Analisis Perbandingan Pola Arsitektur MVI vs. MVVM: Evaluasi Kinerja

1st Yohanes Kenny
Pradita University
Tangerang, Indonesia
yohanes.kenny@student.pradita.ac.id

2nd Muhammad Abdi Reinanda
Pradita University
Tangerang, Indonesia
muhammad.abdi@student.pradita.ac.id

3rd Joaquin Camilo Tololiu
Pradita University
Tangerang, Indonesia
joaquin.camilo@student.pradita.ac.id

4th Revaldo Sugianto
Pradita University
Tangerang, Indonesia
revaldo.sugianto@student.pradita.ac.id

5th Ronald Bryan Alfredo
Pradita University
Tangerang, Indonesia
ronald.bryan@student.pradita.ac.id

Abstract—Lorem Ipsum is simply dummy text of the printing and typesetting industry. Lorem Ipsum has been the industry's standard dummy text ever since the 1500s, when an unknown printer took a galley of type and scrambled it to make a type specimen book. It has survived not only five centuries, but also the leap into electronic typesetting, remaining essentially unchanged. It was popularised in the 1960s with the release of Letraset sheets containing Lorem Ipsum passages, and more recently with desktop publishing software like Aldus PageMaker including versions of Lorem Ipsum.

Index Terms—MVI, MVVM, pola arsitektur, evaluasi kinerja, kecepatan, memori.

I. Pendahuluan

Pendahuluan kurang panjang, tambahkan seperti sejarah MVVM sejarah MVI ataupun hal lain sebagai landasan mengapa penelitian ini dilakukan

Dalam pengembangan perangkat lunak, pemilihan pola arsitektur yang tepat memainkan peran krusial dalam menentukan kualitas, keterbacaan, dan kinerja keseluruhan aplikasi. Dua pola arsitektur yang sering dibandingkan adalah Model-View-Intent (MVI) dan Model-View-ViewModel (MVVM). MVI menekankan pada unidirectional data flow dan pengelolaan kejadian (events), sementara MVVM fokus pada pemisahan antara tampilan (view) dan logika bisnis (view model).

Penelitian ini bertujuan untuk mengidentifikasi dan memahami perbedaan kinerja antara MVI dan MVVM dalam konteks pengembangan aplikasi Android. Tujuan utama dari eksperimen ini adalah untuk mengevaluasi dan membandingkan performa kedua pola arsitektur tersebut dalam aspek kecepatan dan penggunaan memori. Dengan pemahaman yang mendalam tentang kelebihan dan kekurangan masing-masing pendekatan, pengembang perangkat lunak akan dapat membuat keputusan yang lebih terinformasi dalam memilih pola arsitektur yang sesuai dengan kebutuhan aplikasi mereka.

Melalui eksperimen ini, kami berharap dapat memberikan kontribusi yang berharga dalam penelitian tentang pola arsitektur perangkat lunak dan membantu pengembang untuk memilih pola arsitektur yang optimal sesuai dengan tujuan dan karakteristik proyek mereka.

Hindari pengunaan kata saya, kami, sebaiknya gunakan kata penulis

II. Kajian Terkait

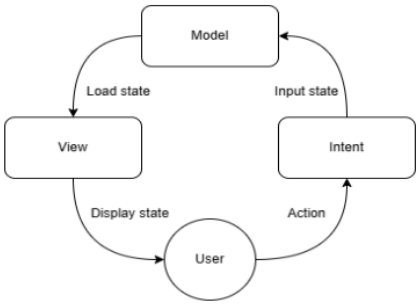
A. MVI

MVI adalah sebuah pola desain yang menitikberatkan pada abstraksi tugas View terhadap Model, sehingga dalam konteks ini, View dianggap sebagai representasi langsung dari Model.

MVI menitikberatkan perannya pada Model sebagai Sumber Kebenaran dari View. Karena Model tidak memiliki perilaku dan mudah didistribusikan [1], replikasi View dapat dilakukan tanpa memengaruhi keadaan (state) dari setiap salinan secara bersamaan.

Pola MVI sepenuhnya menyerahkan proses representasi Model kepada View, terutama dalam konteks Fragment. Model yang diterima oleh Fragment dari ViewModel bersifat Immutable, menjadikan Model sebagai abstraksi dari View itu sendiri [4]. Meskipun pendekatan ini lebih sederhana dalam interaksi daripada MVVM, namun kelemahan MVI terletak pada kemungkinan sebagian Model berada di luar kendali Fragment. Hal ini menyebabkan Fragment kesulitan dalam mengubah Model tanpa melewati serangkaian proses yang kompleks. Perbedaan kewenangan antara Model dan Fragment juga sulit dideskripsikan secara jelas oleh Model itu sendiri, setidaknya sampai saat ini.

Salah satu keunggulan utama dari MVI adalah penerapan prinsip Single Source of Truth (SSOT), di mana kode inti ditempatkan dalam satu file. Hal ini menyebabkan penulisan program yang lebih sederhana dan dapat dimodifikasi dengan mudah, serta memastikan bahwa semua data dapat diakses dengan tepat, yang nantinya akan mempermudah proses perawatan (maintenance) di masa mendatang. [2].MVI memiliki keunggulan yang unik dalam konsep Separation of Concernsnya, yang memungkinkan



Gambar 1. Alur arsitektur MVI

penyederhanaan manajemen keadaan (state management) dalam aplikasi program.

B. MVVM

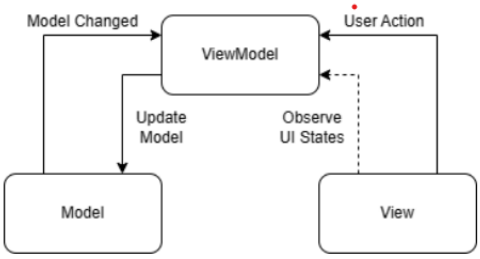
Model-View-View-Model (MVVM) adalah paradigma desain arsitektur baru untuk menangani persetujuan pengguna saat mengembangkan aplikasi Android yang berfungsi sebagai antarmuka pengguna (back-end antarmuka pengguna itu front end...). MVVM juga merupakan varian modern dari Model-View-Controller (MVC) dan tujuan intinya adalah memisahkan antara komponen Model dan komponen View [5]. Sejak Google pertama kali mengumumkan MVVM pada Google I/O 2018, Google selalu mendorong setiap developer untuk mengerjakan aplikasi Android Studio. [3]

ViewModel tidak dapat ditansaksikan. Hal ini memberikan konsekuensi dimana state dari View (secara design) tidak dapat dicopy atau didistribusikan [6]. Oleh karena itu merepresentasikan ulang state dari View bisa menjadi sebuah tantangan. Walaupun secara teknis hal ini dapat dilakukan, Akan tetapi menggunakan Shared ViewModel yang Mutable membuat source of truth dipertanyakan kebenarannya.

MVVM diclaim memudahkan kita untuk melakukan view management. Arsitektur MVVM memiliki keunggulan dalam modifiabilitas [1]. Dengan ViewModel mengatakan pada Fragment bagaimana dia harus merepresentasikan model, manajemen View memang bisa menjadi lebih mudah. Akan tetapi hal ini kadang membuat ViewModel menjadi bulky dan complex. Bahkan bisa lebih kompleks lagi jika kita berbicara mengenai Shared ViewModel. Konsekuensi dan batasan bagaimana ViewModel harus berperilaku seringnya tidak jelas. Secara prinsib, Fragment seharusnya tidak perlu lagi memikirkan bagaimana View harus di representasikan, akan tetapi pada praktiknya, beberapa Event dan Interaksi yang kompleks lebih mudah dihandle langsung oleh Fragment tanpa melalui ViewModel. Hal tersebut menjadi Gray Area antara tugas Fragment dan ViewModel vs Effort dan Kompleksitas, dan pada akhirnya kita memiliki 2 buah class yakni Fragment dan ViewModel yang sama-sama kompleks, dengan Gray Area yang dilematis.

Komponen dalam MVVM adalah:

1) Model
   Bagian ini mewakili inti dari logika bisnis dan data. Ini berarti bahwa logika bisnis menentukan cara data dimanipulasi dan disimpan dalam Model [5].
2) View
   Komponen ini merupakan representasi antarmuka pengguna dan pada dasarnya berisi elemen antarmuka seperti layout XML dan sejenisnya [5].
3) ViewModel
   ViewModel adalah elemen kunci dalam arsitektur ini karena membantu memisahkan tampilan dari logika bisnis. Ini bertindak sebagai penghubung antara View dan Model, menjaga keduanya terpisah namun memungkinkan interaksi dan koordinasi di antara keduanya. ViewModel juga menyimpan instruksi dan metode untuk mempertahankan status tampilan dan mengelola Model sesuai dengan tindakan yang dilakukan pada View. Selain itu, ViewModel juga memfasilitasi pemicuan peristiwa di dalam View itu sendiri [5].



Gambar 1. Alur arsitektur MVVM

III. Metodologi

Lorem Ipsum is simply dummy text of the printing and typesetting industry. Lorem Ipsum has been the industry's standard dummy text ever since the 1500s, when an unknown printer took a galley of type and scrambled it to make a type specimen book. It has survived not only five centuries, but also the leap into electronic typesetting, remaining essentially unchanged. It was popularised in the 1960s with the release of Letraset sheets containing Lorem Ipsum passages, and more recently with desktop publishing software like Aldus PageMaker including versions of Lorem Ipsum.

References

[1] Firmansyah Firdaus Anhar, Made Hanindia Prami Swari, and Firza Prima Aditiawan. Analisis perbandingan implementasi clean architecture menggunakan mvp, mvi, dan mvvm pada pengembangan aplikasi android native. Jupiter: Publikasi Ilmu Keteknikan Industri, Teknik Elektro dan Informatika, 2(2):181–191, 2024.

[2] Muh Alif Al Gibran Arif, Dana Sulistyo Kusumo, and Shinta Yulia Puspitasari. Optimasi pengembangan aplikasi cross-platform berbasis flutter menggunakan pendekatan arsitektur model mvi (model-view-intent). eProceedings of Engineering, 8(5), 2021.

[3] Fikri Maulana, Rita Afyenni, and Aldo Erianda. Aplikasi manajemen laboratorium menggunakan metode mvvm berbasis android. JITSI: Jurnal Ilmiah Teknologi Sistem Informasi, 3(3):88–93, 2022.
[4] Robin Nunkesser. Choosing a global architecture for mobile applications. Authorea Preprints, 2023.
[5] Naufal Rasyid and Aaqila Dhiyaanisafa Goenawan. Sistem pencatatan keuangan aplikasi android dengan menggunakan design pattern model-view-view-model (mvvm), room database, sharedpreference dan api. Jurnal Teknik Mesin, Industri, Elektro dan Informatika, 1(2):52–63, 2022.
[6] Yudistiro Septian Saputro et al. IMPLEMENTASI ANDROID ARCHITECTURE COMPONENTS DENGAN PATTERN MVVM (MODEL-VIEW-VIEW MODEL) PADA APLIKASI PELAPORAN KEMISKINAN DI KABUPATEN BANTUL. PhD thesis, STMIK AKAKOM Yogyakarta, 2020.