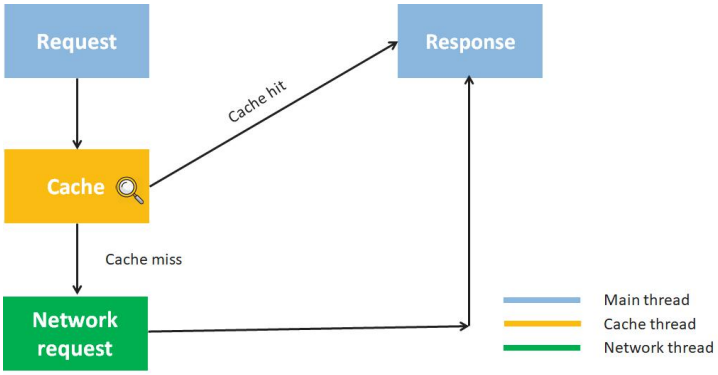
BAB III
TINJAUAN PUSTAKA

3.1. REST API Client

Pada perkembangan teknologi internet, terdapat REST yang merupakan sebuah arsitektur digunakan dalam membangun sebuah web. Teknologi ini bergantung pada Uniform Resource Identifier atau URI pada Hyper Text Transfer Protocol (HTTP) untuk berkomunikasi satu sama lain [1], dimana pada HTTP terdapat metode seperti GET atau POST untuk berkomunikasi. Hingga saat ini kegunaan REST telah menjadi dasar dalam prinsip pembuatan API. Application Programming Interface atau disingkat API merupakan penengah yang digunakan untuk program dan komputer saling berkomunikasi satu sama lain. Dapat disimpulkan bahwa REST API berkomunikasi antara *client* dan *server* melalui sebuah URI dapat terpisah satu sama lain. Dengan adanya perkembangan REST API, hal ini tentu membuka kesempatan bagi aplikasi Android untuk menggunakan teknologi tersebut.

Dalam aplikasi Android terdapat library Volley [2] dan Retrofit [3] yang dapat digunakan sebagai REST API Client. Volley sendiri merupakan *networking library* yang diciptakan oleh Google yang sangat berguna untuk *network request*. Library ini dapat memberikan request secara asinkronus, sinkronus, caching, hingga debugging dan tracing. Dengan banyaknya fitur yang diberikan, pengembang dapat dengan mudah melakukan berbagai macam operasi untuk berkomunikasi dengan API. Volley memiliki arsitektur sebagai berikut .



Gambar 3.1 Arsitektur Volley

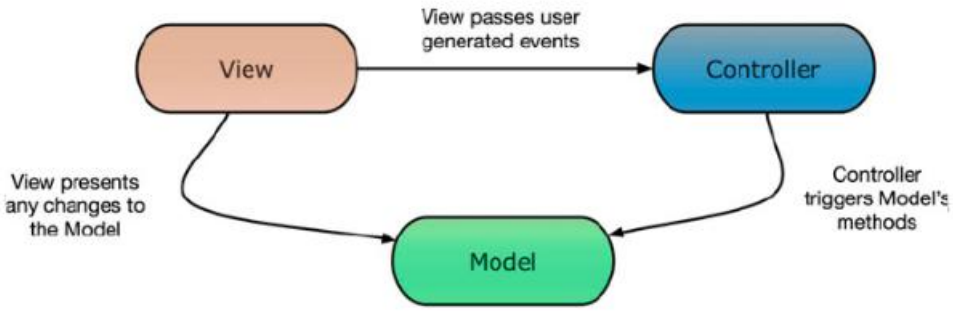
Retrofit merupakan library yang diciptakan oleh Square. Library ini memberikan fitur yang lebih ringan dan mudah digunakan. Retrofit juga dapat melakukan operasi sinkronus, asinkronus dengan automatic JSON scanning yang memudahkan pengguna untuk mengubah response dari API menjadi objek pada pemograman Android. Kedua library ini tentu terdapat kelebihan dan kekurangan yang dibahas pada penelitian berikut [4]. Pengembang aplikasi Android dapat mempertimbangkan dari kedua REST API Client sesuai kebutuhan.

3.2. Arsitektur Pemograman

Dalam pemograman seringkali ditemukan istilah arsitektur pemograman. Arsitektur pemograman sendiri merupakan sebuah struktur yang terdiri dan elemen perangkat lunak, properti dari elemen tersebut yang terlihat, dan hubungan antara hal tersebut [5]. Arsitektur pemograman berguna untuk memberikan gambaran aplikasi yang akan dibuat. Dengan adanya arsitektur pemograman, kemungkinan permasalahan dapat ditemukan. Selain itu, arsitektur dapat memberikan dampak yang signifikan pada kualitas aplikasi agar lebih mudah dimengerti terutama pada perencanaan alokasi *resource* [6].

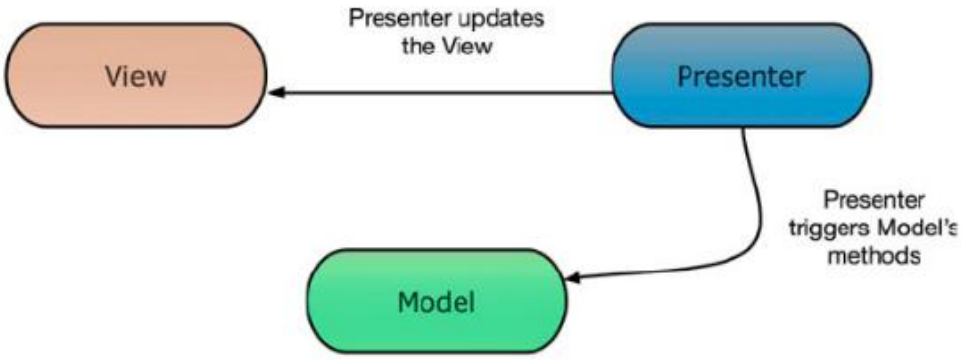
Pada pemograman Android terdapat jenis arsitektur yang seringkali digunakan. Arsitektur tersebut terdiri dari Model View Controller (MVC), Model View Presenter (MVP), dan Model View View Model (MVVM). Ketiga arsitektur ini memiliki kesamaan yaitu pada prinsip yang dimiliki yaitu Three-Tier Application Architecture. Prinsip tersebut terdiri dari Presentation Layer yang memberikan tampilan kepada pengguna, Business

Layer yang mengatur validasi data, dan Data Access Layer yang mengatur sumber data [7]. MVC terdiri dari Model yang berfungsi untuk menampung data, View yang memberikan tampilan data dari Model dan memberikan event kepada Controller, dan Controller penerima event dari pengguna dan menjalankan fungsi pada model.



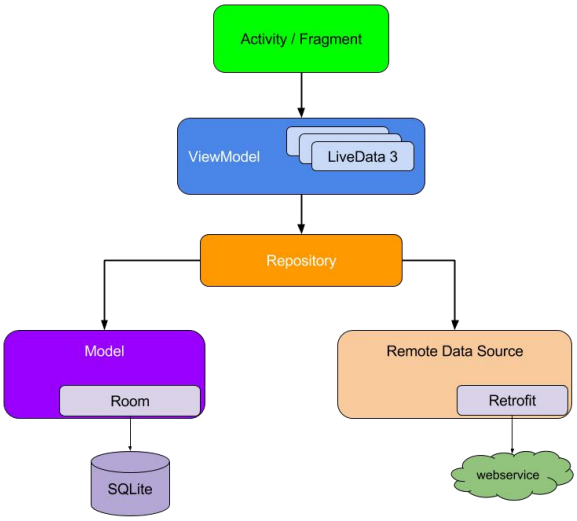
Gambar 3.2 Arsitektur MVC

Berbeda dengan sebelumnya, pada MVP View tidak memiliki akses langsung pada Model dan hal itu juga berlaku untuk Model, oleh karena terdapat Presenter yang mengatur perubahan. Presenter pada arsitektur MVP berperan sebagai otak yang menerima input kemudian mengatur perubahan data pada View.



Gambar 3.3 Arsitektur MVP

Selanjutnya terdapat arsitektur MVVM yang terdiri dari Model, View Model yang menggantikan Presenter dan Controller, serta View. MVVM merupakan arsitektur yang disarankan untuk pengembang aplikasi Android [8]. Berikut tampilan struktur arsitektur MVVM pada panduan pengembang aplikasi Android :



Gambar 3.4 Arsitektur MVVM

Untuk melakukan penerapan arsitektur MVVM, pengembang dapat menggunakan *library* LiveData sebagai penampung data dinamis yang akan mempertahankan nilainya meskipun terdapat perubahan keadaan pada perangkat. Kemudian terdapat ViewModel sebagai penampung data serta mengatur pergantian data. Selanjutnya terdapat Repository yang berfungsi sebagai penghubung serta mengatur aliran data dari sumber. Dengan menggunakan Repository, pengembang dapat mengatur sumber data mana yang dapat digunakan.

3.3. API Testing

Selama melakukan pengembangan pada aplikasi, pengembang perlu melakukan percobaan untuk memastikan sistem berjalan dengan lancar. Percobaan ini diperlukan terutama dalam sistem API yang merupakan hal terpenting dalam pembuatan sebuah aplikasi online. Untuk melakukan percobaan pada API dapat digunakan API Client yang didesain khusus. Salah satu API Client tersebut adalah Postman. Postman memberikan panduan untuk membuat request, melakukan testing otomatis, hingga memonitor API yang digunakan [9]. API Client ini juga memberikan fitur seperti Workspace yang memungkinkan untuk pengguna saling menghubungkan Postman-nya satu sama lain.

3.4. User Interface and User Experience Design

Dalam mengembangkan suatu aplikasi, seorang pengembang perlu mengetahui konsep desain tampilan aplikasi. Dalam pengembangan aplikasi Android hal ini menjadi konsep dasar yang perlu diketahui saat melakukan desain tampilan layout. Untuk membantu pengembang, Google membuat sebuah laman yang bernama Material Design [10]. Laman tersebut menyediakan prinsip desain tampilan aplikasi hingga komponen berupa library yang memiliki contoh implementasi untuk program Android Studio. Material Design juga memberikan contoh aplikasi yang memiliki desain *best-case* untuk pembelajaran desain.

3.5. Android Studio

Untuk membuat sebuah aplikasi Android, diperlukan sebuah aplikasi ataupun framework. Salah satu aplikasi tersebut adalah Android Studio. Android Studio merupakan sebuah Integrated Development Environment (IDE) yang dibuat berdasarkan program IntelliJ IDEA [11]. Program Android Studio mendukung berbagai macam fitur dimulai dari Layout Editor yang berfungsi untuk membuat tampilan laman aplikasi secara interaktif, Emulator sebagai emulasi aplikasi Android yang sedang dibuat, hingga Intelligent Code Editor yang dapat melakukan *autocomplete* pada proses menulis kode.

3.6. Navigation Component

Navigation Component merupakan salah satu *library* pada Android Studio untuk membantu sistem navigasi antar laman pada aplikasi [12]. Library ini terdiri dari 3 komponen yaitu :

- Navigation Graph yang memberikan tampilan grafis mengenai hal - hal yang berhubungan dengan Navigation seperti salah satu laman aplikasi hingga alur antar laman.
- NavHost merupakan kontainer yang menampilkan laman tujuan yang ada pada Navigation Graph.

- NavController yang digunakan untuk mengatur navigasi aplikasi yang ada pada NavHost.

3.7. RxJava

Untuk melakukan penerapan LiveData pada arsitektur dapat digunakan *library* tambahan yaitu RxJava. RxJava berfungsi untuk membantu membuat suatu sistem yang asinkronus dan bergantung pada *event* menggunakan *observable sequence* [13]. Dengan menggunakan RxJava, aplikasi dapat dengan mudah memproses data dari API untuk menjadi LiveData. Tak hanya mempermudah proses perubahan data, RxJava juga memiliki level abstraksi yang lebih baik untuk memanggil suatu API.