cpifp Los Enlaces

PROGRAMACIÓN MULTIMEDIA Y DISPOSITIVOS MÓVILES

2º DESARROLLO DE APLICACIONES MULTIPLATAFORMA

TEMA 5. RESUMEN RECYCLERVIEW

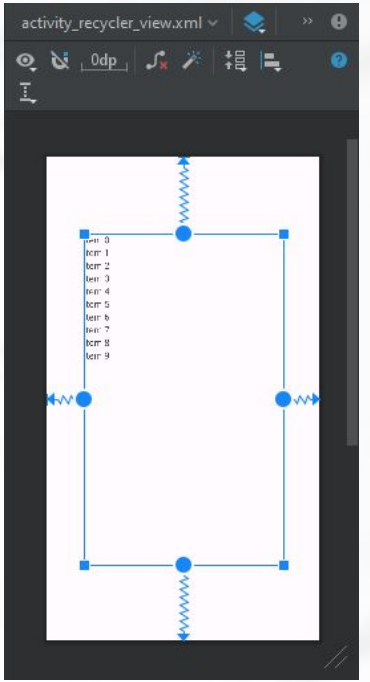
cpi fp Los Enlaces

T5. RESUMEN RECYCLERVIEW

1. LAYOUT PRINCIPAL

Dentro del layout de la actividad en la que queramos implementar el RecyclerView insertamos la etiqueta correspondiente definiendo su altura, anchura e id.

```
<androidx.recyclerview.widget.RecyclerView
    android:id="@+id/rvEjemplo"
    android:layout_width="300dp"
    android:layout_height="500dp"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintTop_toTopOf="parent"
    app:layout_constraintBottom_toBottomOf="parent"/>
```



cpifp Los Enlaces

T5. RESUMEN RECYCLERVIEW

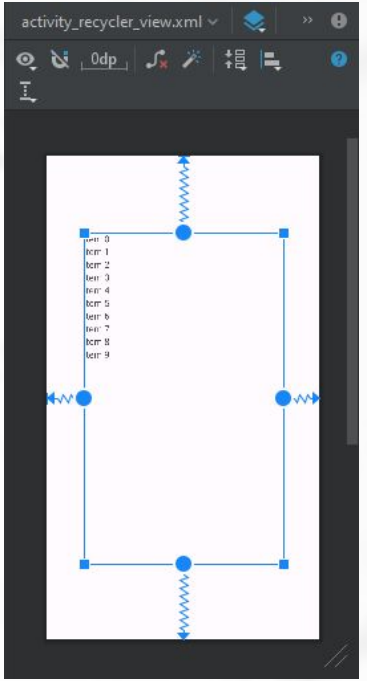
1. LAYOUT PRINCIPAL

En la lógica de la actividad asociada a este layout declaramos las variables necesarias para interactuar con el RecyclerView: objetos RecyclerView y Adapter.

```
class RecyclerViewActivity : AppCompatActivity() {

    private lateinit var rvEjemplo: RecyclerView
    private lateinit var rvAdapter: EjemploAdapter

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_recycler_view)
    }
}
```



T5. RESUMEN RECYCLERVIEW

2. LAYOUT ÍTEM

El Adapter necesita recibir una lista de objetos que implementar en el RecyclerView. Por ejemplo, cuadros de texto con un color de letra determinado.

Para ello, creamos un layout independiente desde el directorio *res* → *layout*, clickando con botón derecho y eligiendo *New* → *Layout Resource File*.

Nombramos dicho archivo de manera que se identifique fácilmente a qué RecyclerView va asociado. En este caso, *item_ejemplo.xml*.

cpifp Los Enlaces

T5. RESUMEN RECYCLERVIEW

2. LAYOUT ÍTEM

Definimos el TextView que vamos a usar como ejemplo dejando un margen vertical para que los elementos del RecyclerView se muestren separados.

```
<TextView
    android:id="@+id/tvEjemplo"
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="50dp"
    android:layout_marginVertical="4dp"
    android:gravity="center"
    tools:textColor="#3949AB"
    tools:text="Prueba de ejemplo" />
```

T5. RESUMEN RECYCLERVIEW

3. CLASES

Debemos crear ahora 3 clases: Adapter, ViewHolder y otra para los objetos que conformarán la lista que recibe el Adapter.

Esta última puede ser de cualquier tipo. La más sencilla de implementar es la *data class*.

Con botón derecho sobre el directorio en el que estemos trabajando, en mi caso *com.ruben.aplicacionespmdm* → *resumenRecyclerView*, creamos las distintas clases siguiendo la ruta *New* → *Kotlin Class/File*.

cpifp Los Enlaces

T5. RESUMEN RECYCLERVIEW

3. CLASES: OBJETOS LISTA

Creamos una clase *TextoEjemplo.kt* que define objetos con dos propiedades que se podrán modificar desde el ViewHolder: el texto a mostrar y su color.

```
package com.ruben.aplicacionespmdm.resumenRecyclerView

data class TextoEjemplo (val texto: String, val colorTexto: Int)
```

T5. RESUMEN RECYCLERVIEW

3. CLASES: OBJETOS LISTA

Una vez definida esta clase, podemos crear en la actividad principal la lista que debe recibir el Adapter. Si la lista se va a poder modificar haríamos *mutableListOf( )*.

```
private val listaEjemplo = listOf(
    TextoEjemplo("Primer elemento", -871890688),
    TextoEjemplo("Segundo elemento", -1509921024),
    TextoEjemplo("Tercer elemento", -1131230976),
    TextoEjemplo("Cuarto elemento", -1509883935),
    TextoEjemplo("Quinto elemento", -858717953)
)
```

cpi'fp Los Enlaces

T5. RESUMEN RECYCLERVIEW

3. CLASES: ADAPTER

Creamos entonces la clase Adapter, le pasamos la lista anterior de objetos *TextoEjemplo* y le indicamos cómo se va a llamar el ViewHolder asociado al RecyclerView.

```
class EjemploAdapter (private val ejemplos: List<TextoEjemplo>) :
RecyclerView.Adapter<EjemploViewHolder> () {

}
```

cpifp Los Enlaces

T5. RESUMEN RECYCLERVIEW

3. CLASES: ADAPTER

Al definir una clase tipo Adapter, Android Studio nos facilita la creación de la misma con las acciones contextuales: *Alt+Intro → Implement members*.

```
class EjemploAdapter (private val ejemplos: List<TextoEjemplo>) :
RecyclerView.Adapter<EjemploViewHolder> () {
   override fun onCreateViewHolder(parent: ViewGroup, viewType:
Int): EjemploViewHolder{}

   override fun onBindViewHolder(holder: EjemploViewHolder,
position: Int) {}

   override fun getItemCount(): Int {}
}
```

cpifp Los Enlaces

T5. RESUMEN RECYCLERVIEW

3. CLASES: ADAPTER

Cada uno de estos métodos tiene su función. El primero de ellos toma el layout que hemos creado para los elementos del RecyclerView y se lo pasa al ViewHolder.

```
override fun onCreateViewHolder(parent: ViewGroup, viewType:
Int): EjemploRecyclerView {
    val view =
        LayoutInflater.from(parent.context).inflate(R.layout.item_e
        jemplo, parent, false)
    return EjemploViewHolder(view)
}
```

cpi fp Los Enlaces

T5. RESUMEN RECYCLERVIEW

3. CLASES: ADAPTER

El segundo se encarga de ejecutar las acciones del ViewHolder sobre cada uno de los elementos de la lista. Y el tercero determina el tamaño de la lista.

```
override fun onBindViewHolder(holder: EjemploViewHolder,
position: Int) {
    holder.pintarViews(ejemplos[position])
}

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

cpi fp Los Enlaces

T5. RESUMEN RECYCLERVIEW

3. CLASES: VIEWHOLDER

La clase ViewHolder recibe un View (layout de los ítems) y se encarga de pintarlo en la pantalla. Debemos declarar los elementos de dicho layout (solo 1 en este caso).

```
class EjemploViewHolder (view: View) :
RecyclerView.ViewHolder(view) {
   private val tvEjemplo: TextView =
     view.findViewById(R.id.tvEjemplo)

}
```

cpifp Los Enlaces

T5. RESUMEN RECYCLERVIEW

3. CLASES: VIEWHOLDER

En el método *onBindViewholder* del Adapter hemos definido una función *pintarViews* que debemos construir dentro del ViewHolder.

```
class EjemploViewHolder (view: View) :
RecyclerView.ViewHolder(view) {
   private val tvEjemplo: TextView =
     view.findViewById(R.id.tvEjemplo)

     fun pintarViews(item: TextoEjemplo){
        tvEjemplo.text = item.texto
        tvEjemplo.setTextColor(item.colorTexto)
     }
}
```

cpifp Los Enlaces

T5. RESUMEN RECYCLERVIEW

3. CLASES: VIEWHOLDER

Esta función pintará cada uno de los elementos de la lista según los atributos de cada objeto que hayamos definido en dicha lista.

```
class EjemploViewHolder (view: View) :
RecyclerView.ViewHolder(view) {
   private val tvEjemplo: TextView =
     view.findViewById(R.id.tvEjemplo)

     fun pintarViews(item: TextoEjemplo){
        tvEjemplo.text = item.texto
        tvEjemplo.setTextColor(item.colorTexto)
     }
}
```

cpifp Los Enlaces

T5. RESUMEN RECYCLERVIEW

4. LÓGICA

Solo queda inicializar las variables de la actividad principal y asignarle un *layoutManager* al RecyclerView con el scroll que queramos (horizontal o vertical).

```
override fun onCreate(savedInstanceState: Bundle?) {
    super.onCreate(savedInstanceState)
    setContentView(R.layout.activity_recycler_view)
    rvEjemplo = findViewById(R.id.rvEjemplo)
    rvAdapter = EjemploAdapter(listaEjemplo)

    rvEjemplo.layoutManager = LinearLayoutManager(this,
LinearLayoutManager.VERTICAL, false)
    rvEjemplo.adapter = rvAdapter
}
```

cpifp Los Enlaces

T5. RESUMEN RECYCLERVIEW

4. ESQUEMA