FRAGMENT LIST (RECYVLERVIEW)

Este será el caso de que nuestra aplicación requiera cargar un fragmento en el que se van a cargar una lista de elementos, el ejemplo más cercano que podemos comentar es la lista de contactos de nuestra agenda.

Paso 1:

Creamos un nuevo proyecto con un Empty Activity.

Añadimos un fragmento: new → Fragment→Fragment (List), donde nos van a aparecer muchas opciones:

- Object Kind: nombre de la clase que vamos a usar, en el ejemplo anterior sería Contactos, en el ejemplo de la Moodle, sobre un taller serían Avería. Son por tanto los objetos que contendrá nuestra lista.
- Fragment class name: nombre del fragmento
- Column cont: número de columnas en las que se mostraran los elementos
- Object content layout filename: donde se define la estructura para un elemento
- List layout filename: definición de la lista que contendrá todos los elementos.

Cuando creamos el fragmento analizar principalmente dos aspectos:

- En el onCreateView: Al cargar el layout con el inflater no lo devuelve con un return, si no que guarda dicha vista en un View para poder hacer un par de modificaciones según como se carguen la lista de elementos. Además podemos ver que al crear esa vista, el archivo .xml que carga accediendo a él, vemos que es del tipo RecyclerView, que es un tipo especial de Android para listar elementos. Por último vemos que comprueba la vista y si es del tipo indicado, en función de el número de columnas con que se quiera mostrar usará una lista o un grid, y en dicha lista cargará unos objetos determinados. En principio los llama DummyContent pero esta sería la clase que nosotros tenemos que implementar para nuestra clase concreta que queramos cargar.
- En este también tenemos un método static que nos permitirá que al ser llamado le pasaremos el número de columnas con que queremos mostrar nuestra vista y él se encargará de dicho acción.

```
public static AveriaFragment newInstance(int columnCount) {
    AveriaFragment fragment = new AveriaFragment();
    Bundle args = new Bundle();
    args.putInt(ARG_COLUMN_COUNT, columnCount);
    fragment.setArguments(args);
    return fragment;
}
```

Paso 2:

Volvemos al MainActivity.java, y dentro de este en su activity_main.xml borramos el TextView y en lugar de un ConstraintLayout lo cambiamos por un FrameLayout. No olvidar añadir el id:

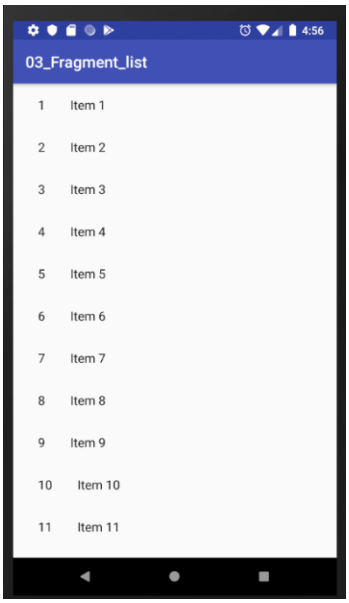
```
android:id="@+id/contenedor"
```

Ahora ya podemos cargar en ese contenedor el fragmento:

```
public class MainActivity extends AppCompatActivity {

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

        getSupportFragmentManager()
                .beginTransaction()
                .add(R.id.contenedor, new AveriaFragment())
                .commit();
    }
}
```

Para ejecutar este código que llevamos implementado, pero no nos de problemas, debemos comentar código que se nos ha generado, pero que nos puede dar problemas, ya que sirve, como veremos en el siguiente ejemplo, para pasar información entre el fragmento y la activity, cosa que ahora no sucede.

```
@Override
public void onAttach(Context context) {
    super.onAttach(context);
    if (context instanceof OnListFragmentInteractionListener) {
        mListener = (OnListFragmentInteractionListener) context;
    } else {
        throw new RuntimeException(context.toString()
                + " must implement OnListFragmentInteractionListener");
    }
}

@Override
public void onDetach() {
    super.onDetach();
    mListener = null;
}
```



Como vemos se ejecuta con una lista con todos los elementos. Ahora mismo con valores por defecto, cuando le carguemos nuestra clase tendrá formato adecuado.

Paso 3: Modificar número de columnas.

En el paso 1 hablamos de un método static que nos serviría para modificar el número de columnas con que se cargase en nuestra aplicación los distintos elementos.

Para hacer uso de esto lo que tenemos que hacer es cambiar en el MainActivity la forma de crear el fragmento que cargar en el contenedor:

```
public class MainActivity extends AppCompatActivity {

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

        getSupportFragmentManager()
                .beginTransaction()
                .add(R.id.contenedor, AveriaFragment.newInstance(2))
                .commit();
    }
}
```

int columnCount

Es decir, en lugar de cargar una nueva instancia de avería, invocamos el método AveriaFragment.newInstance, este método que es estático, recibe un parámetro que representa el número de columnas, como ya vimos anteriormente, con que va a ser representado en pantalla.