* 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.

As the name implies, RecyclerView recycles those individual elements. When an item scrolls off the screen, RecyclerView doesn't destroy its view. Instead, RecyclerView reuses the view for new items that have scrolled onscreen. RecyclerView improves performance and your app's responsiveness, and it reduces power consumption.

* [**RecyclerView**](https://developer.android.com/reference/androidx/recyclerview/widget/RecyclerView) is the [ViewGroup](https://developer.android.com/reference/android/view/ViewGroup) that contains the views corresponding to your data. It's a view itself, so you add RecyclerView to 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](https://developer.android.com/reference/androidx/recyclerview/widget/RecyclerView.ViewHolder).
* The RecyclerView requests views, and binds the views to their data, by calling methods in the *adapter.* You define the adapter by extending [RecyclerView.Adapter](https://developer.android.com/reference/androidx/recyclerview/widget/RecyclerView.Adapter).
* The *layout manager* arranges the individual elements in your list. You can use one of the layout managers provided by the RecyclerView library, or you can define your own. Layout managers are all based on the library's [LayoutManager](https://developer.android.com/reference/androidx/recyclerview/widget/RecyclerView.LayoutManager) abstract class.

