RecyclerView (Tema8 p4)

1. Crear una clase pojo con constructor *Ej. Usuario.kt*

```
class Usuario(nombre:String apellidos:String) {
    var nombre: String
    var apellidos: String
    init {
        this.nombre = nombre
        this.apellidos = apellidos
    }
}
```

2. Añadir el RecyclerView al layout que se va a mostrar *Ej. main_activity.xml*

```
<androidx.constraintlayout.widget.ConstraintLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">
<androidx.recyclerview.widget.RecyclerView
    android:id="@+id/recyclerList"
    android:background="@color/azul"
    android:layout_width="match_parent"
    android:layout_height="match_parent"/>
</androidx.constraintlayout.widget.ConstraintLayout>
```

3. Crear un layout generico para las vistas de los elementos del recycler *Ej. recyclerlayout.xml*

```xml
<?xml version="1.0" encoding="utf-8"?>
<androidx.cardview.widget.CardView
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:card_view="http://schemas.android.com/apk/res-auto"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    card_view:cardCornerRadius="4dp"
    card_view:cardUseCompatPadding="true"
    card_view:cardElevation="2dp">
    <LinearLayout
        android:padding="8dp"
        android:background="#493DEC"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:gravity="center"
        android:orientation="horizontal">
        <LinearLayout
            android:layout_width="0dp"
            android:layout_height="wrap_content"
            android:layout_weight="0.75"
            android:orientation="vertical">
            <TextView
                android:id="@+id/textView"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:text="Large Text"
                android:textColor="@android:color/white"
                android:textSize="20sp" />
            <TextView
                android:id="@+id/textView2"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:text="Medium Text"
                android:textColor="@android:color/white"
                android:textSize="15sp" />
        </LinearLayout>
    </LinearLayout>
</androidx.cardview.widget.CardView>
```

4. Crear una clase Holder.kt que reciba una vista y herede de RecylclerView.ViewHolder * Ej. Holder.kt*

```
0
        val textNombre: TextView
        val textApellido: TextView

        fun bind(entity: Usuario) {
6
7
        }
        init {
10
11
        }
    }
```

- **Linea 0** clase que extiende de **RecylclerView.ViewHolder** con los atributos que necesitemos
- **Líneas 10 y 11** hinchamos los atributos con las vistas
- **Líneas 6 y 7** asignamos el valor de la clase pojo a las propiedades de esas vistas.

5. Creamos una clase que herede de RecyclerView.Adapter nos obliga a sobreescribir 3 metodos *Ej. Adaptor*

```
0
                                    RecyclerView.Adapter<Holder>()
    {
        override fun onCreateViewHolder(viewGroup: ViewGroup, i: Int):Holder
        {
6
                .inflate(R.layout.recyclerlayout, viewGroup, false)
8
        }
        override fun onBindViewHolder(holder: Holder, position: Int) {
            val item: Usuario = datos[position]
            holder.bind(item)
        }
        override fun getItemCount(): Int {
            return datos.size
        }
    }
```

- **Linea 0:** preguntar por Internal constructor
- **onCreateViewHolder:** **Línea 6** inflamos la vista del recyclerlayout.xml **Línea 8** llamamos al constructor de Holder.kt pasandole la vista y lo devolvemos.

- **onBindViewHolder:** recuperar el objeto correspondiente a la posición recibida como parámetro y **llamar al método bind desde el ViewHolder recibido como parametro**.
- **getItemCount():** devuelve el tamaño del ArrayList *datos*.

6. Asignar el adaptador al ReciclerView en nuestra MainActivity

```
class MainActivity : AppCompatActivity() {
    override fun onCreate(savedInstanceState: Bundle?)
    {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)
6
7
8
9
10
            LinearLayoutManager(this, LinearLayoutManager.VERTICAL,
                                false)
    }
14
    {
        var datos = ArrayList<Usuario>()
        for (i in 0..19)
            datos.add(Usuario("nombre$i", "apellido1$i Apellido2$i"))
        return datos
    }
}
```

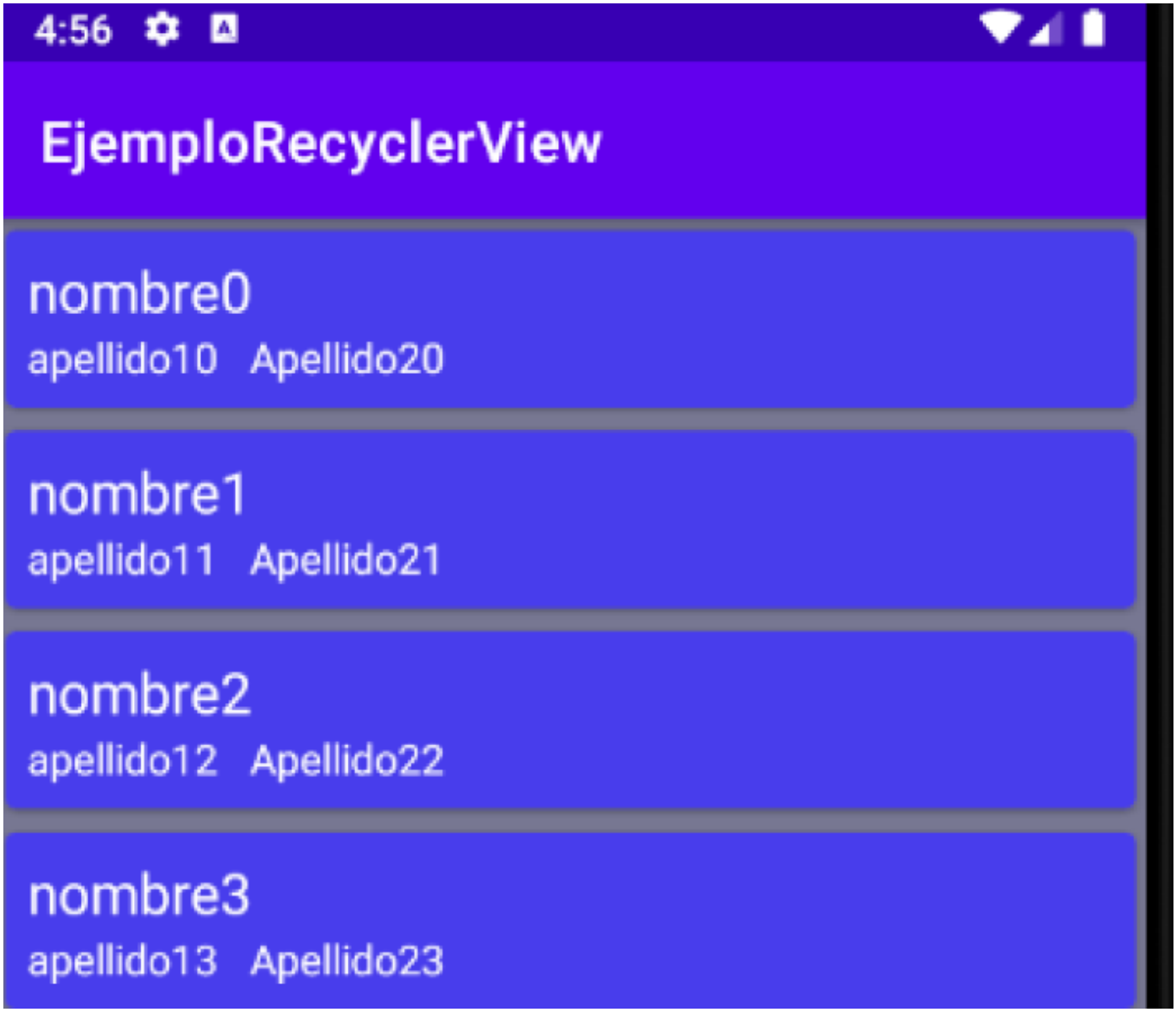
- **Línea 14 / 6:** función para crear el ArrayList de objetos / variable a la que se asigna
- **Línea 7:** hinchamos la vista del RecyclerView
- **Línea 8:** llamamos al constructor de Adaptador.kt y le pasamos el array de usuarios
- **Línea 9:** asignamos el adaptador al RecyclerView
- **Línea 10:** asignamos el LayoutManager llamando al constructor de LinearLayoutManager, le indicamos que la orientacion y desplazamiento sea VERTICAL. **Si no usasemos un layoutManager predefinido nos tocaria implementarlo**.

Otras propiedades (Tema8 p11)

tipos de LayoutManager

- **LinearLayoutManager:** para la visualización como lista vertical u horizontal
- **GridLayoutManager:** para la visualización como tabla tradicional ()
- **StaggeredGridLayoutManager:** que visualiza los elementos como una tabla apilada o de celdas no alineadas.

  A los 2 ultimos hay que pasarles el numero de columnas a mostrar




```
recyclerView.setLayoutManager(new GridLayoutManager(this, 2));
```

```
recyclerView.setLayoutManager(new StaggeredGridLayoutManager(2,1 ))
```

ItemDecoration e ItemAnimation

- **ItemDecoration:** Se usa para personalizar el aspecto con divisores o separadores por ejemplo.
- **ItemAnimation:** define animaciones al realizar acciones comunes sobre elementos(añadir, eliminar, mover, modificar) se implementa por defecto con **DefaultItemAnimator**.

Mas RecyclerView

7. Click sobre un elemento de la lista

RecyclerView no tiene un evento onItemClick() hay que crearlo en el ViewHolder

```
    class Adaptador internal constructor(val datos: ArrayList<Usuario>) :
2
    {
4
      override fun onCreateViewHolder(viewGroup: ViewGroup, i: Int):Holder{
            val itemView: View = LayoutInflater.from(viewGroup.context)
                    .inflate(R.layout.recyclerlayout, viewGroup, false)
8
            return Holder(itemView)
    }
    override fun onBindViewHolder(holder: Holder, position: Int) {
        val item: Usuario = datos[position]
        holder.bind(item)
    }
    override fun getItemCount(): Int {
        return datos.size
    }
18
        this.listenerClick=listener
    }
21
        listenerClick?.onClick(p0)
    }
    }
```

- **Línea 2:** hacemos que el adaptador herede View.OnClickListener.
- **Línea 4:** declaramos una lateinit var de tipo View.OnClickListener.
- **Línea 8:** ponemos un escuchador sobre el itemView para que se detecte la pulsación.
- **Línea 21:** anular el metodo onclick asignandole la propiedad de este tipo que hemos declara en la línea 4
- **Línea 18:** Para que esta propiedad no sea nula, tendremos que crear un método al que le llegue una variable de este tipo y le sea asignada.

8. Llamar al metodo desde donde queramos utilizarlo Ej. MainActivity

```
adaptador.onClick(View.OnClickListener { v ->
        Toast.makeText(
        this@MainActivity,
        "Has pulsado" + recyclerView.getChildAdapterPosition(v),
        Toast.LENGTH_SHORT
    ).show()
})
```

> 📌 Con el objeto adaptador asignado al recycler podemos llamar a la función onClick y pasar un anónimo de tipo OnClickListener, que será invocado al pulsar sobre un elemento de la lista. Con la vista que entra podemos saber que posición a sido pulsada a través del método getChildAdapterPosition() de la clase RecyclerView.