

INDICE

<u>1.1.</u>	<u>INTRODUCCIÓN</u>	<u>1</u>
<u>1.2.</u>	<u>AÑADIENDO LAS LIBRERÍAS</u>	<u>2</u>
<u>1.3.</u>	<u>CASO PRÁCTICO</u>	<u>3</u>
<u>1.3.1.</u>	<u>CREANDO EL DISEÑO DEL CARDVIEW</u>	<u>4</u>
<u>1.3.2.</u>	<u>CREANDO EL ADAPTADOR</u>	<u>7</u>
<u>1.3.3.</u>	<u>DEFINIENDO LA CLASE VIEWHOLDER</u>	<u>9</u>
<u>1.3.4.</u>	<u>COMPLETANDO EL FUNCIONAMIENTO DEL ADAPTADOR</u>	<u>10</u>
<u>1.3.4.1.</u>	<u>INSTANCIANDO EL VIEWHOLDER</u>	<u>10</u>
<u>1.3.4.2.</u>	<u>AÑADIENDO DATOS AL VIEWHOLDER</u>	<u>11</u>
<u>1.3.4.3.</u>	<u>IMPLEMENTANDO O MÉTODO GETITEMCOUNT</u>	<u>11</u>
<u>1.3.5.</u>	<u>PASO FINAL</u>	<u>11</u>
<u>1.3.6.</u>	<u>GESTIONAR EVENTOS DE CLICK'S SOBRE LOS ELEMENTOS</u>	<u>13</u>
<u>1.3.6.1.</u>	<u>PASANDO INFORMACIÓN</u>	<u>15</u>
<u>1.3.7.</u>	<u>AÑADIENDO-BORRANDO-ACTUALIZANDO ELEMENTOS</u>	<u>18</u>

1.1. Introducción

- ✓ El RecyclerView tiene como función mostrar al usuario en forma de lista con scroll.

La diferencia con respecto al ListView (visto anteriormente) o GridView (no veremos), es que con cantidades altas de datos, este componente es más eficiente que los dos anteriores.

- El RecyclerView genera todos los elementos de la lista que se manden inicialmente, pero si añadimos nuevos elementos o borramos no tiene que volver a recrear toda la lista, solamente recrea los que sufran algún cambio.
- En operaciones de eliminación, el RecyclerView reutiliza los view's que conforman cada uno de los elementos de la lista (de ahí viene el nombre de 'recycler') de tal forma que al añadir nuevos elementos 'reutiliza' los view's creados previamente para añadir nuevos datos.
- Un ArrayAdapter genera los views de todos los elementos de la lista estén visualmente o no visibles en la lista y si hay algún cambio tiene que volver a recrearlos todos.
- ✓ Por otro lado banda, el componente CardView sirve para diseñar la forma en cómo se presenta la información de cada componente de la lista.

Por ejemplo, podemos indicar que cada elemento de la lista que se muestra haciendo uso del RecyclerView esté formado por una ImageView y un TextView.

Este componente permite presentar información en forma de grupos, empleando una forma de 'tarjeta'.

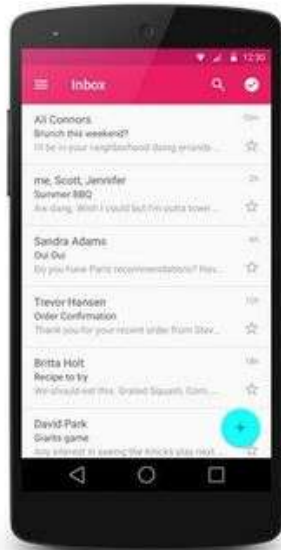


Figure 1. A list using RecyclerView



Figure 2. A list also using CardView



Ejemplo de cardview obtenida de este [enlace](#)

1.2. Añadiendo las librerías

- ✓ En el caso de querer utilizar las bibliotecas de compatibilidad v7, lo primero que tenemos que hacer es añadir las librerías que van permitir que hagamos uso de dichos componentes.

Si arrastramos gráficamente los componentes, dichas librerías ya se añaden automáticamente al archivo **build.gradle (nivel de módulo)**

```
1 implementation 'com.android.support:recyclerview-v7:28.0.0'  
2 implementation 'com.android.support:cardview-v7:28.0.0'
```

1.3. Caso Práctico

NOTA: Todos los ejemplos que siguen están implementados utilizando las bibliotecas de compatibilidad androidx.

✓ Vamos a crear la siguiente Activity.



- ✓ Adjunto archivo **Planetas.zip** que contiene las imágenes de los planetas.
- ✓ Dentro del paquete **Adaptadores** crear una nueva 'Empty Activity' de nombre: **UD04_01_RecyclerViewCardView** de tipo Launcher.

Modifica el archivo **AndroidManifest.xml** y añade una label a la activity.

El layout asociado a dicha activity tendrá un RecyclerView ocupando el 100% del espacio con un margen superior de 24dp's:

Versión empleando las bibliotecas de compatibilidad androidx:

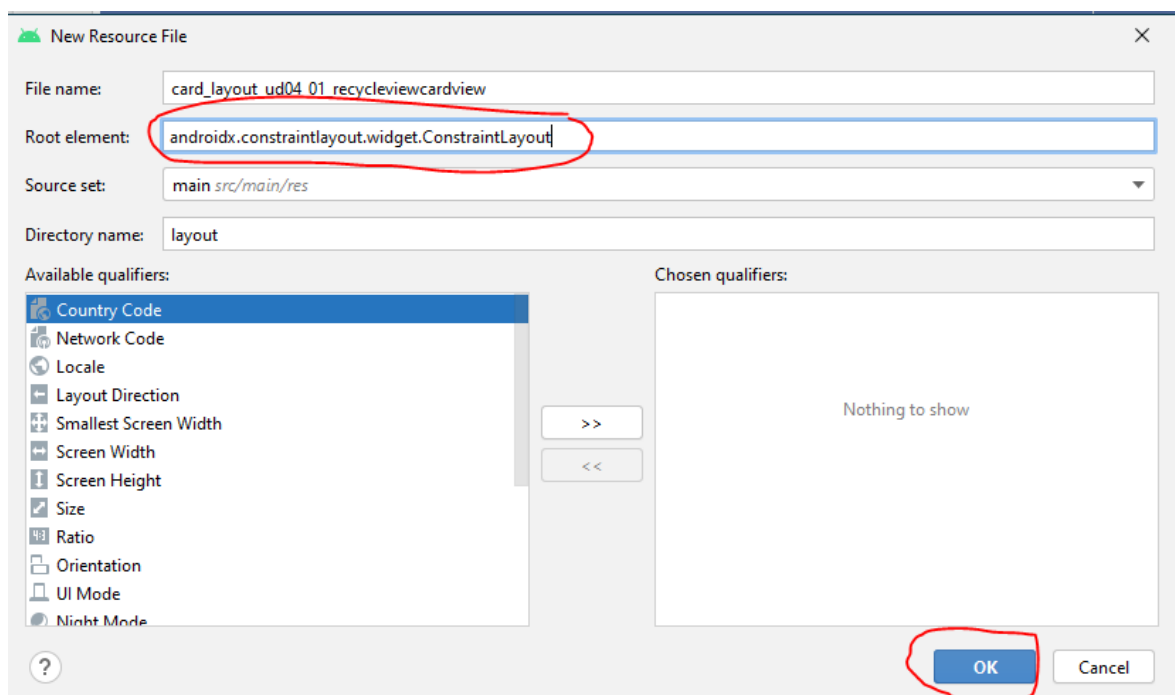
```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".UD04_01_RecyclerViewCardView">

    <androidx.recyclerview.widget.RecyclerView
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:layout_marginTop="24dp"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

</androidx.constraintlayout.widget.ConstraintLayout>
```

1.3.1.Creando el diseño del CardView

- ✓ El CardView viene a ser como un layout que representa como se van visualizar **cada una de las filas que conforman el RecyclerView**.
- ✓ Por lo tanto creamos un archivo en /res/layout, de nombre '*card_layout_ud04_01_recycleviewcardview*'.



- ✓ Ahora añadimos el layout que queremos que tenga cada fila y los componentes que queremos que aparezcan.
- ✓ Tendrá que añadirse el layout dentro de la etiqueta <CardView>.
- ✓ En el ejemplo vamos a hacer que aparezca una imagen y un texto asociado a dicha imagen y vamos a usar un ConstraintLayout para hacerlo.

✓ Fijarse que:

- En la altura del layout y del `CardView`, vamos a poner 'wrap_content' para que solamente ocupe el alto del componente que sea más alto (en este caso la foto).
- El ancho y alto de la imagen va a ser 100x100 dp's y se desplaza a la izquierda.
- El texto ocupa todo el espacio restante y está centrado con respecto a la foto. Tiene una separación izquierda con respecto a la foto y un tamaño de texto grande.
- El texto tiene el color del S.O. suponiendo que empleamos el aspecto Holo (@android:color/holo_purple) (podéis poner el que os guste más).

```
<androidx.cardview.widget.CardView
    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="5dp"
    app:cardBackgroundColor="#81C784"
    app:cardCornerRadius="12dp"
    app:cardElevation="3dp"
    app:contentPadding="4dp">

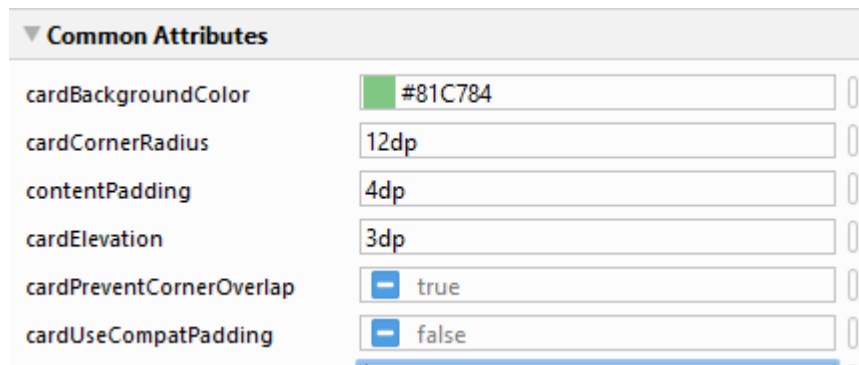
    <androidx.constraintlayout.widget.ConstraintLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content">

        <ImageView
            android:id="@+id/imgImaxe_UD04_01_CardLayout"
            android:layout_width="100dp"
            android:layout_height="100dp"
            android:contentDescription="Imaxe"
            app:layout_constraintEnd_toStartOf="@+id/tvTexto_UD04_01_CardView"
            app:layout_constraintStart_toStartOf="parent"
            app:layout_constraintTop_toTopOf="parent"
            app:srcCompat="@mipmap/ic_launcher" />

        <TextView
            android:id="@+id/tvTexto_UD04_01_CardView"
            android:layout_width="0dp"
            android:layout_height="wrap_content"
            android:layout_marginTop="16dp"
            android:paddingStart="5dp"
            android:text="Texto a mostrar por fila"
            android:textColor="@android:color/holo_purple"
            android:textSize="24sp"
            app:layout_constraintEnd_toEndOf="parent"
            app:layout_constraintStart_toEndOf="@+id/imgImaxe_UD04_01_CardLayout"
            app:layout_constraintTop_toTopOf="parent" />
    </androidx.constraintlayout.widget.ConstraintLayout>

</androidx.cardview.widget.CardView>
```

- ✓ Fijarse que tenemos una serie de propiedades específicas de cardview:



- ✓ Podemos consultar la lista completa en este [enlace](#).

Entre las que vamos a modificar están:

- cardBackgroundColor: Color de fondo de la tarjeta.
- cardCornerRadius: El perfil redondeado de las tarjetas.
- cardElevation: Elevación (Material Design)
- contentPadding: Padding de todo el contenido que está dentro de la tarjeta.
- Para separar una tarjeta de otra podemos hacer uso de los márgenes a nivel de CardView.

```
<androidx.cardview.widget.CardView 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="5dp"
    app:cardBackgroundColor="#81C784"
    app:cardCornerRadius="12dp"
    app:cardElevation="3dp"
    app:contentPadding="4dp">
```

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

```
<ImageView
    android:id="@+id/imgImaxe_UD04_01_CardLayout"
    android:layout_width="100dp"
    android:layout_height="100dp"
    android:contentDescription="Imaxe"
    app:layout_constraintEnd_toStartOf="@+id/tvTexto_UD04_01_CardView"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent"
    app:srcCompat="@mipmap/ic_launcher" />
```

```
<TextView
    android:id="@+id/tvTexto_UD04_01_CardView"
    android:layout_width="0dp"
```

```

        android:layout_height="wrap_content"
        android:layout_marginTop="16dp"
        android:paddingStart="5dp"
        android:text="Texto a mostrar por fila"
        android:textColor="@android:color/holo_purple"
        android:textSize="24sp"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toEndOf="@+id/imgImaxe_UD04_01_CardLayout"
        app:layout_constraintTop_toTopOf="parent" />
    </androidx.constraintlayout.widget.ConstraintLayout>

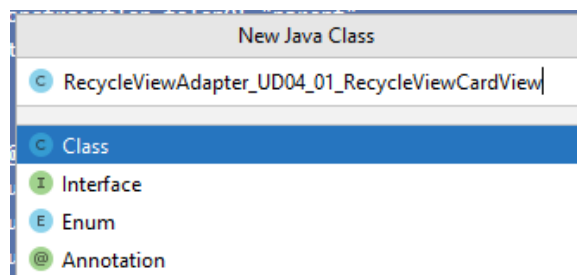
</androidx.cardview.widget.CardView>

```

1.3.2.Creando el Adaptador

✓ Un adaptador:

- Es un elemento intermediario entre una fuente de datos (XML, Arrays, Ficheros, BBDD) y un interface de usuario que muestra esos datos, por ejemplo un Spinner, una lista de Selección, etc.
- Por cada dato crea una View y la representa.
- Es el responsable de generar todos los elementos de representación asociados a los datos. Imaginar que cada ítem estuviera formad por dos subcomponentes: nombre de persona y foto. El adaptador debe ser quien pueda representar eso.
- En el del RecyclerView, va a emplear un adaptador que deriva de la clase RecyclerView.Adapter.
- Por lo tanto vamos a crear una clase que derive de ella.



Al hacerlo dará un error en el que nos informa que debemos de implementar una serie de métodos.

```

package com.example.adaptadores;

import android.view.ViewGroup;

import androidx.annotation.NonNull;
import androidx.recyclerview.widget.RecyclerView;

public class RecyclerViewAdapter_UD04_01_RecyclerViewCardView extends RecyclerView.Adapter {
    @NonNull
    @Override
    public RecyclerView.ViewHolder onCreateViewHolder(@NonNull ViewGroup viewGroup, int i) {
        return null;
    }

    @Override
    public void onBindViewHolder(@NonNull RecyclerView.ViewHolder viewHolder, int i) {

```

```

    }

    @Override
    public int getItemCount() {
        return 0;
    }
}

```

Como vemos se implementan los siguientes métodos:

- **getItemCount():** Tiene que devolver el número de elementos que se van a visualizar en la lista.
 - **onCreateViewHolder():** Este método tiene que devolver un objeto de la clase ViewHolder la cual es empleada para visualizar el contenido de la lista. Será en la clase ViewHolder donde 'cargaremos' el diseño de fila hecho en el paso anterior.
 - **onBindViewHolder():** Este método recibe el ViewHolder del método anterior y asociamos a cada componente gráfico de dicho View el dato que queremos que visualice.
- ✓ En este adaptador es donde tendremos que obtener los datos que conformarán las filas.
- ✓ Podríamos obtenerlos de una consulta a una base de datos o de cualquiera otra fuente (internet, un archivo,...).

En el ejemplo vamos a definirlos localmente...

En nuestro ejemplo tendremos que descargar imágenes de varios planetas y guardarlas en la carpeta /res/drawable

Los datos serán los siguientes (adaptarlos a vuestro caso):

```

package com.example.adaptadores;

import android.view.ViewGroup;

import androidx.annotation.NonNull;
import androidx.recyclerview.widget.RecyclerView;

public class RecyclerViewAdapter_UD04_01_RecyclerViewCardView extends RecyclerView.Adapter {

    private String[] textos = {"Mercurio", "Venus", "Tierra", "Jupiter", "Saturno"};
    private int[] imagenes = {
        R.drawable.mercurio, R.drawable.venus, R.drawable.tierra, R.drawable.jupiter, R.drawable.saturno};

    @NonNull
    @Override
    public RecyclerView.ViewHolder onCreateViewHolder(@NonNull ViewGroup viewGroup, int i) {
        return null;
    }
    .....
}

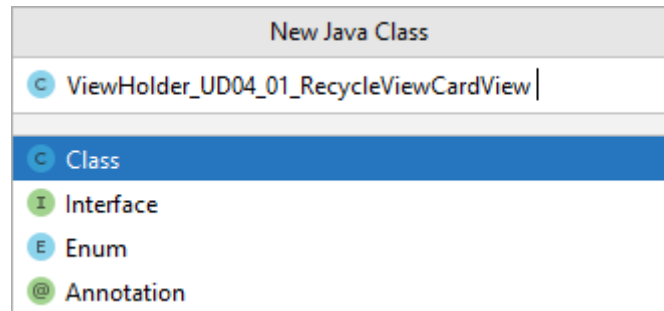
```

Nota: Fijarse en la orden y como el número de elemento dentro del array se corresponde a dos arrays con el mismo concepto (en la posición 0 del array está el texto y la imagen de Mercurio).

1.3.3. Definiendo la clase ViewHolder

- ✓ ViewHolder va a ser lo que emplee el adaptador para 'cargar' los datos en los elementos gráficos que conforman cada fila (en nuestro ejemplo, una imagen y un texto).

Creemos por tanto una clase que derive de RecyclerView.ViewHolder (podríamos crearla dentro de la clase RecyclerView.Adapter):



Al hacerlo nos dará un error de que hay que implementar un constructor. Lo haremos con el asistente.

```
package com.example.adaptadores;

import android.view.View;

import androidx.annotation.NonNull;
import androidx.recyclerview.widget.RecyclerView;

public class ViewHolder_UD04_01_RecyclerViewCardView extends RecyclerView.ViewHolder {
    public ViewHolder_UD04_01_RecyclerViewCardView(@NonNull View itemView) {
        super(itemView);
    }
}
```

- ✓ Esta clase va a ser la que nos permite acceder a los elementos gráficos definidos en el CardView (lo que está definido en el layout que representa cada fila).
- ✓ De esta forma desde el Adapter vamos a poder cargar con datos los elementos gráficos.
- ✓ Al constructor de esta clase se le pasará (siguiente paso) una instancia de 'card_layout_ud04_01_recycleviewcardview'.
- ✓ Simplemente referenciaremos dichos controles para poder manejarlos desde el adaptador.

Modificamos el código como sigue:

```
package com.example.adaptadores;

import android.view.View;
import android.widget.ImageView;
import android.widget.TextView;

import androidx.annotation.NonNull;
import androidx.recyclerview.widget.RecyclerView;

public class ViewHolder_UD04_01_RecyclerViewCardView extends RecyclerView.ViewHolder {
    public ImageView itemImagen;
    public TextView itemTexto;

    public ViewHolder_UD04_01_RecyclerViewCardView(@NonNull View itemView) {
```

```

        super(itemView);

        itemImagen = itemView.findViewById(R.id.imgImaxe_UD04_01_CardLayout);
        itemTexto = itemView.findViewById(R.id.tvTexto_UD04_01_CardView);
    }
}

```

1.3.4.Completando el funcionamiento del adaptador

1.3.4.1. Instanciando el ViewHolder

- ✓ Como dijimos antes, el método onCreateViewHolder tiene que 'devolver' un objeto ViewHolder en el que se encuentran los elementos gráficos que conforman la fila.

La clase en la que se basará va a ser la que hicimos en los pasos anteriores y que deriva de ViewHolder (*ViewHolder_UD04_01_RecyclerViewCardView*).

Para hacerlo tenemos que realizar un proceso de 'inflar' el layout que conforma cada fila (*card_layout_ud04_01_recycleviewcardview.xml*) a un objeto de la clase View.

Y después crear un objeto de la clase *ViewHolder_UD04_01_RecyclerViewCardView* en base al View obtenido anteriormente.

- ✓ Para hacer esto primero tenemos que 'convertir' el layout donde están definidos los elementos gráficos que conforman la fila a una View.

Para eso empleamos la [clase LayoutInflater](#).

- Primero obtenemos una referencia a dicha clase de una de las siguientes formas:

- `LayoutInflater mInflater = (LayoutInflater) viewGroup.getContext().getSystemService(Context.LAYOUT_INFLATER_SERVICE);`
- `LayoutInflater mInflater = LayoutInflater.from(viewgroup.getContext());`

- Después llamamos al método 'inflate' para que devuelva por programación una referencia al layout donde están definidos los elementos gráficos de una fila (un View):

- `View v = mInflater.inflate(R.layout.card_layout_ud04_01_recycleviewcardview,viewGroup,false);`

- Y por último creamos el ViewHolder creando un objeto de la clase ViewHolder creada por nosotros previamente (*ViewHolder_UD04_01_RecyclerViewCardView*) y pasando como parámetro al constructor el View anterior:

- `RecyclerView.ViewHolder viewHolder = new ViewHolder_UD04_01_RecyclerViewCardView(v);`

```

@NonNull
@Override
public RecyclerView.ViewHolder onCreateViewHolder(@NonNull ViewGroup viewGroup, int i) {

    LayoutInflater mInflater = (LayoutInflater) viewGroup.getContext().getSystemService(Context.LAYOUT_INFLATER_SERVICE);
    //View view = (LayoutInflater.from(viewGroup.getContext()))

```

```
View v = inflater.inflate(R.layout.card_layout_ud04_01_recycleviewcardview,viewGroup,false);
RecyclerView.ViewHolder viewHolder = new ViewHolder_UD04_01_RecyclerViewCardView(v);

return viewHolder;
}
```

1.3.4.2. Añadiendo datos al ViewHolder

- ✓ Los datos, recordar, que están definidos en dos arrays, pero podéis tener un único arrays de objetos de una clase y cogerlos de ese array.

Lo que siempre tendremos es al menos un array de elementos con información.

- ✓ Otro método que vimos que se creaba cuando creamos el RecyclerView.Adapter era el **onBindViewHolder**.

En este método vamos a 'meter' los datos que queremos que aparezcan en el ViewHolder creado en el paso anterior.

Para eso temos que emplear o parámetro 'i' que indica o número de elemento do array sobre o que temos que buscar a información e después asociarla a los elementos gráficos.

Por otra banda como el objeto de la clase ViewHolder pertenece realmente a la clase ViewHolder_UD04_01_RecyclerViewCardView, podemos acceder sus elementos gráficos ya que están declarados como 'public' en la definición da clase.

- ✓ O código queda como sigue:

```
@Override
public void onBindViewHolder(@NonNull RecyclerView.ViewHolder viewHolder, int i) {
    ViewHolder_UD04_01_RecyclerViewCardView viewHolderMeu = (ViewHolder_UD04_01_RecyclerViewCardView)
    viewHolder;
    viewHolderMeu.itemImagen.setImageResource(imagenes[i]);
    viewHolderMeu.itemTexto.setText(textos[i]);
}
```

1.3.4.3. Implementando o método getItemCount

- ✓ Como indicamos anteriormente, dicho método tiene que devolver el número de elementos de la lista, que debe de coincidir con el número de elementos del array donde están los datos.

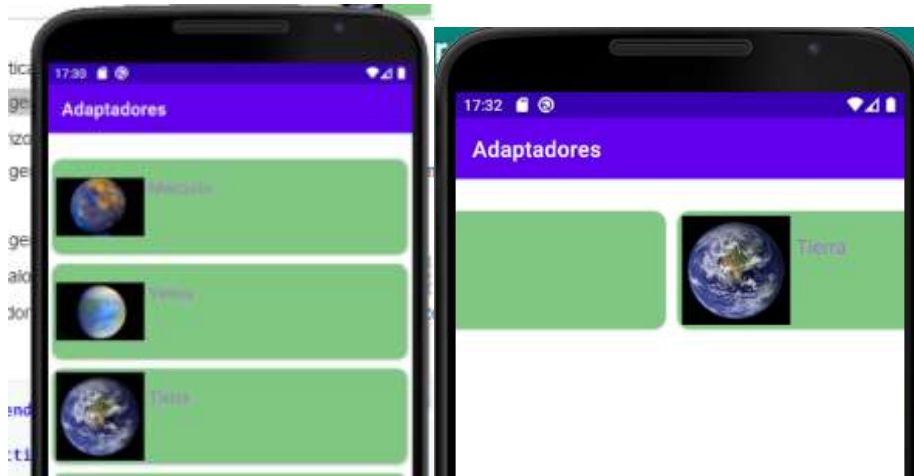
Por lo tanto:

```
@Override
public int getItemCount() {
    return textos.length;
}
```

1.3.5.Paso final

- ✓ Por último tenemos que hacer 2 cosas:


- **1º Paso:** El RecyclerView necesita un Layout Manager que le diga cómo van distribuidos cada uno de los elementos que van a aparecer.
- La forma más común de hacerlo es con un LinearLayout y que vayan en forma de en forma de fila, pero podemos emplear otras distribuciones.



- En la imagen anterior vemos:
 - Ejemplo de LinearLayout con distribución vertical (por defecto):


```
RecyclerView.LayoutManager layoutManager = new LinearLayoutManager(this);
```
 - Ejemplo de LinearLayout con distribución horizontal:


```
RecyclerView.LayoutManager layoutManager = new LinearLayoutManager(this,LinearLayoutManager.HORIZONTAL,false);
```

- Ejemplo de GridLayout con dos columnas:
 RecyclerView.LayoutManager layoutManager = new GridLayoutManager(this,2);
- Una vez creado el LayoutManager hay que asociarlo al RecyclerView.

- **2º Paso:** Hay que crear una instancia del adaptador creado por nosotros (RecyclerViewAdapter_UD04_01_RecyclerViewCardView) y asociarlo al RecyclerView.

```
package com.example.adaptadores;

import androidx.appcompat.app.AppCompatActivity;
import androidx.recyclerview.widget.GridLayoutManager;
import androidx.recyclerview.widget.RecyclerView;

import android.os.Bundle;

public class UD04_01_RecyclerViewCardView extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_u_d04_01__recycle_view_card_view);

        RecyclerViewAdapter_UD04_01_RecyclerViewCardView recycleAdapter = new RecyclerViewAdap-
ter_UD04_01_RecyclerViewCardView();

        RecyclerView.LayoutManager layoutManager = new GridLayoutManager(this,2);
        RecyclerView recyclerView = findViewById(R.id.rvwRecyclerView);
        recyclerView.setLayoutManager(layoutManager);
        recyclerView.setAdapter(recycleAdapter);
    }
}
```

1.3.6.Gestionar eventos de Click's sobre los elementos

- ✓ Podemos modificar un par de propiedades del layout asociado al cardview para que cuando pulsemos aparezca un efecto de 'selección' sobre el elemento pulsado.

```
<androidx.cardview.widget.CardView 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="5dp"
    android:foreground="?android:attr/selectableItemBackground"
    android:clickable="true"
    android:focusable="true"
    app:cardBackgroundColor="#81C784"
    app:cardCornerRadius="12dp"
    app:cardElevation="3dp"
    app:contentPadding="4dp">
```

- ✓ Podéis leer más acerca de este tipo de animaciones en [este enlace](#).

- ✓ Se queremos gestionar el evento del Click lo podemos hacer:
 - Sobre todo el conjunto de elementos que conforman la interface gráfica de la fila.
 - Sobre algunos de los elementos que conforman la interface gráfica de la fila.
- ✓ **Caso a) Gestión del Click sobre todo el conjunto.** En primer lugar tenemos que hacer uso del objeto itemView que se encuentra disponible en la clase ViewHolder y que representa el CardView con todo:

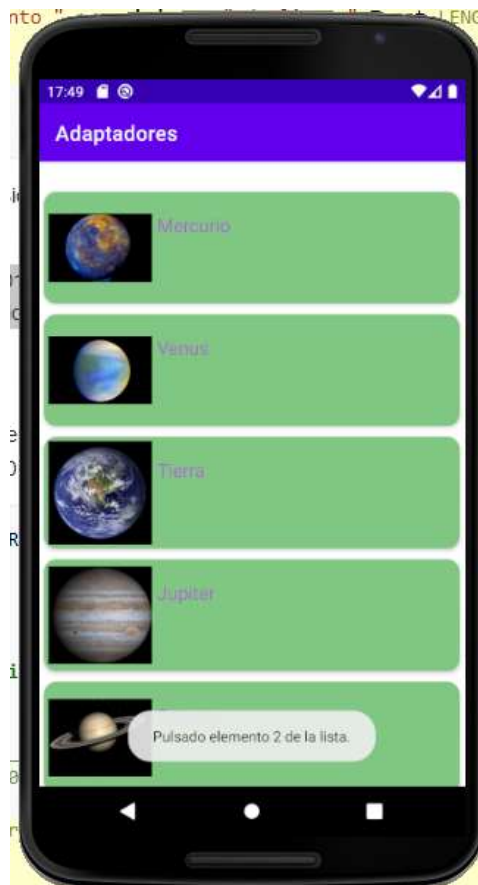
```
public class ViewHolder_UD04_01_RecyclerViewCardView extends RecyclerView.ViewHolder {
    public ImageView itemImagen;
    public TextView itemTexto;
```

```
    public ViewHolder_UD04_01_RecyclerViewCardView(@NonNull View itemView) {
        super(itemView);
```

```
        itemImagen = itemView.findViewById(R.id.imgImaxe_UD04_01_CardLayout);
        itemTexto = itemView.findViewById(R.id.tvTexto_UD04_01_CardView);
```

```
        itemView.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                int posicion = getAdapterPosition();
                Toast.makeText(v.getContext(), "Pulsado elemento " + posicion + " de la lista.", Toast.LENGTH_SHORT).show();
            }
        });
    }
}
```

- ✓ **Línea 15:** Disponemos del método `getAdapterPosition()` que nos devuelve la posición del elemento seleccionado de la lista.



Nota: Esto mismo lo podemos hacer desde la clase 'RecyclerViewAdapter_UD04_01_RecyclerViewCardView' ya que el view que pasamos como parámetro al constructor del ViewHolder es el que empleamos en el ejemplo anterior.

- ✓ **Caso b) Gestión del Click sobre un view del conjunto.** En este caso debemos de registrar el evento de Click sobre el view que nos interese. Lo podemos hacer en la clase ViewHolder o en la clase RecyclerViewAdapter_UD04_01_RecyclerViewCardView.

Nota: Fijarse que ahora solamente responderá al evento de hacer click sobre la imagen.

1.3.6.1. Pasando información

- ✓ Esto ya está comentado en el ImageView.
- ✓ Todos view's disponen de una propiedad Tag asociada en la que se puede guardar cualquier tipo de información, ya que permite guardar un objeto de cualquier clase.

Por lo tanto podemos emplear dicha propiedad para 'asociar' información a un view.

En un ejemplo real, tendríamos una clase donde guardaríamos los datos de cada elemento de la lista. Por ejemplo, una clase Planeta, con datos como el nombre, ruta y nombre de la imagen en la sd, dimensiones, velocidad de rotación,...y muchos más datos.

En el adaptador del RecyclerView tendríamos un ArrayList de la clase Planeta y no dos arrays separados como tenemos ahora.

En la lista solamente aparecería la imagen y nombre del planeta, pero nos queremos acceder a todos sus datos y no solamente a los que aparecen 'visualmente'.

- ✓ Vamos a modificar el ejemplo para emplear esta clase Planeta:

Archivo: Planeta_UD04_01_RecyclerViewCardView.java

```
package com.example.adaptadores;

import android.content.Context;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;

import androidx.annotation.NonNull;
import androidx.recyclerview.widget.RecyclerView;

import java.util.ArrayList;

public class RecyclerViewAdapter_UD04_01_RecyclerViewCardView extends RecyclerView.Adapter {

    private ArrayList<Planeta_UD04_01_RecyclerViewCardView> planetas = new ArrayList<Planeta_UD04_01_RecyclerViewCardView>();

    public RecyclerViewAdapter_UD04_01_RecyclerViewCardView(){

        planetas.add(new Planeta_UD04_01_RecyclerViewCardView(1,"MERCURIO",false,R.drawable.mercurio));
        planetas.add(new Planeta_UD04_01_RecyclerViewCardView(2,"VENUS",false,R.drawable.venus));
        planetas.add(new Planeta_UD04_01_RecyclerViewCardView(3,"TIERRA",false,R.drawable.tierra));
        planetas.add(new Planeta_UD04_01_RecyclerViewCardView(4,"JUPITER",false,R.drawable.jupiter));
        planetas.add(new Planeta_UD04_01_RecyclerViewCardView(5,"SATURNO",false,R.drawable.saturno));

    }

    @NonNull
    @Override
    public RecyclerView.ViewHolder onCreateViewHolder(@NonNull ViewGroup viewGroup, int i) {

        LayoutInflater mInflater = (LayoutInflater) viewGroup.getContext().getSystemService(Context.LAYOUT_INFLATER_SERVICE);
        //View view = (LayoutInflater.from(viewGroup.getContext()))
        View v = mInflater.inflate(R.layout.card_layout_ud04_01_recycleviewcardview,viewGroup,false);
        RecyclerView.ViewHolder viewHolder = new ViewHolder_UD04_01_RecyclerViewCardView(v);

        return viewHolder;
    }

    @Override
    public void onBindViewHolder(@NonNull RecyclerView.ViewHolder viewHolder, int i) {
        ViewHolder_UD04_01_RecyclerViewCardView viewHolderMeu = (ViewHolder_UD04_01_RecyclerViewCardView) viewHolder;
        viewHolderMeu.itemImagen.setImageResource(planetas.get(i).getFotoId());
        viewHolderMeu.itemTexto.setText(planetas.get(i).getNombre());
    }

    @Override
    public int getItemCount() {

        return planetas.size();
    }
}
```



```
}
}
```

- ✓ Modificamos la clase adaptadora y empleamos un ArrayList de objetos pertenecientes a la clase anterior.

Estos datos podrían venir de una base de datos...

Clase RecyclerViewAdapter_UD04_01_RecyclerViewCardView

```
package com.example.adaptadores;

import android.content.Context;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;

import androidx.annotation.NonNull;
import androidx.recyclerview.widget.RecyclerView;

import java.util.ArrayList;

public class RecyclerViewAdapter_UD04_01_RecyclerViewCardView extends RecyclerView.Adapter {

    private ArrayList<Planeta_UD04_01_RecyclerViewCardView> planetas = new ArrayList<Planeta_UD04_01_RecyclerViewCardView>();

    public RecyclerViewAdapter_UD04_01_RecyclerViewCardView(){

        planetas.add(new Planeta_UD04_01_RecyclerViewCardView(1,"MERCURIO",false,R.drawable.mercurio));
        planetas.add(new Planeta_UD04_01_RecyclerViewCardView(2,"VENUS",false,R.drawable.venus));
        planetas.add(new Planeta_UD04_01_RecyclerViewCardView(3,"TIERRA",false,R.drawable.tierra));
        planetas.add(new Planeta_UD04_01_RecyclerViewCardView(4,"JUPITER",false,R.drawable.jupiter));
        planetas.add(new Planeta_UD04_01_RecyclerViewCardView(5,"SATURNO",false,R.drawable.saturno));

    }

    @NonNull
    @Override
    public RecyclerView.ViewHolder onCreateViewHolder(@NonNull ViewGroup viewGroup, int i) {

        LayoutInflater mInflater = (LayoutInflater) viewGroup.getContext().getSystemService(Context.LAYOUT_INFLATER_SERVICE);
        //View view = (LayoutInflater.from(viewGroup.getContext()))
        View v = mInflater.inflate(R.layout.card_layout_ud04_01_recyclerviewcardview,viewGroup,false);
        RecyclerView.ViewHolder viewHolder = new ViewHolder_UD04_01_RecyclerViewCardView(v);

        return viewHolder;
    }

    @Override
    public void onBindViewHolder(@NonNull RecyclerView.ViewHolder viewHolder, int i) {
        ViewHolder_UD04_01_RecyclerViewCardView viewHolderMeu = (ViewHolder_UD04_01_RecyclerViewCardView) viewHolder;
        viewHolderMeu.itemImagen.setImageResource(planetas.get(i).getFotoId());
        viewHolderMeu.itemTexto.setText(planetas.get(i).getNombre());
    }

    @Override
    public int getItemCount() {
```

```

    }
    return planetas.size();
}

```

- ✓ Ahora queremos pasar de alguna forma el objeto seleccionado (y fijarse que digo objeto) al view, de tal forma que cuando pulse sobre el elemento de la lista, pueda recuperar la información completa del objeto.

Para eso hacemos uso del [método setTag\(\) y getTag\(\)](#).

Clase RecyclerViewAdapter_UD04_01_RecyclerViewCardView

```

@Override
public void onBindViewHolder(@NonNull RecyclerView.ViewHolder viewHolder, int i) {
    ViewHolder_UD04_01_RecyclerViewCardView viewHolderMeu = (ViewHolder_UD04_01_RecyclerViewCardView) view-
Holder;
    viewHolderMeu.itemView.setTag(planetas.get(i));
    viewHolderMeu.itemImagen.setImageResource(planetas.get(i).getFotoId());
    viewHolderMeu.itemTexto.setText(planetas.get(i).getNombre());
}

```

Nota: No se puede hacer en el método 'onCreateViewHolder' ya que el 'int i' no se corresponde con el índice del elemento que se va a visualizar.

Clase ViewHolder_UD04_01_RecyclerViewCardView

```

public class ViewHolder_UD04_01_RecyclerViewCardView extends RecyclerView.ViewHolder {
    public ImageView itemImagen;
    public TextView itemTexto;

    public ViewHolder_UD04_01_RecyclerViewCardView(@NonNull View itemView) {
        super(itemView);

        itemImagen = itemView.findViewById(R.id.imgImaxe_UD04_01_CardLayout);
        itemTexto = itemView.findViewById(R.id.tvTexto_UD04_01_CardView);

        itemImagen.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                Planeta_UD04_01_RecyclerViewCardView planetaSeleccionado = (Plane-
ta_UD04_01_RecyclerViewCardView) v.getTag();
                Toast.makeText(v.getContext(), "Pulsado elemento " + getAdapterPosition() + " de la lis-
ta.", Toast.LENGTH_SHORT).show();
            }
        });
    }
}

```

1.3.7. Añadiendo-Borrando-Actualizando elementos

- ✓ Se hacemos cualquiera de estas operaciones tendremos que informar al adaptador que hay cambios para que vuelva a cargar los elementos de la lista.

Para eso tendremos que llamar al [método notifyDataSetChanged\(\)](#).

El problema que tiene esta llamada es que es poco eficiente ya que reconstruye todos los view's del RecyclerView.

Normalmente se llama cuando tenemos muchas operaciones o cambios en la estructura de los elementos.

Es más eficiente llamar a los siguientes métodos:

- notifyDataSetChanged(int)
- notifyItemInserted(int)
- notifyItemRemoved(int)
- notifyItemRangeChanged(int, int)
- notifyItemRangeInserted(int, int)
- notifyItemRangeRemoved(int, int)

Podéis consultar lo que hace cada uno [en este enlace](#).

- ✓ Siguiendo con nuestro ejemplo, vamos a agregar un botón en la activity del RecyclerView (en mi caso añadí un FloatingActionButton):

Layout activity_ud04_01_recycle_view_card_view

```
<?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=".UD04_01_RecyclerViewCardView">

    <androidx.recyclerview.widget.RecyclerView
        android:id="@+id/rvwRecyclerView"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:layout_marginTop="24dp"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

    <com.google.android.material.floatingactionbutton.FloatingActionButton
        android:id="@+id/fabAdd_ud04_01_recyclecardview"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_margin="20dp"
        android:clickable="true"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:srcCompat="@android:drawable/ic_input_add" />

</androidx.constraintlayout.widget.ConstraintLayout>
```

- ✓ Ahora los datos van a venir desde la Activity y no van a estar definidos en la clase Adaptadora.

Para pasar los datos a dicha clase, haremos uso del constructor de la clase Adaptadora.

Clase RecyclerViewAdapter_UD04_01_RecyclerViewCardView

```
.....
public class RecyclerViewAdapter_UD04_01_RecyclerViewCardView extends RecyclerView.Adapter {

    private ArrayList<Planeta_UD04_01_RecyclerViewCardView> planetas = new Array-
List<Planeta_UD04_01_RecyclerViewCardView>();

    public RecyclerViewAdapter_UD04_01_RecyclerViewCardView(ArrayList<Planeta_UD04_01_RecyclerViewCardView>
planetas){
        this.planetas = planetas;
    }
    .....
}
```

- ✓ En la activity ahora es donde tendríamos que buscar los datos a una base de datos, por ejemplo, y pasarlos a la clase Adaptadora en el constructor.

Activity UD04_01_RecyclerViewCardView

```
package com.example.adaptadores;

import androidx.appcompat.app.AppCompatActivity;
import androidx.recyclerview.widget.LinearLayoutManager;
import androidx.recyclerview.widget.RecyclerView;

import android.os.Bundle;
import android.view.View;

import com.google.android.material.floatingactionbutton.FloatingActionButton;
import java.util.ArrayList;

public class UD04_01_RecyclerViewCardView extends AppCompatActivity {

    private RecyclerViewAdapter_UD04_01_RecyclerViewCardView recycleAdapter;

    private ArrayList<Planeta_UD04_01_RecyclerViewCardView> planetas;

    private void inicializarDatosPlanetas(){

        planetas = new ArrayList<>();

        planetas.add(new Planeta_UD04_01_RecyclerViewCardView(1,"MERCURIO",false,R.drawable.mercurio));
        planetas.add(new Planeta_UD04_01_RecyclerViewCardView(2,"VENUS",false,R.drawable.venus));
        planetas.add(new Planeta_UD04_01_RecyclerViewCardView(3,"TIERRA",false,R.drawable.tierra));
        planetas.add(new Planeta_UD04_01_RecyclerViewCardView(4,"JUPITER",false,R.drawable.jupiter));
        planetas.add(new Planeta_UD04_01_RecyclerViewCardView(5,"SATURNO",false,R.drawable.saturno));

    }

    private void inicializarRecyclerView(){

        recycleAdapter = new RecyclerViewAdapter_UD04_01_RecyclerViewCardView(planetas);
    }
}
```

```

RecyclerView.LayoutManager layoutManager = new LinearLayoutManager(this);
RecyclerView recyclerView = findViewById(R.id.rvwRecyclerView);
recyclerView.setLayoutManager(layoutManager);
recyclerView.setAdapter(recycleAdapter);

}

private void xestionarEventos(){

    FloatingActionButton fab = findViewById(R.id.fabAdd_ud04_01_recyclecardview);
    fab.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            planetas.add(new Planeta_UD04_01_RecyclerViewCardView(6,"URANO",false,R.drawable.urano));
            recycleAdapter.notifyItemInserted(planetas.size());
        }
    });
}

@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_u_d04_01__recycle_view_card_view);

    inicializarDatosPlanetas();
    inicializarRecyclerView();
    xestionarEventos();
}
}

```

- ✓ **Línea 36:** Ahora pasamos los datos al Adaptador mediante el constructor.
- ✓ **Línea 53:** Fijarse como ahora, cuando añadimos un nuevo planeta al ArrayList, llamamos al método notifyItemInserted, indicando la posición del elemento añadido.