Analisis Perbandingan Performa Pola Arsitektur MVC dan MVVM dalam Mengeksekusi Operasi

Andrew Kurniawan Gianto
Informatika
Pradita University
Tangerang, Indonesia
andrew.kurniawan@student.pradita.ac.id

Ryo Ferdinand
Informatika
Pradita University
Tangerang, Indonesia
ryo.ferdinand@student.pradita.ac.id

Eka Rifail Saipuddin Rachman
Informatika
Pradita University
Tangerang, Indonesia
eka.rifail@student.pradita.ac.id

Verrel Novendra Sulu
Informatika
Pradita University
Tangerang, Indonesia
verrel.novendra@student.pradita.ac.id

Hezel Anthonie Norman Piter Papia
Informatika
Pradita University
Tangerang, Indonesia
hezel.anthonie@student.pradita.ac.id

***Abstract*—Abstrak akan diisi setelah makalah sudah lengkap dengan pendahuluan, kajian terkait, metodologi, hasil dan pembahasan, serta kesimpulannya.**

***Index Terms*—MVC, MVVM, kecepatan, memori**

I. Pendahuluan

Perangkat lunak telah menjadi dasar bagi banyak aplikasi dan sistem yang digunakan dalam berbagai hal dalam era teknologi informasi yang terus berkembang. Karena itu, perangkat lunak dibangun agar tidak hanya beroperasi dengan efisien, tetapi juga dapat berkembang dan berubah. Pengembang harus mengatur kode mereka sehingga mudah dipahami dan diselesaikan karena sistem yang kompleks dan masalah berulang. Pola desain antarmuka pengguna dapat membantu mengurangi kekacauan sistem yang kompleks. Penggunaan pola ini akan membuat aplikasi lebih terorganisir dan dapat diskalakan, sehingga tidak terelakkan untuk mempelajari pola antarmuka pengguna dan membuat keputusan yang tepat tentang pola mana yang paling cocok [1]. *Model-View-Controller* (MVC) dan *Model-View-ViewModel* (MVVM) adalah dua arsitektur yang akan dibahas dalam artikel ini.

Tujuan dari penelitian ini adalah untuk membandingkan kinerja arsitektur MVC dengan MVVM, khususnya dalam hal kecepatan dan penggunaan memori. MVC telah menjadi arsitektur yang paling umum digunakan dalam pengembangan perangkat lunak dengan metode prototype, sedangkan MVVM adalah arsitektur yang paling umum digunakan dalam pengembangan aplikasi mobile [1]. Penelitian akan menentukan arsitektur mana yang paling efisien dan paling sesuai dengan berbagai kebutuhan pengembangan perangkat lunak melalui eksperimen untuk masing-masing *Model-View* (MV).

Penelitian ini diharapkan dapat membantu pengembang perangkat lunak memahami kelebihan dan kelemahan arsitektur melalui eksperimen yang dapat memberikan pemahaman yang lebih baik. Dengan demikian, pengembang dapat memperoleh informasi yang lebih baik dan membuat keputusan yang lebih baik dalam memilih arsitektur yang tepat untuk proyek mereka, sehingga menghasilkan aplikasi yang lebih baik, efisien, dan dapat diandalkan.

II. Kajian Terkait

A. Model-View-Controller (MVC)

Model-View-Controller atau MVC adalah arsitektur perangkat lunak untuk membuat sebuah aplikasi dengan memisahkan data (*Model*) dari tampilan (*View*) dan cara bagaimana memprosesnya (*Controller*) [1] [3] . MVC menekankan tiga elemen penting: perhatian, tanggung jawab, dan logika. Ini mempercepat kinerjanya.

Berikut adalah penjelasan dari masing-masing komponen:

a Bagian *model* biasanya digunakan untuk mengambil data dari database atau menyimpan data ke dalam database.
b Bagian *view* adalah komponen yang menampilkan antarmuka pengguna aplikasi, yang dibangun berdasarkan data model.
c Bagian *control* adalah bagian yang menangani interaksi pengguna, bekerja dengan model, dan menyimpan data ke dalam database.

1) Keuntungan:

- Pemisahan tanggung jawab antara *Model, View,* dan *Control* membuat aplikasi lebih mudah dipahami, diuji, dan dikembangkan.
- Fleksibilitas dalam mengganti atau memodifikasi salah satu komponen tanpa memengaruhi komponen lainnya.

- Memungkinkan pengembangan paralel antara komponen-komponen aplikasi.

2) Kerugian:

- Adanya kompleksitas dalam pengelolaan koneksi antara 3 komponen tersebut.
- Adanya kesulitan dalam pengujian unit pada *View* dan *Controller*.
- Kesulitan dalam menangani situasi di mana komponen-komponen tersebut harus berinteraksi secara kompleks.

B. Model-View-View-Model (MVVM)

Model View ViewModel (MVVM) adalah sebuah arsitektur perangkat lunak yang memisahkan antara kode untuk logika bisnis dan tampilan aplikasi [2]. Dalam kata lain, MVVM adalah sebuah pattern desain yang terdiri dari tiga komponen: *Model, View, dan ViewModel* [6].

Berikut adalah penjelasan dari masing-masing komponen:

a Bagian *model* merupakan representasi dari data yang digunakan dalam logika bisnis.

b Bagian *view* adalah komponen yang terdiri dari layout sumber daya file dan aktivitas/fragmen. Activity/Fragment secara dinamis mengontrol tampilan pada layout sumber daya file.

c Bagian *ViewModel* berinteraksi dengan Model dan menyiapkan variabel yang akan diamati oleh *View*. *ViewModel* bersifat lifecycle-aware, sehingga kelas ini akan hidup ketika sebuah kelas View telah melalui tahapan create dan belum melalui tahapan destroy.

1) Keuntungan:

- Pemisahan antara *View* dan *ViewModel* sehingga membuat pengujian dan pemeliharaan kode lebih mudah
- Kode MVVM biasanya digerakkan oleh peristiwa, yang berarti kode dapat diuji secara terpisah.

2) Kerugian:

- Komponen antarmuka MVVM harus dibuat observable yang berarti dapat memakan waktu dan rumit.
- Banyak kode yang harus ditulis dan dikelola untuk setiap komponen UI sehingga aplikasi dapat menjadi lebih sulit untuk digunakan dan dipahami.

C. Perbandingan MVC dan MVVM

Dalam penelitian "Perbandingan Kinerja Pola Perancangan MVC, MVP, dan MVVM Pada Aplikasi Berbasis Android (Studi Kasus: Aplikasi Laporan Hasil Belajar Siswa SMA BSS)" [5], dibahas analisis perbandingan tiga pola perancangan arsitektur perangkat lunak yang populer: Model-View-Controller, Model-View-Presenter, dan Model-View-ViewModel (MVVM). Aplikasi laporan hasil belajar siswa SMA Brawijaya Smart School digunakan sebagai basis untuk penelitian ini. Tujuan dari penelitian ini adalah untuk menemukan pola perancangan yang paling hemat energi dan memori untuk aplikasi Android. Penelitian dimulai dengan rekayasa untuk menentukan persyaratan fungsional dan non-fungsional. Persyaratan ini kemudian digunakan sebagai dasar untuk desain dan pelaksanaan penelitian. Ketiga pola desain ini digunakan untuk memulai implementasi Java. Setelah tahap implementasi, setiap aplikasi yang menggunakan pola ini diuji. Pengujian dilakukan sebanyak lima kali untuk mendapatkan data penggunaan energi dan memori. Hasil pengujian menunjukkan bahwa ketiga pola perancangan memiliki penggunaan energi yang sederhana, dengan penggunaan memori rata-rata 59,7 MB untuk MVC, 59 MB untuk MVP, dan 73,2 MB untuk MVVM. Selain itu, tahap pengujian fungsional dengan metode pengujian blackbox memberikan hasil validitas sebesar 100% untuk semua fungsi.

Dalam penelitian "A Comparison of Android Native App Architecture—MVC, MVP, and MVVM" [4], dibahas mengenai masalah efisiensi dan kualitas pengembangan aplikasi Android yang dipengaruhi oleh arsitektur aplikasi. Tujuan dari artikel ini adalah untuk memberikan analisis menyeluruh untuk menentukan apakah arsitektur MVP (Model-View-Presenter) dan MVVM (Model-View-ViewModel) lebih unggul dari arsitektur aplikasi Android native bawaan, yaitu MVC (Model-View-Controller). Dengan menggunakan Metode Analisis Tradeoff Arsitektur untuk menjawab pertanyaan ini dan menetapkan tiga kriteria: ketepatan, modifikasi, dan kinerja. Berdasarkan kriteria ini, mereka mengenali faktor-faktor penting untuk setiap atribut kualitas dan melakukan perbandingan. Hasil analisis dan eksperimen menunjukkan bahwa MVP dan MVVM lebih unggul daripada MVC dalam hal testability, modifiability (tingkat coupling yang rendah), dan performance (mengonsumsi memori yang lebih sedikit).

Dalam penelitian "Performance Comparison of Native Android Application on MVP and MVVM" [7], dibahas analisis komparatif terhadap arsitektur MVP (*Model-View-Presente*r) dan MVVM (*Model-View-ViewModel*) dalam konteks pengembangan aplikasi Android asli dilakukan dalam makalah yang ditulis oleh Wisnuadhi, Munawar, dan Wahyu. Tujuan dari penelitian ini adalah untuk mengevaluasi kinerja kedua arsitektur tersebut dengan mempertimbangkan tiga faktor: penggunaan CPU, penggunaan memori, dan waktu eksekusi. Hasil eksperimen menunjukkan bahwa arsitektur MVVM lebih baik dalam penggunaan CPU dan waktu eksekusi, sedangkan MVP unggul dalam penggunaan memori. Aplikasi MVVM memiliki penggunaan CPU yang lebih rendah dengan perbedaan rata-rata 0,55% dan waktu eksekusi yang lebih cepat dengan perbedaan rata-rata 126.21 ms. Aplikasi MVP juga memiliki penggunaan memori yang lebih rendah dengan perbedaan rata-rata 0.92 Mb1.

III. Metodologi

IV. Hasil dan Pembahasan

V. Kesimpulan

References

[1] Gloria Arcos-Medina, Jorge Menéndez, and Javier Vallejo. Comparative study of performance and productivity of mvc and mvvm design patterns. *KnE Engineering*, pages 241–252, 1 2018.

[2] Arief Rahman Fajri and Septia Rani. Penerapan design pattern mvvm dan clean architecture pada pengembangan aplikasi android (studi kasus: Aplikasi agree partner). *Automata: Disemasi Tugas Akhir Mahasiswa*, (16):8, 2022.

[3] Laberto Kelen. Implementasi model-view-controller (mvc) pada ujian online melalui penerapan framework codeigniter. *Jurnal Pendidikan Teknologi Informasi (JUKANTI)*, 1(1):10–16, Mar. 2018.
[4] Tian Lou. A comparison of android native app architecture – mvc, mvp and mvvm. 2016.
[5] Bahrur Rizki Putra Surya, Agi Putra Kharisma, and Novanto Yudistira. Perbandingan kinerja pola perancangan mvc, mvp, dan mvvm pada aplikasi berbasis android (studi kasus : Aplikasi laporan hasil belajar siswa sma bss). *Jurnal Pengembangan Teknologi Informasi dan Ilmu Komputer*, 4(11):4089–4095, Nov 2020.
[6] Irya Muhammad Riyadhi, Intan Purnamasari, and Kamal Prihandani. Penerapan pola arsitektur mvvm pada perancangan aplikasi pengaduan masyarakat berbasis android. *INFOTECH journal*, 2023.
[7] Bambang Wisnuadhi, Ghifari Munawar, and Ujang Wahyu. Performance comparison of native android application on mvp and mvvm. pages 276–282, 2020.