ArrayAdapter<>( context: this,
            android.R.layout.simple_list_item_1, lista);
    lv.setAdapter(cosas);
}// fin mostrarLista
```

```
private void mostrarImagen(boolean hay, int opcion)
{
    if(hay)
    {
        switch(opcion) {
            case 1: img.setImageResource(R.drawable.ic_libros); break;
            case 2: img.setImageResource(R.drawable.ic_musica); break;
            case 3: img.setImageResource(R.drawable.ic_pelis);  break;
            default:  img.setImageResource(R.drawable.ic_nada); break;
        }
        img.setVisibility(View.VISIBLE);
    }
    else img.setVisibility(View.GONE);
}// fin mostrarImagen
```

Pasando Información entre Activities

Ahora podemos probar la ejecución:

Pantalla Inicial	Llegamos a Final sin datos	Volvemos a Inicio y elegimos
		

Final Libros+Imagen	Final Música+Imagen	Final Peliculas+Imagen
		

FASE 4

Vamos a pulir unos detallitos que nos faltan.

1º - ¿Cómo conseguimos esa flechita que nos permite volver de Final a Main?

2º - ¿Cómo hacer que aparezca durante unos milisegundos este mensaje de despedida cuando pulsamos el botón de salir?

Pistas -- Handler

Post Delayed