


Open · Opened 3 days ago by  **Rubén Montero**

RecyclerView

Resumen

- Comprenderemos la idea esencial del componente visual **RecyclerView**
- Añadiremos un `<RecyclerView>` a `activity_main.xml`. De momento, no mostrará nada

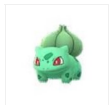
Descripción

Mostrar una *lista* es relativamente común en la mayoría de aplicaciones: La lista de correos de la bandeja de entrada, lista de *posts* de Facebook, lista de noticias...

Un componente esencial en Android es [RecyclerView](#). Análogo a [UITableView](#) de iOS, se dedica a *reciclar* las vistas asociadas a cada celda de la *lista*.

[RecyclerView](#), ¿por qué?

Hoy en día existen 890 pokemon.



Si quisiéramos desarrollar una actividad Android que los mostrase a todos, podríamos emplear una implementación [demasiado sencilla](#) del primitivo componente [ListView](#) en cuyo caso almacenaríamos en memoria RAM todos los datos.

Suponiendo que mostramos un PNG de 1024x1024 (~500Kb) por cada uno, necesitaremos:

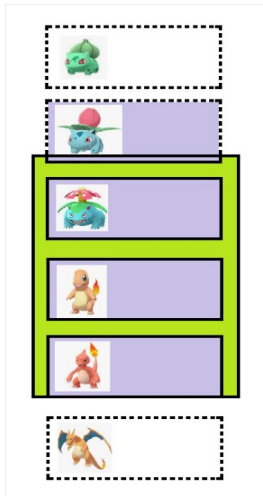
$$500 \text{ Kb} * 890 = 445\,000 \text{ Kb} = 434,57 \text{ Mb}$$

Esto es...

...más del doble de memoria RAM que tenía [el primer dispositivo Android](#). ¡Demasiado!

RecyclerView reduce el uso de memoria RAM *reutilizando* (reciclando) las *celdas* visibles en pantalla.

Por ejemplo, si sólo caben 4 celdas en pantalla:



...la memoria RAM necesaria para tener las imágenes será como mucho de:

$$500 \text{ Kb} * 4 = 2\,000 \text{ Kb} = 1,95 \text{ Mb}$$

¡Vendido! Quiero un **RecyclerView**

Implementarlo llevará ciertos pasos:

1. Añadir un `<RecyclerView>` al XML de la actividad.

2. Añadir un nuevo fichero XML que represente una *celda*.
3. Añadir una clase Java vinculada al XML de la celda. Esta clase Java representa una *celda*, pero en código Java. Hereda de `ViewHolder`.
4. Añadir una clase *adaptador*. Esta clase heredará de `RecyclerView.Adapter`.

Para esta tarea, sólo añadiremos `<RecyclerView>` al XML.

La tarea

En `activity_main.xml`, **escribe** una nueva etiqueta `<RecyclerView`, y **dale** a ENTER para que Android Studio autocomplete:

```
<androidx.recyclerview.widget.RecyclerView
    android:layout_width=""
    android:layout_height=""
```

Añade al `<androidx.recyclerview.widget.RecyclerView>` los atributos necesarios para que esté ajustado al contenedor, de tal forma que ocupe toda la pantalla. Puedes usar:

- `match_parent` en `layout_width` y `layout_height`
- `0dp` en `layout_width` y `layout_height`, y *constraints* `Start_toStart`, `End_toEnd`, `Top_toTop` y `Bottom_toBottom`

...lo que prefieras.

Para terminar, **añade** un `android:id` al `<RecyclerView>`. El que quieras. Lo usaremos más adelante.

(No te preocupes porque teóricamente el `<RecyclerView>` y el `<ProgressBar>` se solapan. Ambas vistas no estarán visibles simultáneamente).

Por último

Sube tus cambios al repositorio en un nuevo *commit*.



Rubén Montero @ruben.montero changed milestone to [%Sprint 2](#) 3 days ago