



# UT-2

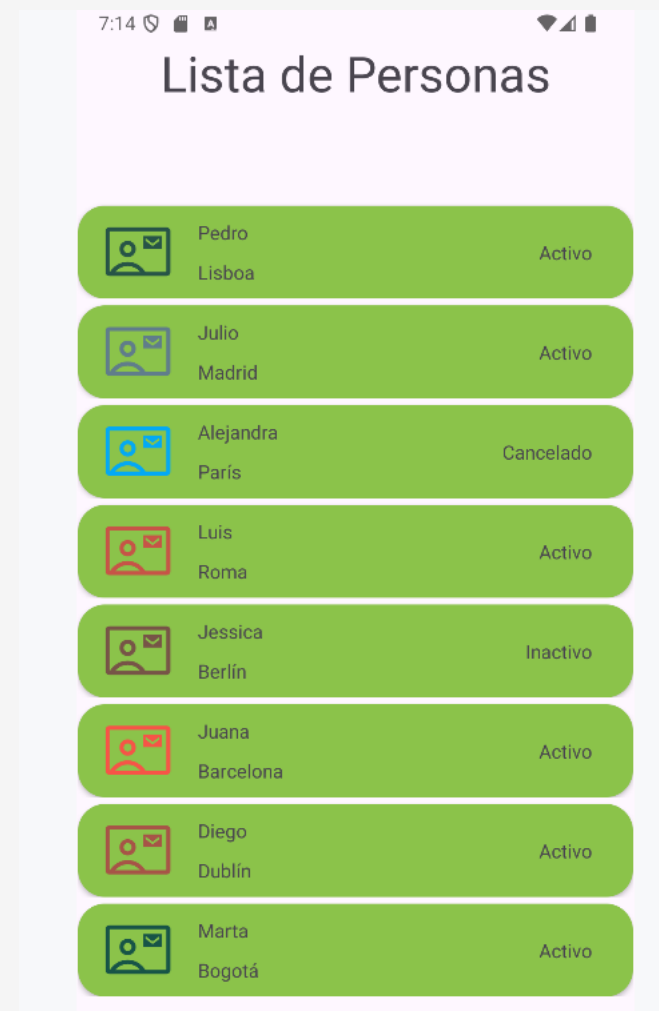


Programación de aplicaciones Android  
Interfaz de Usuario – RecyclerView



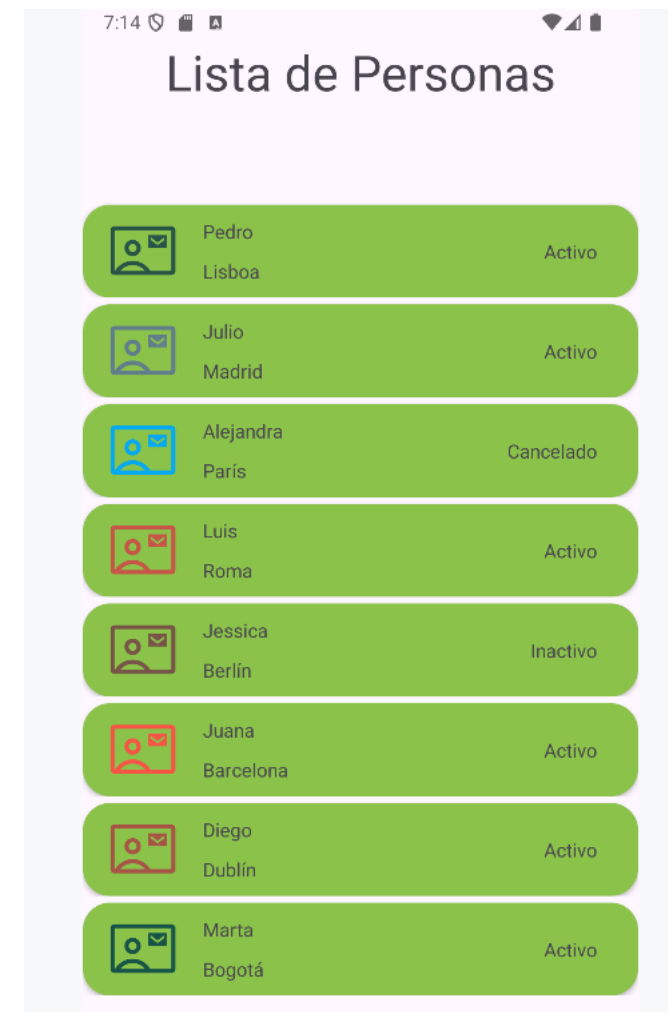
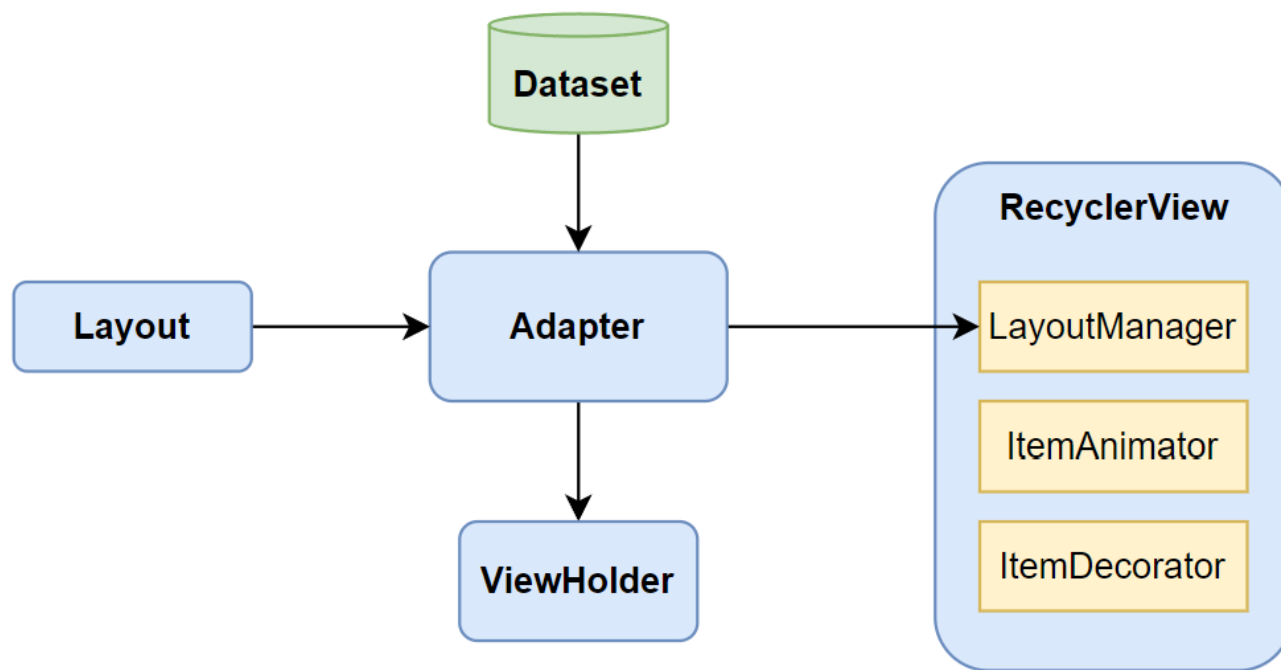
# RecyclerView

- Componente que muestra listas (o grids)
- Reutiliza vistas para ser más eficiente
- Es el sucesor de ListView





# RecyclerView - Estructura

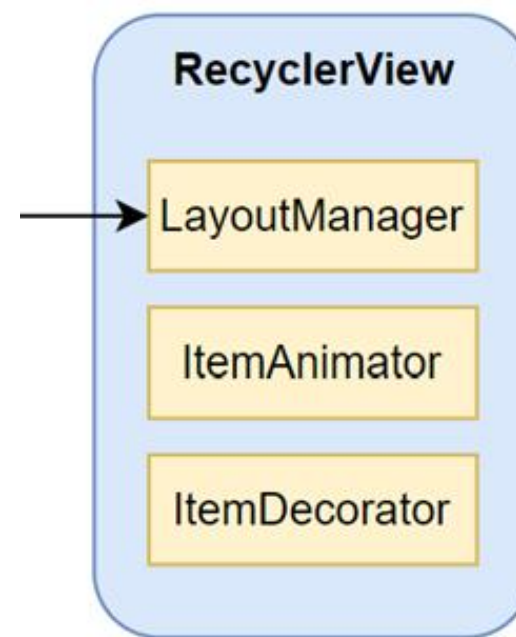




# RecyclerView - Estructura

## RecyclerView

- **ViewGroup** que contiene las vistas.
- Hay que asignarle un **LayoutManager**:
  - **LinearLayoutManager**: lista horizontal o vertical
  - **GridLayoutManager**: cuadrícula.
  - **StaggeredGridLayoutManager**: cuadrícula escalonada.

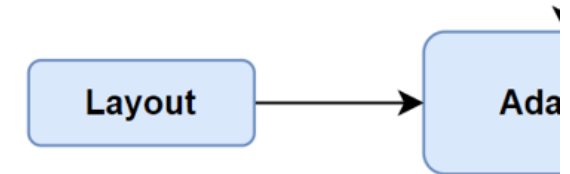




# RecyclerView - Estructura

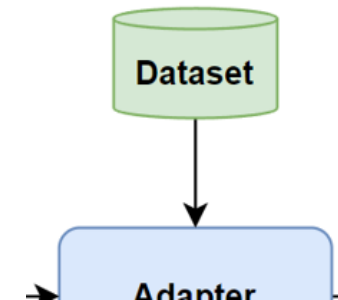
## Layout

- Diseño de la vista de un elemento individual a mostrar.
- Puede ser tan complejo como se quiera



## Dataset

- Origen de datos para los elementos
- Puede ser un array, un fichero, una base de datos etc





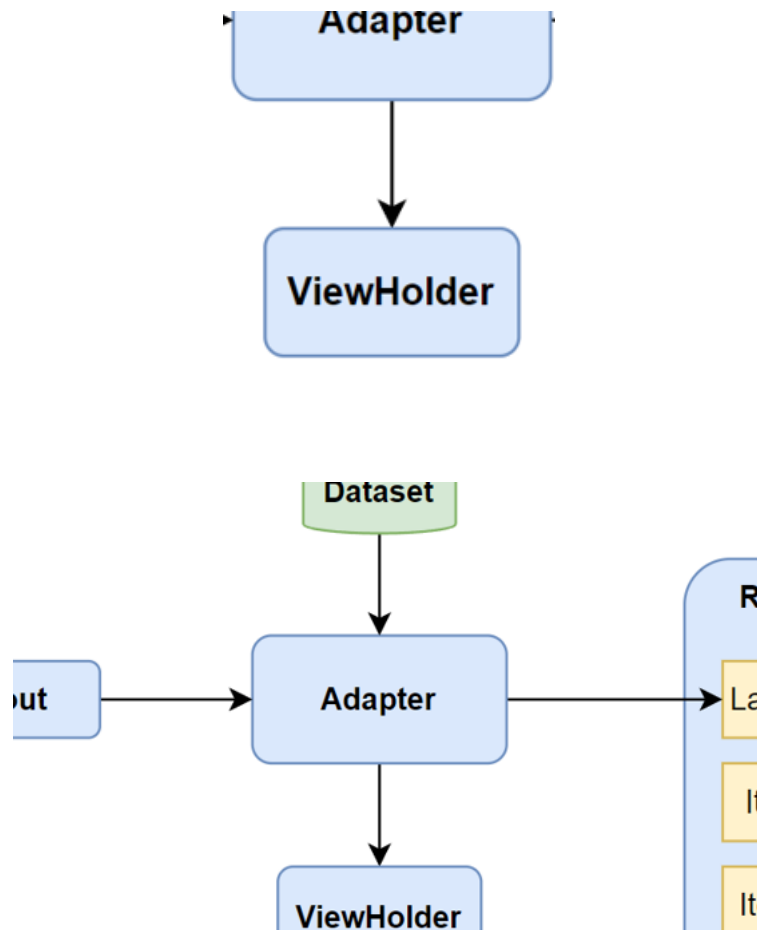
# RecyclerView - Estructura

## ViewHolder

- Contiene la vista de un elemento concreto del Dataset.
- Extiende la clase **RecyclerView.ViewHolder**

## Adapter

- Adapta los datos del Dataset a lo que finalmente verá el usuario
- Extiende la clase **RecyclerView.Adapter**





# RecyclerView - Estructura

## Adapter: sobrescribimos al menos 3 métodos

- **onCreateViewHolder()**
  - Crea un viewHolder
  - No le asocia datos
- **onBindViewHolder:**
  - Asocia a un viewHolder datos
- **getItemCount()**
  - Devuelve el número de elementos que contiene la fuente de datos



# RecyclerView - Ejemplo

## Creamos un nuevo proyecto con un Empty Views

New Project

**Empty Views Activity**

Creates a new empty activity

Name

Package name

Save location

Language

Minimum SDK

**i** Your app will run on approximately **98,6%** of devices.  
[Help me choose](#)

Build configuration language

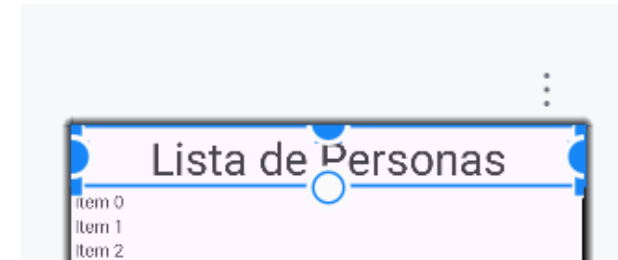




# RecyclerView – Ejemplo: RecyclerView

## En el layout activity\_main:

- Colocamos el textView en la parte superior.
  - Texto: “Lista de personas”
  - Tamaño: 36sp
  - Altura que se ajuste al contenido
  - Alineación: centrada



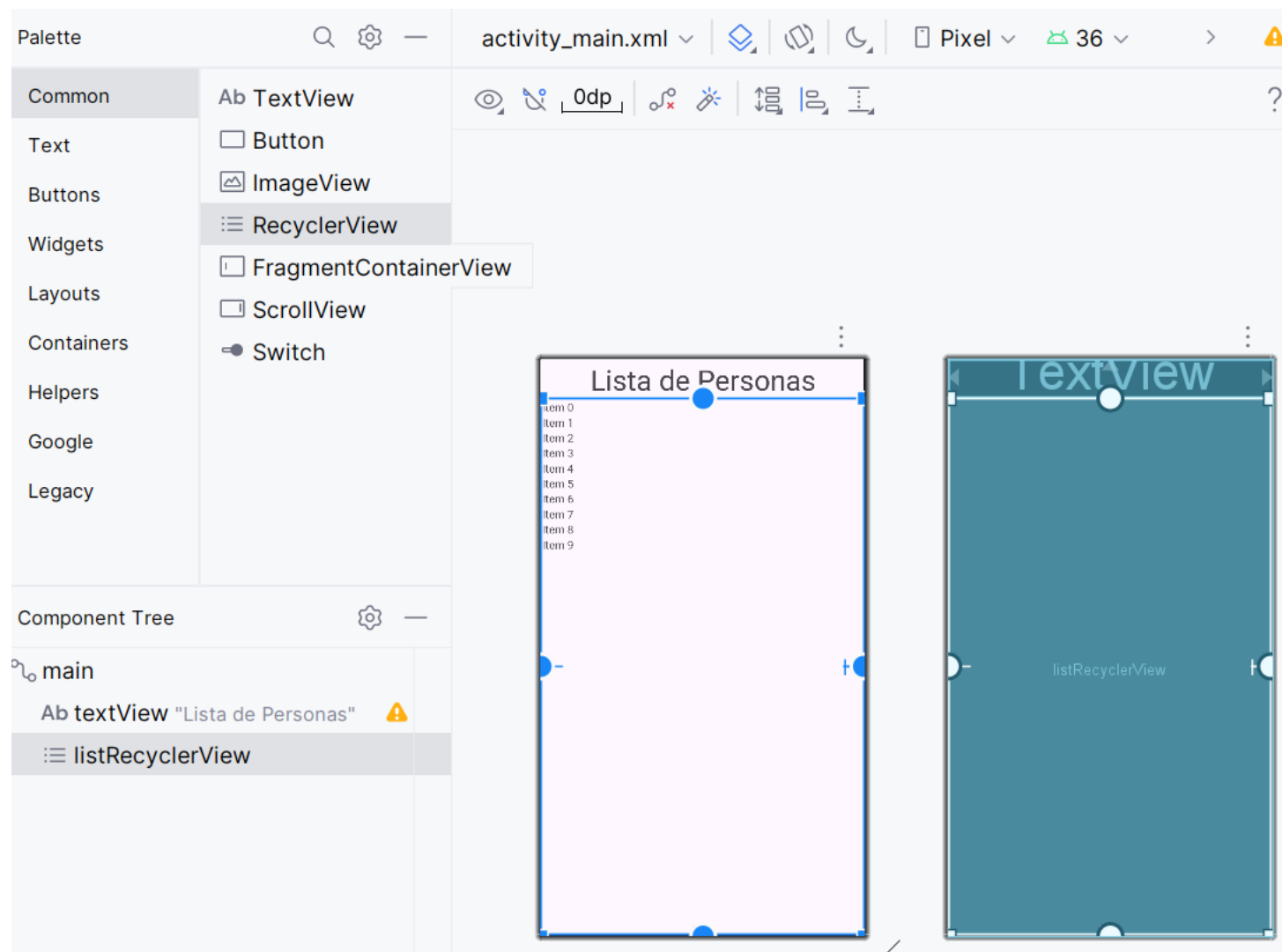
```
<TextView
    android:id="@+id/textView"
    android:layout_width="0dp"
    android:layout_height="wrap_content"
    android:text="Lista de Personas"
    android:textAlignment="center"
    android:textSize="36sp"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent" />
```



# RecyclerView – Ejemplo: RecyclerView

## En el layout activity\_main:

- Añadimos un RecyclerView
- Lo ajustamos para que ocupe el resto del espacio
- Le ponemos un nombre significativo:  
**rvListaPersonas**

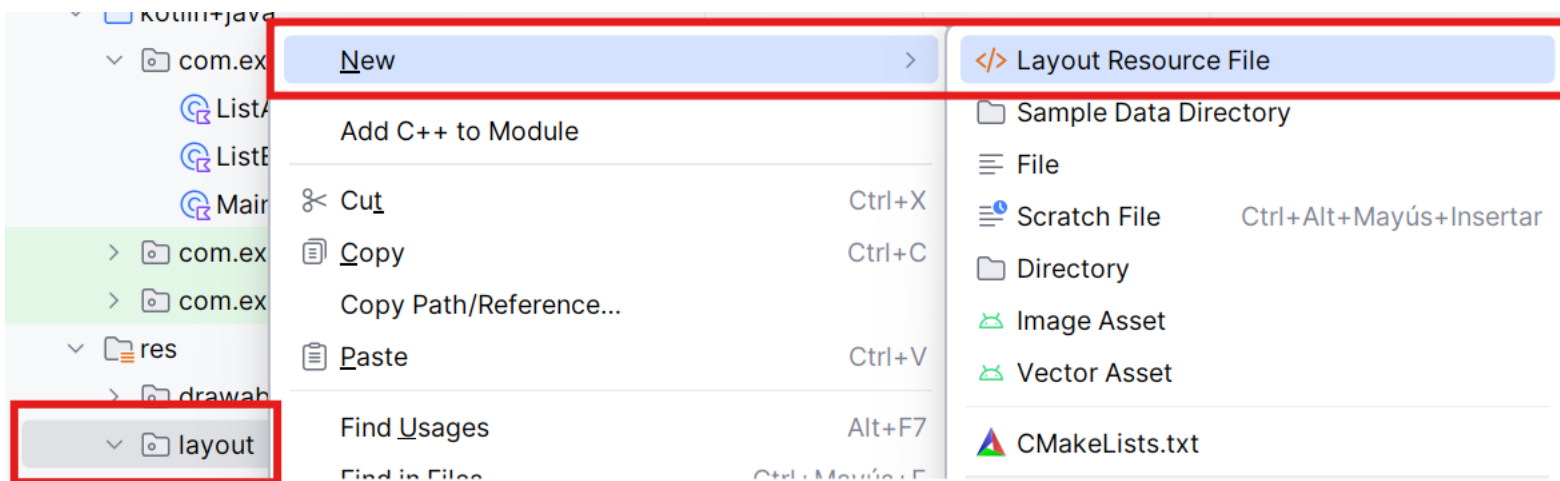




# RecyclerView – Ejemplo: Layout

## Creamos un layout para representar un elemento

- Pinchando sobre la carpeta layout con el botón derecho elegimos **New -> Layout Resource File**

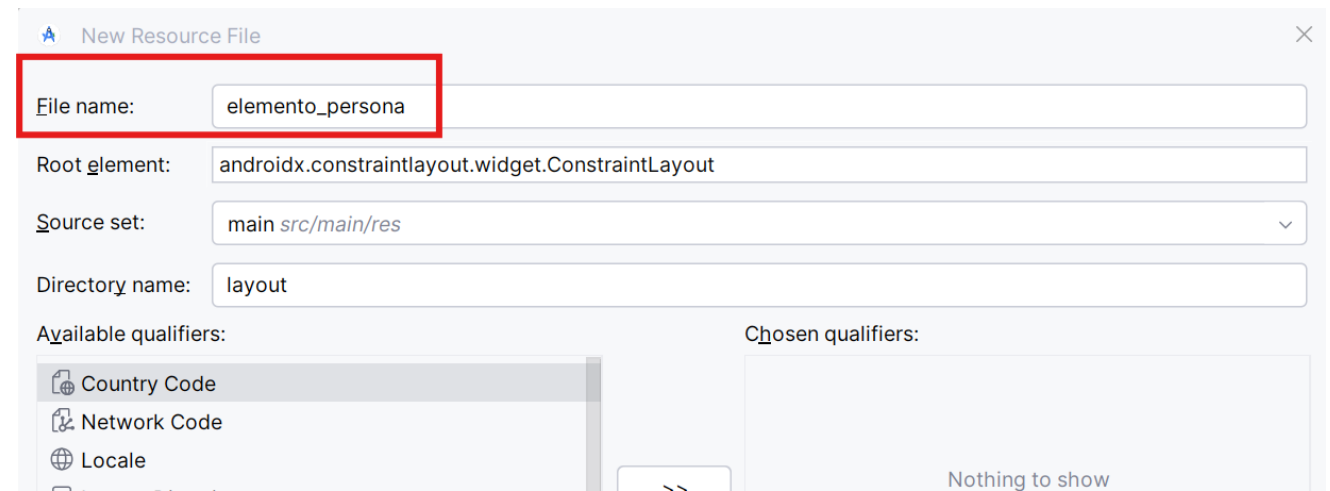
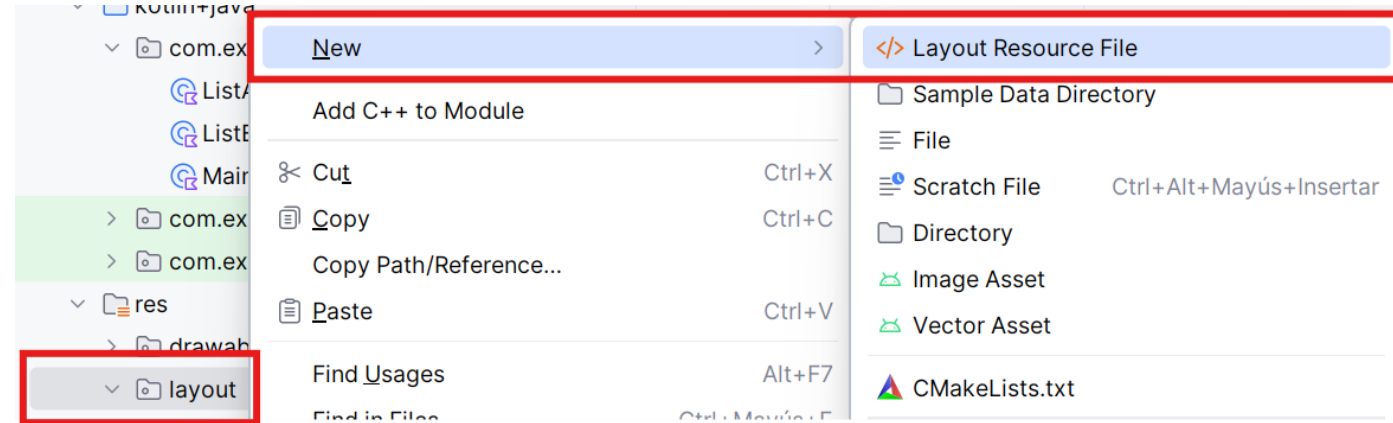




# RecyclerView – Ejemplo: Layout

## Creamos un layout para representar un elemento

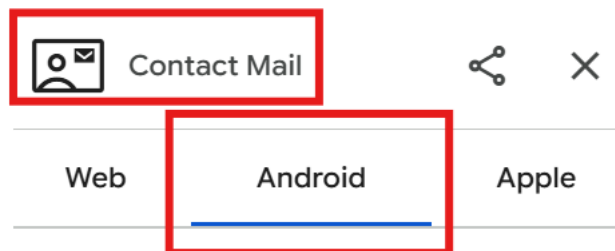
- Pinchando sobre la carpeta layout con el botón derecho elegimos **New -> Layout Resource File**
- Le llamamos **elemento\_persona**





# RecyclerView – Ejemplo: Layout

- Descargamos el icono en la siguiente URL: <https://fonts.google.com/icons>
- Buscamos el icono **Contact Mail** y lo descargamos en formato android



Use with Views

Download the Vector Drawable and follow the [import instructions](#).

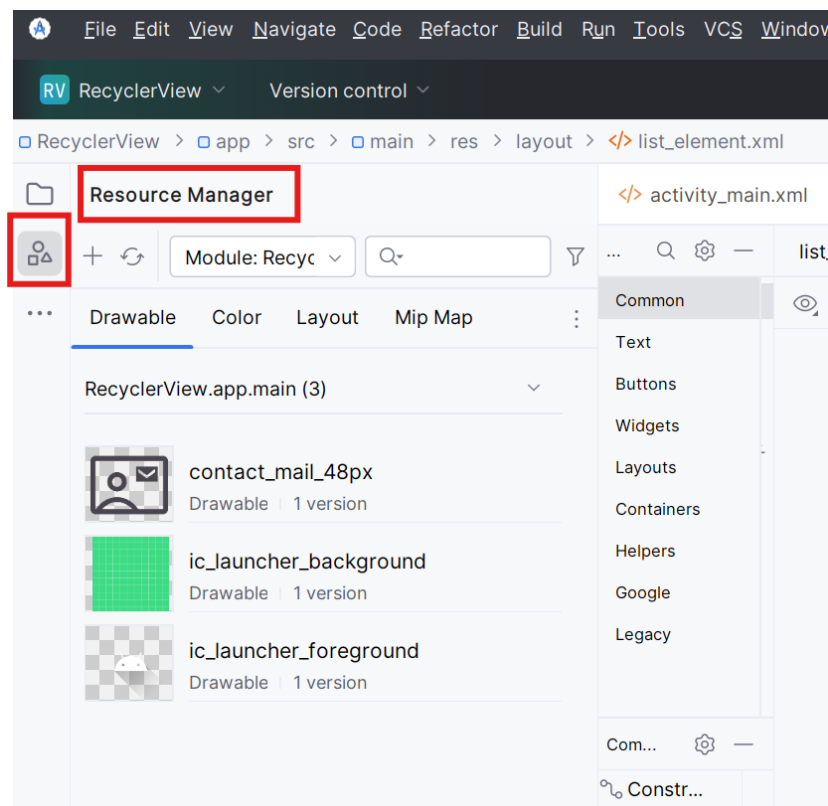
```
<ImageView ...  
    android:src="@drawable/contact_mi  
/>
```

 Copy code



# RecyclerView – Ejemplo: Layout

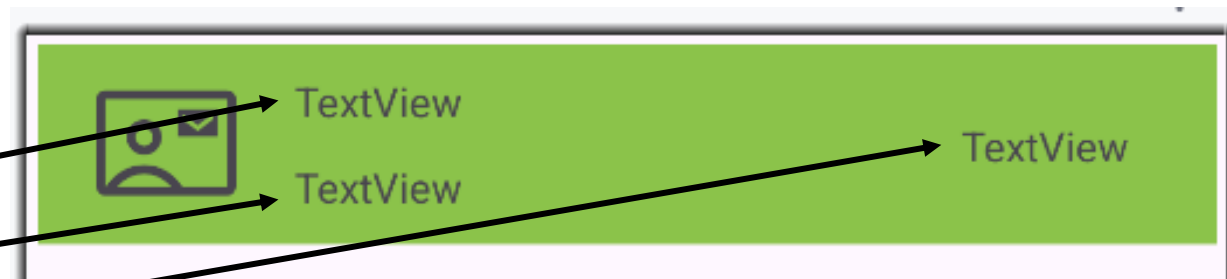
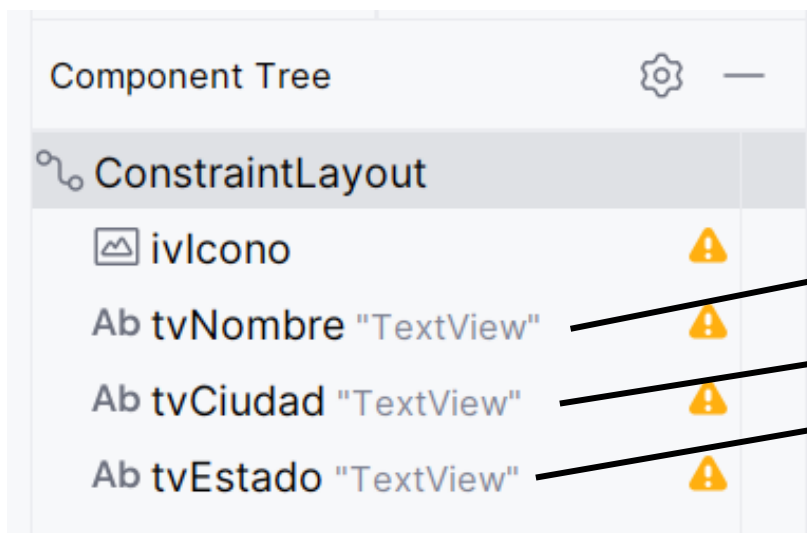
- Podemos agregarlo al proyecto arrastrando el archivo a la ventana **Resource Manager** y aceptando las opciones por defecto





# RecyclerView – Ejemplo: Layout

- Añadimos 3 TextView y un ImageView
- Les asignamos los nombres que se ven abajo
- Los posicionamos con constraints como se ve en la imagen





# RecyclerView – Ejemplo: Layout

- Al constraintlayout le ponemos un margen para que los elementos no se muestren pegados en el RecyclerView
- Le cambiamos también el color de fondo

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout xmlns:andro
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_margin="3dp"
    android:background="#8BC34A">
```





# RecyclerView – Ejemplo: Layout

- Al imageView **ivIcono** le ponemos restricción superior e izquierda respecto al padre
- Le ponemos un margen izquierdo de 20dp y un margen superior de 10dp

```
<ImageView  
    android:id="@+id/ivIcono"  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content"  
    android:layout_marginStart="20dp"  
    android:layout_marginTop="10dp"  
    app:layout_constraintStart_toStartOf="parent"  
    app:layout_constraintTop_toTopOf="parent"  
    app:srcCompat="@drawable/contact_mail_48px" />
```



# RecyclerView – Ejemplo: Layout

- Al textView **tvNombre** le ponmos restricción superior respecto al padre e izquierda respecto al imageView
- Le ponemos un margen izquierdo de 20dp y un margen superior de 10dp

```
<TextView
    android:id="@+id/tvNombre"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginStart="20dp"
    android:layout_marginTop="10dp"
    android:text="TextView"
    app:layout_constraintStart_toEndOf="@+id/ivIcono"
    app:layout_constraintTop_toTopOf="parent" />
```



# RecyclerView – Ejemplo: Layout

- Al textView **tvCiudad** le ponemos restricción inferior respecto al padre, izquierda respecto al imageView y superior respecto a tvNombre
- Le ponemos un margen izquierdo de 20dp y un margen superior de 10dp(este margen es respecto a tvNombre) y un margen inferior de 10dp

```
<TextView
    android:id="@+id/tvCiudad"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginStart="20dp"
    android:layout_marginTop="10dp"
    android:layout_marginBottom="10dp"
    android:text="TextView"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintStart_toEndOf="@+id/ivIcono"
    app:layout_constraintTop_toBottomOf="@+id/tvNombre" />
```



# RecyclerView – Ejemplo: Layout

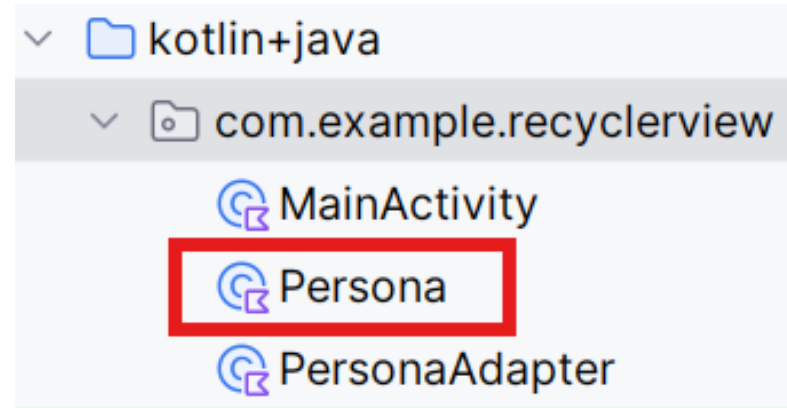
- Al textView tvEstado le ponemos restricción superior e izquierda respecto al padre.
- Le ponemos un margen derecho de 30dp y un margen superior de 30dp

```
<TextView  
    android:id="@+id/tvEstado"  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content"  
    android:layout_marginTop="25dp"  
    android:layout_marginEnd="30dp"  
    android:text="TextView"  
    app:layout_constraintEnd_toEndOf="parent"  
    app:layout_constraintTop_toTopOf="parent" />
```



# RecyclerView – Ejemplo: Dataset

- Creamos la clase persona



```
package com.example.recyclerview
```

```
class Persona (var color: String = "", var nombre: String = "", var ciudad: String = "", var estado: String = "")
```



# RecyclerView – Ejemplo: Dataset

- En **mainActivity** creamos una lista de personas que será nuestro Dataset:

```
val personas=ListOf(  
    Persona("#275447", "Pedro", "Lisboa", "Activo"),  
    Persona("#607d8b", "Julio", "Madrid", "Activo"),  
    Persona("#03a9f4", "Alejandra", "París", "Cancelado"),  
    Persona("#C75447", "Luis", "Roma", "Activo"),  
    Persona("#775447", "Jessica", "Berlín", "Inactivo"),  
    Persona("#F75447", "Juana", "Barcelona", "Activo"),  
    Persona("#A75447", "Diego", "Dublín", "Activo"),  
    Persona("#175447", "Marta", "Bogotá", "Activo")  
)
```



# RecyclerView – Ejemplo: Adapter

- Creamos la clase **PersonaAdapter**
- Tiene como atributo una lista de personas
- Hereda de RecyclerView.Adapter
  - Parametrizamos la clase indicando que utilizara el tipo de ViewHolder **PersonaAdapter.ViewHolder** (lo definiremos dentro de esta misma clase y como todavía no lo hemos definido se mostrará un error)

```
package com.example.recyclerview
```

```
import androidx.recyclerview.widget.RecyclerView
```

```
class PersonaAdapter(private val lista: List<Persona>): RecyclerView.Adapter<PersonaAdapter.PersonaViewHolder>() {  
}
```



# RecyclerView – Ejemplo: Adapter

- Definimos dentro de la clase PersonaAdapter la clase **PersonaViewHolder**
- Recibe en su constructor un View y hereda de RecyclerView.ViewHolder
- Dentro de la clase declaramos propiedades que recojan una referencia a los elementos que hemos definido en el layout del elemento.

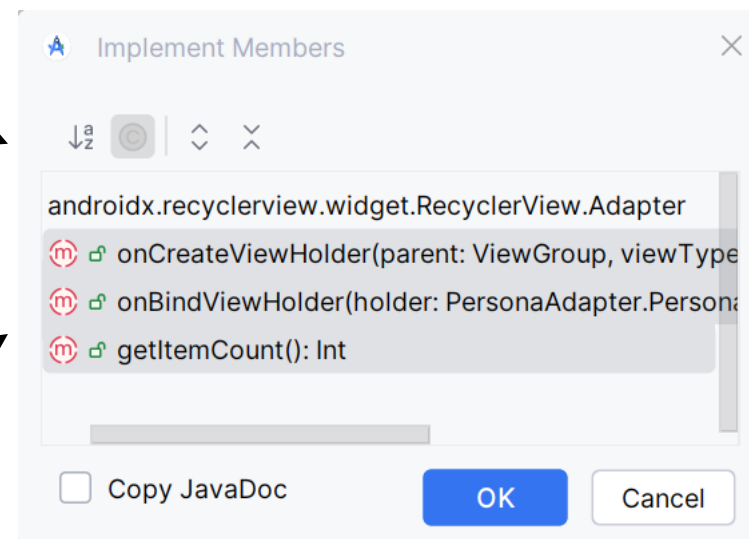
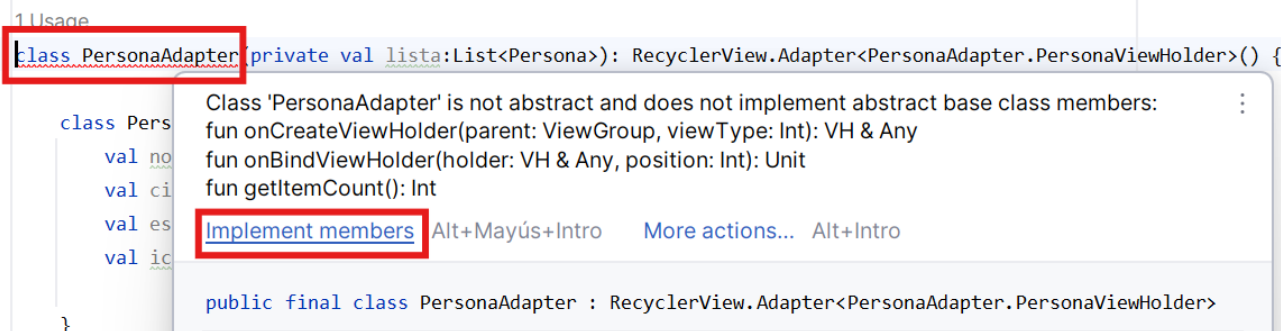
```
class PersonaViewHolder(itemView: View) : RecyclerView.ViewHolder(itemView) {  
    val nombre: TextView? = itemView.findViewById(R.id.tvNombre)  
    val ciudad : TextView?= itemView.findViewById(R.id.tvCiudad)  
    val estado : TextView? = itemView.findViewById(R.id.tvEstado)  
    val icono : ImageView? = itemView.findViewById(R.id.ivIcono)  
  
}
```



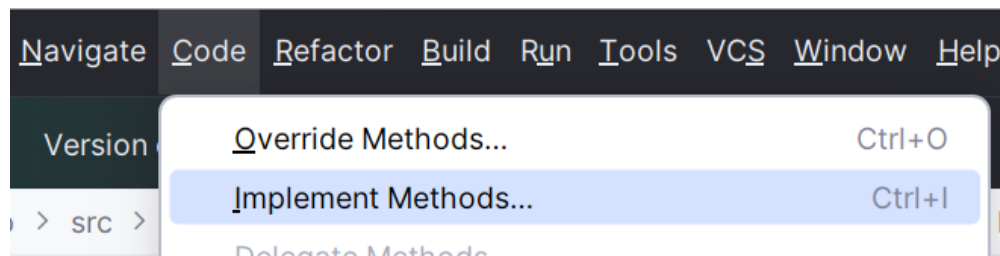


# RecyclerView – Ejemplo: Adapter

- Implementamos los 3 métodos obligatorios, para ello podemos pasar el ratón por encima del error y pinchar en **Implement members**



- También podemos ir al menú Code -> Implement Methods





# RecyclerView – Ejemplo: Adapter

- Método **onCreateViewHolder**(parent: ViewGroup, viewType: Int ) : PersonaViewHolder
- Este método devuelve un ViewHolder en este caso PersonaViewHolder sin datos asociados.
- “Infla” el layout con el diseño del element y a partir de la view obtenida crea un PersonaViewHolder.

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



# RecyclerView – Ejemplo: Adapter

- Método **onBindViewHolder**(holder: PersonaViewHolder, position: Int )
- Recibe un **PersonaViewHolder** y una posición y debe rellenar el **PersonaViewHolder** con los datos del **Dataset** situados en la posición que indica el parámetro **position**

```
override fun onBindViewHolder(  
    holder: PersonaViewHolder,  
    position: Int  
) {  
    val persona = lista[position]  
    holder.nombre?.text = persona.nombre  
    holder.ciudad?.text = persona.ciudad  
    holder.estado?.text = persona.estado  
    holder.icono?.setColorFilter(persona.color.toColorInt())  
}
```



# RecyclerView – Ejemplo: Adapter

- Método **getItemCount(): Int**
- Debe devolver el número de elementos que contiene el Dataset.
- En este caso al ser el Dataset una lista de objetos basta con obtener la propiedad size de la lista.

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



# RecyclerView – Ejemplo: Configurar el RecyclerView

- Configuramos el RecyclerView en MainActivity
- Le asignamos un LayoutManager de tipo LinearLayoutManager que mostrará los elementos en forma de lista.
- Creamos un Adapter con la lista de personas que creamos previamente y se lo asignamos al RecyclerView

*//Obtenemos una referencia al RecyclerView*

```
val rv = findViewById<RecyclerView>(R.id.rvListaPersonas)
```

*//Le asignamos un LayoutManager al RecyclerView de tipo LinearLayoutManager*

```
rv.layoutManager = LinearLayoutManager(this)
```

*//Asignamos un adapter al RecyclerView*

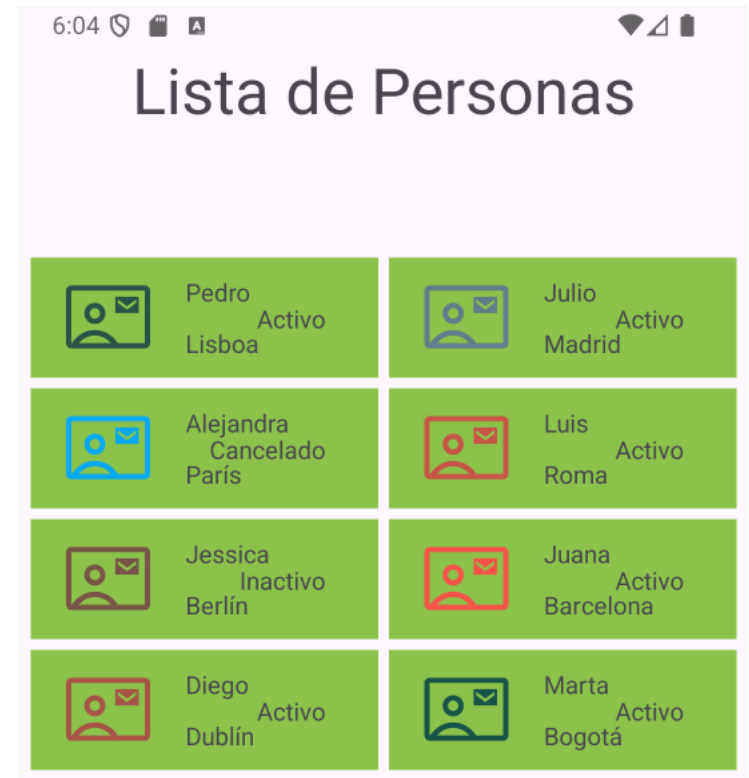
```
rv.adapter = PersonaAdapter(personas)
```



# RecyclerView – Ejemplo: Configurar el RecyclerView

- Programamos un GridLayoutManager
- En el constructor además del contexto hay que indicar cuantas columnas queremos.

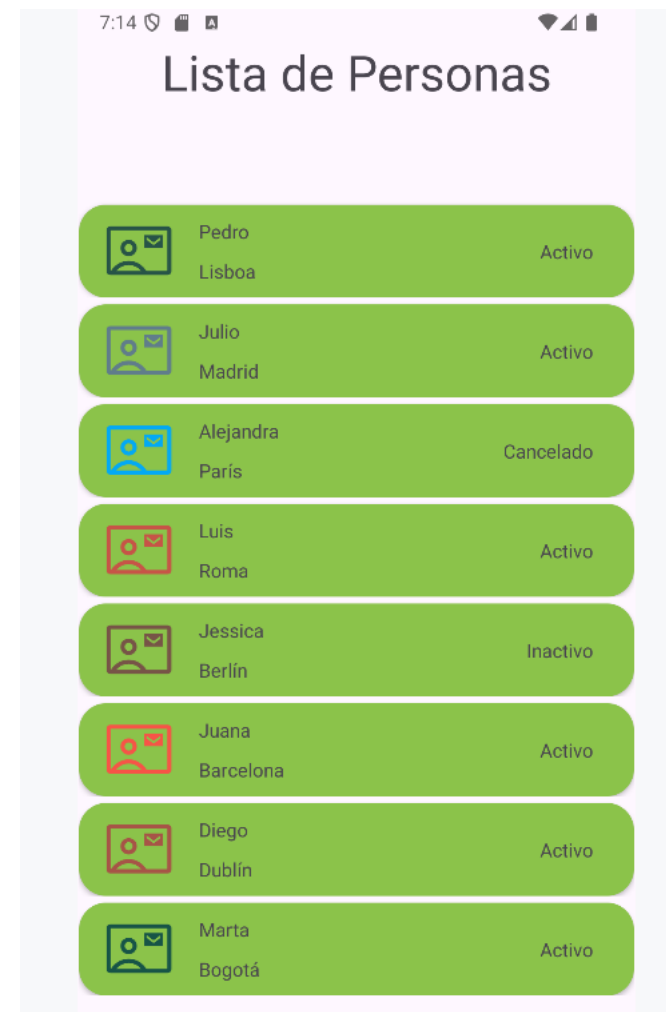
```
rv.LayoutManager = GridLayoutManager(this, 2)
```





# RecyclerView – Ejemplo: Ejercicio Propuesto 1

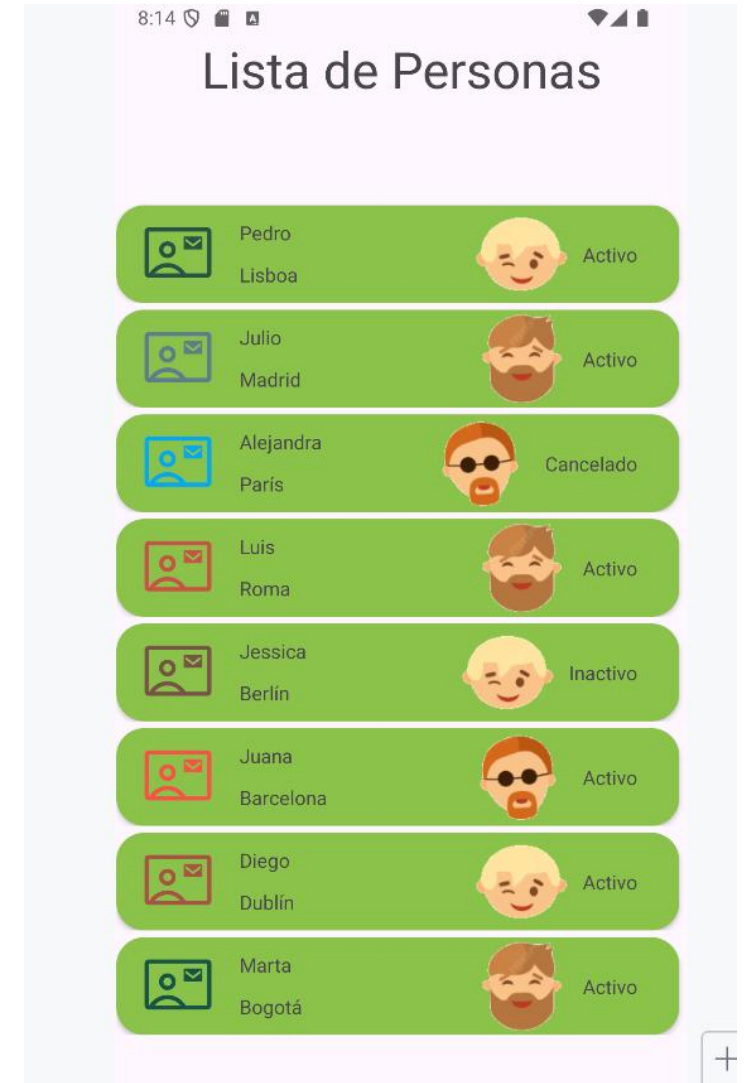
- Modifica el layout elemento\_persona y añade un **CardView** al diseño para que los bordes del elemento se vean redondeados.
- Será necesario añadir un segundo ConstraintLayout anidado dentro del CardView para poder ordenar los elementos dentro del CardView.





# RecyclerView – Ejemplo: Ejercicio Propuesto 2

- Modifica el ejercicio para que las personas tengan un atributo más que sea una imagen de una cara.







## RecyclerView – Ejemplo: Ejercicio Propuesto 3

- Si modificamos la clase persona y dejamos la cabecera de la siguiente manera:

```
class Persona(color: String, nombre: String, ciudad: String, estado: String)
```

- ¿Qué ha cambiado?
- Sin modificar la cabecera añade el código necesario para que la clase funcione como antes.