Instituto Politécnico Nacional

Omar Montoya Romero

7CM1

Desarrollo de Aplicaciones Móviles Nativas

Sistemas Computacionales

Actividad II: ListView y RecyclerView

Mtro. Efraín Arredondo Morales

02/10/2023

LISTVIEW

El control ListView es un ItemsControl que se deriva de ListBox. Habitualmente, sus elementos son miembros de una recopilación de datos y se representan como objetos ListViewItem. Un objeto ListViewItem es un ContentControl y solo puede contener un único elemento secundario. Sin embargo, el elemento secundario puede ser cualquier elemento visual.

El control ListView contiene ListViewItem objetos, que representan los elementos de datos que se muestran. Puede usar las propiedades siguientes para definir el contenido y el estilo de los elementos de datos:

- En el control ListView, use las propiedades ItemTemplate, ItemTemplateSelector y ItemContainerStyle.
- En el control ListViewItem, use las propiedades ContentTemplate y ContentTemplateSelector.

Para evitar problemas de alineación entre celdas en una GridView, no use ItemContainerStyle para establecer las propiedades o añadir contenido que afecte a la anchura de un elemento en un ListView. Podría producirse un problema de alineación cuando establece la propiedad Margin en ItemContainerStyle, por ejemplo. Para especificar las propiedades o definir el contenido que afecta a la anchura de los elementos en GridView, use las propiedades de la clase GridView y sus clases relacionadas, como GridViewColumn.

Ejemplo. –

```
<ListView ItemsSource="{Binding Source={StaticResource
EmployeeInfoDataSource}}">
   <ListView.View>
      <GridView AllowsColumnReorder="true" ColumnHeaderToolTip="Employee
Information">
         <GridViewColumn DisplayMemberBinding="{Binding Path=FirstName}"
Header="First Name" Width="100"/>
         <GridViewColumn DisplayMemberBinding="{Binding Path=LastName}"
Width="100">
            <GridViewColumnHeader>Last Name
               <GridViewColumnHeader.ContextMenu>
```

```
                <ContextMenu MenuItem.Click="LastNameCM_Click"
Name="LastNameCM">
                    <MenuItem Header="Ascending" />
                    <MenuItem Header="Descending" />
                </ContextMenu>
            </GridViewColumnHeader.ContextMenu>
        </GridViewColumnHeader>
    </GridViewColumn>
    <GridViewColumn DisplayMemberBinding="{Binding
Path=EmployeeNumber}" Header="Employee No." Width="100"/>
  </GridView>
 </ListView.View>
</ListView>
```

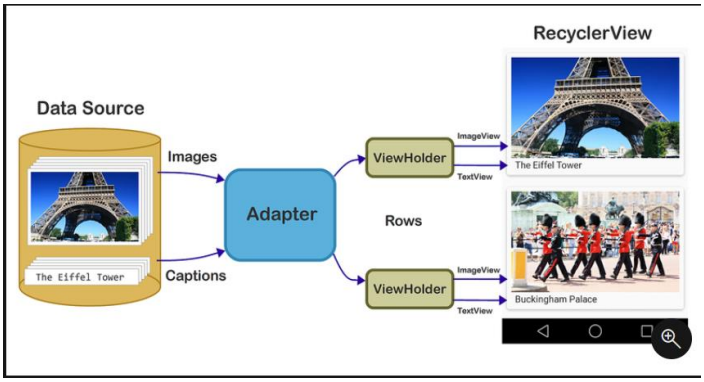
First Name	Last Name	Employee No.
Jesper	Aaberg	12345
Dominik	Paiha	98765
Yale	Li	23875
Muru	Subramani	49392

[1]

RECYCLERVIEW

RecyclerView controla algunas tareas internamente (como el desplazamiento y el reciclaje de vistas), pero esencialmente es un administrador que coordina las clases auxiliares para mostrar una colección. RecyclerView delega las tareas en las siguientes clases auxiliares:

- **Adapter**: infla los diseños de elementos (crea instancias del contenido de un archivo de diseño) y enlaza los datos a las vistas que se muestran dentro de .RecyclerView El adaptador también notifica eventos de clic en elementos.



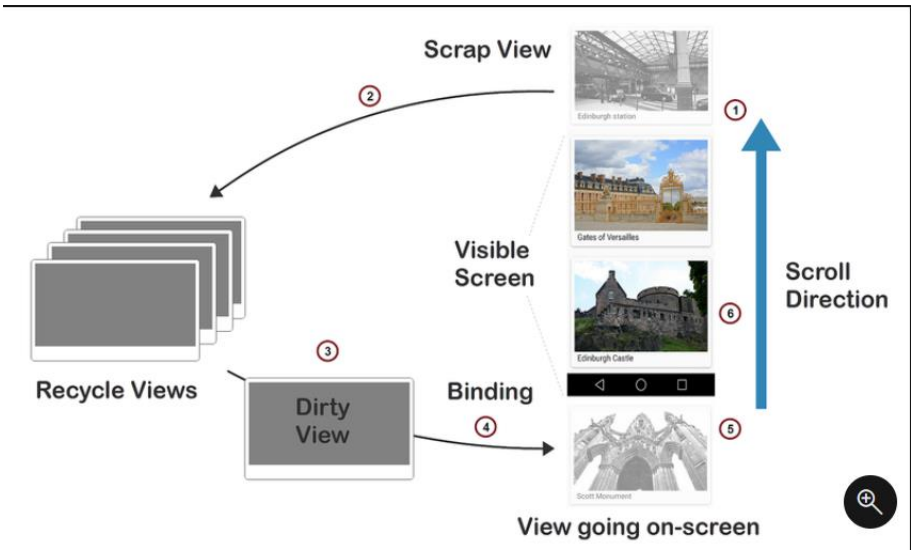
- **LayoutManager** – Mide y coloca las vistas de elementos dentro de y RecyclerView administra la directiva para el reciclaje de vistas.

- **ViewHolder** : busca y almacena las referencias de vista. El titular de la vista también ayuda a detectar clics de vista de elementos.

- **ItemDecoration** : permite que una aplicación agregue desplazamientos especiales de dibujo y diseño a vistas específicas para los divisores de dibujo entre elementos, resaltados y límites de agrupación visual.

- **ItemAnimator** : define las animaciones que tienen lugar durante las acciones del elemento o a medida que se realizan cambios en el adaptador.

Ejemplo del cómo funciona un RecyclerView:



1. Cuando una vista se desplaza fuera de la vista y ya no se muestra, se convierte en una *vista de extracción*.

2. La vista de extracción 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.

4. Dado que esta vista debe volver a enlazarse con el adaptador antes de mostrarse, se denomina *vista desfasada*.

5. La vista desfasada se recicla: el adaptador localiza los datos del siguiente elemento que se va a mostrar 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.

6. La vista reciclada se agrega a la lista de elementos de que RecyclerView están a punto de ir en pantalla.

7. La vista reciclada se pone en pantalla a medida que el usuario desplaza hasta RecyclerView el siguiente elemento de la lista. Mientras tanto, otra vista se desplaza fuera de la vista y se recicla según los pasos anteriores. [2]

Ejemplo 2 RecyclerView. –

Primero importamos las siguientes dependencias

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

Despues definimos el RecyclerView en nuestro layout

```
<androidx.recyclerview.widget.RecyclerView
    android:id="@+id/recyclerView"
    android:layout_width="match_parent"
    android:layout_height="wrap_content" />
```

Una vez implementado obtenemos la referencia, le asignamos el layout manager y le asociamos un adapter en el main

```
class MainActivity : AppCompatActivity() {
    private val myDataSet = arrayOf(
        "PHP",
        "Javascript",
        "Go",
        "Python"
    )

    private val mAdapter by lazy {
        MyAdapter(myDataSet)
    }

    public override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
```

```
        setContentView(R.layout.my_activity);

        recyclerView.setHasFixedSize(true)

        val layoutManager = LinearLayoutManager(baseContext)
        recyclerView.layoutManager = layoutManager

        recyclerView.adapter = mAdapter
    }
}
```

Un adapter:

- Contiene una **clase interna** ViewHolder, que permite obtener referencias de los *componentes visuales* (views) de cada elemento de la lista.

- Presenta un **constructor** y/o **métodos para gestionar el Data Set** (añadir, editar o eliminar elementos).

- Contiene un método onCreateViewHolder que infla el layout (archivo xml) que representa a nuestros elementos, y devuelve una instancia de la clase ViewHolder que antes definimos.

- Contiene un método onBindViewHolder que enlaza nuestra data con cada ViewHolder.

- Contiene un método getItemCount que devuelve un entero indicando la cantidad de elementos a mostrar en el RecyclerView.

```
class MyAdapter (private val mDataSet: Array)
    : RecyclerView.Adapter() {

    // En este ejemplo cada elemento consta solo de un nombre
    class ViewHolder(var textView: TextView) :
RecyclerView.ViewHolder(textView)

    // El layout manager invoca este método para renderizar cada elemento
    override fun onCreateViewHolder(parent: ViewGroup, viewType: Int):
ViewHolder {
        val v = LayoutInflater.from(parent.context)
                .inflate(R.layout.item_text_view, parent, false) as TextView
```

```
•
•           // Aquí podemos definir tamaños, márgenes, paddings, etc
•           return ViewHolder(v)
•       }
•
•       // Este método asigna valores para cada elemento de la lista
•       override fun onBindViewHolder(holder: ViewHolder, position: Int) {
•           holder.textView.text = mDataSet[position]
•       }
•
•       // Cantidad de elementos del RecyclerView
•       // Puede ser más complejo (por ejm, si implementamos filtros o búsquedas)
•       override fun getItemCount() = mDataSet.size
•   }
```

DataSet
Como puedes observar, dentro del adapter tenemos un atributo llamado mDataSet.
En este caso, la fuente de datos que espera recibir el adaptador es solo un arreglo de objetos String.
Pero esta fuente de datos puede ser más compleja, según se requiera.

ViewHolder
En el método onCreateViewHolder hacemos uso de la clase LayoutInflater para "inflar" un layout XML.
Para este simple ejemplo, donde **listamos lenguages de programación**, el layout se representa solo por un TextView.
Lo podemos definir como item_text_view.xml dentro de la carpeta res\layout:

```
<TextView xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/tvName"
    android:padding="12dp"
    android:layout_height="wrap_content"
    android:layout_width="match_parent" />
```

En resumen debe quedar asi:

```
<android.support.v7.widget.CardView
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:card_view="http://schemas.android.com/apk/res-auto"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    card_view:cardUseCompatPadding="true"
```

```
card_view:cardElevation="4dp"
card_view:cardCornerRadius="3dp"
android:layout_margin="6dp">
<LinearLayout
   android:id="@+id/linearLayout"
   android:layout_width="match_parent"
   android:layout_height="wrap_content"
   android:orientation="vertical"
   android:padding="6dp"
   android:gravity="center">
   <TextView
      android:id="@+id/tvInformId"
      android:textColor="@color/colorPrimary"
      android:layout_width="match_parent"
      android:layout_height="wrap_content"
      android:text="INFORME 777" />
   <TextView
      android:id="@+id/tvUserName"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:textAppearance="@android:style/TextAppearance"
      android:text="Juan Ramos" />
   <TextView
      android:id="@+id/tvCreatedAt"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:text="Creado el día 01/02/2017"/>
   <TextView
      android:id="@+id/tvFromDate"
      android:textColor="@color/colorPrimaryDark"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
     android:textAppearance="@style/Base.TextAppearance.AppCompat.Medium"
      android:text="Desde 17/03/2017" />
   <TextView
      android:id="@+id/tvToDate"
      android:textColor="@color/colorPrimaryDark"
      android:layout_width="wrap_content"
     android:textAppearance="@style/Base.TextAppearance.AppCompat.Medium"
      android:layout_height="wrap_content"
      android:text="Hasta 17/07/2017" />
```

```
    <LinearLayout
      android:layout_width="match_parent"
      android:layout_height="wrap_content"
      android:orientation="horizontal">
      <Button
        android:id="@+id/btnGoToReports"
        style="@style/Widget.AppCompat.Button.Borderless.Colored"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_weight="1"
        android:text="Ver reportes" />
      <Button
        style="@style/Widget.AppCompat.Button.Borderless.Colored"
        android:layout_width="0dp"
        android:layout_weight="1"
        android:layout_height="wrap_content"
        android:text="Editar informe"
        android:id="@+id/btnEditInform" />
    </LinearLayout>
  </LinearLayout>
</android.support.v7.widget.CardView>
```

Y el resultado es el siguiente:

[3]

Conclusión

El listview es una versión mas vieja que el recyclerview, es mejor usar el recyclerview por que como dice el nombre reutiliza la vista haciendo que la carga sea menor ocasionando menos lag, aparte de que es mas flexible, permite colocar mejores diseños,

efectos, el único lado complejo que tiene el recyclerview es que es complicado de implementar, pero una vez comprendido todo se vuelve más fácil.

Referencias

[1] adegeo, «Microsoft Learn: Build skills that open doors in your career,» Microsoft, 03 05 2023. [En línea]. Available: https://learn.microsoft.com/es-es/dotnet/desktop/wpf/controls/listview-overview?view=netframeworkdesktop-4.8. [Último acceso: 03 10 2023].

[2] «Microsoft Learn: Build skills that open doors in your career,» Microsoft, 13 07 2023. [En línea]. Available: https://learn.microsoft.com/es-es/xamarin/android/user-interface/layouts/recycler-view/parts-and-functionality. [Último acceso: 03 10 2023].

[3] «Programación y más | Aprende desarrollo web y móvil,» [En línea]. Available: https://programacionymas.com/blog/listas-dinamicas-android-usando-recycler-view-card-view. [Último acceso: 03 10 2023].