Simulando un ListView - Instrucciones

1. **Un voluntario será el Listview:**
   a. Su tarea es pedirle al adapter una **celda completa y cargada con datos** para mostrar en pantalla. Pedirá cuatro celdas, de a una por vez.
2. **Un voluntario será el Adapter:**
   a. Su tarea es coordinar a cuatro voluntarios más, para que armen la **celda completa** que le pide el listview.

Simulando un ListView - Instrucciones

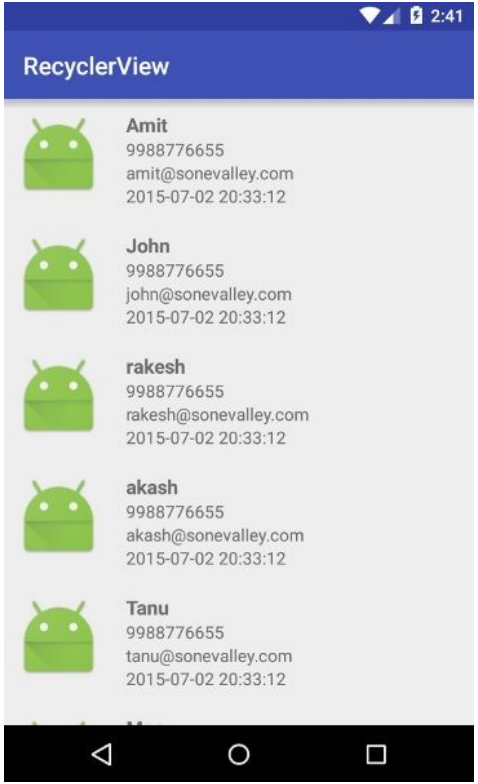
3. **Cuatro voluntarios serán el Adapter.**
    a. **Dos voluntarios** serán los encargados de inflar la celda y armar su estructura. La celda deberá tener **un imageview y dos textview**.
    b. **El tercer voluntario** será el encargado de buscar el lugar donde se insertarán los datos, señalándole al **cuarto voluntario** donde debe pegar los datos.
    c. El **cuarto voluntario** buscará los datos y los pegará en la celda. Luego le dará la celda completa al adapter

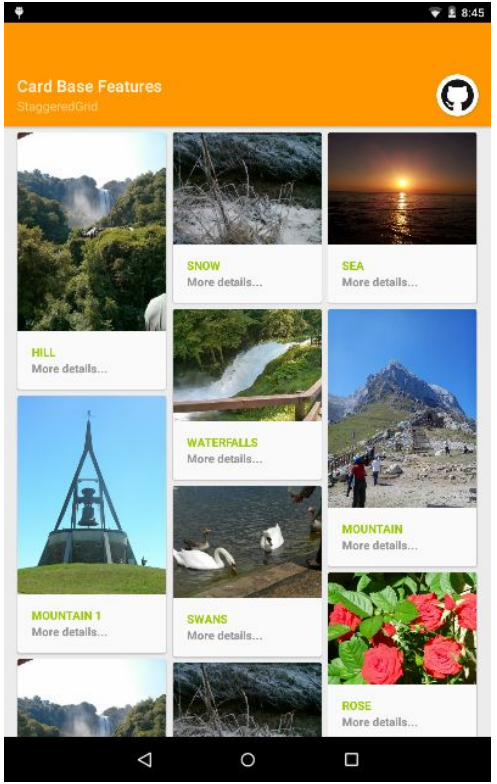
Analicemos el juego...

¿Qué pasó?

RecyclerView

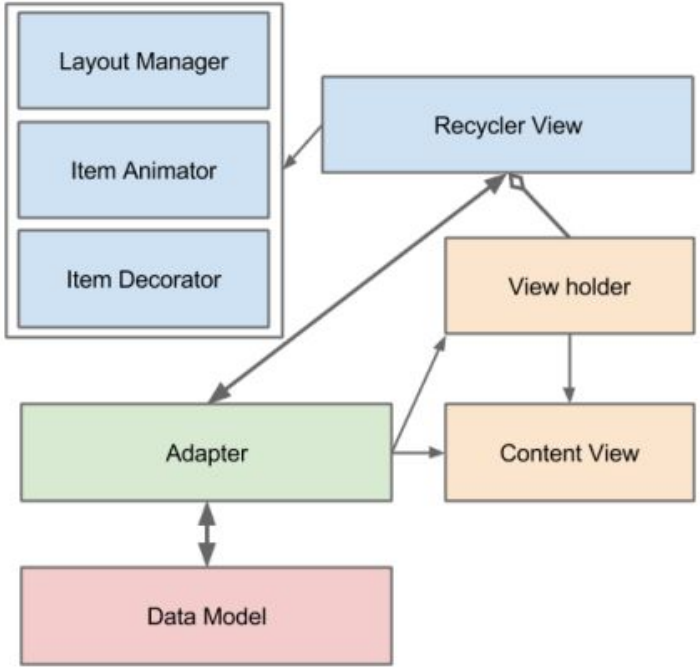
- Nos permite mostrar en pantalla grandes colecciones de **datos**
- Similar al ListView





Componentes

1. RecyclerView.Adapter
2. RecyclerView.ViewHolder
3. LayoutManager
4. ItemDecoration e ItemAnimator



1. Adapter

- Tiene la misma funcionalidad que tenía en el ListView, es decir, **es el encargado de unir los datos con la vista**.

- Nuestro adapter ahora extenderá de la clase **RecyclerView.Adapter.**

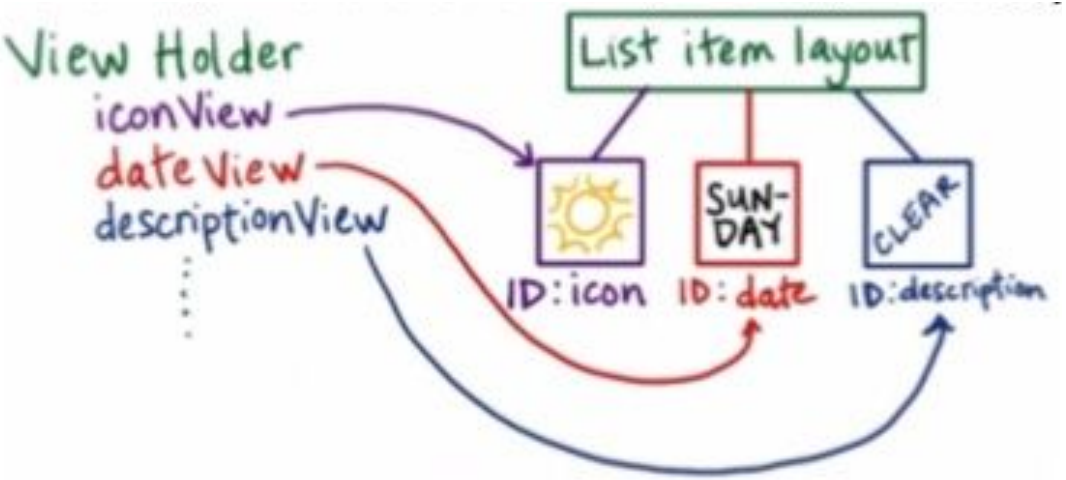
2. ViewHolder - Motivación

- **Obtener cada componente** que conforma nuestra celda mediante el método findViewById() **es costoso**.

- Si utilizamos muchas veces la función findViewById() la **performance** de nuestra aplicación **se degrada**.

2. ViewHolder - Solución

Guardar una referencia a los componentes que forman un layout para no tener que volver a buscarlos.



2. ViewHolder - Pattern

- Crearemos una clase, MiViewHolder, que tendrá ahí dentro todos los componentes de nuestra celda.

- La idea será **crear e inicializar el objeto ViewHolder una única vez**. Posteriormente, podemos recuperar ese ViewHolder y reutilizarlo fácilmente.

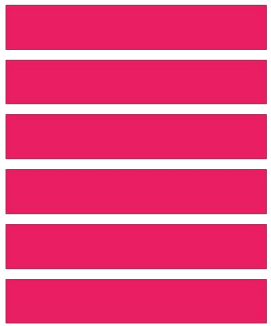
2. ViewHolder - Pattern

```
private class ViewHolder{
    private TextView textViewNombrePersonaje;

    public ViewHolder(View view){
        //De esta manera cuando se inicializa el ViewHolder que esta relacionado con una vista
        //Guarda en ese mismo momento las referencias de los elementos de la misma
        textViewNombrePersonaje = (TextView)view.findViewById(R.id.textViewNombrePersonaje);
    }

    public void cargarPersonaje(PersonajeDeSerie unPersonaje){
        textViewNombrePersonaje.setText(unPersonaje.getNombre());
    }
}
```

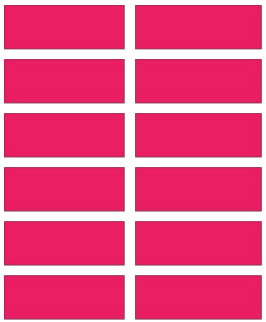
3. Layout Manager

- Determina **la forma** en que se van a mostrar en pantalla los elementos de nuestra colección.
- Android incorpora de serie dos LayoutManagers:

LinearLayoutManager



GridLayoutManager



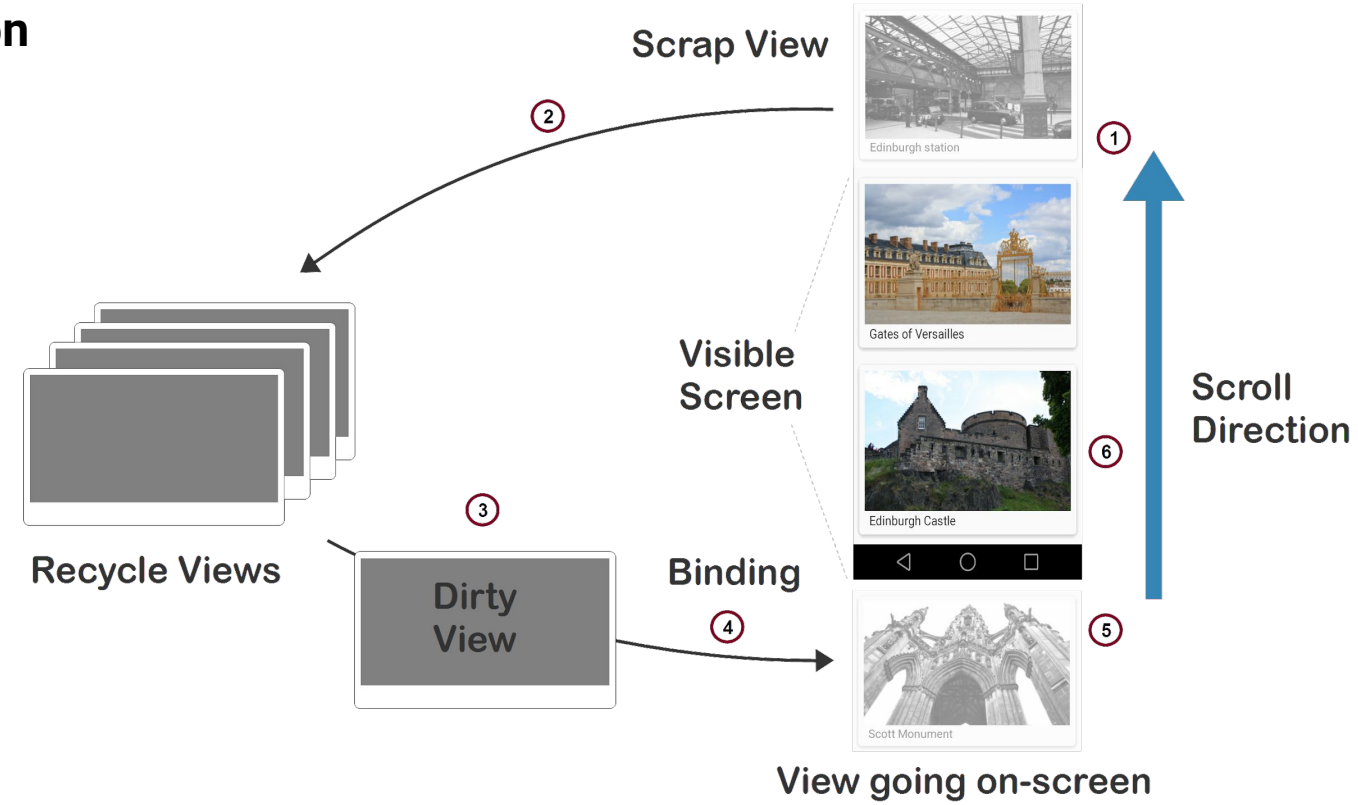
4. Item Decorator - Item Animator

Estos componentes se encargan de definir cómo se representan algunos aspectos visuales de nuestra lista.

Ejemplo:

- Marcadores.
- Separadores de elementos.
- Animaciones al añadir o eliminar elementos.

Conclusión


Scrap View
2
Edinburgh station
1
Gates of Versailles
Visible Screen
Edinburgh Castle
6
Scroll Direction
3
Recycle Views
Dirty View
Binding
4
5
Scott Monument
View going on-screen