Fabián Zamora Ramírez
Jeff Schmidt Peralta
Noviembre, 2016

Programación de aplicaciones Android

Dia 3

Fabián Zamora Ramírez
Jeff Schmidt Peralta
Noviembre, 2016

Fabián Zamora Ramírez
Jeff Schmidt Peralta

Listas y tarjetas **3**
- Adapter 3
- ViewHolder 3
- LayoutManager 3
- CardView 4

FileProvider **5**
- Cómo definir un File Provider 5
- Especificación de archivos disponibles 6

Referencias **7**

Fabián Zamora Ramírez
Jeff Schmidt Peralta
Noviembre, 2016

I. Listas y tarjetas

Para crear listas siguiendo los principios de Material Design en las aplicaciones, Android proporciona los widgets **RecyclerView** y **CardView** de la biblioteca de soporte-v7-appcompat.

RecyclerView es un contenedor para mostrar grandes conjuntos de datos que se pueden desplazar de manera muy eficiente **al mantener una cantidad limitada de vistas existentes en memoria al mismo tiempo**.

Para usar el widget **RecyclerView**, se tiene que especificar un adaptador (**Adapter**) y un administrador de diseño (**LayoutManager**) como se muestra en la Figura 1.

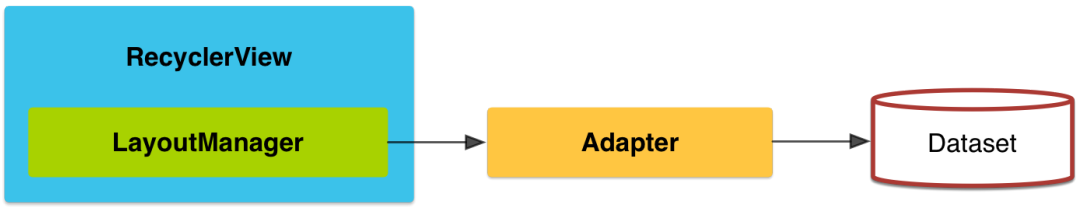

Figura 1. Elementos necesarios para rellenar un RecyclerView

A. Adapter

Un adapter se encarga de crear filas y mapear los datos en una lista. Para crear un adaptador para un RecyclerView se hereda la clase **RecyclerView.Adapter** y se sobreescriben los métodos que describen y manejan los datos. Los detalles de la implementación dependen de las especificaciones del conjunto de datos y los tipos de vistas.

B. ViewHolder

Un ViewHolder describe el view de una fila de la lista y metadatos sobre su lugar dentro del RecyclerView. Los ViewHolder pertenecen a un adapter y deben poseer campos para almacenar en caché resultados potencialmente caros de *findViewById (int)*. Los ViewHolder se crean heredando de la clase RecyclerView.ViewHolder.

Fabián Zamora Ramírez
Jeff Schmidt Peralta
Noviembre, 2016

C. LayoutManager

Un administrador de diseño (**LayoutManager**) posiciona las filas dentro de un RecyclerView y determina cuándo volver a usar los view de las filas que ya no están visibles para el usuario. Para reutilizar (o reciclar) una vista, un administrador de diseño puede solicitar al adaptador que reemplace el contenido del view de una fila con un elemento diferente del conjunto de datos. De esta manera, cuando se reciclan las vistas se mejora el rendimiento al evitar la creación de vistas innecesarias o realizar búsquedas costosas de *findViewById()*. El RecyclerView proporciona tres administradores de diseño incorporados:

- **LinearLayoutManager:** muestra elementos en una lista de desplazamiento horizontal o vertical.
- **GridLayoutManager:** muestra elementos en una cuadrícula.
- **StaggeredGridLayoutManager:** muestra elementos en una cuadrícula escalonada.

D. CardView

Es un FrameLayout con esquinas redondeadas y sombra. Utiliza el atributo **android:elevation** para Android Lollipop (y recientes) y utiliza sombras emuladas en plataformas anteriores.

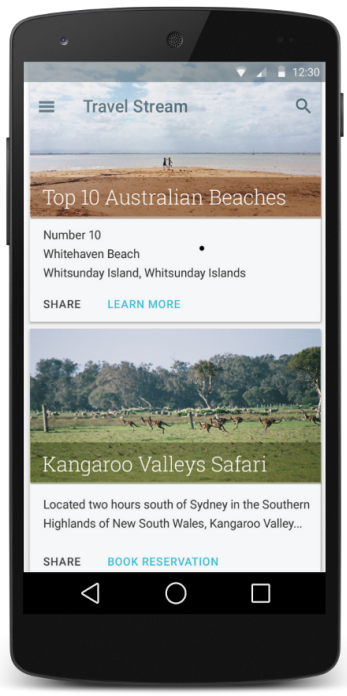

Figura 2. Ejemplo de CardView.

Fabián Zamora Ramírez
Jeff Schmidt Peralta

II. FileProvider

Es una subclase de ContentProvider que facilita compartir archivos de forma segura entre aplicaciones al crear un Uri de contenido (**content://**) en vez de un Uri de archivo (**file:///**).
Cuando se crea un Uri de contenido se permite leer y escribir sobre el archivo mediante el uso de permisos temporales que viven mientras la Activity o el Service que recibe el archivo estén vivos.
Al contrario cuando se maneja un archivo con un Uri de archivo se tienen que modificar los permisos del sistema del archivo los cuales están disponibles a todas las aplicaciones lo que es inseguro.

A. Cómo definir un File Provider

Un FileProvider se puede definir completamente en XML. Para especificar el componente FileProvider, se debe añadir un elemento <provider> al AndroidManifest.xml y establecer los siguientes atributos:

- **android:name:** en android.support.v4.content.FileProvider.
- **android:authorities:** en un dominio que usted controla; Por ejemplo, si controla el dominio mydomain.com debe usar la autoridad com.mydomain.fileprovider.
- **android:export:** en falso; El FileProvider no necesita ser público.
- **android:grantUriPermissions:** en true, para permitir conceder acceso temporal a los archivos.

```
<manifest>
    ...
    <application>
        ...
        <provider
            android:name="android.support.v4.content.FileProvider"
            android:authorities="com.mydomain.fileprovider"
            android:exported="false"
            android:grantUriPermissions="true">
            ...
        </provider>
        ...
    </application>
</manifest>
```

Figura 3. Ejemplo de declaración de un FileProvider.

Fabián Zamora Ramírez
Jeff Schmidt Peralta
Noviembre, 2016

B. Especificación de archivos disponibles

Un FileProvider sólo puede generar un URI de contenido para archivos en directorios que se especifiquen de antemano. Para especificar un directorio, se debe crear un xml donde se definen las rutas.

```
<paths xmlns:android="http://schemas.android.com/apk/res/android">
    <files-path name="my_images" path="images/"/>
    ...
</paths>
```

Figura 4. Ejemplo de declaración de rutas de un FileProvider.

Fabián Zamora Ramírez
Jeff Schmidt Peralta
Noviembre, 2016

III. Referencias

- *Portions of this page are reproduced from work created and shared by the Android Open Source Project and used according to terms described in the Creative Commons 2.5 Attribution License.*
- *Portions of this page are modifications based on work created and shared by the Android Open Source Project and used according to terms described in the Creative Commons 2.5 Attribution License.*
- *La lista completa de sitios se enlista a continuación:*
    - *https://developer.android.com*
    - *https://developer.android.com/training/material/lists-cards.html?hl=es*
    - *https://developer.android.com/reference/android/support/v7/widget/CardView.html*
    - *https://developer.android.com/reference/android/support/v4/content/FileProvider.html*