Wiki.Local

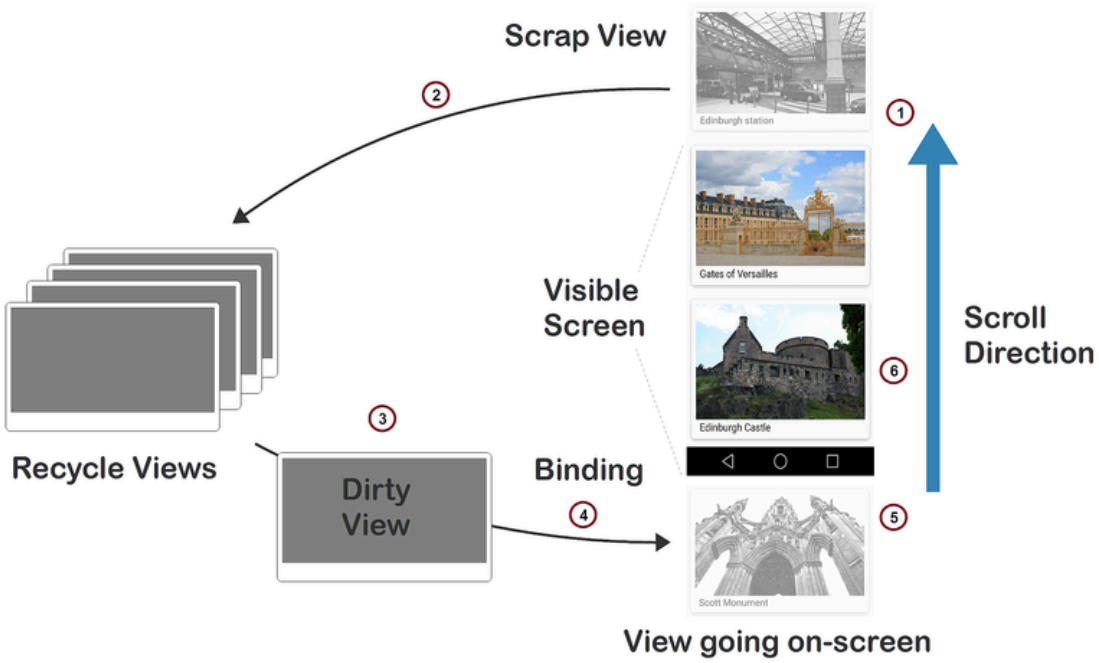
El Componente RecyclerView

OBJETIVO

En esta guía de laboratorio aprenderemos a manejar el componente RecyclerView para que a futuro puedas implementarlo en tus aplicaciones.

RecyclerView

RecyclerView es otro componente que permite crear listas al igual que ListView o GridView, este componente es más flexible y versatil que el ListView. A diferencia de ListView no asigna una vista a cada elemento del origen de datos. En su lugar asigna solamente el número de vistas que caben en la pantalla y reutiliza el diseño conforme el usuario se va desplazando como se muestra en la siguiente figura:



1. Cuando una vista se desplaza fuera de la vista y ya no se muestra, se convierte en una vista de rechazo (scrap view)
2. La vista de rechazo (scrap view) se coloca en un grupo y se convierte en una vista de reciclaje. Este grupo es una caché de vistas que muestran el mismo tipo de datos
3. Cuando se va a mostrar un nuevo elemento, se toma una vista del grupo de reciclaje para su reutilización. Dado que el adaptador debe volver a enlazar esta vista antes de mostrarse, se denomina vista sucia (dirty view)
4. La vista sucia (Dirty View) se recicla: el adaptador localiza los datos del siguiente elemento que se mostrará y copia estos datos en las vistas de este elemento. Las referencias para estas vistas se recuperan del soporte de vista asociado a la vista reciclada.
5. La vista reciclada se agrega a la lista de elementos de RecyclerView que están a punto de pasar a la pantalla
6. La vista reciclada va en pantalla a medida que el usuario se desplaza en el RecyclerView al siguiente elemento de la lista. Mientras tanto, otra vista se desplaza fuera de la vista y se recicla de acuerdo con los pasos anteriores.

Además de la reutilización de elementos de la vista, RecyclerView utiliza un elemento de optimización conocido como ViewHolder. Un ViewHolder es una clase especial que almacena en cache las referencias de vista. Cada vez que el Adapter infla un archivo de diseño (Layout), tambien se crea un ViewHolder correspondiente. El ViewHolder utiliza el método findViewById para referenciar las componentes view del archivo de diseño de cada item cada vez que se recicla el diseño para mostrar nuevos datos.

Componentes de RecyclerView

RecyclerView esta compuesto de los siguientes elementos:

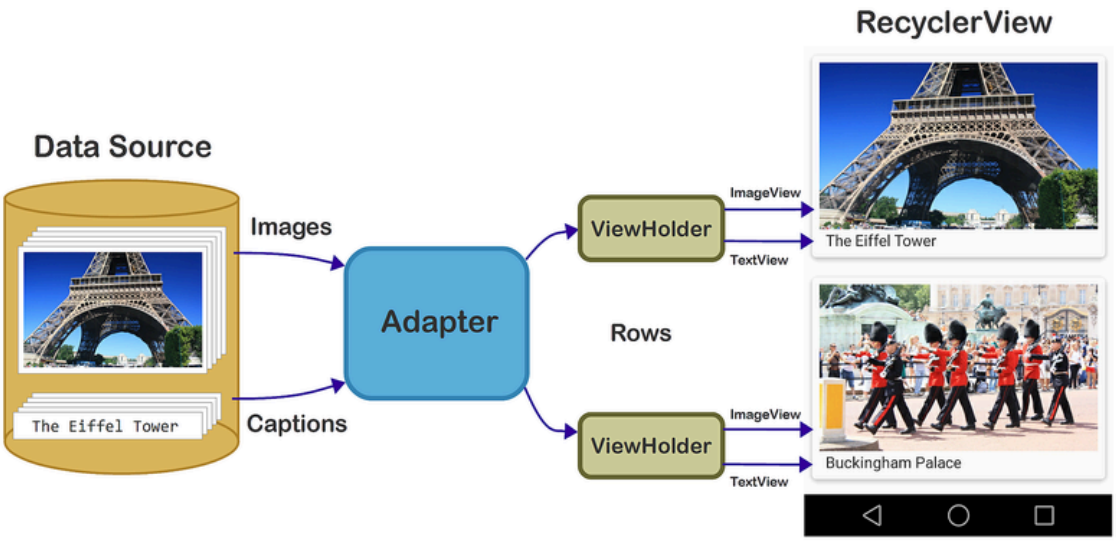


Figure 1: Ilustración de como asigna el Adapter el contenido a cada fila del RecyclerView a través de ViewHolders

El ViewHolder

El ViewHolder es una clase que almacena en caché las referencias de las vistas. El Adapter utiliza esta clase para enlazar la vista a su contenido, de tal forma que cada elemento tiene asociado una referencia o instancia de ViewHolder que se almacena en caché.

El Adapter

Es el componente que infla los diseños de elementos (crea una instancia del contenido de un archivo de diseño) y enlaza los datos a las vistas que se muestran dentro de un archivo.

LayoutManager

Es el responsable de colocar elementos en la pantalla; determina el tipo de presentación (una lista o una cuadrícula), la orientación (si los elementos se muestran vertical u horizontalmente) y qué elementos de dirección deben mostrarse (en orden normal o en orden inverso).

Pasos para Implementar una lista con RecyclerView

Para mostrar una lista de datos usando RecyclerView se siguen los siguientes pasos:

1. Crear un modelo de datos
2. Crear el diseño de la actividad principal (activity_main.xml)
3. Crear el diseño de los elementos del RecyclerView (items_recycler.xml)
4. Crear el ViewHolder y el Adapter
5. Vincular el Adapter al RecyclerView

1. Crear un modelo de datos (Data Class)

Para este ejemplo usaremos una lista de objetos de tipo Persona con los siguientes campos: nombre, apellido, imagen, telefono.

Persona.kt

```
data class Persona(
    var nombre: String? = null,
    var apellido: String? = null,
    var telefono: String? = null,
    var imagen: String? = null
) {
    constructor(nombre: String, apellido: String, telefono: String) : this(nombre, apellido, telefono, null)

    override fun toString(): String {
        return nombre ?: ""
    }
}
```

2. Crear el diseño de la actividad principal (activity_main.xml)

Agregamos el elemento `androidx.constraintlayout.widget.ConstraintLayout` dentro de la actividad principal:

activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout 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:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    <androidx.recyclerview.widget.RecyclerView
```

```
            android:id="@+id/recycler1"
            android:layout_width="match_parent"
            android:layout_height="match_parent"
            app:layout_constraintTop_toTopOf="parent"
            app:layout_constraintStart_toStartOf="parent"
            app:layout_constraintEnd_toEndOf="parent"
            app:layout_constraintBottom_toBottomOf="parent"
            />
    </androidx.constraintlayout.widget.ConstraintLayout>
```

3. Crear el diseño de los elementos del RecyclerView (item_persona.xml)

Para ello creamos un archivo de tipo Layout con el nombre item_persona.xml, cambiamos el layout a `androidx.cardview.widget.CardView` o `com.google.android.material.card.MaterialCardView`. Añadimos los elementos de tal forma que el diseño se vea de la siguiente forma:

item_persona.xml

```
    <?xml version="1.0" encoding="utf-8"?>
    <com.google.android.material.card.MaterialCardView xmlns:android="http://schemas.android.com/apk/res/android"
        xmlns:app="http://schemas.android.com/apk/res-auto"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_margin="10dp"
        app:cardElevation="10dp"
        >

        <androidx.constraintlayout.widget.ConstraintLayout
            android:layout_width="match_parent"
            android:layout_height="wrap_content">


            <ImageView
                android:id="@+id/imageLogo"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:src="@tools:sample/avatars"
                app:layout_constraintEnd_toEndOf="parent"
                app:layout_constraintStart_toStartOf="parent"
                app:layout_constraintTop_toTopOf="parent" />

            <TextView
                android:id="@+id/txtTitulo"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:text="TextView"

                android:textAppearance="@style/TextAppearance.AppCompat.Display1"
                app:layout_constraintEnd_toEndOf="parent"
                app:layout_constraintStart_toStartOf="parent"

                app:layout_constraintTop_toBottomOf="@id/imageLogo" />

            <TextView
                android:id="@+id/txtDescripcion"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:text="TextView"

                app:layout_constraintEnd_toEndOf="parent"
                app:layout_constraintStart_toStartOf="parent"
                app:layout_constraintTop_toBottomOf="@id/txtTitulo" />

            <Button
                android:id="@+id/btnModificar"
                android:layout_width="0dp"
                android:layout_height="wrap_content"
                android:text="Modificar"

                app:layout_constraintBottom_toBottomOf="parent"
                app:layout_constraintEnd_toStartOf="@id/btnEliminar"
                app:layout_constraintStart_toStartOf="parent"
                app:layout_constraintTop_toBottomOf="@id/txtDescripcion" />

            <Button
                android:id="@+id/btnEliminar"
                style="?attr/materialButtonOutlinedStyle"

                android:layout_width="0dp"
                android:layout_height="wrap_content"
                android:text="Eliminar"
                app:layout_constraintBottom_toBottomOf="parent"
                app:layout_constraintEnd_toEndOf="parent"
                app:layout_constraintStart_toEndOf="@id/btnModificar"

                app:layout_constraintTop_toBottomOf="@id/txtDescripcion" />


        </androidx.constraintlayout.widget.ConstraintLayout>
    </com.google.android.material.card.MaterialCardView>
```

4. Crear el Adapter y ViewHolder

Para implementar el Adapter se debe crear una clase que herede de la clase `RecyclerView.Adapter`. y a su vez esta debe contener un `ViewHolder` que extienda a `RecyclerView.ViewHolder`.

Un Adapter esta constituido de los siguientes métodos y objetos:

- **onCreateViewHolder**, este método se encarga de inflar el layout para cada elemento de la lista del modelo de datos.
- **OnBindViewHolder**, este método es llamado por el `RecyClerView` para mostrar los datos en la posición especificada, internamente desde este método se debe referenciar a los componentes del ViewHolder.
- Por lo general en el adapter tambien se encuentran los métodos para gestionar el modelo de datos, agregar, modificar y eliminar elementos del RecyclerView.
- getItemCount, este método debe devolver el tamaño de los datos
- inner class `ViewHolder` que contiene la instancia de las vista de cada item

Para nuestro ejemplo el adapter (`PersonasAdapter`) queda de la siguiente forma:

PersonasAdapter.kt

```
package com.example.lab4ejemplo

import android.content.Context
import android.view.LayoutInflater
import android.view.View
import android.view.ViewGroup
import android.widget.ImageView
import android.widget.TextView
import androidx.recyclerview.widget.RecyclerView
import com.bumptech.glide.Glide

class PersonasAdapter (ctx: Context, private val personasModel: List<Persona> ):RecyclerView.Adapter<PersonasAdapter.PersonasViewHolder>(){

    inner class PersonasViewHolder(itemView: View):RecyclerView.ViewHolder(itemView){
        var imageUrl = itemView.findViewById<ImageView>(R.id.imageLogo)
        var titulo = itemView.findViewById<TextView>(R.id.txtTitulo)
        var descripcion = itemView.findViewById<TextView>(R.id.txtDescripcion)
    }

    override fun onCreateViewHolder(parent: ViewGroup, viewType: Int): PersonasViewHolder {
        val v = LayoutInflater.from (parent.context).inflate(R.layout.item_persona,parent,false)
        return PersonasViewHolder(v)
    }

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

    override fun onBindViewHolder(holder: PersonasViewHolder, position: Int) {
        val i = personasModel[position]
        holder.titulo.text = i.nombre
        holder.descripcion.text = i.apellido

        Glide.with(holder.itemView.context)
            .load(i.imagen
            ,)
            .centerCrop()
            //.placeholder(R.drawable.loading_spinner)
            .into(holder.imageUrl)


    }

}
```

* Para agregar la librería Glide se añade en el archivo `Build.gradle.kt` del `Project:nombreproy` la siguiente linea :

```
plugins {
    id("com.android.application") version "8.2.2" apply false
    id("org.jetbrains.kotlin.android") version "1.9.22" apply false
    id("com.google.devtools.ksp") version "1.8.10-1.0.9" apply false

}
```

En el archivo `AndroidManifest.xml`

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

En el archivo `build.gradle.kts` del `Module:app`

```
    implementation("com.github.bumptech.glide:glide:4.16.0")
```

5. Vincular el Adapter al RecyclerView

Este ultimo paso consiste en crear una instancia de PersonasAdapter enviando como paráemetro el Contexto y el modelo de datos:

MainActivity.kt

```
package com.example.lab4ejemplo

import androidx.appcompat.app.AppCompatActivity
import android.os.Bundle
import androidx.recyclerview.widget.LinearLayoutManager
import androidx.recyclerview.widget.RecyclerView

class MainActivity : AppCompatActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)
        val r = findViewById<RecyclerView>(R.id.recycler1)
        val personasList = mutableListOf<Persona>(
            Persona("RAFA","GORGORI","74859632"
                ,"https://fastly.picsum.photos/id/237/200/300.jpg?hmac=TmmQSbShHz9CdQm0NkEjx1Dyh_Y984R9LpNrpvH2D_U"
```

```
                ),
                Persona("MARIBEL","CONDORI","74859632"
                    ,"https://fastly.picsum.photos/id/237/200/300.jpg?hmac=TmmQSbShHz9CdQm0NkEjx1Dyh_Y984R9LpNrpvH2D_U"
                ),
                Persona("JUANA","AGUIRRE","74859632",
                    "https://fastly.picsum.photos/id/237/200/300.jpg?hmac=TmmQSbShHz9CdQm0NkEjx1Dyh_Y984R9LpNrpvH2D_U"
                ),
                Persona("GUADALUPE","MEJIA","74859632"
                    ,"https://fastly.picsum.photos/id/237/200/300.jpg?hmac=TmmQSbShHz9CdQm0NkEjx1Dyh_Y984R9LpNrpvH2D_U"
                ),
                Persona("SILVANA","GUTIERREZ","74859632"
                    ,"https://fastly.picsum.photos/id/237/200/300.jpg?hmac=TmmQSbShHz9CdQm0NkEjx1Dyh_Y984R9LpNrpvH2D_U"
                ),
                Persona("MARIO","MARTINEZ","74859632"
                    ,"https://fastly.picsum.photos/id/237/200/300.jpg?hmac=TmmQSbShHz9CdQm0NkEjx1Dyh_Y984R9LpNrpvH2D_U"
                ),

                )

            val personasAdapter = PersonasAdapter(applicationContext,personasList)
            r.adapter = personasAdapter

            val layout = LinearLayoutManager(applicationContext)
            layout.orientation = LinearLayoutManager.VERTICAL
            r.layoutManager = layout



    }
}
```

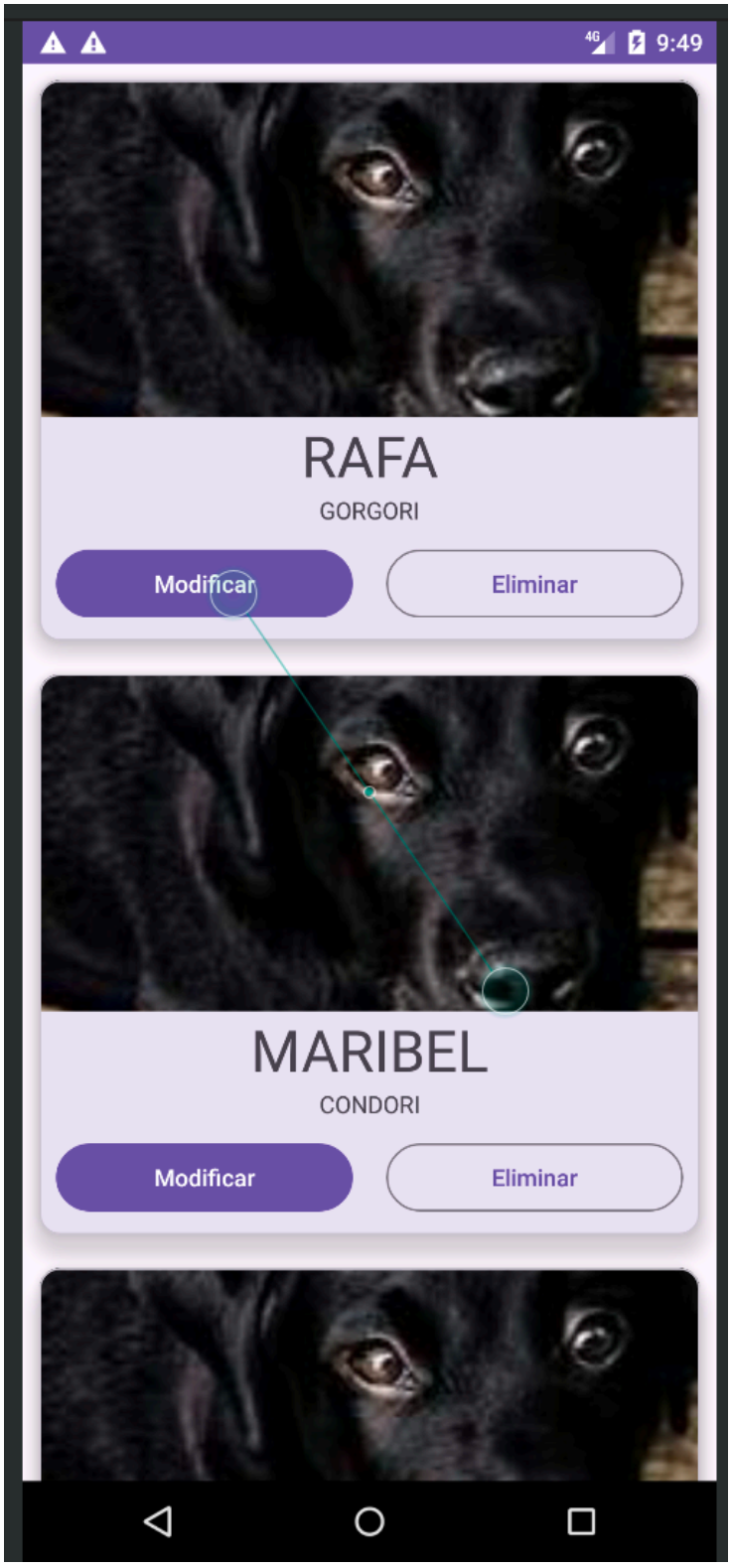
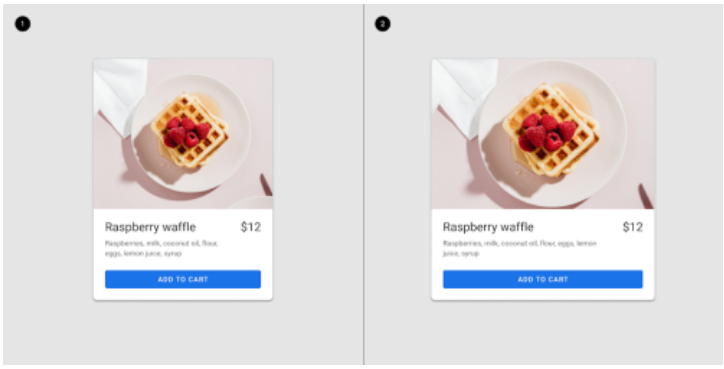


Figure 2: RecyclerView en ejecución

Laboratorio Nro.3 (parte 2)

Usando el componente RecyclerView elaborar un modelo de datos que permita listar los productos de un local de cómida rápida con sus 4 operaciones CRUD y una base de datos SQLite. Tal como se ilustra en la siguiente figura:



materias/tecnologias-emergentes/unidad_1/08_recyclerview_kotlin.txt · Última modificación: 2024/04/02 23:41 por ::1