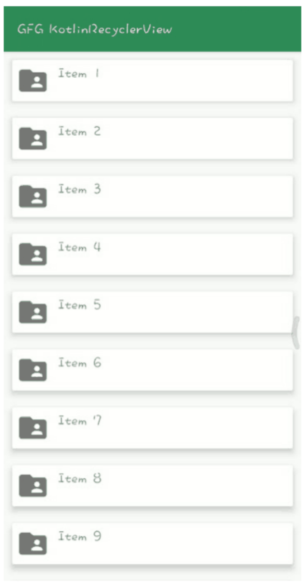
Create dynamic lists with RecyclerView

RecyclerView

RecyclerView makes it easy to efficiently display large sets of data. You supply the data and define how each item looks, and the RecyclerView library dynamically creates the elements when they're needed.

When you have a long list of items to show you can use RecyclerView. It has the ability to reuse its views. In RecyclerView when the View goes out of the screen or not visible to the user it won't destroy it, it will reuse these views



Key classes

Several different classes work together to build your dynamic list.

- **`RecyclerView`** is the **`ViewGroup`** that contains the views corresponding to your data. It's a view itself, so you add **`RecyclerView`** into your layout the way you would add any other UI element.

- Each individual element in the list is defined by a view holder object. When the view holder is created, it doesn't have any data associated with it. After the view holder is created, the **`RecyclerView`** binds it to its data. You define the view holder by extending **`RecyclerView.ViewHolder.`**

- The **`RecyclerView`** requests those views, and binds the views to their data, by calling methods in the adapter. You define the adapter by extending **`RecyclerView.Adapter.`**

- The layout manager arranges the individual elements in your list.

Steps for implementing your RecyclerView

- First of all, decide what the list or grid is going to look like. Ordinarily you'll be able to use one of the RecyclerView library's standard layout managers.

- Design how each element in the list is going to look and behave. Based on this design, extend the ViewHolder class. Your version of **`ViewHolder`** provides all the functionality for your list items. Your view holder is a wrapper around a **`View`**, and that view is managed by **`RecyclerView.`**

- Define the **`Adapter`** that associates your data with the **`ViewHolder`** views.

Plan your layout

The items in your RecyclerView are arranged by a LayoutManager class. The RecyclerView library provides three layout managers, which handle the most common layout situations:

- `LinearLayoutManager` arranges the items in a one-dimensional list.

- `GridLayoutManager` arranges all items in a two-dimensional grid:
  - If the grid is arranged vertically, `GridLayoutManager` tries to make all the elements in each row have the same width and height, but different rows can have different heights.
  - If the grid is arranged horizontally, `GridLayoutManager` tries to make all the elements in each column have the same width and height, but different columns can have different widths.

- `StaggeredGridLayoutManager` is similar to GridLayoutManager, but it does not require that items in a row have the same height (for vertical grids) or items in the same column have the same width (for horizontal grids). The result is that the items in a row or column can end up offset from each other.

Implementing your adapter and view holder

Once you've determined your layout, you need to implement your Adapter and ViewHolder. These two classes work together to define how your data is displayed.

The ViewHolder is a wrapper around a View that contains the layout for an individual item in the list. The Adapter creates ViewHolder objects as needed, and also sets the data for those views. The process of associating views to their data is called binding.

When you define your adapter, you need to override three key methods:

- **`onCreateViewHolder():`** RecyclerView calls this method whenever it needs to create a new ViewHolder. The method creates and initializes the ViewHolder and its associated View, but does not fill in the view's contents—the ViewHolder has not yet been bound to specific data.
- **`onBindViewHolder():`** RecyclerView calls this method to associate a ViewHolder with data. The method fetches the appropriate data and uses the data to fill in the view holder's layout. For example, if the RecyclerView displays a list of names, the method might find the appropriate name in the list and fill in the view holder's TextView widget.
- **`getItemCount()`**: RecyclerView calls this method to get the size of the data set. For example, in an address book app, this might be the total number of addresses. RecyclerView uses this to determine when there are no more items that can be displayed.