RECYCLERVIEW

Integrantes

Sara Martín, Javier Aragoneses, Sergio Muñoz, Marta Hernández

Introducción

En el desarrollo de aplicaciones es necesario conocer cuáles son las diferentes vistas que nos provee un sistema operativo que, además, nos sirven para representar adecuadamente el contenido que queremos presentarle al usuario en la interfaz.

Por ello, en este documento, queremos explorar en detalle la funcionalidad de RecyclerView en el entorno de desarrollo de Android Studio. Además de explicar su importancia, utilidad y proporcionar métodos detallados para su implementación, proporcionaremos una comprensión clara de cómo RecyclerView ha revolucionado la forma en que manejamos y mostramos datos en aplicaciones Android.

RecyclerView recicla elementos individuales, además de mejorar el rendimiento y la capacidad de respuesta de tu app.

¿Qué es RecyclerView ?

El RecyclerView en Android es un componente gráfico que facilita la presentación de listas de elementos en una aplicación. A diferencia de las antiguas listas en Android, los RecyclerView ofrecen un rendimiento y fluidez significativamente mejorados.
Esto se logra mediante la optimización de vistas, por lo que solo se muestran en pantalla los elementos visibles para el usuario en un momento dado, evitando la sobrecarga innecesaria de recursos de computación.

Este componente es parte de la categoría de View Group en Android, actuando como un contenedor de otras vistas. Se utiliza para listados de datos cuyo tamaño no se conoce de antemano, ya que los datos pueden generarse dinámicamente durante la ejecución de la aplicación

Además, el RecyclerView permite la reutilización de vistas, lo que significa que las vistas que ya no son visibles se pueden reciclar y utilizar para mostrar nuevos datos en lugar de crear nuevas vistas desde cero. Esto mejora significativamente el rendimiento de la aplicación y la experiencia del usuario al reducir el consumo de memoria y optimizar la velocidad de desplazamiento.

RecyclerView en Android ofrece una manera eficiente y dinámica de manejar listas de elementos, mejorando el rendimiento y la fluidez de las aplicaciones al mostrar solo los elementos necesarios en pantalla y al reutilizar vistas para conservar recursos del sistema.

Clases clave:

Varias clases trabajan juntas para compilar tu lista dinámica.

- RecyclerView es el que contiene las vistas correspondientes a tus datos.Por lo que agregas RecyclerView a tu diseño de la misma manera en que agregarías cualquier otro elemento de la interfaz del usuario.
- Cada elemento individual de la lista está definido por un objeto contenedor. Cuando lo crea, este contenedor no tiene datos. Al crearlo,RecyclerView lo vincula a sus datos. Para definir el contenedor, debes extender RecyclerView.ViewHolder.
- El RecyclerView solicita vistas y las vincula a sus datos. Para definir el adaptador, extiende RecyclerView.Adapter.
- El administrador de diseño organiza los elementos individuales de tu lista. Puedes usar uno de los administradores de diseño proporcionados por la biblioteca RecyclerView o puedes definir el tuyo. Todos los administradores de diseño se basan en la clase LayoutManager de la biblioteca.

La clase LayoutManager proporciona tres tipos de LayoutManagers integrados:

- LinearLayoutManager: se utiliza para mostrar la lista horizontal y vertical
- GridLayoutManager: se utiliza para mostrar elementos en forma de cuadrículas
- StaggeredGridLayoutManager: se utiliza para mostrar elementos en forma de cuadrículas escalonadas

Funcionalidades de RecyclerView

- Reutilización de Vistas: RecyclerView recicla vistas para ahorrar recursos y mejorar el rendimiento.
- Manejo Eficiente de Datos: Capacidad para gestionar conjuntos de datos sin sacrificar rendimiento.
- Flexibilidad de Diseño: Muestra datos en diferentes formatos usando Layout Managers (listas, cuadrículas, etc.).
- Gestos de Deslizamiento y Arrastre: Permite acciones como eliminar o reorganizar elementos mediante gestos intuitivos.
- Divisores y Decoraciones: Mejora la presentación visual con divisores y decoraciones entre elementos.
- Escucha de Eventos: Maneja eventos de clic y largo clic en elementos para interacciones del usuario.
- Selección y Resaltado de Elementos: Implementa fácilmente la selección y resaltado visual de elementos en la lista.

Implementación en Android Studio

Si vas a usar RecyclerView, debes realizar algunas acciones. Se tratarán en detalle en las siguientes secciones.

- En primer lugar, decide cómo será la lista o cuadrícula. Por lo general, podrás usar uno de los administradores de diseño estándar de la biblioteca RecyclerView.
- Diseña el aspecto y el comportamiento de cada elemento de la lista. En función de este diseño, extiende la clase ViewHolder. Tu versión de ViewHolder proporciona todas las funcionalidades para tus elementos de lista. El contenedor de vistas es un wrapper alrededor de una View, y RecyclerView la administra.
- Define el Adapter que asocia tus datos con las vistas del ViewHolder.

Puedes adaptar los elementos de RecyclerView para satisfacer tus necesidades específicas. Las clases estándar que se utilizan para crear listas dinámicas con RecyclerView ofrecen la funcionalidad básica necesaria para la mayoría de los desarrolladores.
En la mayoría de los casos, la personalización se limita al diseño de la vista para cada elemento de la lista y a escribir el código para actualizar esas vistas con los datos correspondientes. Sin embargo, si tu aplicación tiene requisitos particulares, tienes la flexibilidad de modificar el comportamiento estándar de varias maneras.

Opciones de personalización:

- Modificar el diseño:

  El RecyclerView utiliza un administrador de diseño para posicionar los elementos en pantalla y decidir cuándo reciclar las vistas que ya no son visibles. Puedes elegir entre tres administradores de diseño estándar: LinearLayoutManager (para listas unidimensionales), GridLayoutManager (para cuadrículas bidimensionales) y StaggeredGridLayoutManager (para cuadrículas con columnas ligeramente desplazadas).

  Además, puedes crear tu propio administrador de diseño personalizado extendiendo la clase abstracta RecyclerView.LayoutManager.

- Agregar animaciones a los elementos:

  Cuando un elemento cambia, RecyclerView utiliza un animador para modificar su apariencia. De forma predeterminada, se utiliza DefaultItemAnimator.

Sin embargo, puedes proporcionar tus propias animaciones personalizadas creando un objeto animador que extienda RecyclerView.ItemAnimator.

- Habilitar la Selección de Elementos de Listas:

  Para permitir que los usuarios seleccionen elementos en un RecyclerView, puedes utilizar la biblioteca recyclerview-selection.

  Para habilitar la selección, necesitas crear un ItemKeyProvider que determine el tipo de clave de selección que se usará (Parcelable, String o Long). Luego, implementa ItemDetailsLookup para permitir que la biblioteca acceda a información sobre los elementos de RecyclerView.

  Debes actualizar las vistas de los elementos para reflejar si han sido seleccionados o no. Además, puedes usar ActionMode para proporcionar herramientas al usuario para interactuar con los elementos seleccionados.

- Realizar Acciones Secundarias Interpretadas:

  La biblioteca de selección también permite interpretar acciones secundarias, como activar un elemento al presionarlo o realizar un arrastrar y soltar con elementos seleccionados. Para manejar estas acciones, necesitas registrar un objeto de escucha correspondiente.

- Reunir Todo con SelectionTracker.Builder:

  Finalmente, para utilizar la funcionalidad de selección en un RecyclerView, debes crear una instancia de SelectionTracker usando SelectionTracker.Builder. Debes proporcionar el mismo objeto RecyclerView.Adapter que se utilizó para inicializar el RecyclerView. También es necesario incluir la selección en los eventos del ciclo de vida de la actividad para mantener el estado de selección persistente.

Optimización y Mejores Prácticas

Al utilizar RecyclerView, se pueden considerar estrategias para optimizar su rendimiento. Algunas prácticas recomendadas incluyen:

- Uso Eficiente de ViewHolder: Aprovecha al máximo el ViewHolder para minimizar la sobrecarga en la creación y actualización de vistas.
- Carga Diferida de Datos: Implementa la carga diferida de datos para evitar la carga masiva de información al iniciar la aplicación, mejorando los tiempos de respuesta.

- Implementación de Administradores de Diseño Personalizados: En ciertos casos, crear administradores de diseño personalizados puede ser beneficioso para adaptarse a requerimientos específicos de diseño.

Demo de una aplicación

Para crear una demo utilizando el RecyclerView hemos pensado en hacer una aplicación donde podamos usar una lista de acciones u objetos que en cualquier momento podamos editar para añadir o quitar de la lista. De esa manera estaremos utilizando lo principal a aprender sobre este contenido. La creación, la implementación y sus diferentes formas de uso.
En este ejemplo hemos pensado en hacer una aplicación para apuntar las actividades que un usuario debe de hacer en su dia y con la finalidad que al final del dia ese mismo usuario vea cuales son las actividades que ha completado.
Para empezar la creación de está aplicación empezamos buscando ejemplos ya creados y ver su funcionalidad al completo en una aplicación funcional
En este caso enseñaremos la implementación ya explicada anteriormente con un ejemplo utilizando de un RecyclerView para mostrar una lista de actividades en dos secciones:

Una para actividades pendientes (recyclerViewActividades).

Una para actividades completadas (recyclerViewCompletadas).

Esta clase utiliza RecyclerView junto con adaptadores personalizados para gestionar listas de actividades pendientes y completadas. La interfaz de usuario se actualiza dinámicamente cuando se agregan nuevas actividades, se marcan como completadas o se eliminan.

Aquí hay una explicación de las principales funciones y el flujo de la clase:

Importación de las librerías:

```
import androidx.recyclerview.widget.LinearLayoutManager;
import androidx.recyclerview.widget.RecyclerView;
```

Declaración de Variables:

recyclerViewActividades y recyclerViewCompletadas: Objetos RecyclerView que se utilizan para mostrar las listas de actividades.

```
3 usages
private RecyclerView recyclerViewActividades;
3 usages
private RecyclerView recyclerViewCompletadas;
```

adapterActividades y adapterCompletadas: Adaptadores personalizados (MyAdapter) para vincular los datos de las listas de actividades con los elementos de la interfaz de usuario.

```
private MyAdapter adapterActividades;
3 usages
private MyAdapter adapterCompletadas;
```

listaActividades y listaCompletadas: Listas que contienen objetos de la clase Actividad para las actividades pendientes y completadas, respectivamente.

```
private List<Actividad> listaActividades;
4 usages
private List<Actividad> listaCompletadas;
```

Configuración Inicial:

Se establece el diseño de la actividad utilizando setContentView.
Se obtienen las referencias de los RecyclerView mediante findViewById.
Se configura un LinearLayoutManager para cada RecyclerView. Este administrador de diseño manejará la disposición lineal de los elementos en la lista.
Inicialización de Adaptadores:

Se crean instancias de los adaptadores (MyAdapter) y se asocian con las listas de actividades correspondientes.
Se establece un listener (setOnActividadCheckedChangeListener) para el adaptador de actividades pendientes, de modo que cuando una actividad se marca como completada, se llame al método completarActividad().
Configuración de Adaptadores en RecyclerViews:

Se establecen los adaptadores en los RecyclerView mediante setAdapter.
Botón para Nueva Actividad:

Se configura un botón (buttonNuevaActividad) para abrir un cuadro de diálogo (AlertDialog) al hacer clic. Este cuadro de diálogo permite al usuario ingresar el nombre de una nueva actividad.

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

    recyclerViewActividades = findViewById(R.id.recyclerView);
    recyclerViewCompletadas = findViewById(R.id.recyclerView_eliminadas);

    recyclerViewActividades.setLayoutManager(new LinearLayoutManager( context: this));
    recyclerViewCompletadas.setLayoutManager(new LinearLayoutManager( context: this));

    listaActividades = new ArrayList<>();
    listaCompletadas = new ArrayList<>();

    adapterActividades = new MyAdapter(listaActividades);
    adapterCompletadas = new MyAdapter(listaCompletadas);

    adapterActividades.setOnActividadCheckedChangeListener((position, isChecked) -> {
        if (isChecked) {
            completarActividad(position);
        }
    });

    recyclerViewActividades.setAdapter(adapterActividades);
    recyclerViewCompletadas.setAdapter(adapterCompletadas);

    Button buttonNuevaActividad = findViewById(R.id.button_nuevaactividad);
    buttonNuevaActividad.setOnClickListener(v -> solicitarNuevaActividad());

    inicializarActividades();
}
```

Método solicitarNuevaActividad():

Se crea un cuadro de diálogo que solicita al usuario ingresar el nombre de una nueva actividad.
Cuando el usuario hace clic en "Guardar", se crea una nueva instancia de Actividad, se agrega a la lista de actividades pendientes, y se notifica al adaptador que se insertó un nuevo elemento.

```
private void solicitarNuevaActividad() {
    AlertDialog.Builder builder = new AlertDialog.Builder( context: this);
    builder.setTitle("Nueva Actividad");

    final EditText input = new EditText( context: this);
    input.setInputType(InputType.TYPE_CLASS_TEXT);
    builder.setView(input);

    builder.setPositiveButton( text: "Guardar", (dialog, which) -> {
        String nuevoNombreActividad = input.getText().toString();
        if (!nuevoNombreActividad.isEmpty()) {
            Actividad nuevaActividad = new Actividad(nuevoNombreActividad);
            listaActividades.add(nuevaActividad);
            adapterActividades.notifyItemInserted( position: listaActividades.size() - 1);
        } else {
            Toast.makeText( context: MainActivity.this,  text: "Por favor, ingresa un nombre para la actividad", Toast.LENGTH_SHORT).show();
        }
    });

    builder.setNegativeButton( text: "Cancelar", (dialog, which) -> dialog.cancel());
    builder.show();
}
```

Método inicializarActividades():

Se inicializan algunas actividades de ejemplo y se agregan a la lista de actividades pendientes.

```
private void inicializarActividades() {
    listaActividades.add(new Actividad( tarea: "Hacer los deberes"));
    listaActividades.add(new Actividad( tarea: "Ir al gimnasio"));
    listaActividades.add(new Actividad( tarea: "Lavar los platos"));
}
```

Método completarActividad(int position):

Este método se llama cuando se marca una actividad como completada.
La actividad se elimina de la lista de actividades pendientes y se agrega a la lista de actividades completadas.
Se notifica a los adaptadores que se realizaron cambios en los conjuntos de datos, por lo que la interfaz de usuario se actualiza automáticamente.

```
private void completarActividad(int position) {
    Actividad actividad = listaActividades.remove(position);
    listaCompletadas.add(actividad);
    adapterActividades.notifyItemRemoved(position);
    adapterCompletadas.notifyItemInserted( position: listaCompletadas.size() - 1);
}
```

Conclusión / Resumen

En resumen, RecyclerView en Android es un componente gráfico fundamental que optimiza la presentación de listas de elementos en aplicaciones.

A diferencia de las antiguas listas, RecyclerView mejora el rendimiento y la fluidez al reciclar vistas y mostrar solo los elementos visibles en pantalla.

Se utiliza como un contenedor de vistas y ofrece diversas funcionalidades, como reutilización de vistas, manejo eficiente de datos y flexibilidad en el diseño.

Soporta gestos para acciones como eliminar o reorganizar elementos. Permite mejorar la presentación visual mediante divisores y decoraciones entre elementos y además maneja fácilmente eventos de clic y largo clic en elementos para interacciones del usuario.

Recomendaciones

- Exploración de Bibliotecas Complementarias: Investigar y utilizar bibliotecas complementarias que puedan mejorar aún más la funcionalidad de RecyclerView a un nivel superior. Algunas bibliotecas ofrecen características avanzadas y personalizaciones que pueden beneficiar tu aplicación.

- Adopción de Prácticas Recomendadas: Seguir las prácticas recomendadas para la implementación de RecyclerView, asegurando una integración efectiva y eficiente de RecyclerView. Mantente actualizado sobre nuevas recomendaciones y actualizaciones de la biblioteca para garantizar un desarrollo continuo y sin problemas.

- Utilización de ejemplos ya creados: Para implementar algunos aspectos de tu aplicación, no siempre debes crearlo todo desde cero, muchas veces el tiempo que utilizamos puede ser invertido en otro aspecto de la aplicación y si puedes utilizar una funcionalidad que te ahorrará tiempo, no lo desperdicies, eso sí, teniendo en cuenta este aspecto debes de informarte bien que estás utilizando para luego poder retocar y adaptar a tu gusto para mejorar tu aplicación. Es por eso que la información que obtienes de algunos ejemplos y videos debes saber como funciona para implementarla en tu aplicación

Referencias o Bibliografía

-Cómo crear listas dinámicas con RecyclerView | Desarrolladores de Android | Android Developers

-Somos hackers de la programación

-ChatGPT (openai.com)

-Android: Listas dinámicas usando RecyclerView

Ejemplo de código:

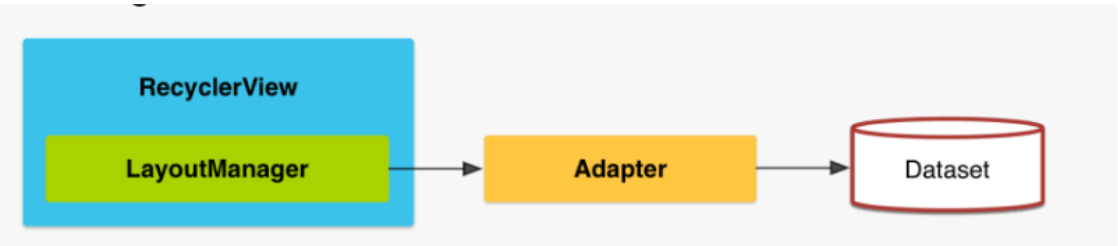
```
private RecyclerView recyclerView;

private ItemAdapter itemAdapter;


import androidx.recyclerview.widget.LinearLayoutManager;

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

Ejemplo de imagen explicada:



RecyclerView: Nuestro RecyclerView se va a "pintar" en función al LayoutManager que reciba como parámetro. También hará uso de un Adapter, que funcionará de acuerdo a un Dataset.

LayoutManager: Este "gestor del diseño" va a definir la disposición de los elementos. Es decir, si van formando una lista vertical u horizontal, si van formando una cuadrícula, u otra variante.

Adapter: El adaptador se encargará de adaptar el dataset a lo que finalmente verá el usuario. Es el encargado de traducir datos en UI.

Dataset: Es el conjunto de datos que se espera mostrar en el RecyclerView. Se puede representar por un simple array de objetos String; o ser algo más elaborado, como un ArrayList de objetos que presentan múltiples atributos.

Videos explicativos:

[Curso de Android con Java: Control RecycleView]

Curso de Android con Java: **Control RecyclerView** – Agregar y eliminar items. diego moisset de espanes•5.8K views · 4:31 · Go to channel ·...

YouTube · diego moisset de espanes · 23 feb 2021

[RecyclerView en menos de 12 minutos. Con ejemplo y código. Android Studio. Onclick elementos RV.]