Deezer project review

API

alex jacobs

Contents

[Introduction 1](#_Toc530116835)

[API 1](#_Toc530116836)

[Development process 1](#_Toc530116837)

[User interfaces 2](#_Toc530116838)

[Loading screen 2](#_Toc530116839)

[Home activity 3](#_Toc530116840)

[Search activity 3](#_Toc530116841)

[Kotlin scripts 4](#_Toc530116842)

[Search button 4](#_Toc530116843)

[Gson 4](#_Toc530116844)

[Picasso 4](#_Toc530116845)

[Recycler view 4](#_Toc530116846)

2018

# Introduction

This aim of this report is to explain decisions made during the development process of my API application. The API which I chose to use for this assignment is the Deezer API, “Deezer is an Internet-based music streaming service. It allows users to listen to music content from record labels including Sony Music, Universal Music Group, and Warner Music Group on various devices online or offline” (wikipedia, 2018).

# API

“API’s provide mechanisms for customers to access and manipulate data stored by the API provider” (medium.com, 2018) , in terms of my application what an API allows me to do is request data surrounding Artists, groups and tracks.

The response I will receive will be in the form of a JSON file, the data we receive will be completely dependant on the request sent, for example if I request a JSON file containing all the information surrounding Daft Punk I will receive the response as seen below.

{  
  "id": "[27](javascript:void(0);)",  
  "name": "Daft Punk",  
  "link": "<https://www.deezer.com/artist/27>",  
  "share": "https://www.deezer.com/artist/27?utm\_source=deezer&utm\_content=artist-27&utm\_term=0\_1542318756&utm\_medium=web",  
  "picture": "<https://api.deezer.com/artist/27/image>",  
  "picture\_small": "https://e-cdns-images.dzcdn.net/images/artist/f2bc007e9133c946ac3c3907ddc5d2ea/56x56-000000-80-0-0.jpg",  
  "picture\_medium": "https://e-cdns-images.dzcdn.net/images/artist/f2bc007e9133c946ac3c3907ddc5d2ea/250x250-000000-80-0-0.jpg",  
  "picture\_big": "https://e-cdns-images.dzcdn.net/images/artist/f2bc007e9133c946ac3c3907ddc5d2ea/500x500-000000-80-0-0.jpg",  
  "picture\_xl": "https://e-cdns-images.dzcdn.net/images/artist/f2bc007e9133c946ac3c3907ddc5d2ea/1000x1000-000000-80-0-0.jpg",  
  "nb\_album": 32,  
  "nb\_fan": 3664588,  
  "radio": true,  
  "tracklist": "https://api.deezer.com/artist/27/top?limit=50",  
  "type": "artist"  
}

# Development process

Regarding the development process we used android studio to make the GUI and code the supporting Kotlin files.

“Android Studio is the official integrated development environment (IDE) for Google's Android operating system, built on JetBrains' IntelliJ IDEA software and designed specifically for Android development.” (wikipedia, 2018)”

# User interfaces

The approach I took to user interfaces was to generate wireframes which can be found within the project file on GitHub, I then created the activities for each separate wireframe within android studio. As the application needed very little in terms of navigation and content pages the whole application consist of only three pages.

In terms of responsive design, I used the existing content and resized and moved based on the resolution of the device it being displayed upon. This was done using the parameter view within android studio which displays the different layout based on the resolution provided by the device.

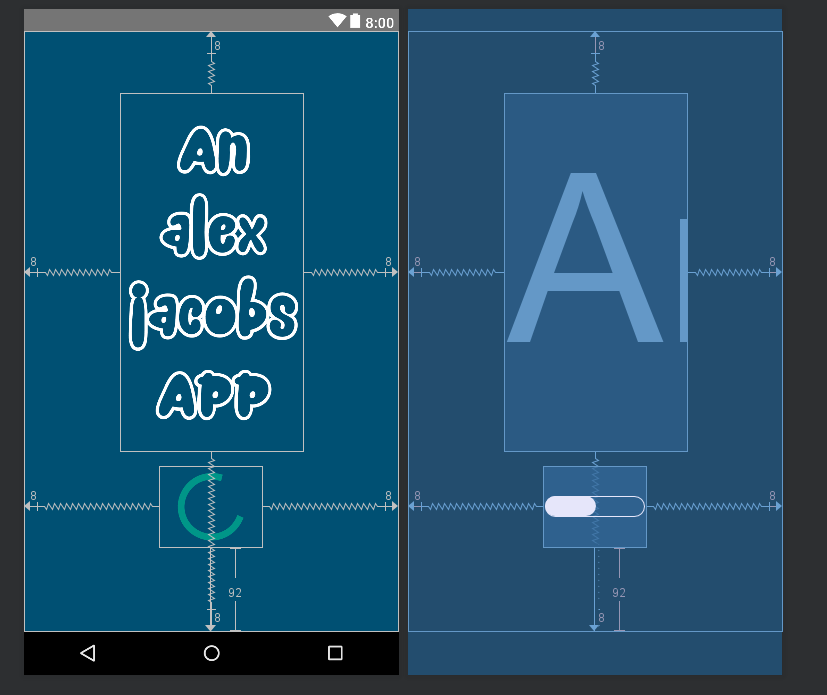
The versions I tried to accommodate for include;

1. Android small 480x800 (Portrait and landscape)
2. Android medium 740x1280 (Portrait and landscape)
3. Android Tablet 2560x1800 (Portrait and landscape)

## Loading screen

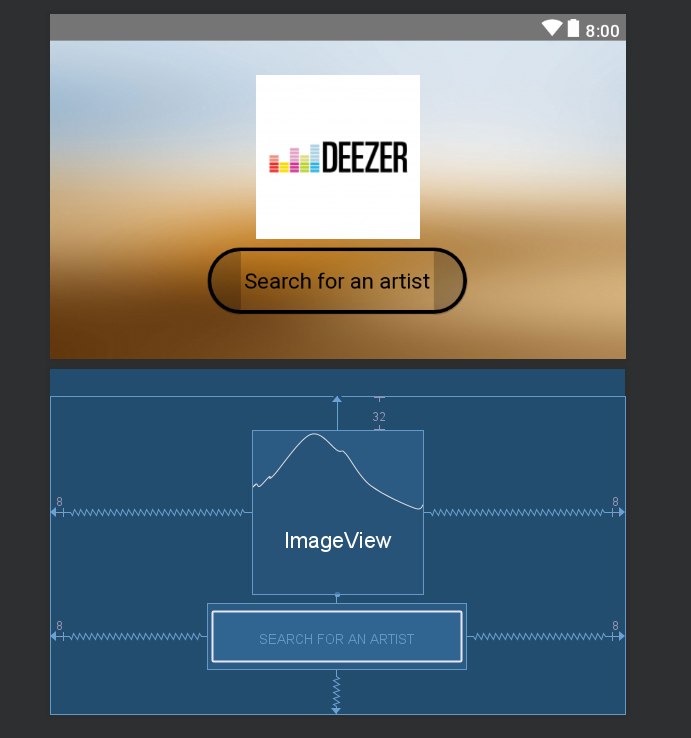
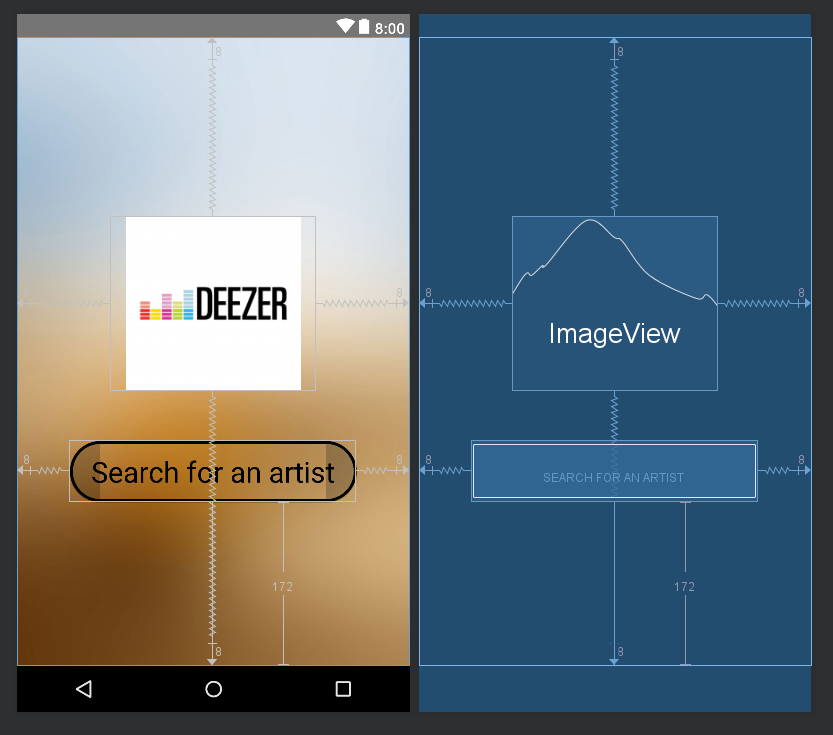
The loading screen consists of a text view with a custom font, and a progress spinner, although the spinner is on a timer instead of waiting to load application files. Getting the timer to work was a difficult task as initially it paused the activity before the UI has loaded then progresses straight onto the next activity without displaying anything.

This activity is locked to portrait within the xml file, so the device only displays in portrait mode, this isn’t really an issue as there are no interactions needed from this page.



## Home activity

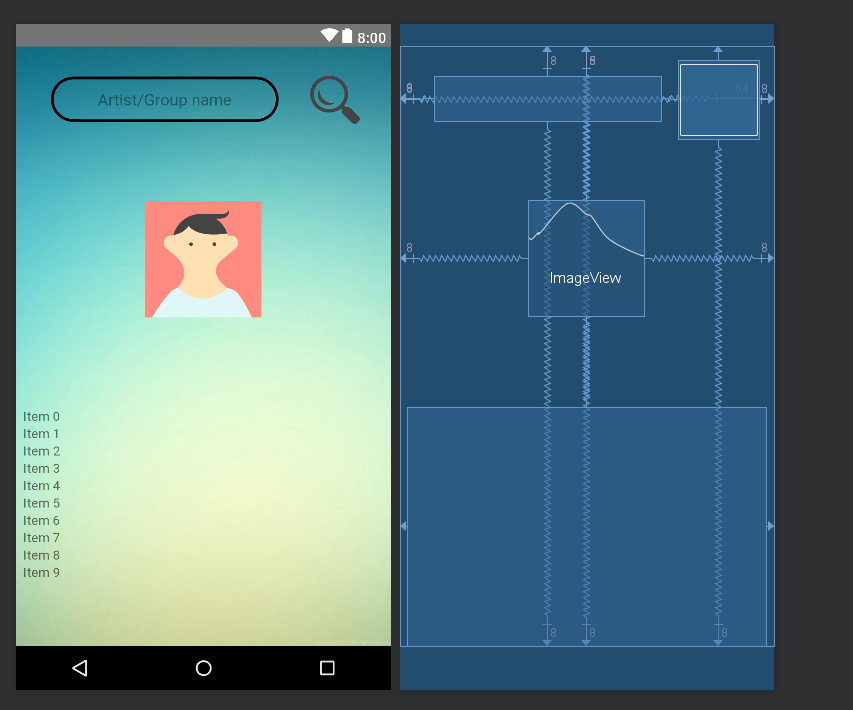
This activity just breaks up the application, it serves as a midpoint between the application loading the search functionality, initially there was also another navigation option taking the user to a forth activity explaining the application but as time progresses adding another activity and coding functionality was not achievable.

This activity just has a single image holder referencing an PNG image and a navigation button which has a background set to a drawable. Although the button doesn’t have a background, I had to use one to allow me to achieve the rounded corner effect I wanted.

This is the first activity I created a landscape design for which continues the same design and features but with a slightly altered layout. Furthermore, this application was also the first activity to support tablet view as well as small and medium devices.

## Search activity

This activity was by far the most complicated of the three activities I created, I designed this initially for medium sized android devices e.g. Pixel 2, however I was unsure how the JSON images would scale so I worked the activity in such a way that the small and large layouts were the final task of the project.



# Kotlin scripts

Kotlin is a [statically typed](https://en.wikipedia.org/wiki/Statically_typed) programming language that runs on the Java virtual machine and also can be compiled to JavaScript source code (wikipedia., 2018)

Kotlin being the primary language used in the creation of this application was an entirely new language for myself but is very similar in terms of syntax to Java.

## Search button

The search button triggers an on click event, this event is the main function within this application, it uses the text within the search box to form a Get request which then pulls the information down based upon the Artist or Group name. the response we receive is a JSON response which needs to be parsed for the data to be useable.

### Gson

“Gson (also known as Google Gson) is an [open source](https://en.wikipedia.org/wiki/Open_source) [Java](https://en.wikipedia.org/wiki/Java_(programming_language)) library to [serialize](https://en.wikipedia.org/wiki/Serialize) and deserialize Java objects to (and from) [JSON](https://en.wikipedia.org/wiki/JSON).” (wikipedia, october ). This library is what enables me to covert raw JSON data into a defined class.

For example, within my application I receive two main json responses these are;

1. Artist, defining an artist class was easy as the Deezer documentation provided me with all the fields within the class and the data types.
2. Top 5 tracks, this data class was slightly more complicated as this response is composed of a class made of the track classes, meaning I had to define tracks before I could parse this JSON file.

## Picasso

Picasso is an image loading library which enables the user to reference image file URLs and use them within a kotlin script. Using this library, I was able to use the artist image JSON response, parse it into a string and reference the image URL and use this within my project.

## Recycler view

Using a recycler view was a complicated idea for me to get my head around, as it involved a lot of back end coding to enable it to work. Recycler view is fundamentally a list that dynamically creates and removes entities based on the current scroll position in order to better manage memory.

The adapter that goes with the recycler view is what binds the content to the view holder and passes the it to the recycler view

# Bibliography

medium.com. (2018, Jan 11). *medium.com*. Retrieved from medium.com: https://medium.com/@TebbaVonMathenstien/what-is-an-api-and-why-should-i-use-one-863c3365726b

wikipedia. (2018, October 30 ). *wikipedia*. Retrieved from wikipedia.org: https://en.wikipedia.org/wiki/Deezer

wikipedia. (2018, october 16). *wikipedia.org*. Retrieved from wikipedia.org: https://en.wikipedia.org/wiki/Android\_Studio

wikipedia. (october , 2018 6). *wikipedia.org*. Retrieved from wikipedia.org: https://en.wikipedia.org/wiki/Gson

wikipedia. (2018, november 14). *wikipedia.org*. Retrieved from wikipedia.org: https://en.wikipedia.org/wiki/Kotlin\_(programming\_language)