LISTVIEW PERSONALIZADO

En este ejemplo vamos a usar un adaptador personalizado, creando nuestro propio diseño, no usando plantillas.

Paso1:

Creamos un proyecto con un Empty Activity.

En nuestro *activity_main.xml*, igual que en el ejemplo anterior, tendrá un elemento listView que ocupa todo el ancho y alto, con un id y que será el que almacene los elementos.

En *MainActivity.java* volvemos a crear un atributo de clase *ListView lista* y en *onCreate* conectamos dicha lista con el ListView del layout mediante el método *findViewById.*

Ahora en nuestra clase crearemos una lista, que será la que almacene los elementos a cargar, pero en este caso no almacenará String, si no objetos de una clase que hemos de implementar

```
List<Averia> averiaList;
```

Esta clase en Android recibe el nombre de clase pollo.

Paso 2:

Implementamos la clase definiendo sus atributos y métodos. Como en otros entornos de desarrollo podemos generar dichos métodos. Será suficiente con un constructor sin parámetros y otro con todos, y los getters y setters correspondientes a los atributos de clase.

```
class Averia {
    private String titulo;
    private String modeloCoche;
    private String urlFoto;
    private int numeroPresupuestos;

    public Averia() {
    }

    public Averia(String titulo, String modeloCoche, String urlFoto, int numeroPresupuestos) {
        this.titulo = titulo;
        this.modeloCoche = modeloCoche;
        this.urlFoto = urlFoto;
        this.numeroPresupuestos = numeroPresupuestos;
    }
```

Ahora si podemos en onCreate implementar un ArrayList donde almaceno los objetos de mi nueva clase.

```
averiaList = new ArrayList<>();
averiaList.add(new Averia( titulo: "Espejo roto", modeloCoche: "Audi - A4", urlFoto: "http://blog.mister-auto.es/wp-content/uploads/2014/09/23116650_m.jpg",
averiaList.add(new Averia( titulo: "Paragolpes delantero", modeloCoche: "Citroen - C4", urlFoto: "", numeroPresupuestos: 0));
averiaList.add(new Averia( titulo: "Embrague", modeloCoche: "Seat - Ibiza", urlFoto: "", numeroPresupuestos: 0));
averiaList.add(new Averia( titulo: "Cambio de aceite", modeloCoche: "Seat - Toledo", urlFoto: "", numeroPresupuestos: 1));
```

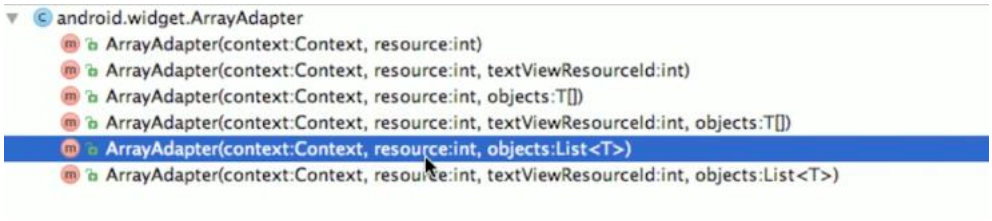
Paso3:

Ya solo queda generar mi adaptador para conectar el listView con mi lista de objetos.

```
MiAdaptadorAverias adaptadorAverias = new MiAdaptadorAverias();
```

Como no existe los creo como hice con la clase (new→java class), y al crearla le diré que extienda de ArrayAdapter<Averias>

Para eliminar el error que aparece, debemos crear un constructor. Si desplegamos las opciones de posibles constructores, podemos elegir varios, pero nosotros seleccionaremos el que se muestra marcado en la siguiente imagen, porque en caso de que solo se quiera implementar uno, este es el más básico.



Como vemos recibe:

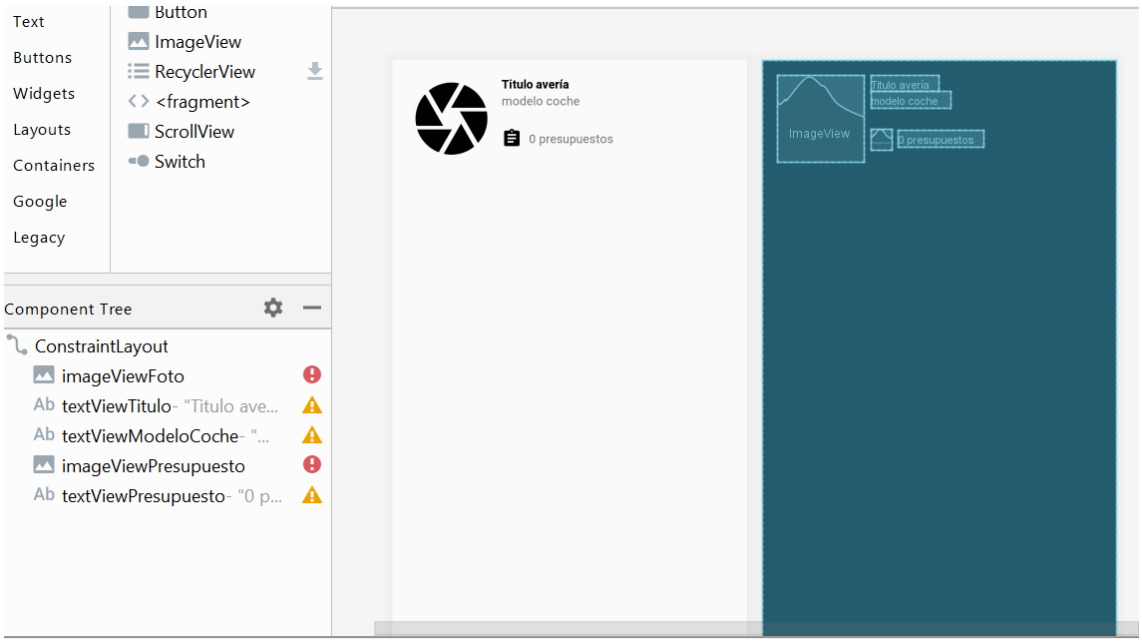
- El contexto
- El layout para dibujar cada elemento de la lista
- La lista de elementos

Ya podemos terminar de implementar mi adaptador en el onCreate (solo faltaba decirle los parámetros que recibía por cabecera). Lo último será vincular la lista con el adaptador, tal como vemos en la última línea de esta imagen anterior.

```
MiAdaptadorAverias adaptadorAverias = new MiAdaptadorAverias(
        this,
        R.layout.averia_item,
        averiaList
);

lista.setAdapter(adaptadorAverias);
```

Pero vemos que al especificar el layout (segundo parámetros), este aún no existe y debemos crearlo también (averia_item.xml), ya que es la plantilla que especifica cómo se mostrarán los elementos, elementos que son los atributos de mi clase, tal como queremos que se muestren los objetos que queremos mostrar.



Lo implementamos con un ConstraintLayout como contenedor y le asigno los elementos según quiera que se muestren en la vista.

Paso4:

Personalizar el adaptador.

Los primeros pasos que debemos hacer será:

- Crear en dicha clase los atributos de clase e inicializar en el constructor sus parámetros a los recibidos por cabecera.

```
class MiAdaptadorAverias extends ArrayAdapter<Averia> {
    Context ctx;
    int layoutTemplate;
    List<Averia> averiaList;


    public MiAdaptadorAverias(@NonNull Context context, @LayoutRes int resource, @NonNull List<Averia> objects) {
        super(context, resource, objects);
        this.ctx = context;
        this.layoutTemplate = resource;
        this.averiaList = objects;
    }
```

- Sobrescribimos el método getView que se lanzará automáticamente como un bucle, una vez por cada elemento de la Lista que se ha pasado como parámetro en el constructor. Cada vez que se invoca este método, recibe una posición, que es la que ocupa el elemento en cuestión en la lista de averías

```
@NonNull
@Override
public View getView(int position, @Nullable View convertView, @NonNull ViewGroup parent) {
    //crea un layout donde cargar cada objeto Averia
    View v = LayoutInflater.from(ctx).inflate(layoutTemplate, parent,  attachToRoot: false);

    // Obtener la información del elemento de la lista que estoy iterando en este momento
    Averia elementoActual = averiaList.get(position);

    // Rescatar los elementos de la IU de la template
    TextView textViewTitulo = (TextView) v.findViewById(R.id.textViewTitulo);
    TextView textViewModelo = (TextView) v.findViewById(R.id.textViewModeloCoche);
    TextView textViewPresupuestos = (TextView) v.findViewById(R.id.textViewPresupuesto);
    ImageView imageViewFoto = (ImageView) v.findViewById(R.id.imageViewFoto);

    // HAcer un set de la info del elemento Actual en los elementos de la IU
    textViewTitulo.setText(elementoActual.getTitulo());
    textViewModelo.setText(elementoActual.getModeloCoche());
    textViewPresupuestos.setText(elementoActual.getNumeroPresupuestos()+" presupuestos");

    if(!elementoActual.getUrlFoto().isEmpty()) {//si el elemento URL no está vacio cargo una foto
        Glide.with(ctx)
                .load(elementoActual.getUrlFoto())
                .into(imageViewFoto);
    }//si está vacio pongo la definida por defecto

    return v;
}
```

En la imagen anterior se pueden ver paso a paso especificados todos los casos a implementar en este método para cargar cada uno de los elementos con los valores correspondientes.

Nota1: recuerda activar los permisos adecuados

```
<uses-permission android:name="android.permission.INTERNET"></uses-permission>
```

Nota2: recuerda que para añadir imágenes con Glide tenemos que añadir algunas líneas de código a gradle (copiamos y pegamos de dicha página en nuestro código, tal como se vio en el apartado de imágenes). Tenéis los enlaces a la librería en el final del tema 2 y como ejemplos podéis encontrarlos en la teoría del tema dos, dentro de la carpeta *Códigos de clase IU*