

Práctica 05

Añadir Elementos a una Lista



En esta práctica vamos a contar con una única Activity de tipo Basic, así pues tendremos los 2 layouts iniciales:

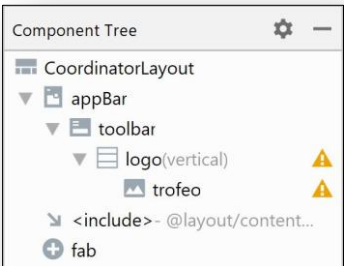



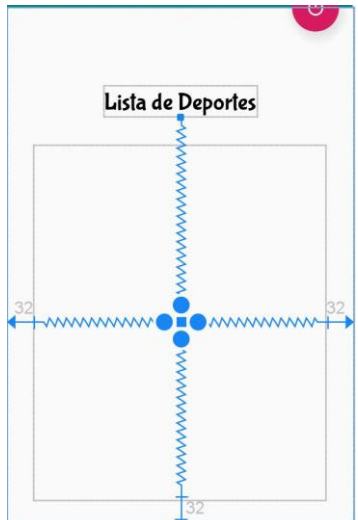
- activity_main.xml → coordinador de layouts donde tenemos:
 - o Toolbar → donde aparecerá nuestro menú
 - o Floating Action Button → que vamos a cambiar de posición para que no moleste a la hora de ver las listas.
- content_main.xml → donde tendremos:
 - o TextView para el título
 - o ListView → Visible
 - o GridView → Oculto (gone)

La idea es que cuando entremos a la app nos enseñe una lista inicial con los deportes que practicamos, indicándonos el nombre, el nivel que creemos que tenemos y la imagen de la estación del año en la que más lo practicamos.

Mediante los botones del menú el usuario elije si quiere verlo como lista, como rejilla y si quiere añadir un deporte nuevo a la lista. Por último mediante el botón flotante el usuario puede salir de la app.

FASE 1

Vamos a ver los layouts que tenemos

MAINACTIVITY			
	ACTIVITY_MAIN		CONTENT_MAIN
	Como se ve la parte de activity_main	<p>Así se ve en diseño el content_main Se aprecian los 3 elementos que lo componen</p> <p>Vemos los constraints del Grid que está oculto</p>	
	Así se verá en ejecución		
			



FASE 2

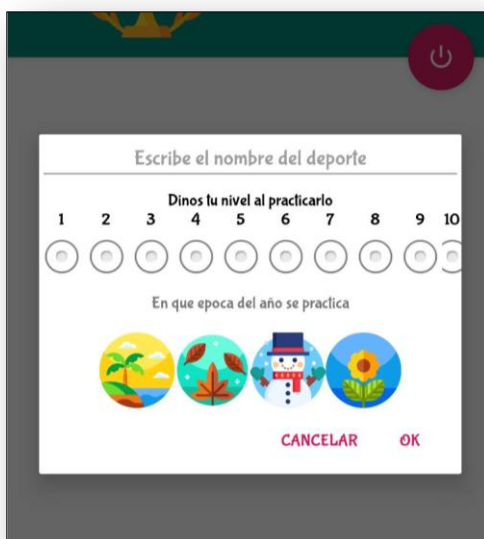
Ahora vamos a ver los pantallazos en ejecución:



LISTA



REJILLA



DIALOGO



DESPEDIDA



FASE 3

Vamos a centrarnos en el código.

1º.- Debemos elaborar el ArrayList con los DEPORTES que integran nuestra lista

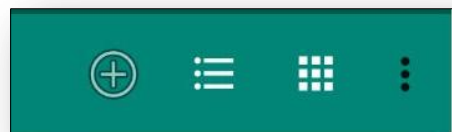
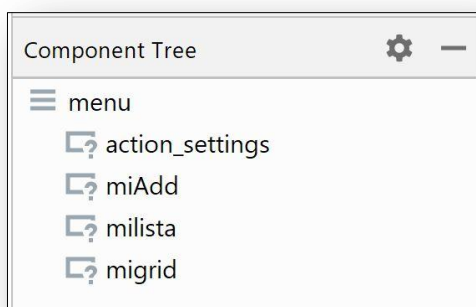
- Se trata de una lista de DEPORTES, por tanto tenemos que definir en primer lugar como va a ser el objeto Deporte:

DEPORTE.JAVA

Creamos un objeto con los siguientes campos y constructores:

```
public class Deporte {  
    int codigo;  
    String estacion;  
    String nombre;  
    int nivel;  
  
    public Deporte() {  
        // Constructor vacío  
    }  
  
    public Deporte(int codigo, String estacion, String nombre, int nivel) {  
        this.codigo = codigo;  
        this.estacion = estacion;  
        this.nombre = nombre;  
        this.nivel = nivel;  
    }  
}  
//fin Deporte
```

- Después generaremos mediante un método la lista inicial con la información que tenemos en nuestros **String-array** .
- También tenemos que dar las opciones que tendremos en el **MENÚ** de nuestra app (menu_main.xml) que aparecerá en la ToolBar .

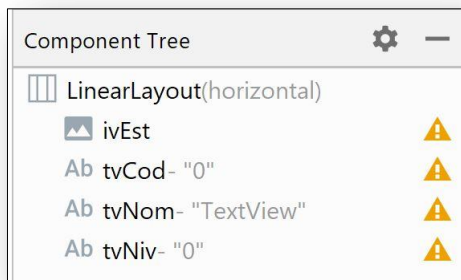




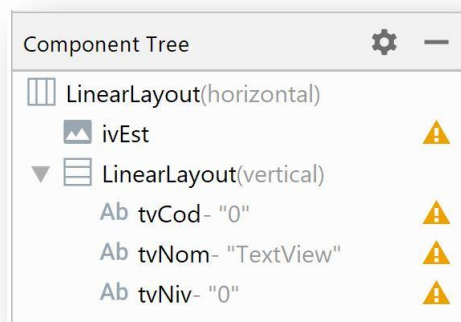
2º.- Necesitamos Crear nuestro Adapter para mostrar la lista en cualquiera de los 2 formatos que el usuario quiera seleccionar.

Por tanto, también necesitamos crear los minilayouts de fila y de celda:

FILA.XML -(MINILAYOUT DE LA FILA)



CELDA.XML -(MINILAYOUT DE LA CELDA)



El código del adapter y la forma de crearlo es totalmente equivalente al de la práctica 04 cambiando eso sí los nuevos elementos que hemos definido aquí, lo ideal sería que se llamara **DeporteAdapter.java** .



3º.- Necesitamos Crear nuestro cuadro de diálogo personalizado, así que lo primero es saber que información queremos que nos dé el usuario y después organizar nuestro minilayout para el cuadro de diálogo.

Queremos la siguiente info del usuario:

- Nombre del deporte
- Valoración del nivel que tiene el usuario (1 a 10)
- Seleccionar la estación del año en que se practica

Recordamos que estamos trabajando con una app móvil, por tanto debemos facilitar al máximo la introducción de datos, el nombre del deporte nos lo tiene que escribir el usuario, pero el nivel y la estación se los pediremos mediante radiogroups con radiobuttons personalizados.

DIALOGO_NUEVO.XML -(MINILAYOUT DEL CUADRO DE DIÁLOGO)



Escribe el nombre del deporte									
Dinos tu nivel al practicarlo									
1	2	3	4	5	6	7	8	9	10
<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
En que epoca del año se practica									

Aquí podemos ver la estructura del layout con sus componetes e identificadores y a la derecha vemos como queda el diseño.



El cuadro de diálogo debe tener los mismos componentes que en la práctica 03, pero con la variante de que le pasamos un layout personalizado.

Veamos como lo declaramos:

```
public class DeporteNuevo extends DialogFragment
    implements
        RadioGroup.OnCheckedChangeListener
{
    private Context entorno;
    private int nivel;
    private String estacion;
    private View vista;
    private EditText etDeporte;
    private RadioGroup rgNivel, rgEstacion;
    private RadioButton rbn, rbe;
    Deporte uno;
    CrearNuevo unomas;

    public DeporteNuevo(Context contexto) { this.entorno = contexto; }
```

Debemos implementar el método del Listener de los radiogroups:

```
@Override
public void onCheckedChanged(RadioGroup radioGroup, int i) {
    switch (radioGroup.getId())
    {
        case R.id.rgNivel: rbn = vista.findViewById(i);
                        nivel = Integer.parseInt(rbn.getText().toString());
                        break;
        case R.id.rgEstacion: rbe = vista.findViewById(i);
                        estacion = rbe.getContentDescription().toString();
                        break;
        default: nivel = 0;
                estacion = "";
                break;
    }
}
```



Ahora nos centramos en el método que genera la vista:

```
public Dialog onCreateDialog(Bundle savedInstanceState){
    AlertDialog.Builder constructor = new AlertDialog.Builder(entorno);
    LayoutInflater bombin = requireActivity().getLayoutInflater();
    vista = bombin.inflate(R.layout.dialogo_nuevo, root: null);
    etDeporte = vista.findViewById(R.id.etDeporte);
    rgNivel = vista.findViewById(R.id.rgNivel);
    rgNivel.setOnCheckedChangeListener(this);
    rgEstacion = vista.findViewById(R.id.rgEstacion);
    rgEstacion.setOnCheckedChangeListener(this);
    constructor.setView(vista)
        .setPositiveButton(R.string.dialog_ok,
            new DialogInterface.OnClickListener()
            {
                @Override
                public void onClick(DialogInterface dialog, int id)
                {
                    uno = new Deporte();
                    uno.codigo = 0;
                    uno.estacion = estacion;
                    uno.nombre = etDeporte.getText().toString();
                    uno.nivel = nivel;
                    unomas.alCrear(uno);
                    dialog.dismiss();
                }
            }
        )
        .setNegativeButton(R.string.dialog_can, (dialog, id) → {
            unomas.alCrear( otro: null);
            dialog.dismiss();
        });
    return constructor.create();
}
```

Recordamos que debemos definir los métodos de onAttach y la interfaz igual que hicimos en la práctica 03.



MAINACTIVITY

Ya hemos preparado todas las cosas que necesitamos para nuestra app así que vamos a organizar el código de la Activity que controla todo:

1. Necesitamos definir e inicializar los elementos que interactúan con el layout, esto lo hacemos como en las prácticas anteriores.
2. Debemos poder llamar al cuadro de diálogo cuando el usuario pulsa el botón del menú así que vamos a ver como controlamos el menú de opciones:

```
@Override
public boolean onCreateOptionsMenu(Menu menu) {
    // Inflate the menu; this adds items to the action bar if it is present.
    getMenuInflater().inflate(R.menu.menu_main, menu);
    return true;
}

@Override
public boolean onOptionsItemSelected(MenuItem item) {
    // Handle action bar item clicks here. The action bar will
    // automatically handle clicks on the Home/Up button, so long
    // as you specify a parent activity in AndroidManifest.xml.
    switch (item.getItemId())
    {
        case R.id.miAdd: addReg(); break;
        case R.id.migrid:
```

Estos dos métodos vienen implementados cuando utilizamos una Activity de tipo Basic, se puede observar, que para crear el menú lo que hacemos es inflarlo, y después tenemos un método que funciona como listener de las opciones del menú, funciona exactamente igual que cuando hacemos el listener de nuestros botones en las prácticas anteriores.

3. Vemos que el método usado para llamar al cuadro de diálogo es addReg():

```
public void addReg()
{
    DeporteNuevo pregunta = new DeporteNuevo( contexto: MainActivity.this);
    pregunta.show(getSupportFragmentManager(), tag: "Nuevo Deporte");
} //fin addReg
```


Práctica 05

Añadir Elementos a una Lista



4. Ahora necesitamos implementar el método de la interfaz, para saber que hacer con el Deporte nuevo que resulta tras la ejecución del cuadro de diálogo

```
public void alCrear(Deporte otro) {  
    if (otro != null) {  
        uno = new Deporte();  
        uno.codigo = 1+lista.size();  
        uno.estacion = otro.estacion;  
        uno.nombre = otro.nombre;  
        uno.nivel = otro.nivel;  
        lista.add(uno);  
        adapter.notifyDataSetChanged();  
    }  
    else  
    {  
        Toast.makeText( context: this, text: "No hay Deporte nuevo", Toast.LENGTH_LONG).show();  
    }  
}  
} //fin alCrear
```

Pues esta ha sido la parte nueva de esta práctica, lo que falta para acabar de implementar la app os lo dejo a vosotros.

A programar



FASE 4

Vamos a ultimar detalles que faltan

- 1º - Conseguir que en el menú aparezca solamente el botón de cambio de vista, es decir, si estamos viendo la lista que en el menú aparezca el botón de cambiar a grid solamente y si estamos viendo la rejilla, que en el menú aparezca solo el de cambiar a lista, no los dos como está en los pantallazos.
- 2º - Conseguir que el fab aparezca entre la ActionBar y el content.
- 3º - Conseguir poner nuestro icono para que aparezca al lado de las opciones de menú
- 4º - Conseguir que los radioButtons tengan solamente imagen o imagen y texto personalizado.