RecyclerView

Es una clase de vista avanzada y flexible en Android que es parte del Android Support Library y se utiliza para mostrar grandes conjuntos de datos que se pueden desplazar muy eficientemente manteniendo una cantidad limitada de vistas. Es una mejora y más avanzada en comparación con la antigua ListView.

Características y funcionalidades

- Reutiliza las vistas que ya no son visibles para el usuario al desplazarse. Esto significa que no crea una nueva vista para cada elemento de datos; en su lugar, recicla las vistas que ya no están en la pantalla y las rellena con nuevos datos.
- Requiere un LayoutManager para gestionar la disposición de sus elementos. Esto determina si los elementos se organizan en una lista vertical u horizontal (LinearLayoutManager), en una cuadrícula (GridLayoutManager), o de alguna otra manera personalizada (StaggeredGridLayoutManager o tu propio LayoutManager).
- Necesita un adaptador, una clase que extiende RecyclerView.Adapter, para proporcionar vistas y enlazar datos a esas vistas. Dentro del adaptador, se utiliza un ViewHolder para mantener referencias a las vistas relevantes dentro de cada elemento de la lista para que no tengan que ser buscadas cada vez que se dibuja un elemento.
- Viene con animaciones predeterminadas para operaciones comunes, como agregar o eliminar elementos. También puedes proporcionar tus propias animaciones personalizadas.
- Soporta la adición de gestores de toques sin necesidad de una implementación personalizada complicada, facilitando la detección de clics o gestos en los elementos.

Ejemplo explicado del uso de RecyclerView

En este ejemplo vamos a crear una aplicación de reproductor de música.

Aplicación

Esta clase se utiliza para inicializar datos globales que se requieren en toda la aplicación.

1. Es importante tener en cuenta que esta clase extiende de Application asi que puede ser usada para mantener un estado global y realizar inicializaciones cuando la aplicación se crea.
2. El método onCreate se sobrescribe del Application y se llama cuando la aplicación se está creando. Aquí es donde se inicializan los recursos que se necesitan para toda la aplicación.
3. Dentro del método onCreate, se inicializa el vectorCanciones con un conjunto de canciones de ejemplo. Se asume que Cancion.ejemploCanciones() es un método estático que devuelve un Vector de objetos Cancion.

```
public class Aplicacion extends Application {
    private Vector<Cancion> vectorCanciones;
    private AdaptadorCanciones adaptador;

    @Override
    public void onCreate() {
        super.onCreate();
```

```
        vectorCanciones = Cancion.ejemploCanciones();
        adaptador = new AdaptadorCanciones(this, vectorCanciones); //crea una
instancia de AdaptadorCanciones, pasando el contexto de la aplicación y el Vector
de canciones para inicializar el adaptador.
    }

    public AdaptadorCanciones getAdaptador() {
        return adaptador;
    }

    public Vector<Cancion> getVectorCanciones() {
        return vectorCanciones;
    }
}
```

Recuerda que deberás definir esta clase en tu archivo AndroidManifest.xml para indicarle al sistema que utilice tu clase personalizada como la clase de aplicación al iniciar la aplicación.

```
android:name=".Aplicacion"
```

Canción

Esta clase es un modelo de datos que representa una canción con atributos como título, autor, recurso de imagen, URL de audio, género y si es una novedad o si ha sido escuchada.

```
public class Cancion {
    public String titulo;
    public String autor;
    public int recursoImagen;
    public String urlAudio;
    public String genero;
    public Boolean novedad;
    public Boolean escuchado;

    public final static String G_TODOS = "Todos los géneros";
    public final static String G_ROCK = "Rock, R&R y Metal";
    public final static String G_CLASICO = "Música clásica";
    public final static String G_INDIE = "Género indie";

    public final static String[] G_ARRAY = new String []{G_TODOS, G_ROCK,
G_CLASICO, G_INDIE};

    public Cancion(String titulo, String autor, int recursoImagen,
                   String urlAudio, String genero, Boolean novedad, Boolean
escuchado) {
        this.titulo = titulo;
        this.autor = autor;
        this.recursoImagen = recursoImagen;
        this.urlAudio = urlAudio;
```

```
        this.genero = genero;
        this.novedad = novedad;
        this.escuchado = escuchado;
    }


    //Este método estático dentro de la clase Cancion genera un conjunto de
canciones de ejemplo para usar en la aplicación.
    public static Vector<Cancion> ejemploCanciones() {
        final String SERVIDOR = "https://audionautix.com/Music/";
        Vector<Cancion> canciones = new Vector<Cancion>();
        canciones.add(new Cancion("Falling Sky", "Jason", R.drawable.falling,
                                  SERVIDOR + "FallingSky.mp3", Cancion.G_INDI,
false, false));
        // Añadir más canciones aquí si es necesario
        return canciones;
    }
}
```

AdaptadorCanciones y ViewHolder

Esta clase es un adaptador para RecyclerView que gestiona un conjunto de datos de tipo Cancion y los presenta en la RecyclerView. El patrón ViewHolder es una práctica recomendada en Android para el manejo eficiente de listas, y colocar el ViewHolder dentro del adaptador forma parte de este patrón para garantizar un código bien organizado, de alto rendimiento y fácil de mantener.

```
public class AdaptadorCanciones extends
RecyclerView.Adapter<AdaptadorCanciones.ViewHolder> {
    private LayoutInflater inflador;
    protected Vector<Cancion> vectorCanciones;
    private Context contexto;
    private View.OnClickListener onClickListener;

    public AdaptadorCanciones(Context contexto, Vector<Cancion> vectorCanciones) {
        inflador = (LayoutInflater)
contexto.getSystemService(Context.LAYOUT_INFLATER_SERVICE);
        this.vectorCanciones = vectorCanciones;
        this.contexto = contexto;
    }

    // Creación de nuevos views (invocados por el layout manager)
    @Override
    public ViewHolder onCreateViewHolder(ViewGroup parent, int viewType) {
        View view = inflador.inflate(R.layout.elemento_selector, parent, false);
        view.setOnClickListener(onClickListener);
        return new ViewHolder(view);
    }
    // Reemplazando el contenido de un view (invocado por el layout manager)
    @Override
    public void onBindViewHolder(ViewHolder holder, int position) {
        Cancion cancion = vectorCanciones.elementAt(position);
```

```
        holder.portada.setImageResource(cancion.recursoImagen);
        holder.titulo.setText(cancion.titulo);
    }

    // Regresando el tamaño de tu dataset (invocado por el layout manager)
    @Override
    public int getItemCount() {
        return vectorCanciones.size();
    }

    // Método para definir el listener para los elementos
    public void setOnItemClickListener(View.OnClickListener onClickListener) {
        this.onClickListener = onClickListener;
    }

    // Proporciona una referencia a las vistas para cada elemento de datos
    // Los elementos complejos pueden necesitar más de una vista por ítem, y
    // proporcionas acceso a todas las vistas para un ítem en un view holder
    public static class ViewHolder extends RecyclerView.ViewHolder {
        // Cada ítem del dato puede incluir más de una vista
        public ImageView portada;
        public TextView titulo;

        // Constructor del ViewHolder, donde se obtienen las referencias a las
vistas
        public ViewHolder(View itemView) {
            super(itemView);
            portada = itemView.findViewById(R.id.portada);
            titulo = itemView.findViewById(R.id.titulo);
        }
    }
}
```

RecyclerView en MainActivity

MainActivity es donde se inicializa y configura la RecyclerView con el AdaptadorCanciones.

```
public class MainActivity extends AppCompatActivity {
    private RecyclerView recyclerView;
    private RecyclerView.LayoutManager layoutManager;

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

        Aplicacion app = (Aplicacion) getApplication();
        recyclerView = (RecyclerView) findViewById(R.id.recycler_view);
        recyclerView.setAdapter(app.getAdaptador());
        layoutManager = new LinearLayoutManager(this);
        recyclerView.setLayoutManager(layoutManager);
```

```
        app.getAdaptador().setOnItemClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                int itemPosition = recyclerView.getChildLayoutPosition(view);
                Cancion cancion =
app.getVectorCanciones().elementAt(itemPosition);
                Toast.makeText(MainActivity.this, "Seleccionado: " +
cancion.titulo, Toast.LENGTH_SHORT).show();
            }
        });
    }
}
```