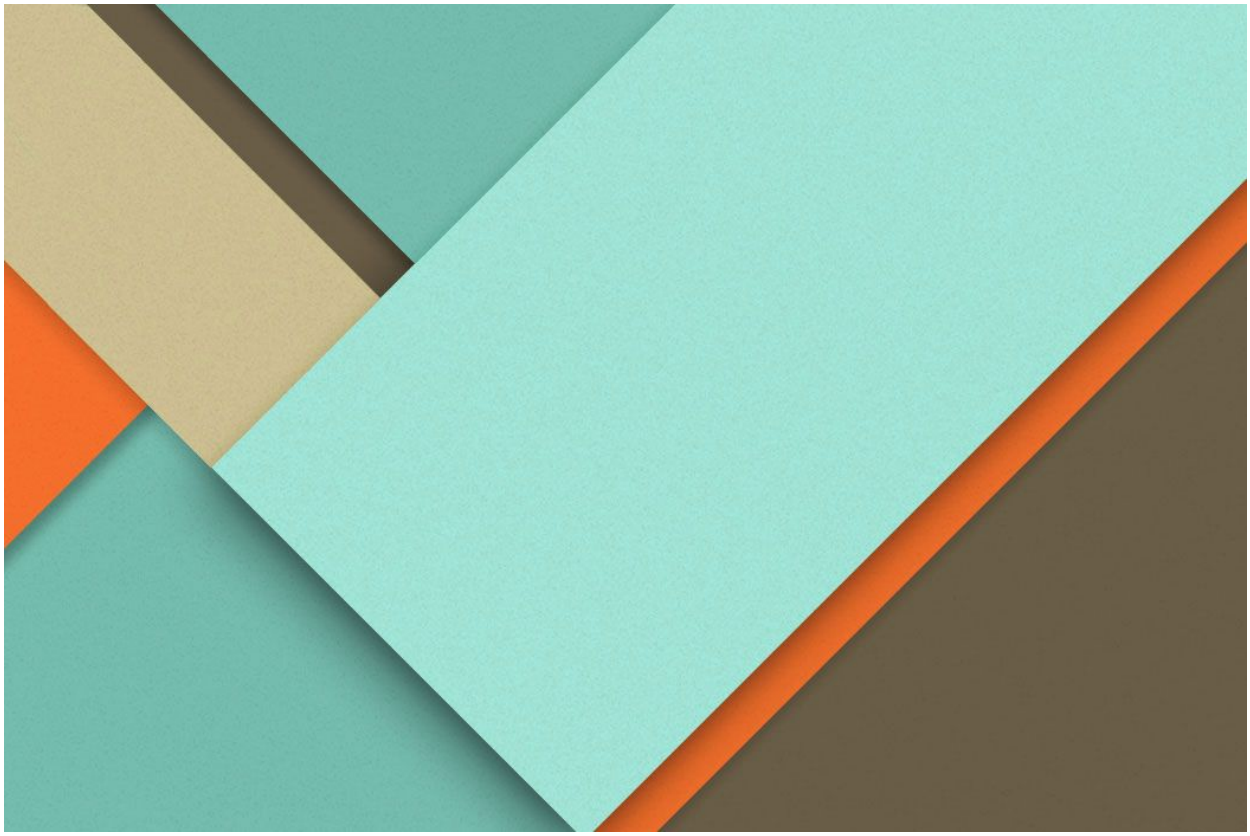


# Character Viewer

01.02.2018

Mert Vurgun

# Overview

This is a small scale sample project which provides functionality of MVP architecture, consuming network and image libraries, multi pane device configuration over phone and tablet.

# Goals

1. Creating MVP structure boilerplate
2. Third party library decision
3. Kotlin first
4. Product flavoring
5. Listing as flat list and grid

# Milestones

## I. Creating MVP structure

Model-View-Presenter is one of the powerful architecture that can split the logic from activities and fragments. It is also helpful to solve strongly coupled classes problems.

There are different ways of implementing this architecture which can be picked based on project size, used language or compatibility with different libraries. In this project, I tried to keep it as simple as possible while some activities or fragments do not even need.

## II. Third Party Libraries

**Network**

Nowadays, almost in every project we use strong and stable third party libraries. When its asked that can we still implement our way, yes of course. However, most of these libraries are being used for a long time and some are supported by Google such as Volley networking library.

I prefer using Retrofit as networking library. One reason is that usage is very simple and we can easily modularize. The other reason is what you only need is a plain java object. Retrofit handles all conversion for us. The final and my very personal opinion is they way to call APIs can easily be shrinked by using Kotlin.

### Image Loading

We have a very large variety of image libraries such as Picasso, Glide, Fresco, AQuery and many more. We can compare them based on their size, caching methodology, image quality or download speed.

In the last few projects, I choose to use AQuery and Picasso. AQuery could be pretty good if images displayed on each item of the list and Picasso is a powerful and stable one to pick for more quality images.

However, I used Glide image loading library in this project. One reason is that the usage is very similar to Picasso and by using Kotlin extension, main code stays very clean. The other reason is Glide loads exact size of the image view while Picasso downloads and resize (*). Also, while I was checking the JSON responses, I saw few .gif image urls. I thought it would be really good to display gifs especially in tablet mode.

### Kotlin first

Because of this is fresh project, I completely used Kotlin. It gives few warning about type parameters but my focus again clean code as possible.

### Product flavoring

Based on requirements, there are two flavors which indicates different data, but both using same app sources and json structure. Luckily, provided APIs are similar.

**http://api.duckduckgo.com/?q=simpsons+characters&format=json**

**http://api.duckduckgo.com/?q=the+wire+characters&format=json**

As we can see above, only change is a string query parameter which can easily be placed in app flavors. Sample below:

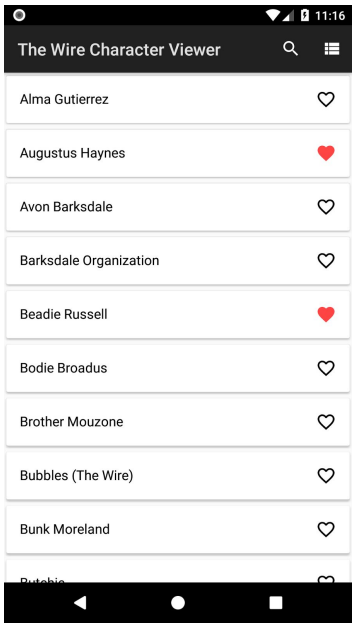
```
productFlavors {
    simsons { resValue "string", "character_type", "simpsons+characters"  }
    thewire { resValue "string", "character_type", "the+wire+characters" }
  }
```

### List2Grid - Grid2List

Again there are many ways to approach this task such as using different layouts or fragments, changing Linear to Grid Layout Manager. I prefer using Grid Layout Manager and play with columns to change between layouts. There is a slight performance effect, but this is a small project and simple sounds elegant.

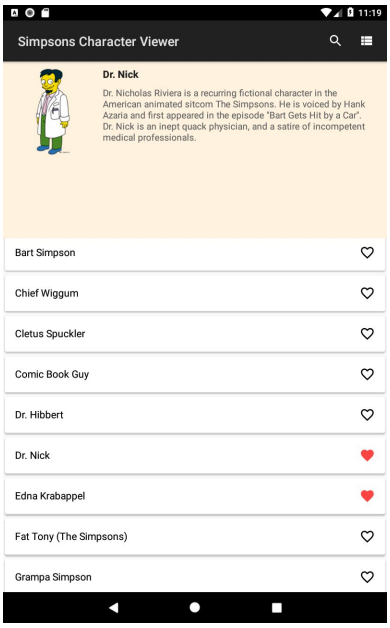
### Screenshots

Phone




Phone


Tablet




### Resources

*

https://inthecheesefactory.com/blog/get-to-know-glide-recommended-by-google/en