UI: RecyclerView

Desarrollo para plataformas móviles

Borja Martin Herrera
Borja.herrera@u-tad.com

RecyclerView

- RecyclerView:
  - Al igual que ListView/GridView es una lisa de datos que nos permite navegar por la app.
  - Es más flexible y eficiente que ListView/GridView.

RecyclerView

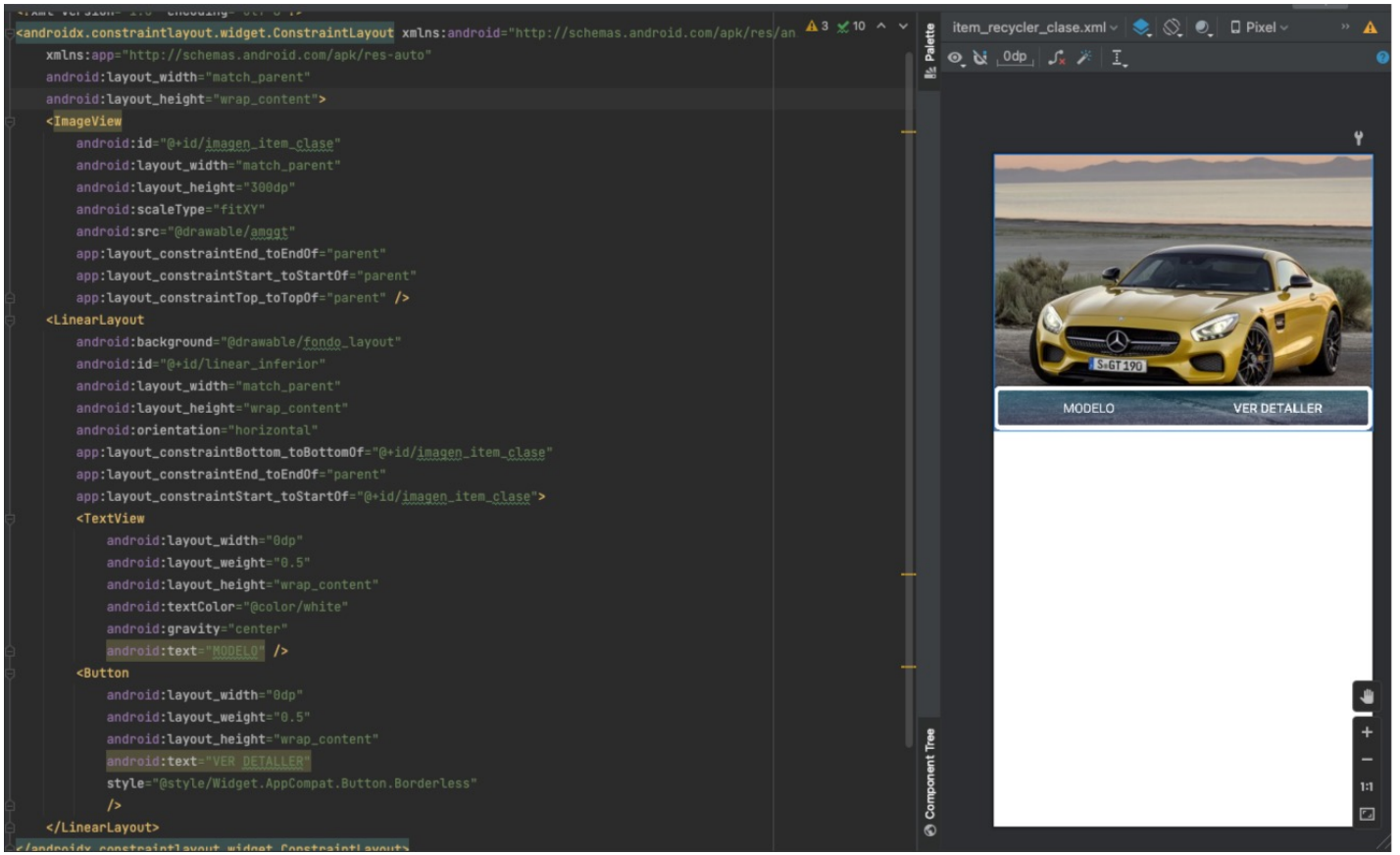
- RecyclerView:
  - Necesitamos varios componentes para hacer funcionar el RecyclerView.
  - Por un lado, tenemos que usar la clase *LayoutManager* para que el RecyclerView pueda autorellenarse (define cómo se tiene que rellenar, lineal, en grid, etc).
  - Por otro lado, necesitamos usar la clase RecyclerView.*ViewHolder* para definir cada uno de los elementos de la lista (cada elemento de la lista es un objeto de esa clase).
  - Estos *ViewHolder* son manejados por un *Adapter* al igual que en los *ListView*. El *Adapter* crea los *ViewHolders* necesarios y los rellena con sus datos correspondientes.

RecyclerView

- RecyclerView:
  - Además, necesitamos definir un *layout* para cada uno de los elementos de la lista.
  - Y necesitaremos un conjunto de datos para rellenarla.
  - Vamos a crear el RecyclerView paso a paso:

RecyclerView

- El layout de cada ítem (*recycler_item_custom.xml*):



(Está en la carpeta app/res/layout)

RecyclerView

- La clase que representa el modelo de datos es la siguiente:

```
data class Coche (var marca: String, var modelo: String, var cv: Int, var precio: Int, var tipo: String, var imagen: Int,)
```

- Al no necesitar funcionalidad adicional se declara como data class
- Especial interés en el atributo imagen, que es de tipo int ya que Android trata los recursos como imágenes

RecyclerView

- Una vez hecho esto, podemos añadir el elemento RecyclerView al *Layout* de la actividad.

```xml
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:orientation="vertical"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    <androidx.recyclerview.widget.RecyclerView
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:id="@+id/recycler"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

</LinearLayout>
```

RecyclerView

- Construimos el adapter. Heredamos de la clase RecyclerView.Adapter para ello:

```
class AdapterCoche(var context: Context, var lista: ArrayList<Coche>) : RecyclerView.Adapter<AdapterCoche.MyHolder>() {
```

- Creamos una clase dentro del adapter que hereda de RecyclerView.ViewHolder para proveer a cada ítem de una referencia a su Layout

```
inner class MyHolder(itemView: View) : RecyclerView.ViewHolder(itemView) {

    var imagen: ImageView
    var modelo: TextView
    var boton: Button

    init {
        imagen = itemView.findViewById(R.id.imagen_item)
        modelo = itemView.findViewById(R.id.modelo_item)
        boton = itemView.findViewById(R.id.boton_item)
    }
}
```

RecyclerView

- Sobrescribimos el método *onCreateViewHolder* para inflar el *layout* que tenemos de cada *item*:

```
override fun onCreateViewHolder(parent: ViewGroup, viewType: Int): MyHolder {
    val view: View = LayoutInflater.from(context).inflate(R.layout.item_recycler, parent,  attachToRoot: false)
    return MyHolder(view)
}
```

RecyclerView

- Sobrescribimos el método *onBindViewHolder* para rellenar los elementos de la vista:

```
override fun onBindViewHolder(holder: MyHolder, position: Int) {
    val coche = lista.get(position)
    holder.modelo.text = coche.modelo
    holder.imagen.setImageResource(coche.imagen)
}
```

- Por último sobrescribimos el método que devuelve el último de los elementos:

```
override fun getItemCount(): Int {
    return lista.size
}
```

RecyclerView

- Tenemos que añadir código a la actividad para que sea capaz de manejar el nuevo tipo de lista (Lo hacemos en *onCreate*), añadiendo los datos iniciales que se quieren mostar:

```
coches = ArrayList();

coches.add(Coche( marca: "Mercedes", modelo: "AMG GT", cv: 500, precio: 200000, tipo: "Deportivo", R.drawable.amggt))
coches.add(
    Coche(
        marca: "Bentley",
        modelo: "Continental",
        cv: 400,
        precio: 300000,
        tipo: "Berlina deportivo",
        R.drawable.continental
    )
)
coches.add(Coche( marca: "Jaguar", modelo: "FType", cv: 300, precio: 150000, tipo: "Deportivo", R.drawable.ftype))
coches.add(Coche( marca: "Ford", modelo: "GT40", cv: 500, precio: 300000, tipo: "Deportivo clasico", R.drawable.gt40))
coches.add(Coche( marca: "Nissan", modelo: "GTR", cv: 300, precio: 200000, tipo: "Deportivo", R.drawable.gtr))
coches.add(Coche( marca: "Porche", modelo: "Huayara", cv: 600, precio: 400000, tipo: "Deportivo", R.drawable.huayra))
coches.add(Coche( marca: "Lexus", modelo: "LC", cv: 200, precio: 100000, tipo: "Deportivo", R.drawable.lc))
coches.add(Coche( marca: "Ferrari", modelo: "Le ferrari", cv: 600, precio: 600000, tipo: "Deportivo", R.drawable.leferrari))
coches.add(Coche( marca: "McLaren", modelo: "MC600", cv: 500, precio: 450000, tipo: "Deportivo", R.drawable.mc600))
coches.add(Coche( marca: "Toyota", modelo: "Supra", cv: 300, precio: 150000, tipo: "Deportivo", R.drawable.supra))
coches.add(Coche( marca: "Porche", modelo: "Taycan", cv: 350, precio: 250000, tipo: "Deportivo", R.drawable.taycan))
var adapterCoche = AdapterCoche( context: this, coches)
binding.recycler.adapter = adapterCoche;
binding.recycler.layoutManager =
    LinearLayoutManager( context: this, LinearLayoutManager.VERTICAL, reverseLayout: false)
```

RecyclerView

- En el caso de querer detectar la pulsación del recycler es necesario hacerlo en la clase adaptador

```
override fun onBindViewHolder(holder: MyHolder, position: Int) {
    val coche = lista.get(position)
    holder.modelo.text = coche.modelo
    holder.imagen.setImageResource(coche.imagen)
    holder.boton.setOnClickListener {  it: View!

    }
}
```

RecyclerView

- Si se quiere hacer una comunicación con la activity, los pasos son los siguientes

1. Definición en el origen de una interfaz con el dato que se quiera comunicar, en este caso en el adaptador

```
// 1. Origen de los datos creo una interfaz

interface OnRecyclerUsuarioListener{
    fun comunicaUsuarioSelected(usuario: Usuario)
    fun comunicaUsuarioSelected(usuario: Usuario, posicion: Int)
}
```

RecyclerView

- Si se quiere hacer una comunicación con la activity, los pasos son los siguientes

2. Se crea un objeto del tipo de la interfaz, se instancia y se utiliza cuando se quiera comunicar

```
// 2. Creo un objeto de la interfaz para poder utilizarlo
private lateinit var listener: OnRecyclerUsuarioListener

init {
    // OnRecyclerUsuarioListener =  OnRecyclerUsuarioListener
    listener = context as OnRecyclerUsuarioListener;
}
```

```
holder.nombre.setOnClickListener { it: View!
    // 2. Utilizo el método de la interfaz
    listener.comunicaUsuarioSelected(usuarioFila,position)
}
```

RecyclerView

- Si se quiere hacer una comunicación con la activity, los pasos son los siguientes

3. Se crea implementa la interfaz en el destino, obligando a sobreescribir los métodos y por lo tanto dando por concluida la comunicación

```
// 3. En el destino de los datos implementar la interfaz
class MainActivity : AppCompatActivity(), AdaptadorRecycler.OnRecyclerUsuarioListener {
```

```
override fun comunicaUsuarioSelected(usuario: Usuario) {
    // Utilizar los datos
    Snackbar.make(binding.listaRecycler,
        text: "Comunicado ${usuario.nombre}", Snackbar.LENGTH_SHORT).show()
}

override fun comunicaUsuarioSelected(usuario: Usuario, posicion: Int) {
    // Utilizar los datos
    Snackbar.make(binding.listaRecycler,
        text: "Comunicado ${usuario.nombre} en posicion ${posicion}",
        Snackbar.LENGTH_SHORT).show()
}
```

Ejercicio

Ejercicio: el *RecyclerView* hecho donde al pulsar una fila se pase el coche seleccionado a otra pantalla y se vean sus detalles