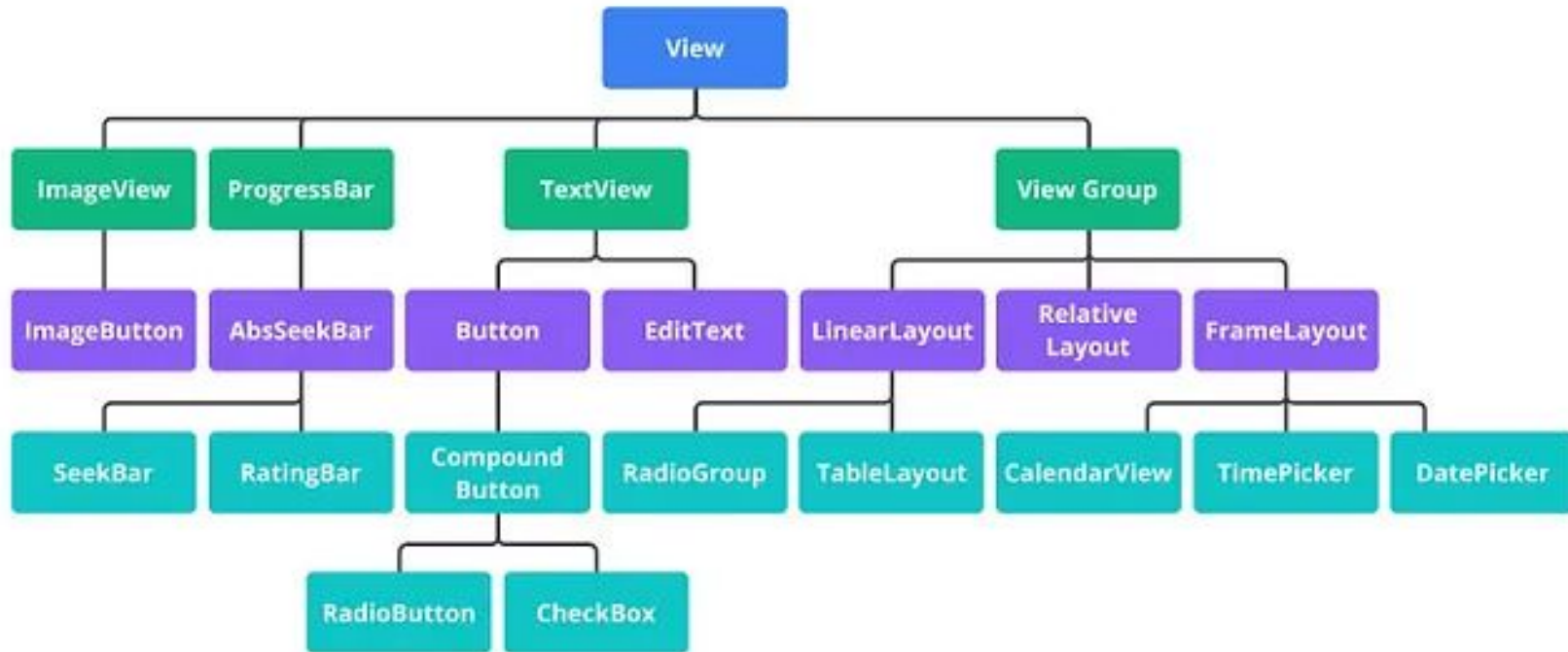


Contenedores y componentes










Conceptos: View, ViewGroup, Layout, ListView, RecyclerView

Contenedores y componentes extienden de View









Contenedores(Layouts) y componentes (Widgets)

Layout

-  ConstraintLayout
-  GridLayout
-  FrameLayout
-  LinearLayout (hor
-  LinearLayout (vert
-  RelativeLayout
-  TableLayout
-  TableRow
-  <fragment>

Container

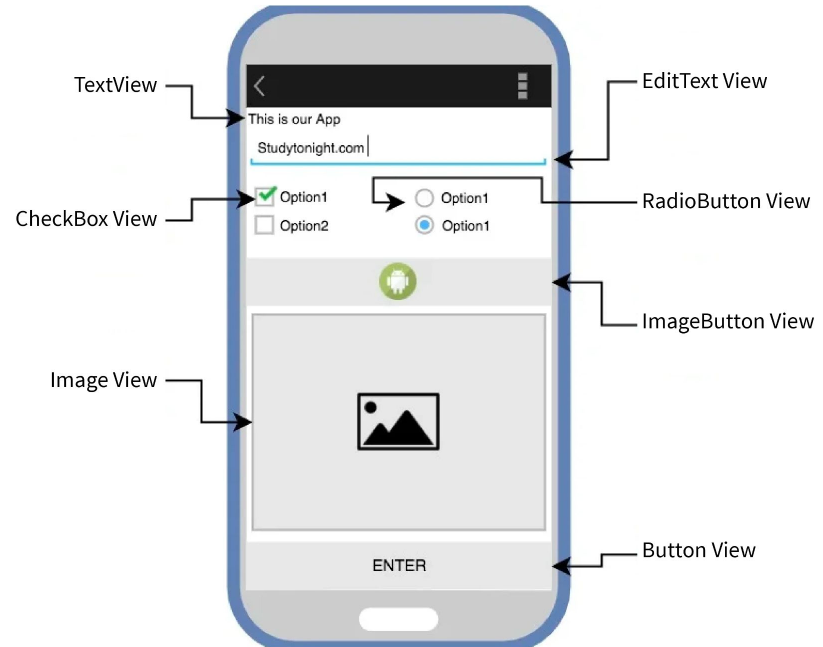
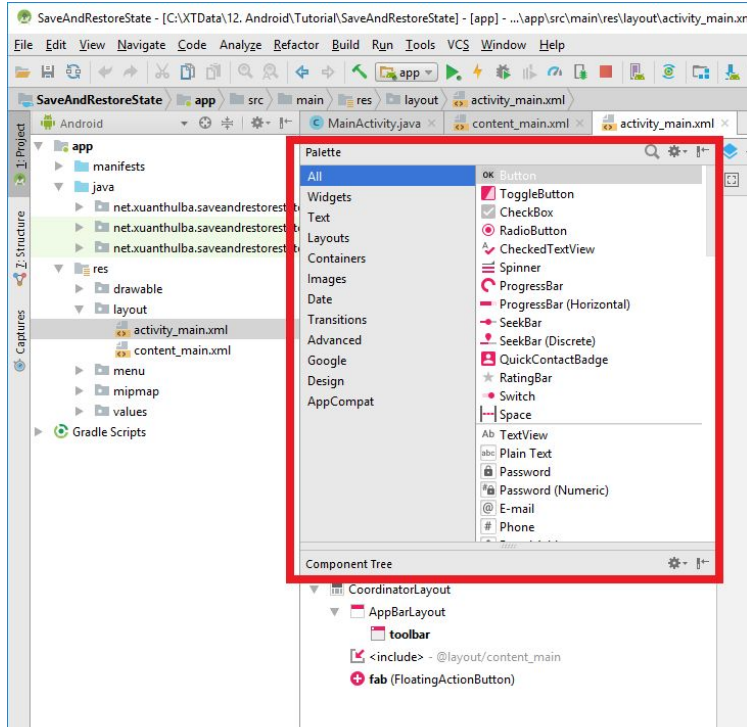
-  RadioGroup
-  ListView
-  GridView
-  ExpandableListView
-  ScrollView
-  HorizontalScrollView

-Un **layout** es un esquema o modelo que **define cómo se disponen los elementos visuales** dentro de un container.

-Un **container** es un componente que **contiene y agrupa** otros elementos de la interfaz de usuario, como botones, textos o imágenes.

Componentes(Widgets) que extienden de View

Son **elementos interactivos** individuales de la interfaz de usuario.



Contenedores(Layouts) extienden de ViewGroup

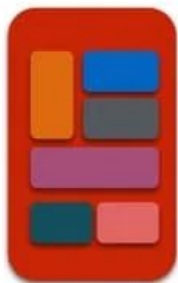
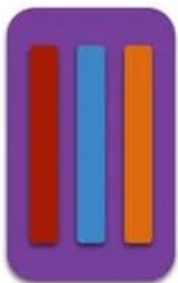
Organizan otros componentes (widgets) dentro de ellos.



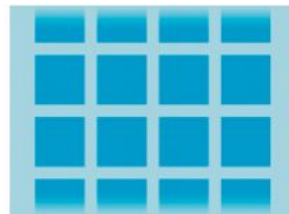
FrameLayout



LinearLayout



RelativeLayout /
ConstraintLayout



GridLayout



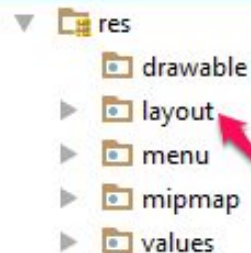
TableLayout

-LinearLayout: Ideal para organizar elementos en una fila o columna.

-FrameLayout: Para apilar elementos uno encima del otro.

-RelativeLayout: Para posicionar elementos en relación unos con otros.

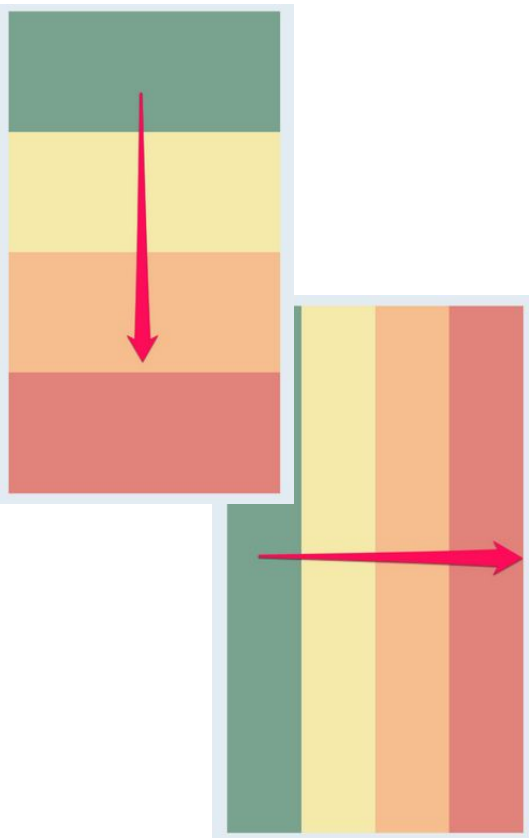
-ConstraintLayout: Para diseños complejos con muchas opciones de alineación y restricciones, mejorando la eficiencia y la simplicidad en layouts jerárquicos.



Aquí van tus layouts

ViewGroup: Es un View que puede contener otros Views (tanto componentes como contenedores).

LinearLayout



```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:padding="48dp">

    <TextView
        android:id="@+id/texto_conectar"
        android:layout_width="wrap_content"
        android:layout_height="0dp"
        android:layout_gravity="center_horizontal"
        android:layout_weight="1"
        android:text="Conectar"
        android:textAppearance="?android:attr/textAppearanceLarge" />

    <EditText
        android:id="@+id/input_usuario"
        android:layout_width="match_parent"
        android:layout_height="0dp"
        android:layout_gravity="center_horizontal"
        android:layout_weight="1"
        android:hint="Correo" />

    <EditText
        android:id="@+id/input_contrasena"
        android:layout_width="match_parent"
        android:layout_height="0dp"
        android:layout_gravity="center_horizontal"
        android:layout_weight="1"
        android:ems="10"
        android:hint="Contraseña"
        android:inputType="textPassword" />

    <Button
        android:id="@+id/boton_iniciar_sesion"
        style="?android:attr/buttonStyleSmall"
        android:layout_width="match_parent"
        android:layout_height="0dp"
        android:layout_gravity="center_horizontal"
        android:layout_weight="1"
        android:text="Iniciar Sesión" />

    <TextView
        android:id="@+id/texto_olvidaste_contrasena"
        android:layout_width="wrap_content"
        android:layout_height="0dp"
        android:layout_gravity="center_horizontal"
        android:layout_weight="1"
        android:gravity="center_vertical"
        android:text="¿Olvidaste tu contraseña?"
        android:textAppearance="?android:attr/textAppearanceMedium"
        android:textColor="#0EBAEE" />

</LinearLayout>
```

Descripción: Coloca los elementos de forma secuencial, ya sea en una dirección vertical u horizontal.

Uso: Se utiliza cuando deseas organizar componentes en una sola columna o fila. Por ejemplo, puedes colocar botones, imágenes o texto uno debajo del otro o uno al lado del otro.

Orientación: Tiene una propiedad llamada **orientation** que define si los elementos estarán organizados de manera vertical (**android:orientation="vertical"**) o horizontal (**android:orientation="horizontal"**).

Ventajas: Es fácil de usar y predecible en la disposición de elementos.

Desventajas: No es eficiente para diseños complejos, ya que puede generar muchas subjerarquías, lo que afecta el rendimiento.

FrameLayout



```
<FrameLayout 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=".ActividadPrincipal">

    <Button
        android:layout_width="match_parent"
        android:layout_height="60dp"
        android:text="Saltar"
        android:id="@+id/boton_saltar"
        android:layout_gravity="center_horizontal|bottom"/>

    <ImageView
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:id="@+id/imagen_background"
        android:layout_gravity="top|center"
        android:src="@drawable/background_frame_layout"
        android:scaleType="centerCrop" />

    <ImageView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/imagen_estadistica"
        android:layout_gravity="center"
        android:src="@drawable/ejemplo_estadistica"
        android:padding="16dp" />

</FrameLayout>
```

Descripción: Un layout simple que coloca todos los elementos uno encima del otro, es decir, apilados.

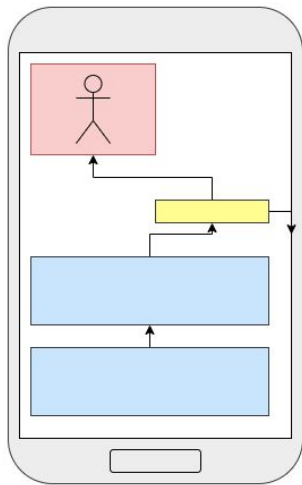
Uso: Es útil cuando tienes un único elemento o quieres superponer varios elementos, como una imagen de fondo y un texto encima.

Características: El último elemento añadido es el que se muestra arriba de los demás.

Ventajas: Muy simple y rápido para layouts sencillos.

Desventajas: No es adecuado para layouts complejos, ya que solo permite apilar elementos.

RelativeLayout



Descripción: Permite posicionar elementos en relación a otros elementos dentro del layout o en relación al contenedor padre.

Uso: Se utiliza cuando quieres un control más detallado sobre la ubicación de los componentes, como colocar un botón debajo de un texto o alinearlo a la derecha de la pantalla.

Propiedades clave: Puedes usar atributos como `layout_below`, `layout_above`, `layout_toLeftOf`, `layout_toRightOf`, etc., para definir las posiciones relativas de los elementos.

Ventajas: Es flexible y permite diseñar layouts complejos sin anidar demasiados elementos.

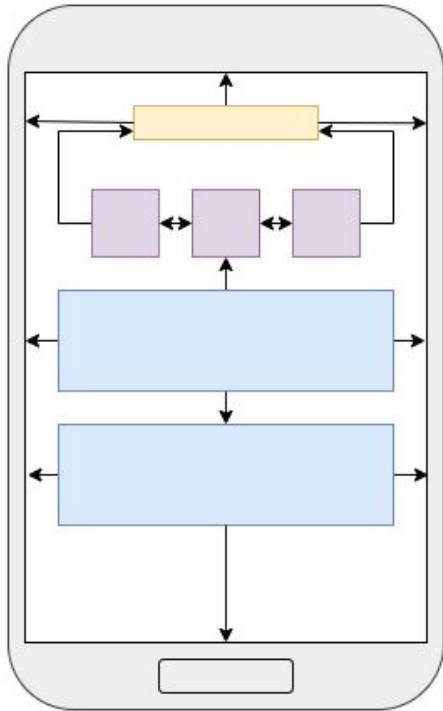
Desventajas: Si no se usa correctamente, puede resultar en jerarquías complicadas y difíciles de mantener.

```
<RelativeLayout 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" android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    android:paddingBottom="@dimen/activity_vertical_margin" tools:context=".ActividadPrincipal">

    <TextView android:text="@string/hello_world" android:layout_width="wrap_content"
        android:layout_height="wrap_content" />

</RelativeLayout>
```


ConstraintLayout



Descripción: Es un layout más flexible y potente que permite posicionar y dimensionar elementos en función de restricciones. Los elementos pueden estar anclados entre sí o a los bordes del contenedor.

Uso: Es ideal para diseños complejos, ya que permite crear layouts con menos anidaciones en comparación con [LinearLayout](#) o [RelativeLayout](#).

Propiedades clave: Utiliza restricciones (constraints) para definir la posición de los elementos. Puedes anclar componentes a los bordes del layout o entre ellos (por ejemplo, que un botón esté siempre centrado horizontalmente o a 20dp del borde superior).

Ventajas: Es muy versátil y puede reemplazar a [RelativeLayout](#) y [LinearLayout](#) en la mayoría de los casos, optimizando el rendimiento.

Desventajas: Aunque es muy flexible, puede ser más complejo de configurar correctamente.

Atributos del diseño XML

Atributo	Objeto
<code>android:id</code>	Especifica un identificador único para una vista, que se puede utilizar para referenciarlo en código Java.
<code>android:text</code>	Estace el contenido de texto de una TextView o Button.
<code>android:layout-width</code>	Especifica la anchura de una vista dentro de un diseño.
<code>android:layout-height</code>	Especifica la altura de una vista dentro de un diseño.
<code>android:layout-margin</code>	Estace los márgenes (espacio vacío) alrededor de una vista.
<code>android:layout-gravity</code>	Estaca la gravedad de una vista dentro de su diseño padre.
<code>android:padding</code>	Define el acolchado (espacio vacío) dentro de una vista.
<code>android: fondo</code>	Establece el color de fondo o dibujar para una vista.
<code>android: visibilidad</code>	Controla la visibilidad de una vista.

Unidades de Medida preferidas

dp (density-independent pixels)

- **Descripción:** Unidad de medida independiente de la densidad de pantalla. Escala automáticamente según la densidad del dispositivo.
- **Uso:** Ideal para definir tamaños de elementos de la interfaz, como márgenes, padding o tamaños de texto.
- **Equivalencia:** 1 dp \approx 1 píxel en una pantalla de 160 dpi.
- **Ejemplo:** `margin: 16dp;`

sp (scale-independent pixels)

- **Descripción:** Similar a `dp`, pero ajusta el tamaño en función de las configuraciones de accesibilidad del texto del usuario.
- **Uso:** Exclusivo para definir tamaños de texto, asegurando que el texto sea legible en cualquier dispositivo.
- **Ejemplo:** `textSize: 18sp;`

ListView vs RecyclerView

ListView

- **Qué es:** Un componente de vista utilizado para mostrar una lista de elementos en una única columna, permitiendo el desplazamiento vertical.
- **Características:**
 - Se usa para listas simples donde cada elemento tiene el mismo tipo de diseño.
 - Ofrece una interfaz básica para manejar datos, con soporte para adaptadores como `ArrayAdapter` o `CursorAdapter`.
 - Aunque es fácil de usar, tiene limitaciones de rendimiento con grandes cantidades de datos porque solo recicla las vistas visibles en pantalla.

RecyclerView

- **Qué es:** Una versión más avanzada y eficiente de `ListView`, diseñada para mostrar listas o cuadrículas de datos de manera más flexible y con mejor rendimiento.
- **Características:**
 - Introduce el patrón `ViewHolder`, que mejora el rendimiento al evitar inflar vistas repetidas veces.
 - Soporta diferentes tipos de diseños (lineales, cuadrículas, o personalizados a través de `LayoutManager`).
 - Es altamente personalizable y ofrece soporte para animaciones, decoraciones, y manejo avanzado del desplazamiento (scroll).
 - Ideal para listas grandes o dinámicas, y permite actualizar solo los elementos necesarios.

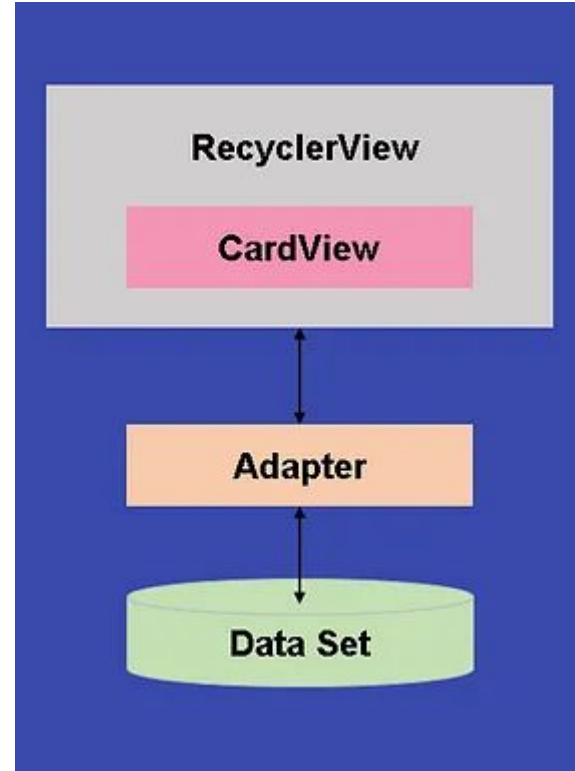
CardView

Es un contenedor con un diseño estilizado que permite mostrar datos en forma de tarjetas, ofreciendo una apariencia visual atractiva con bordes redondeados y sombras.

Características:

- No es una lista por sí misma, sino que envuelve otras vistas (como las que puedes usar en RecyclerView).
- Se utiliza principalmente para crear interfaces más modernas y elegantes, donde los elementos tienen apariencia de tarjetas.
- Es compatible con el uso de RecyclerView para mostrar listas de tarjetas.

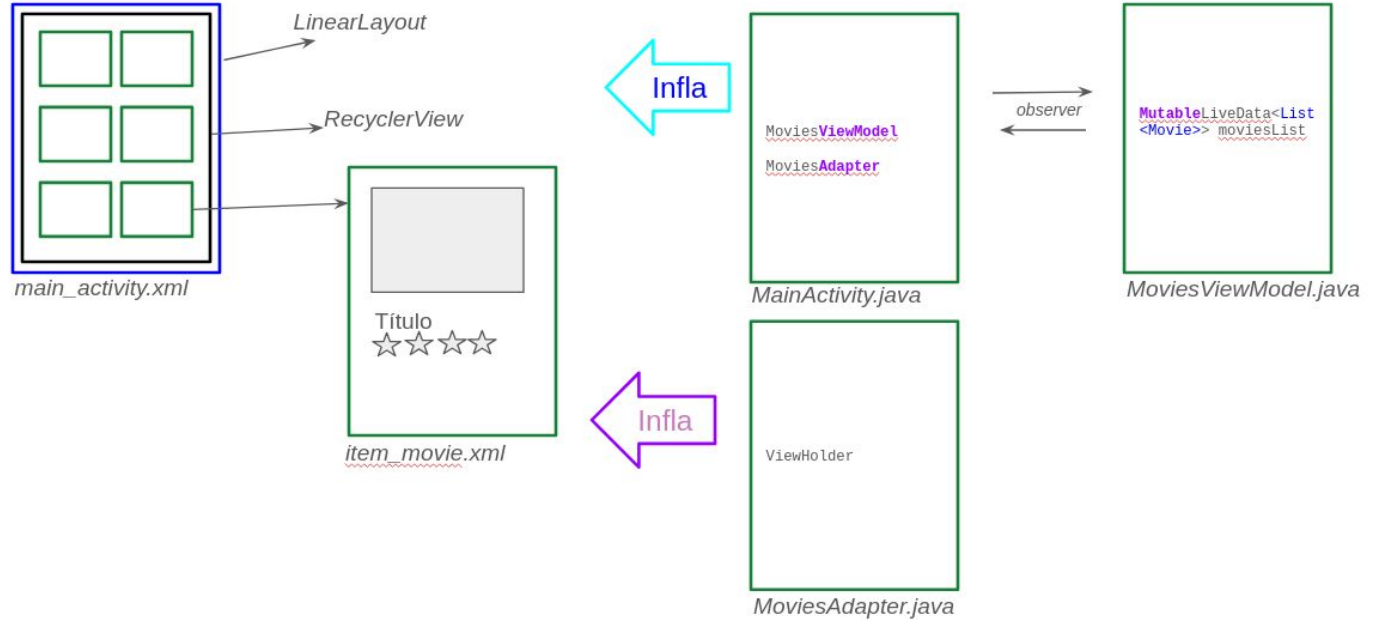
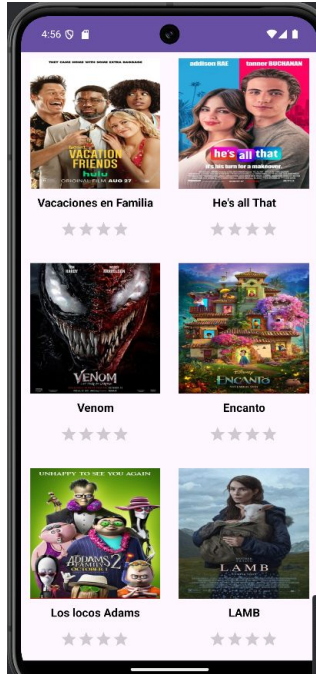
RecyclerView y CardView son dos widgets muy potentes que trabajados en conjunto generan resultados muy efectivos a la hora de mostrar una colección de datos.



ListView, RecyclerView, CardView

ListView	RecyclerView	CardView
Mostrar listas simples de datos.	Mostrar listas más complejas y de alto rendimiento.	Diseñar elementos con apariencia de tarjeta (card).
Menor rendimiento con listas largas (solo recicla vistas visibles).	Alto rendimiento (recicla eficientemente todas las vistas).	Rendimiento según el componente que envuelve.
Introducido en las primeras versiones de Android (antiguo).	Introducido en Android Lollipop (más moderno).	Introducido en Android Lollipop (más moderno).
Menos flexible, tiene limitaciones para listas complejas.	Muy flexible, soporta listas, cuadrículas y diseño avanzado.	Se usa principalmente para envoltura de vistas con estilo.
Usa <code>ArrayAdapter</code> o <code>CursorAdapter</code> .	Usa <code>RecyclerView.Adapter</code> (más personalizable).	No necesita adaptador, solo envuelve vistas.

Ejemplo usando RecyclerView



Adapter

El Adapter es el componente que se encarga de **vincular los datos con las vistas**. Proporciona una interfaz entre los datos y el RecyclerView, permitiendo que los elementos de la lista o cuadrícula se representen visualmente.

- **Crear las vistas (onCreateViewHolder):** El adaptador **crea o infla las vistas** que se mostrarán en el RecyclerView. Estas vistas se reciclan para optimizar el rendimiento, lo que ahorra memoria y recursos.
- **Vincular los datos (onBindViewHolder):** Después de que una vista ha sido creada o reciclada, el adaptador la actualiza con los datos correctos para esa posición específica.
- **Determinar la cantidad de elementos (getItemCount):** Informa al RecyclerView cuántos elementos se deben mostrar, lo que le permite saber el tamaño de la lista o cuadrícula.

El **Adapter maneja cómo los datos se representan en las vistas**, haciendo la conversión entre los datos y la interfaz de usuario.

La función del ViewHolder

El ViewHolder actúa como un contenedor para las vistas que forman parte de cada ítem en el RecyclerView.

Para nuestro ejemplo, tenemos una lista que incluye un TextView y un ImageView, el ViewHolder guardará estas referencias para reutilizarlas sin necesidad de buscarlas nuevamente cuando el usuario haga scroll.

La clase ViewHolder se incluye dentro del adaptador por conveniencia, ya que ambos (Adapter y ViewHolder) están íntimamente relacionados en la funcionalidad del RecyclerView.

El Adapter necesita saber cómo actualizar las vistas del ViewHolder con los datos correspondientes.

Función del **LayoutManager**

Controla la disposición y organización de los elementos en el RecyclerView.

- **Funciones Clave:**
 - Organiza elementos (vertical, horizontal, cuadrícula).
 - Maneja el espaciado y alineación.
 - Mejora el rendimiento al reciclar vistas.

Tipos Comunes:

- **LinearLayoutManager:** Lista vertical u horizontal.
- **GridLayoutManager:** Disposición en cuadrícula.
- **StaggeredGridLayoutManager:** Elementos de tamaños variados.

Interacción: Trabaja junto con el Adapter, que proporciona los datos a mostrar.

Configurar build.gradle.kts

```
buildFeatures {  
    viewBinding = true  
}  
  
dependencies {  
    implementation("androidx.appcompat:appcompat:1.7.0")  
    implementation("com.google.android.material:material:1.12.0")  
    implementation("androidx.activity:activity-ktx:1.7.2")  
    implementation("androidx.constraintlayout:constraintlayout:2.1.4")  
    implementation("androidx.cardview:cardview:1.0.0") // Usar CardView de AndroidX  
    implementation("androidx.recyclerview:recyclerview:1.2.1") // Usar RecyclerView de AndroidX  
  
    // Dependencias de testing  
    testImplementation("junit:junit:4.13.2")  
    androidTestImplementation("androidx.test.ext:junit:1.1.5")  
    androidTestImplementation("androidx.test.espresso:espresso-core:3.5.1")  
  
    // Glide para manejar imágenes  
    implementation("com.github.bumptech.glide:glide:4.12.0")  
    annotationProcessor("com.github.bumptech.glide:compiler:4.12.0")  
}
```

Glide nos permite Cargar imágenes desde la red, el almacenamiento local o recursos de la aplicación.

Mostrar las imágenes en ImageView o vistas similares.

Las Clases xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout 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:id="@+id/main"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    tools:context=".MainActivity">

    <androidx.recyclerview.widget.RecyclerView
        android:id="@+id/recycler"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        app:layoutManager="androidx.recyclerview.widget.GridLayoutManager"
        app:spanCount="2"
        tools:listitem="@layout/item_movie" />
</LinearLayout>
```

main_activity.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout 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="wrap_content"
    android:orientation="vertical"
    android:padding="8dp">

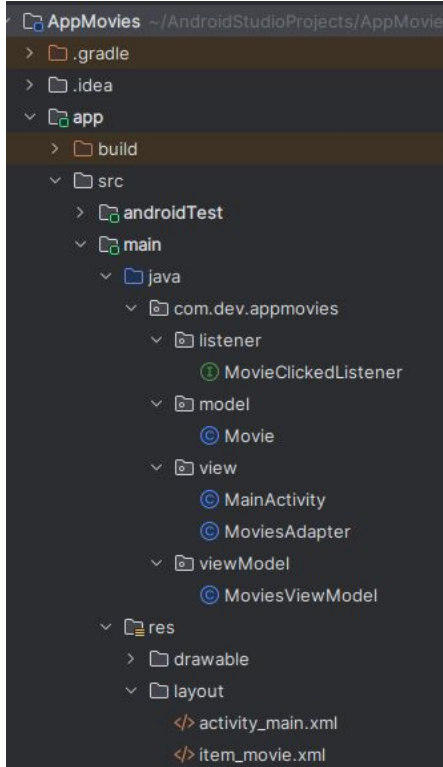
    <ImageView
        android:id="@+id/movie_cover"
        android:layout_width="match_parent"
        android:layout_height="180dp"
        android:scaleType="fitCenter"
        app:srcCompat="@drawable/ic_launcher_background"
        tools:srcCompat="@drawable/ic_launcher_foreground" />

    <TextView
        android:id="@+id/movie_title"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:paddingTop="8dp"
        android:textColor="@android:color/black"
        android:textSize="16sp"
        android:textStyle="bold"
        android:gravity="center" />

    <RatingBar
        android:id="@+id/movie_rating"
        style="?android:ratingBarStyleSmall"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:isIndicator="true"
        android:rating="4"
        android:stepSize="0.5"
        android:layout_gravity="center" />
</LinearLayout>
```

item_movie.xml

La clase Movie.java y una interfáz



```
public class Movie { 12 usages
    private String title, cover; 3 usages
    public Movie(String title, String cover) {...}
    public String getTitle() { return title; }
    public void setTitle(String title) { this.title = title; }
    public String getCover() { return cover; }
    public void setCover(String cover) { this.cover = cover; }
}
```

El adapter dispone de una `List<Movie> movies`;

```
public interface MovieClickListener{
    void onMovieClicked(Movie movie);
}
```

MoviesViewModel.java

```
public class MoviesViewModel extends ViewModel { 4 usages
    private MutableLiveData<List<Movie>> moviesList; 3 usages
    public MoviesViewModel() { no usages
        moviesList = new MutableLiveData<>();
        loadMovies();
    }
    private void loadMovies() { 1 usage
        List<Movie> movies = new ArrayList<>();
        movies.add(new Movie( title: "Vacaciones en Familia", cover: "https://via.assets.so/movie.png?id=1&q=95&w=360&h=360&fit=fill", estrellas: 3.0f));
        movies.add(new Movie( title: "He's all That", cover: "https://via.assets.so/movie.png?id=2&q=95&w=360&h=360&fit=fill", estrellas: 4.0f));
        movies.add(new Movie( title: "Venom", cover: "https://via.assets.so/movie.png?id=3&q=95&w=360&h=360&fit=fill", estrellas: 2.5f));
        movies.add(new Movie( title: "Encanto", cover: "https://via.assets.so/movie.png?id=4&q=95&w=360&h=360&fit=fill", estrellas: 4.0f));
        movies.add(new Movie( title: "Los locos Adams", cover: "https://via.assets.so/movie.png?id=5&q=95&w=360&h=360&fit=fill", estrellas: 4.0f));
        movies.add(new Movie( title: "LAMB", cover: "https://via.assets.so/movie.png?id=6&q=95&w=360&h=360&fit=fill", estrellas: 2.0f));
        moviesList.setValue(movies);
    }
    public LiveData<List<Movie>> getMovies() { return moviesList; }
}
```

[imágenes para el ejemplo](#)

MainActivity.java

```
public class MainActivity extends AppCompatActivity {  
    private ActivityMainBinding binding; 4 usages  
    private MoviesViewModel moviesViewModel; 2 usages  
  
    @Override  
    protected void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        binding = ActivityMainBinding.inflate(getLayoutInflater());  
        setContentView(binding.getRoot());  
  
        // Inicializar el ViewModel con un provider  
        moviesViewModel = new ViewModelProvider( owner: this).get(MoviesViewModel.class);  
        // Configurar el RecyclerView  
        binding.recycler.setLayoutManager(new GridLayoutManager( context: this, spanCount: 2));  
        // Observar los cambios en la lista de películas  
        moviesViewModel.getMovies().observe( owner: this, new Observer<List<Movie>>() {  
            @Override  
            public void onChanged(List<Movie> movies) {  
                if (movies != null && !movies.isEmpty()) {  
                    Log.d( tag: "MoviesList", msg: "Mi lista: " + movies.size());  
                    // Configurar el adaptador del RecyclerView  
                    MoviesAdapter adapter = new MoviesAdapter(movies, new MovieClickListener() {  
                        @Override 1 usage  
                        public void onMovieClicked(Movie movie) {  
                            Toast.makeText( context: MainActivity.this, text: "Eliges: " + movie.getTitle(), Toast.LENGTH_LONG).show();  
                        }  
                    }, context: MainActivity.this);  
                    binding.recycler.setAdapter(adapter);  
                } else {  
                    Log.d( tag: "MoviesList", msg: "Lista vacía");  
                }  
            }  
        });  
    }  
}
```

LayoutManager

Observer

Adapter

Permiso en manifest.xml para internet

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
  xmlns:tools="http://schemas.android.com/tools">
  <uses-permission android:name="android.permission.INTERNET"/>
  <application
    android:allowBackup="true"
    android:dataExtractionRules="@xml/data_extraction_rules"
    android:fullBackupContent="@xml/backup_rules"
    android:icon="@mipmap/ic_launcher"
    android:label="@string/app_name"
    android:roundIcon="@mipmap/ic_launcher_round"
    android:supportsRtl="true"
    android:theme="@style/Theme.AppMovies"
    tools:targetApi="31">
    <activity
      android:name=".MainActivity"
      android:exported="true">
      <intent-filter>
        <action android:name="android.intent.action.MAIN" />

        <category android:name="android.intent.category.LAUNCHER" />
      </intent-filter>
    </activity>
  </application>
</manifest>
```

<https://via.assets.so>

Adapter

```
public class MoviesAdapter extends RecyclerView.Adapter<MoviesAdapter.ViewHolder> { 4 usages
    private List<Movie> movies; 3 usages
    private MovieClickListener movieClickListener; 2 usages
    private Context context; 2 usages
    public MoviesAdapter(List<Movie> movies, MovieClickListener movieClickListener, Context context) { 1 usage
        this.movies = movies;
        this.movieClickListener = movieClickListener;
        this.context = context;
    }
    @NonNull
    @Override
    public ViewHolder onCreateViewHolder(@NonNull ViewGroup parent, int viewType) {
        ItemMovieBinding binding = ItemMovieBinding.inflate(LayoutInflater.from(parent.getContext()), parent, attachToParent: false);
        return new ViewHolder(binding);
    }
    @Override
    public void onBindViewHolder(@NonNull ViewHolder holder, int position) {
        Movie movie = movies.get(position);
        holder.bind(movie);
    }
    @Override
    public int getItemCount() {
        return movies.size();
    }
    // Clase interna ViewHolder
    class ViewHolder extends RecyclerView.ViewHolder { 4 usages
        private final ItemMovieBinding binding; 5 usages
        public ViewHolder(@NonNull ItemMovieBinding binding) { 1 usage
            super(binding.getRoot());
            this.binding = binding;
        }
        // Método para enlazar datos con las vistas
        public void bind(Movie movie) { 1 usage
            binding.movieTitle.setText(movie.getTitle());
            Glide.with(context).load(movie.getCover()).into(binding.movieCover);
            binding.movieRating.setRating(movie.getEstrellas());
            binding.movieCover.setOnClickListener(v -> movieClickListener.onMovieClicked(movie));
        }
    }
}
```

El **ViewHolder** es una clase que se utiliza dentro de un **Adapter** en componentes como **RecyclerView**. Su principal función es almacenar referencias a las vistas que se muestran en cada fila o elemento de la lista, de manera que se puedan reutilizar eficientemente cuando se desplaza el contenido en pantalla.