Perbandingan Arsitektur *Model-View-Intent* (MVI) dan *Model-View-Controller* (MVC) dalam Pengembangan Perangkat Lunak

1 Felicia Audrey Emmanuel
Jurusan Informatika
Universitas Pradita
Tangerang Selatan, Indonesia
felicia.audrey@student.pradita.ac.id

2 David Tulus Halomoan Haryanto
Jurusan Informatika
Universitas Pradita
Tangerang Selatan, Indonesia
david.tulus@student.pradita.ac.id

3 Anantaujas Cipta Adinata
Jurusan Informatika
Universitas Pradita
Tangerang Selatan, Indonesia
anantaujas.cipta@student.pradita.ac.id

4 William Kent
Jurusan Informatika
Universitas Pradita
Tangerang Selatan, Indonesia
william.kent@student.pradita.ac.id

5 Michael Christian Yehuda PutraLeytha
Jurusan Informatika
Universitas Pradita
Tangerang Selatan, Indonesia
michael.christian.yehuda@student.pradita.ac.id

***Abstrak*—**
***Kata kunci*—**

I. Pendahuluan

Pola arsitektur sangat penting dalam bidang pengembangan perangkat lunak karena pola tersebut memainkan peran penting dalam menentukan struktur dan perilaku setiap aplikasi. Model View Controller (MVC) dan Model View Intent (MVI) adalah dua paradigma arsitektur terkemuka yang telah diadopsi secara luas dalam beberapa tahun terakhir. Salah satu paradigma ini dikenal sebagai Model View Controller. Membangun solusi perangkat lunak yang terukur, dapat dipelihara, dan efisien dapat dicapai dengan bantuan arsitektur ini, yang menyediakan kerangka kerja terorganisir bagi pengembang.

Sejak didirikan pada akhir tahun 1970an, MVC telah menjadi komponen yang sangat diperlukan dalam bidang rekayasa perangkat lunak [6]. Akarnya dapat ditelusuri kembali ke Smalltalk. Hal ini dilakukan dengan membagi aplikasi menjadi tiga komponen yang saling berhubungan: Model, yang bertugas mengelola data dan logika bisnis; Tampilan, yang bertugas menyajikan antarmuka pengguna; dan Pengontrol, yang bertindak sebagai perantara antara Model dan View, menangani masukan pengguna dan memperbarui Model sesuai kebutuhan [3]. Memisahkan permasalahan ini satu sama lain akan membantu mendorong modularitas dan membuat pemeliharaan dan pengujian menjadi lebih sederhana. Selain itu, MVI adalah pola arsitektur yang relatif baru yang mendapatkan daya tarik, khususnya dalam konteks pemrograman reaktif dan paradigma fungsional. Hal ini karena MVI merupakan pola yang dikembangkan relatif baru [9].

MVI menekankan pada kekekalan dan aliran data yang dapat diprediksi di seluruh aplikasi. Fondasinya didasarkan pada prinsip aliran data searah. Model, yang bertanggung jawab untuk mewakili keadaan aplikasi saat ini, Tampilan, yang bertanggung jawab untuk merender antarmuka pengguna berdasarkan keadaan saat ini, dan *Intent*, yang merangkum tindakan pengguna sebagai peristiwa yang tidak dapat diubah yang memicu pembaruan keadaan, adalah tiga inti komponen yang membentuk sistem ini [[5]. Melalui penggunaan pendekatan ini, gaya pemrograman yang lebih deklaratif dan dapat diprediksi didorong, yang bermanfaat untuk pengembangan aplikasi yang sangat reaktif dan terukur. Karakteristik kinerja MVC dan MVI terus menjadi topik diskusi dan penyelidikan, meskipun faktanya MVC dan MVI menawarkan keunggulan menarik dalam hal pemeliharaan, skalabilitas, dan pemisahan perhatian. Ada kemungkinan besar bahwa efektivitas pola arsitektur dapat berdampak signifikan terhadap pengalaman pengguna secara keseluruhan, khususnya di lingkungan dengan sumber daya terbatas, seperti perangkat seluler atau browser web [4].

Pengembang perangkat lunak sangat perlu memahami dan mengoptimalkan kinerja aplikasi yang dibangun menggunakan pola arsitektur berbeda, itulah sebabnya penelitian bertajuk ”Unveiling Efficiency: Analyzing Performance in Model View Controller (MVC) vs MVI (Model View Intent)” sangat penting. Pengembang berada di bawah tekanan yang semakin besar untuk menyediakan perangkat lunak yang tidak hanya berfungsi sesuai harapan namun juga berjalan lancar di semua perangkat dan platform seiring kemajuan teknologi dan ekspektasi pengguna yang semakin meningkat [7]. Selain itu, lingkungan dengan sumber daya terbatas menjadi semakin umum, terutama dalam konteks aplikasi seluler dan web, yang menyoroti pentingnya penelitian ini.

Meningkatnya jumlah perangkat seluler dan perangkat *Internet of Things* (IoT) menjadikan pengoptimalan kinerja aplikasi menjadi semakin penting guna memberikan pengala-

man pengguna yang lancar sekaligus meminimalkan beban pada sumber daya sistem seperti memori, CPU, dan masa pakai baterai [1]. Penelitian ini memberikan wawasan praktis kepada pengembang untuk mengatasi masalah kinerja dan meningkatkan kualitas aplikasi dengan menganalisis metrik kinerja penting seperti daya tanggap, pemanfaatan sumber daya, skalabilitas, dan efisiensi baterai. Selain itu, kemampuan pengujian dan kemampuan debug menjadi lebih penting seiring dengan semakin kompleks dan terhubungnya sistem perangkat lunak [5].

II. Kajian Terkait

A. MVC

Desain MVC merupakan salah satu desain yang paling penting dalam bidang ilmu komputer [2].

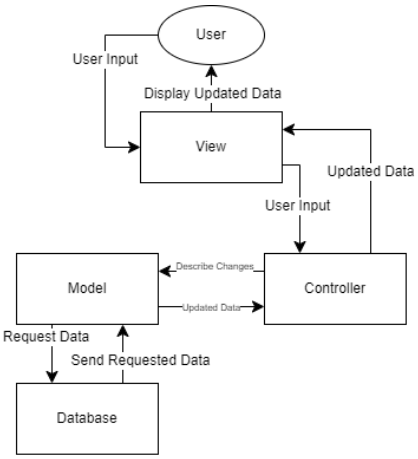

Fig. 1. Arsitektur MVC

MVC digunakan untuk membagi sebuah aplikasi menjadi tiga bagian, yaitu Model, View, dan Controller [10]. Model digunakan untuk pemrosesan data dan hubungan dengan database. Model memberikan data kepada View melalui Controller tanpa mempertimbangkan bagaimana data tersebut dipresentasikan. Bila menerima sebuah notifikasi perubahan atau terjadi sebuah input dari pengguna, Model akan memberi View data yang lebih baru untuk ditampilkan. Tugas Controller adalah untuk menampilkan view berdasarkan input pengguna. Controller menghubungkan Model dengan View. Sebuah input atau pergerakan pada lapisan View akan ditangkap dan diberitakan kepada Controller. Controller akan berinteraksi dengan Model untuk mendapatkan data yang telah diperbaharui dan meneruskan data tersebut pada View. Tugas View adalah untuk menampilkan data tanpa mementingkan proses lain seperti koneksi kepada database. View mempresentasikan data dalam bentuk yang telah diformat dan dapat dibaca oleh pengguna.

MVC berguna untuk sistem yang interaktif, karena memiliki sifat partition-independent, yang berarti bahwa komponen utama dalam MVC dapat beroperasi secara independen satu sama lain, tanpa ada ketergantungan yang kuat di antara mereka. Ini memungkinkan pengembang untuk membuat perubahan tanpa harus khawatir tentang efeknya terhadap komponen yang lain.

MVC, walau bersifat partition-independent, mengalami kesulitan dalam menjalankan proses yang menyebar lapisan pada lokasi yang berbeda, contohnya dengan web application. Pada web application, lapisan View akan berada di device pengguna, namun Controller dan Model bertetap pada server, yang mengakibatkan munculnya permasalahan location-dependant [8]. Komunikasi pada jaringan, pengiriman data, dan permasalahan dalam sinkronisasi membuat implementasi MVC pada aplikasi berbasis web lebih sulit.

B. MVI

Arsitektur MVI, merupakan sebuah arsitektur yang menerapkan aliran data searah (unidirectional flow). Model adalah keadaan sebuah aplikasi, yang masih dalam bentuk sebuah data mentah. View adalah representasi keadaan sebuah aplikasi (dari model) yang ditampilkan kepada pengguna. Intent adalah sebuah tindakan yang dilakukan sistem untuk merespon masukan dari pengguna dan perubahan pada keadaan aplikasi (dari model).

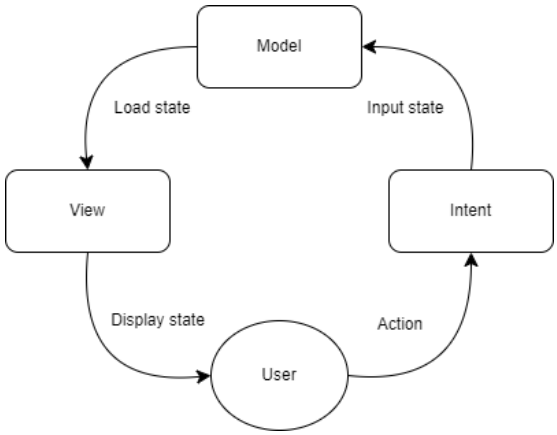

Fig. 2. Arsitektur MVI

Model-View-Intent, atau MVI, menawarkan banyak kelebihan dan kekurangan dalam pengembangan perangkat lunak perangkat. Salah satu fitur utamanya adalah pemisahan yang jelas antara Model, View, dan Intent, yang membuat kode lebih terstruktur dan lebih mudah digunakan. Dengan aliran data searah, MVI mengurangi risiko bug terkait perubahan tak terduga dan memungkinkan penggunaan status persisten dan berkelanjutan. Selain itu, pendekatan ini memanfaatkan penggunaan logika aplikasi yang stabil, tidak dapat diubah, dan mudah dipahami, sehingga meningkatkan kinerja dan kualitas perangkat lunak. Namun pengguna MVI juga memiliki beberapa kelemahan. Penerapannya seringkali rumit, terutama untuk proyek berukuran kecil atau menengah, dan dapat menimbulkan biaya overhead jika tidak digunakan dengan jujur. Selain itu, terbatasnya ketersediaan sumber daya dan dokumentasi ditambah dengan sifat pendidikan yang berbelit-belit dapat menjadi kendala bagi peserta didik yang belum bias terhadap pendidikan.

III. METODOLOGI

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Aliquam commodo ante ut convallis tincidunt. Nulla id nulla tempor nisl euismod condimentum. Sed finibus lectus lacinia metus luctus ultricies. Fusce eros dolor, dapibus quis sagittis quis, sollicitudin ultrices quam. Integer a feugiat nulla, ac pulvinar nunc. Cras bibendum vehicula enim. Donec sollicitudin porta gravida. Donec sit amet pulvinar augue. Suspendisse porttitor sit amet neque feugiat congue. Curabitur luctus sit amet lectus vel consequat. Sed non sodales enim, sit amet congue metus. Donec nisl eros, aliquam aliquet elementum id, sollicitudin ac neque. Aenean luctus ac dolor eu rhoncus.

Sed libero augue, iaculis sit amet bibendum id, porta nec est. Donec varius arcu nec sem elementum ullamcorper. Pellentesque sed purus sit amet nisl lacinia pulvinar in in leo. Pellentesque gravida iaculis ligula id efficitur. Proin suscipit ultricies diam in fringilla. Nam varius pulvinar dolor id consectetur. Nulla dictum porttitor justo eget dictum. Nullam venenatis eleifend justo a efficitur. Morbi eget sagittis orci. Sed tincidunt ut mauris a rhoncus. Cras posuere tristique massa.

IV. HASIL DAN PEMBAHASAN

Before you begin to format your paper, first write and save the content as a separate text file. Complete all content and organizational editing before formatting. Please note sections IV-A–IV-E below for more information on proofreading, spelling and grammar.

Keep your text and graphic files separate until after the text has been formatted and styled. Do not number text heads—LaTeX will do that for you.

A. Abbreviations and Acronyms

Define abbreviations and acronyms the first time they are used in the text, even after they have been defined in the abstract. Abbreviations such as IEEE, SI, MKS, CGS, ac, dc, and rms do not have to be defined. Do not use abbreviations in the title or heads unless they are unavoidable.

B. Units

- Use either SI (MKS) or CGS as primary units. (SI units are encouraged.) English units may be used as secondary units (in parentheses). An exception would be the use of English units as identifiers in trade, such as "3.5-inch disk drive".
- Avoid combining SI and CGS units, such as current in amperes and magnetic field in oersteds. This often leads to confusion because equations do not balance dimensionally. If you must use mixed units, clearly state the units for each quantity that you use in an equation.
- Do not mix complete spellings and abbreviations of units: "Wb/m^2" or "webers per square meter", not "webers/m^2". Spell out units when they appear in text: ". . . a few henries", not ". . . a few H".
- Use a zero before decimal points: "0.25", not ".25". Use "cm^3", not "cc".)

C. Equations

Number equations consecutively. To make your equations more compact, you may use the solidus (/), the exp function, or appropriate exponents. Italicize Roman symbols for quantities and variables, but not Greek symbols. Use a long dash rather than a hyphen for a minus sign. Punctuate equations with commas or periods when they are part of a sentence, as in:

$$a + b = \gamma \tag{1}$$

Be sure that the symbols in your equation have been defined before or immediately following the equation. Use "(1)", not "Eq. (1)" or "equation (1)", except at the beginning of a sentence: "Equation (1) is . . ."

D. LaTeX-Specific Advice

Please use "soft" (e.g., `\eqref{Eq}`) cross references instead of "hard" references (e.g., `(1)`). That will make it possible to combine sections, add equations, or change the order of figures or citations without having to go through the file line by line.

Please don't use the `{eqnarray}` equation environment. Use `{align}` or `{IEEEeqnarray}` instead. The `{eqnarray}` environment leaves unsightly spaces around relation symbols.

Please note that the `{subequations}` environment in LaTeX will increment the main equation counter even when there are no equation numbers displayed. If you forget that, you might write an article in which the equation numbers skip from (17) to (20), causing the copy editors to wonder if you've discovered a new method of counting.

BibTeX does not work by magic. It doesn't get the bibliographic data from thin air but from .bib files. If you use BibTeX to produce a bibliography you must send the .bib files.

LaTeX can't read your mind. If you assign the same label to a subsubsection and a table, you might find that Table I has been cross referenced as Table IV-B3.

LaTeX does not have precognitive abilities. If you put a `\label` command before the command that updates the counter it's supposed to be using, the label will pick up the last counter to be cross referenced instead. In particular, a `\label` command should not go before the caption of a figure or a table.

Do not use `\nonumber` inside the `{array}` environment. It will not stop equation numbers inside `{array}` (there won't be any anyway) and it might stop a wanted equation number in the surrounding equation.

E. Some Common Mistakes

- The word "data" is plural, not singular.
- The subscript for the permeability of vacuum μ_0, and other common scientific constants, is zero with subscript formatting, not a lowercase letter "o".
- In American English, commas, semicolons, periods, question and exclamation marks are located within quotation marks only when a complete thought or name is cited,

such as a title or full quotation. When quotation marks are used, instead of a bold or italic typeface, to highlight a word or phrase, punctuation should appear outside of the quotation marks. A parenthetical phrase or statement at the end of a sentence is punctuated outside of the closing parenthesis (like this). (A parenthetical sentence is punctuated within the parentheses.)

- A graph within a graph is an "inset", not an "insert". The word alternatively is preferred to the word "alternately" (unless you really mean something that alternates).
- Do not use the word "essentially" to mean "approximately" or "effectively".
- In your paper title, if the words "that uses" can accurately replace the word "using", capitalize the "u"; if not, keep using lower-cased.
- Be aware of the different meanings of the homophones "affect" and "effect", "complement" and "compliment", "discreet" and "discrete", "principal" and "principle".
- Do not confuse "imply" and "infer".
- The prefix "non" is not a word; it should be joined to the word it modifies, usually without a hyphen.
- There is no period after the "et" in the Latin abbreviation "et al.".
- The abbreviation "i.e." means "that is", and the abbreviation "e.g." means "for example".

An excellent style manual for science writers is [**?**].

F. Authors and Affiliations

The class file is designed for, but not limited to, six authors. A minimum of one author is required for all conference articles. Author names should be listed starting from left to right and then moving down to the next line. This is the author sequence that will be used in future citations and by indexing services. Names should not be listed in columns nor group by affiliation. Please keep your affiliations as succinct as possible (for example, do not differentiate among departments of the same organization).

G. Identify the Headings

Headings, or heads, are organizational devices that guide the reader through your paper. There are two types: component heads and text heads.

Component heads identify the different components of your paper and are not topically subordinate to each other. Examples include Acknowledgments and References and, for these, the correct style to use is "Heading 5". Use "figure caption" for your Figure captions, and "table head" for your table title. Run-in heads, such as "Abstract", will require you to apply a style (in this case, italic) in addition to the style provided by the drop down menu to differentiate the head from the text.

Text heads organize the topics on a relational, hierarchical basis. For example, the paper title is the primary text head because all subsequent material relates and elaborates on this one topic. If there are two or more sub-topics, the next level head (uppercase Roman numerals) should be used and, conversely, if there are not at least two sub-topics, then no subheads should be introduced.

H. Figures and Tables

a) Positioning Figures and Tables: Place figures and tables at the top and bottom of columns. Avoid placing them in the middle of columns. Large figures and tables may span across both columns. Figure captions should be below the figures; table heads should appear above the tables. Insert figures and tables after they are cited in the text. Use the abbreviation "Fig. 3", even at the beginning of a sentence.

TABLE I
TABLE TYPE STYLES

Table Head	**Table Column Head**		
	Table column subhead	***Subhead***	***Subhead***
copy	More table copy[a]		

[a]Sample of a Table footnote.

Fig. 3. Example of a figure caption.

Figure Labels: Use 8 point Times New Roman for Figure labels. Use words rather than symbols or abbreviations when writing Figure axis labels to avoid confusing the reader. As an example, write the quantity "Magnetization", or "Magnetization, M", not just "M". If including units in the label, present them within parentheses. Do not label axes only with units. In the example, write "Magnetization (A/m)" or "Magnetization {A[m(1)]}", not just "A/m". Do not label axes with a ratio of quantities and units. For example, write "Temperature (K)", not "Temperature/K".

V. KESIMPULAN

DAFTAR PUSTAKA

[1] F. F. Anhar, M. H. P. Swari, and F. P. 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] James Bucanek. Model-view-controller pattern. *Learn Objective-C for Java Developers*, pages 353–402, 2009.

[3] Kshitij Chauhan, Shivam Kumar, Divyashikha Sethia, and Mohammad Nadeem Alam. Performance analysis of kotlin coroutines on android in a model-view-intent architecture pattern. *2021 2nd International Conference for Emerging Technology (INCET)*, May 2021.

[4] J. Deacon. *Model-View-Controller (MVC) Architecture," Computer Systems Development, Consulting Training*, page 1–6, 2009.

[5] Gunawan Gunawan, Armin Lawi, and Adnan Adnan. Analisis arsitektur aplikasi web menggunakan model view controller (mvc) pada framework java server faces. *Scientific Journal of Informatics*, 3(1):55–67, Jun 2016.

[6] A Hidayat and B Surarso. Penerapan arsitektur model view controller (mvc) dalam rancang bangun sistem kuis online adaptif. *Seminar Nasional Teknologi Informasi dan Komunikasi 2012 (SENTIKA 2012)*, page 57–64, 2012.

[7] Khusnul Khotimah. Pengembangan prototipe computer assisted test (cat) menggunakan arsitektur model view controller pada badan kepegawaian negara. *Jurnal Teknologi*, 8(2):53, Jul 2016.

[8] Avraham Leff and James T Rayfield. Web-application development using the model/view/controller design pattern. In *Proceedings fifth ieee international enterprise distributed object computing conference*, pages 118–127. IEEE, 2001.

[9] Luca Mezzalira. Cycle.js and mvi. *Front-End Reactive Architectures*, page 97–127, 2018.

[10] M Qureshi and Fatima Sabir. A comparison of model view controller and model view presenter. *arXiv preprint arXiv:1408.5786*, 2014.