Terjemahan artikel 10.22146/jnteti.v14i2.18985

Analisis Perbandingan Kinerja Pola MVVM dan MVP pada Sistem Dasbor Android

Fajar Pradana[1], Raziqa Izza Langundi[1], Djoko Pramono[1], Nur Ida Iriani[2]

[1] Departemen Sistem Informasi, Fakultas Ilmu Komputer, Universitas Brawijaya, Malang, Jawa Timur 65145, Indonesia
[2] Program Studi Manajemen, Universitas Tribhuwana Tunggadewi, Malang, Jawa Timur 65144, Indonesia

[Diserahkan: 23 Januari 2025, Direvisi: 7 Maret 2025, Diterima: 16 April 2025]
Penulis Korespondensi: Fajar Pradana (email: fajar.prd@gmail.com)

INTISARI — Pertumbuhan pasar Android yang pesat di berbagai negara berkembang telah mendorong permintaan akan aplikasi berkualitas tinggi. Pengembangan aplikasi berbasis Android menghadirkan beberapa tantangan, seperti kebutuhan akan desain responsif dan pengoptimalan untuk perangkat dengan spesifikasi beragam. Pola desain seperti *model-view-controller* (MVC), *model-view-presenter* (MVP), dan *model-view-viewmodel* (MVVM) telah menjadi pendekatan populer untuk mengatasi masalah ini. Namun, studi tentang kinerja pola desain dalam aplikasi Android, terutama dalam bahasa pemrograman modern seperti Kotlin, masih terbatas. Penelitian ini bertujuan untuk membandingkan kinerja pola desain MVP dan MVVM dalam aplikasi manajemen rumah kos berbasis Android, KosGX. Aplikasi ini memanfaatkan Kotlin dan memiliki dasbor interaktif yang membutuhkan sumber daya perangkat yang signifikan. Pengujian dilakukan dengan mengukur kinerja di tiga aspek utama: penggunaan *central processing unit* (CPU), penggunaan memori, dan waktu respons sistem. Hasil studi menunjukkan bahwa MVVM unggul dalam efisiensi CPU, dengan penggunaan rata-rata 8,92% dibandingkan dengan MVP, yaitu 11,15%. Dalam hal penggunaan memori, MVVM juga sedikit lebih efisien, dengan penggunaan rata-rata 121,48 MB dibandingkan dengan 121,55 MB untuk MVP. Namun, MVP unggul dalam waktu respons, dengan rata-rata 236,88 ms, dibandingkan dengan MVVM yang mencapai 252,68 ms. Studi ini menegaskan bahwa pilihan pola desain memengaruhi kinerja aplikasi. MVVM lebih efisien dalam penggunaan CPU dan memori, sedangkan MVP menawarkan waktu respons yang lebih baik. Temuan ini memberikan wawasan berharga bagi pengembang dalam memilih pola desain yang optimal berdasarkan kebutuhan spesifik aplikasi.

KATA KUNCI — Kotlin, Efisiensi CPU, Waktu Respons, MVP, MVVM, Penggunaan Memori, Profil Android.

I. PENDAHULUAN

Saat ini perkembangan pasar Android berada dalam fase pertumbuhan yang positif [1], didorong oleh permintaan yang meningkat di pasar negara berkembang serta adanya inovasi teknologi seperti integrasi AI dalam perangkat bergerak [2]. Menurut laporan International Data Corporation (IDC), pengiriman telepon pintar di seluruh dunia diperkirakan tumbuh sebesar 6,2% *year-over-year* (YoY) pada tahun 2024, mencapai 1,24 miliar unit. Pertumbuhan pesat Android sebesar 7,6% YoY terutama terjadi di wilayah Asia Pasifik (kecuali Jepang), Amerika Latin, Timur Tengah, Afrika, dan China, terutama pada perangkat kelas bawah. Sebaliknya, iOS diperkirakan hanya tumbuh sebesar 0,4% pada tahun 2024.

Platform Android tidak hanya digunakan oleh pengguna perangkat seluler, tetapi juga oleh pengembang dan produsen perangkat lunak di berbagai jenis perangkat, termasuk TV pintar, tablet, perangkat yang dapat dikenakan, dan juga mobil [3]. Selaras dengan keragaman ini, Play Store Android resmi memiliki lebih dari 2 juta aplikasi pada 60 kategori berbeda. Aplikasi-aplikasi ini berkisar dari pendidikan [4], perbankan [5], permainan, medis [6], perjalanan, dan pemantauan kesehatan [7].

Pengembangan aplikasi Android berbeda signifikan dari pengembangan desktop atau web. Aplikasi harus dioptimalkan untuk berbagai macam perangkat dan spesifikasi dibandingkan dengan desktop atau web, yang biasanya melibatkan lingkungan yang lebih stabil dengan sumber daya yang lebih besar [8]. Selain itu, aplikasi Android memerlukan desain yang responsif untuk mengakomodasi berbagai ukuran dan resolusi layar, tidak seperti resolusi dan ukuran layar yang relatif seragam dalam pengembangan desktop dan web [9], [10]. Pengembangan aplikasi desktop dan web telah ada sejak lama, sedangkan pengembangan aplikasi Android adalah bidang yang relatif lebih baru, dengan banyak pengembang pemula [11]. Pengembang pemula sering menghadapi tantangan yang dapat memengaruhi kualitas aplikasi, seperti perencanaan arsitektur yang tidak memadai, penggunaan memori yang berlebihan, implementasi fitur yang buruk, *bug* yang signifikan, dan kesulitan dalam melakukan perbaikan. Oleh karena itu, panduan yang jelas dalam bentuk implementasi pola desain perangkat lunak diperlukan untuk menghasilkan perangkat lunak berkualitas lebih tinggi yang dapat digunakan kembali, dapat dipelihara, dan lebih mudah dikembangkan [12], [13].

Pola desain menawarkan solusi untuk masalah umum yang ditemui dalam pengembangan perangkat lunak [14]. Dengan menerapkan pola desain, pengembang dapat mempercepat alur kerja, meningkatkan kualitas kode, dan membuat sistem yang lebih mudah dirawat. Saat ini, pola desain yang paling banyak digunakan pada platform Android adalah *model-view-controller* (MVC), *model-view-presenter* (MVP), dan *model-view-viewmodel* (MVVM) [15]. MVC adalah pola desain yang paling sering diadopsi karena kesederhanaannya dalam proses pengembangan, tetapi memiliki kekurangan yang nyata dalam bentuk hubungan yang erat antara pengontrol dan tampilan. MVP dan MVVM, di sisi lain, menyediakan pendekatan yang berbeda untuk mengelola data dan interaksi antara komponen aplikasi. Dalam MVP, *presenter* bertindak sebagai perantara antara *view* dan *model*, yang memungkinkan dilakukannya modifikasi pada kedua komponen. Sebaliknya, *viewmodel* MVVM menyediakan aliran data yang dapat dikonsumsi *view*,

sehingga menghilangkan kebutuhan *view* untuk memperbarui data seperti pada MVP [16]. Dengan memisahkan komponen-komponen tersebut, pola-pola ini mengurangi potensi masalah dan meningkatkan kemampuan uji aplikasi. Meskipun MVP dan MVVM diklaim lebih unggul daripada MVC, penelitian terbatas hanya menguji perbedaan kinerja antara kedua pola ini. Penelitian sebelumnya tentang kinerja perangkat seluler terkait penggunaan pola desain dalam aplikasi Android [17] telah melaporkan bahwa MVVM menawarkan penggunaan *central processing unit* (CPU) yang lebih baik dan waktu respons yang lebih cepat dibandingkan dengan MVP. Namun, MVP berkinerja lebih baik dalam manajemen memori. Pengujian ini dilakukan pada aplikasi Point of Sale (PoS) yang dikembangkan di Java, sehingga masih ada kesenjangan penelitian untuk evaluasi kinerja dalam bahasa pemrograman yang lebih modern.

Penelitian ini memperkenalkan pendekatan baru dengan mengevaluasi kinerja aplikasi bernama KosGX, yang dibangun menggunakan Kotlin, melalui analisis komparatif pola desain MVP dan MVVM. Tidak seperti penelitian sebelumnya yang hanya berfokus pada bahasa pemrograman Java dengan kasus penggunaan terbatas, penelitian ini memanfaatkan Kotlin—bahasa pemrograman modern yang secara resmi didukung oleh Android—untuk mengatasi kurangnya penelitian yang mengevaluasi kinerja dalam lingkungan pemrograman yang lebih baru. KosGX, aplikasi manajemen rumah kos dengan elemen interaktif yang kompleks, menyediakan platform pengujian yang komprehensif untuk menilai kinerja pola desain dalam skenario dunia nyata. Aplikasi ini berfungsi sebagai contoh umum sistem berbasis dasbor, yang menyajikan data yang mirip dengan sistem dasbor pada umumnya. Penggunaan memori mencerminkan jumlah *random access memory* (RAM) yang dialokasikan untuk aplikasi, sehingga sangat penting untuk perangkat kelas bawah karena alokasi RAM yang berlebihan dapat menurunkan kinerja [18]. *System response time* (SRT) merupakan faktor utama yang memengaruhi kepuasan pengguna [19].

Studi ini memberikan tiga kontribusi utama. Yang pertama, bukti empiris memberikan bukti ilmiah tentang *trade-off* kinerja antara MVP dan MVVM dalam aplikasi Android berbasis Kotlin, yang mengatasi kesenjangan dalam penelitian sebelumnya tentang pengembangan perangkat lunak kontemporer. Kontribusi kedua adalah implikasi praktis. Penelitian ini menyoroti implikasi praktis dari pemilihan pola desain untuk perangkat lunak dengan keterbatasan sumber daya, memastikan penggunaan memori yang lebih optimal dan waktu respons yang lebih singkat. Lalu, yang ketiga adalah wawasan yang dapat ditindaklanjuti. Penelitian ini menawarkan wawasan yang dapat ditindaklanjuti bagi pengembang untuk meningkatkan kualitas aplikasi dan pengalaman pengguna melalui keputusan arsitektur yang tepat

Dari perspektif masyarakat, temuan ini bermanfaat baik bagi pengembang maupun pengguna akhir. Bagi pengembang, penelitian ini memberikan bekal berupa wawasan yang lebih mendalam tentang kinerja pola desain, yang memungkinkan pengembang membuat aplikasi yang lebih efisien dan mudah dirawat. Bagi pengguna akhir, terutama pengguna yang berada di negara berkembang yang mengandalkan perangkat Android kelas bawah, aplikasi yang dioptimalkan akan berkontribusi pada pengalaman pengguna yang lebih lancar dan masa pakai perangkat yang lebih lama. Lebih jauh lagi, karena solusi digital terus memenuhi kebutuhan masyarakat seperti pendidikan, perawatan kesehatan, dan inklusi keuangan, pengembangan aplikasi berkinerja tinggi menjadi landasan bagi kemajuan teknologi. Penelitian ini bertujuan untuk mengidentifikasi pola desain yang menawarkan penyajian data yang lebih baik dan pemanfaatan sumber daya yang optimal untuk aplikasi KosGX. Jika terdapat perbedaan kinerja yang signifikan, hal ini akan menunjukkan bahwa pilihan pola desain merupakan faktor penting dalam pengujian kinerja. Pemilihan pola desain memengaruhi kinerja perangkat saat menjalankan aplikasi, sehingga penting bagi pengembang untuk memilih pola yang paling sesuai guna memastikan pengalaman pengguna yang positif. Kinerja aplikasi memainkan peran penting dalam kepuasan pengguna dan kinerja yang buruk dapat berdampak buruk pada persepsi pengguna terhadap aplikasi [20]. Akibatnya, pengembang harus mempertimbangkan kinerja sistem sebagai aspek integral dari desain pengalaman pengguna.

II. PENELITIAN TERKAIT

A. POLA DESAIN

Pola desain adalah solusi untuk masalah yang umum dihadapi selama pengembangan sistem, khususnya yang terkait dengan desain, organisasi kode, dan efisiensi sistem [14]. Dalam aplikasi seluler, pengembangan sistem juga memerlukan pola desain untuk memastikan struktur arsitektur lebih terorganisasi dan fungsi serta tujuan setiap baris kode yang ditulis oleh pengembang dapat diidentifikasi dengan mudah. Secara resmi, Android merekomendasikan agar aplikasi terdiri atas dua lapisan, yaitu lapisan presentasi dan lapisan data, dengan lapisan tambahan yang bertindak sebagai perantara untuk memfasilitasi interaksi antara kedua lapisan tersebut [15]. Pola desain sangat penting dalam pengembangan sistem Android untuk membuat aplikasi yang efisien dan dapat dipelihara dengan skalabilitas tinggi [21]. Berbagai jenis pola desain dapat diterapkan dalam pengembangan Android, masing-masing dengan karakteristik, kelebihan, dan kekurangannya yang unik. Misalnya, penerapan pola desain Flyweight di Android telah terbukti meningkatkan kesadaran akan konsumsi memori selama pengembangan aplikasi seluler [22]. Sebuah studi membandingkan pola Flyweight dengan pemrograman berorientasi objek tradisional, yang menunjukkan bahwa pola Flyweight tidak berdampak negatif pada penggunaan memori, sehingga memungkinkan desain perangkat lunak profesional tanpa mengorbankan efisiensi. Studi lain membahas pentingnya peningkatan kualitas dan penggunaan ulang perangkat lunak dalam sistem Android [23]. Hasilnya, sebuah penelitian mengusulkan PatRoid, sebuah kerangka kerja untuk mendeteksi secara otomatis keberadaan pola desain dalam kode sumber. Evaluasi awal menunjukkan bahwa PatRoid berhasil mendeteksi 23 pola desain Gang of Four (GoF) dalam aplikasi Android.

B. MODEL VIEW VIEWMODEL (MVVM) dan MODEL VIEW PRESENTER (MVP)

Pola arsitektur MVVM dan MVP banyak digunakan dalam pengembangan perangkat lunak, khususnya dalam aplikasi Android. Kedua pola tersebut bertujuan untuk memisahkan berbagai masalah serta meningkatkan kemudahan perawatan dan pengujian, tetapi keduanya berbeda dalam pendekatan implementasi dan penanganan data. MVVM adalah variasi dari arsitektur MVC yang dirancang untuk mencapai pemisahan lengkap antara komponen *model* dan *view* [24]. Model adalah kelas yang berisi data, *view* mewakili *user interface* (UI) aplikasi dan bertanggung jawab untuk menampilkan informasi,

sedangkan *viewmodel* menangani logika bisnis aplikasi, menyediakan aliran data ke komponen *view* tanpa terikat langsung dengannya. Dengan kata lain, *viewmodel* tidak memiliki pengetahuan tentang keberadaan *view*.

Sebaliknya, MVP adalah arsitektur yang mirip dengan MVC, tetapi dengan beberapa perbedaan. Alur kerja dimulai dengan *view* yang menangkap masukan pengguna, yang kemudian diteruskan ke *presenter* [25]. Model berisi data yang akan ditampilkan, *view* mewakili antarmuka aplikasi, dan *presenter* mengelola semua interaksi antara *model* dan *view* [26]. Presenter mengambil data dari *model* dan mengirimkannya ke *view*. MVP memanfaatkan kelas antarmuka, yang merupakan definisi fungsi kosong yang dapat diperluas oleh kelas lain. Antarmuka ini digunakan dalam komponen *presenter* dan *view* untuk mengimplementasikan fungsi sebagaimana diperlukan, baik untuk mengambil ataupun mengirim data [27]. Gambar 1 mengilustrasikan alur kerja kedua arsitektur, MVP dan MVVM.

MVVM unggul dalam hal modifikasi, menampilkan indeks modifikasi terendah, sementara MVP berkinerja lebih baik dalam hal kinerja mentah. Kedua arsitektur tersebut merupakan bagian dari paradigma arsitektur bersih, yang meningkatkan pengembangan aplikasi Android dengan memastikan ketergantungan minimal dan peningkatan kegunaan [26].

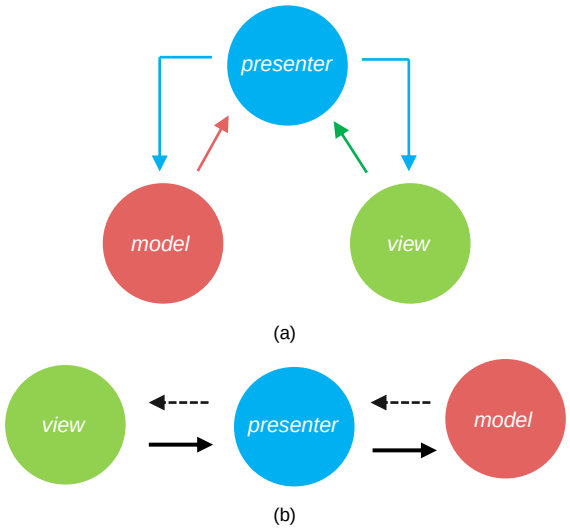
Selain itu, sebuah studi melaporkan bahwa MVVM merupakan yang terbaik dalam hal modifikasi, dengan indeks modifikasi terendah, sedangkan MVP mengungguli dalam hal kinerja dan *model-view-intent* (MVI) unggul dalam cakupan pengujian [28]. Karena terbatasnya penelitian yang mengeksplorasi analisis komparatif MVP dan MVVM dalam platform Android asli, khususnya dalam sistem yang dikembangkan menggunakan Kotlin, studi ini bertujuan untuk mengatasi kesenjangan penelitian yang ada.

C. KINERJA SISTEM

Pengguna telepon pintar modern tidak lagi dibatasi hanya pada aplikasi prainstal yang disediakan oleh produsen perangkat, tetapi juga dapat mengakses aplikasi pihak ketiga yang diunduh melalui berbagai platform distribusi aplikasi [21]. Dalam ekosistem ini, penggunaan aplikasi pihak ketiga sering kali menyebabkan konsumsi memori yang signifikan, yang dapat mengakibatkan memori yang tersedia tidak cukup untuk menjalankan aplikasi ini secara optimal. Masalah ini sangat umum terjadi pada perangkat seluler tingkat rendah dengan kapasitas memori terbatas, sehingga kekurangan memori menjadi masalah yang lebih sering terjadi [29].

Kinerja aplikasi mencerminkan tingkat kecepatan aplikasi berjalan, tingkat kecepatan pemuatan data, dan konektivitas keseluruhannya dengan berbagai operasi. Untuk mengevaluasi kinerja aplikasi Android, beberapa aspek utama perlu dipertimbangkan, termasuk penggunaan CPU, waktu eksekusi, dan konsumsi memori, yang merupakan metrik utama untuk mengukur kinerja aplikasi [30]. Permintaan untuk aplikasi berkinerja tinggi pada platform Android terus tumbuh dengan kemajuan teknologi dan meningkatnya ekspektasi pengguna. Oleh karena itu, penelitian ini bertujuan untuk membandingkan pola desain guna mengidentifikasi pola desain yang memiliki kinerja terbaik pada platform Android.

Dalam penelitian ini, digunakan Android Profiler untuk mengukur kinerja. Selama proses pengujian, perangkat dihubungkan ke Android Profiler untuk merekam penggunaan CPU dan memori saat membuka dasbor. Untuk waktu respons, log dari Android Studio dianalisis untuk menentukan waktu yang dibutuhkan oleh aplikasi untuk menampilkan data.



Gambar 1. Ilustrasi cara kerja pola desain, (a) MVP, (b) MVVM.

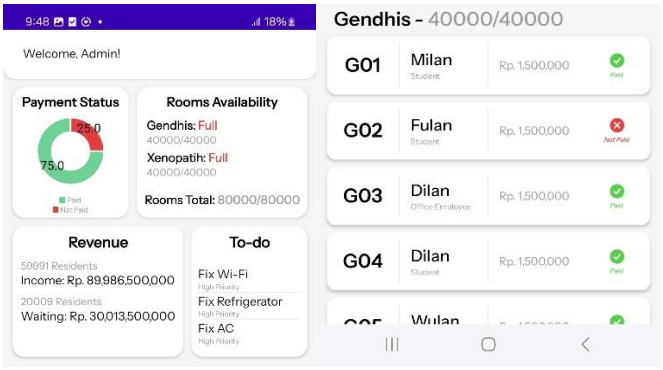
III. METODOLOGI

Penelitian ini menggunakan metodologi yang terinspirasi dari metodologi penelitian dan pengembangan, yang umumnya digunakan untuk mengembangkan produk atau sistem baru melalui proses penelitian, desain, pengembangan, dan evaluasi [31]. Metodologi penelitian dirancang untuk mendukung analisis kinerja komparatif pola desain MVVM dan MVP dalam sistem dasbor Android. Penelitian dimulai dengan fase pengembangan perangkat lunak, yaitu aplikasi yang mengimplementasikan pola desain MVVM dan MVP dikembangkan menggunakan bahasa pemrograman Kotlin. Setelah aplikasi selesai, fase perancangan kasus uji dilakukan untuk membuat skenario pengujian yang mencakup berbagai parameter kinerja, seperti penggunaan CPU, penggunaan memori, dan waktu respons. Fase berikutnya, yaitu eksperimen, melibatkan pelaksanaan skenario pengujian pada aplikasi untuk mengumpulkan data kinerja. Data yang diperoleh dari eksperimen dikumpulkan selama fase pengumpulan data untuk memastikan kelengkapan dan validitasnya. Selanjutnya, data dianalisis selama fase analisis data menggunakan metode statistik yang relevan, seperti uji-*t* sampel independen, untuk mengidentifikasi perbedaan yang signifikan antara kedua pola desain.

A. PENGEMBANGAN PERANGKAT LUNAK

Langkah pertama dalam penelitian ini adalah pengembangan aplikasi berbasis dasbor bernama KosGX, yang dibangun menggunakan Kotlin, bahasa pemrograman modern yang secara resmi didukung oleh Google untuk pengembangan Android. KosGX adalah aplikasi manajemen rumah kos dengan fitur dasbor yang mencakup berbagai komponen visual seperti teks, gambar, grafik, dan elemen interaktif yang membutuhkan sumber daya CPU dan memori yang signifikan.

Aplikasi ini dikembangkan menggunakan pola desain MVP dan MVVM. Antarmuka aplikasi dibuat identik, menampilkan data seperti diagram visual pembayaran, jumlah penghuni, kamar, daftar penghuni, daftar kebutuhan untuk rumah kos, dan ringkasan pendapatan pembayaran. Data tersebut disimpan dalam basis data lokal menggunakan Room, pustaka yang dibangun di SQLite untuk penyimpanan data lokal di Android. Gambar 2 mengilustrasikan antarmuka pengguna aplikasi KosGX, yang berfungsi sebagai objek eksperimen dalam studi perbandingan kinerja pola desain MVVM dan MVP pada aplikasi Android. Aplikasi ini dirancang untuk membantu



Gambar 2. Contoh antarmuka pengguna aplikasi KosGX.

mengelola rumah kos dengan fitur-fitur utama seperti memantau status pembayaran, melacak ketersediaan kamar, manajemen pendapatan, dan daftar tugas.

Di sisi kiri antarmuka, dasbor utama menampilkan ringkasan data penting, termasuk status pembayaran yang direpresentasikan sebagai diagram lingkaran, ketersediaan kamar dengan total kapasitas, total pendapatan, dan daftar tugas berprioritas tinggi untuk pemeliharaan fasilitas. Sementara itu, di sisi kanan ditampilkan tampilan terperinci penghuni kamar, yang menampilkan informasi seperti nama penghuni, profesi, jumlah pembayaran, dan status pembayaran (dibayar atau tidak dibayar), yang ditandai dengan ikon visual.

Aplikasi ini dikembangkan untuk mendukung pengumpulan data yang terkait dengan kinerja aplikasi dalam skenario penggunaan di dunia nyata. Komponen antarmuka pengguna dirancang untuk mengakomodasi kebutuhan pengelola rumah kos, dengan fokus pada efisiensi, penyajian informasi yang jelas, dan interaksi yang sederhana. Dalam percobaan, aplikasi ini digunakan untuk membandingkan kinerja CPU, penggunaan memori, dan waktu respons antara implementasi pola desain MVVM dan MVP.

B. PERANCANGAN KASUS UJI

Skenario pengujian dalam penelitian ini difokuskan pada kinerja aplikasi, yang meliputi evaluasi responsivitas, skalabilitas, stabilitas, dan penggunaan sumber daya [32]. Data tiruan, atau data sintetis, ditampilkan di dasbor aplikasi untuk pengujian. Setiap kasus uji diidentifikasi menggunakan format kode tertentu: "TC-X-Y-Z", dengan X mewakili pola desain, Y mewakili jenis pengujian, dan Z menunjukkan nomor kasus uji. Data untuk setiap kasus uji terdiri atas sepuluh titik data. Tabel I dan Tabel II memperlihatkan kasus uji untuk setiap pola desain.

C. EKSPERIMEN

Pengujian dilakukan dengan menjalankan aplikasi yang diisi dengan data *dummy* sesuai dengan kasus uji yang ditetapkan. Selama pengujian, perangkat dihubungkan ke Android Profiler untuk merekam penggunaan CPU dan memori saat membuka dasbor. Untuk waktu respons, log dari Android Studio digunakan untuk mengukur waktu yang dibutuhkan aplikasi dalam menampilkan data. Sebelum pengujian dimulai, dipastikan tidak ada aplikasi lain yang berjalan di perangkat. Pengujian dilakukan pada perangkat Android, khususnya Samsung Galaxy A52 dengan prosesor Snapdragon 778G, RAM 8GB, dan sistem operasi Android 14.

D. PENGUMPULAN DATA

Pengujian dilakukan dengan menjalankan aplikasi yang telah diisi data *dummy* sesuai dengan kasus uji yang ditetapkan.

TABEL I
KASUS UJI MVVM

Rentang Data	CPU	Memori	Waktu Respons
10.000–19.000	TC-VM-CPU-01	TC-VM-MEM-01	TC-VM-RT-01
20.000–29.000	TC-VM-CPU-02	TC-VM-MEM-02	TC-VM-RT-02
30.000–39.000	TC-VM-CPU-03	TC-VM-MEM-03	TC-VM-RT-03
40.000–49.000	TC-VM-CPU-04	TC-VM-MEM-04	TC-VM-RT-04

TABEL II
KASUS UJI MVP

Rentang Data	CPU	Memori	Waktu Respons
10.000–19.000	TC-P-CPU-01	TC-P-MEM-01	TC-P-RT-01
20.000–29.000	TC-P-CPU-02	TC-P-MEM-02	TC-P-RT-02
30.000–39.000	TC-P-CPU-03	TC-P-MEM-03	TC-P-RT-03
40.000–49.000	TC-P-CPU-04	TC-P-MEM-04	TC-P-RT-04

Selama proses pengujian berlangsung, perangkat dihubungkan dengan Android Profiler untuk merekam penggunaan CPU dan memori saat membuka dasbor, sedangkan untuk waktu respons, log dari Android Studio dibuka untuk melihat waktu yang dibutuhkan aplikasi dalam menampilkan data. Sebelum pengujian dimulai, dipastikan terlebih dahulu bahwa tidak ada aplikasi yang berjalan selain aplikasi yang diuji.

E. ANALISIS DATA

Data kuantitatif dianalisis menggunakan teknik statistik parametrik atau nonparametrik, tergantung pada karakteristik data. Statistik parametrik biasanya digunakan ketika data memenuhi asumsi tertentu, termasuk distribusi normal, linearitas, dan homogenitas varians. Asumsi-asumsi ini penting karena memastikan keandalan dan validitas pengujian seperti uji-*t* atau ANOVA, yang mengandalkan model matematika yang tepat untuk mengevaluasi perbedaan atau hubungan antarvariabel. Misalnya, kenormalan distribusi data sering dinilai menggunakan alat seperti uji Shapiro-Wilk, sedangkan homogenitas varians dapat dievaluasi menggunakan uji Levene.

Jika data gagal memenuhi asumsi ini, teknik statistik nonparametrik digunakan sebagai alternatif. Metode seperti uji Mann-Whitney U atau uji Kruskal-Wallis tidak bergantung pada asumsi ketat tentang distribusi data yang mendasarinya, sehingga lebih kuat untuk menganalisis data yang miring atau nonlinear. Meskipun pengujian nonparametrik kurang sensitif terhadap *outlier* dan ketidakteraturan, pengujian tersebut mungkin tidak memiliki kekuatan statistik pengujian parametrik, yang berarti bahwa pendeteksian efek yang signifikan mungkin memerlukan ukuran sampel yang lebih besar atau perbedaan yang lebih jelas.

Dalam studi ini, hipotesis dirumuskan untuk memeriksa ada tidaknya perbedaan yang signifikan dalam metrik kinerja aplikasi—penggunaan CPU, penggunaan memori, dan waktu respons—antara pola desain MVVM dan MVP. Hipotesis ini diuji menggunakan teknik statistik yang sesuai berdasarkan data penelitian yang diproses. Misalnya, uji-*t* sampel independen diterapkan jika data memenuhi kriteria parametrik, yang memungkinkan perbandingan akurat perbedaan rata-rata antara kedua pola desain. Sebaliknya, jika asumsi dilanggar, uji Mann-Whitney U digunakan untuk membandingkan distribusi metrik kinerja tanpa bergantung pada kenormalan.

Penyelesaian fase analisis data memberikan wawasan penting tentang karakteristik kinerja setiap pola desain. Hasilnya menentukan bahwa perbedaan yang diamati signifikan secara statistik atau hanya karena variasi acak (dalam penggunaan CPU, konsumsi memori, atau waktu respons). Temuan-temuan ini tidak hanya memvalidasi hipotesis, tetapi juga menawarkan kesimpulan yang dapat ditindaklanjuti terkait kesesuaian MVVM dan MVP untuk berbagai skenario aplikasi, yang berkontribusi pada pemahaman yang lebih mendalam tentang *trade-off* kinerja kedua pola desain dalam pengembangan Android.

Dengan menggunakan pendekatan statistik yang ketat dan adaptif, studi ini memastikan bahwa analisis tersebut secara ilmiah kuat dan mampu beradaptasi dengan nuansa data, memberikan bukti yang dapat diandalkan untuk mendukung kesimpulan yang diambil. Metodologi yang cermat ini memperkuat validitas temuan penelitian dan menekankan pentingnya pemilihan teknik statistik yang tepat yang disesuaikan dengan karakteristik data.

F. HASIL DAN PEMBAHASAN

Setelah analisis statistik selesai, akan dihasilkan kesimpulan tentang ada tidaknya perbedaan yang signifikan antara pola desain MVVM dan MVP dalam menampilkan data pada dasbor aplikasi Android. Untuk menentukan pola desain yang optimal, kriteria berikut harus dipenuhi.

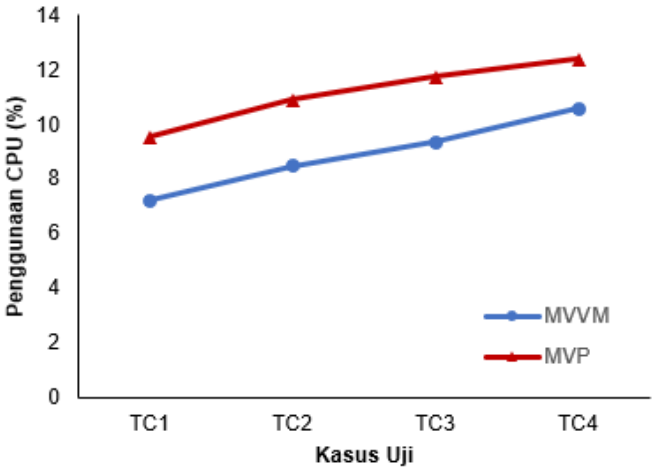
1. Pola desain dianggap lebih baik jika memiliki penggunaan CPU rata-rata yang lebih rendah.
2. Pola desain dianggap lebih baik jika memiliki penggunaan memori rata-rata yang lebih rendah.
3. Pola desain dianggap lebih baik jika menunjukkan waktu respons yang lebih cepat.

IV. HASIL DAN DISKUSI

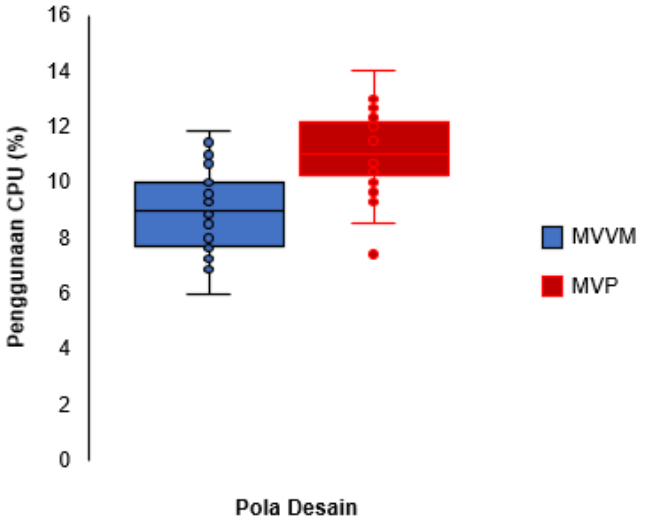
A. HASIL

Pengujian dilakukan pada perangkat Android, khususnya Samsung Galaxy A52 dengan prosesor Snapdragon 778G, RAM 8GB, dan sistem operasi Android 14. Pengujian dibagi menjadi empat kasus uji berdasarkan rentang data yang digunakan pada setiap pengujian. Untuk kasus uji TC-1, yang menggunakan rentang data 10.000–19.000 entri dalam basis data aplikasi, penggunaan CPU rata-rata adalah 7,24% untuk MVVM dan 9,54% untuk MVP. Pada TC-2 (20.000–29.000 data), penggunaan CPU rata-rata adalah 8,49% untuk MVVM dan 10,90% untuk MVP. Pada TC-3 (30.000–39.000 data), penggunaan CPU rata-rata adalah 9,37% untuk MVVM dan 11,76% untuk MVP. Terakhir, dalam TC-4 (40.000–49.000 data), penggunaan CPU rata-rata 10,57% untuk MVVM dan 12,39% untuk MVP. Gambar 3 mengilustrasikan tren peningkatan penggunaan CPU untuk setiap pola desain, dengan MVVM diwakili oleh garis biru dan MVP oleh garis merah. Dalam kategori CPU, MVVM secara konsisten mengungguli MVP dengan menggunakan lebih sedikit CPU dalam semua kasus pengujian.

Distribusi data penggunaan CPU dapat divisualisasikan menggunakan diagram kotak pada Gambar 4. Penggunaan CPU terendah diwakili oleh titik paling bawah, pada 6% untuk MVVM dan 7,43% untuk MVP. Penggunaan CPU tertinggi diwakili oleh titik paling atas, pada 11,86% untuk MVVM dan 14% untuk MVP. Kotak itu sendiri mewakili rentang interkuartil (*interquartile range*, IQR), yang merupakan perbedaan antara kuartil pertama dan ketiga, yang menggambarkan variabilitas data di sekitar median.



Gambar 3. Perbandingan kinerja CPU.

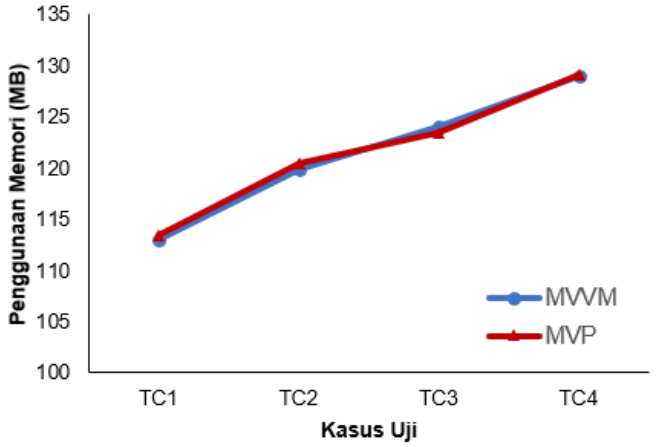


Gambar 4. Diagram *boxplot* penggunaan CPU.

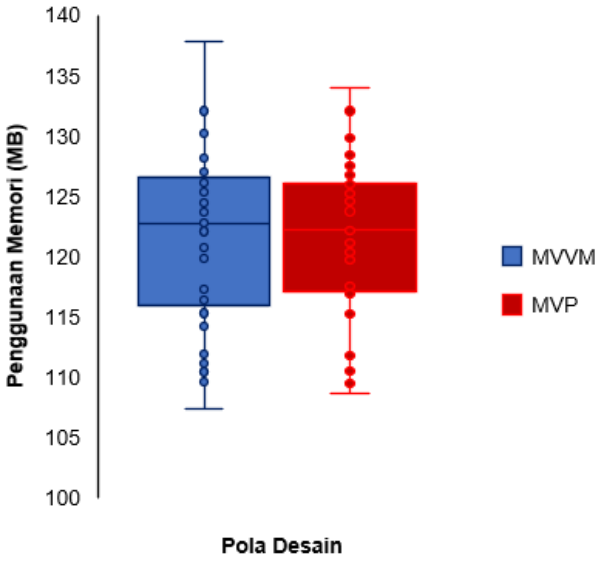
Penggunaan CPU median ditandai oleh garis di dalam kotak, pada 9% untuk MVVM dan 11% untuk MVP. Secara visual, MVVM tampaknya lebih efisien dalam penggunaan CPU dibandingkan dengan MVP.

Pada TC-1, dengan rentang data 10.000–19.000, penggunaan memori rata-rata adalah 113,03 MB untuk MVVM dan 113,35 MB untuk MVP. Pada TC-2, dengan rentang data 20.000–29.000, penggunaan memori rata-rata adalah 119,84 MB untuk MVVM dan 120,39 MB untuk MVP. Untuk TC-3, dengan data 30.000–39.000, penggunaan memori rata-rata adalah 124,10 MB untuk MVVM dan 123,37 MB untuk MVP. Terakhir, pada TC-4, dengan data 40.000–49.000, penggunaan memori rata-rata adalah 128,96 MB untuk MVVM dan 129,10 MB untuk MVP. Visualisasi peningkatan penggunaan memori rata-rata ditunjukkan pada Gambar 5, dengan MVVM diwakili oleh garis biru dan MVP oleh garis merah. Dalam kategori kinerja memori, MVVM dan MVP sangat cocok, dengan MVP berkinerja lebih baik di TC-1, TC-2, dan TC-4, sedangkan MVVM unggul di TC-3.

Distribusi data penggunaan memori dapat divisualisasikan menggunakan *boxplot*, seperti ditunjukkan pada Gambar 6, dengan MVVM diwakili oleh warna biru dan MVP oleh warna merah. Titik paling bawah mewakili penggunaan memori terendah pada 107,37 MB untuk MVVM dan 108,63 MB untuk MVP. Penggunaan memori tertinggi diwakili oleh titik paling atas pada 137,83 MB untuk MVVM dan 133,97 MB untuk



Gambar 5. Perbandingan penggunaan memori.

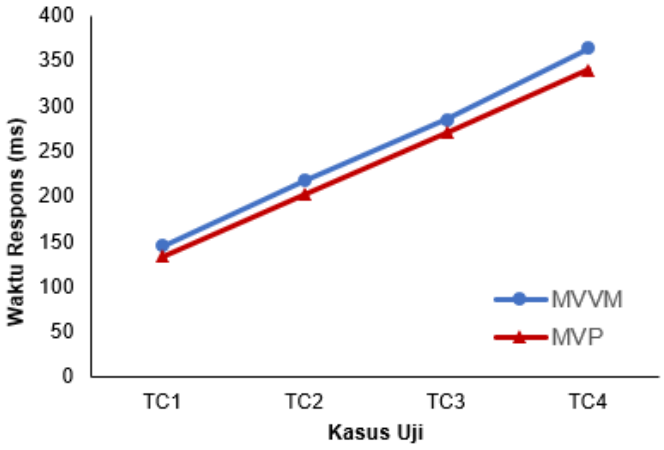


Gambar 6. *Boxplot* diagram penggunaan memori.

MVP. Penggunaan memori median ditunjukkan oleh garis di dalam kotak, pada 122,70 MB untuk MVVM dan 122,23 MB untuk MVP. Secara visual, pola desain MVVM menunjukkan penyebaran data yang lebih luas dan penggunaan memori terendah, sedangkan pola desain MVP memiliki penyebaran data yang lebih sempit.

Pada TC-1, dengan rentang data 10.000–19.000, waktu respons rata-rata adalah 144,79 ms untuk MVVM dan 134,19 ms untuk MVP. Pada TC-2, dengan rentang data 20.000–29.000, waktu respons rata-rata adalah 217,16 ms untuk MVVM dan 202,28 ms untuk MVP. Pada TC-3, dengan rentang data 30.000–39.000, waktu respons rata-rata adalah 285,17 ms untuk MVVM dan 270,49 ms untuk MVP. Terakhir, pada TC-4, dengan rentang data 40.000–49.000, waktu respons rata-rata adalah 363,61 ms untuk MVVM dan 340,57 ms untuk MVP. Visualisasi peningkatan waktu respons rata-rata untuk setiap pola desain ditunjukkan pada Gambar 7, dengan MVVM diwakili oleh garis biru dan MVP oleh garis merah. MVVM mengungguli TC-1, sementara MVP berkinerja lebih baik di TC-2, TC-3, dan TC-4, dengan gap yang relatif dekat antara kedua pola desain tersebut. Secara grafis, MVP lebih cepat dalam menampilkan data dibandingkan dengan MVVM, meskipun perbedaan kecepatan antara kedua pola desain tersebut cukup kecil.

Distribusi data waktu respons dapat divisualisasikan menggunakan diagram kotak, seperti yang ditunjukkan pada



Gambar 7. Perbandingan waktu respons.

Gambar 8; MVVM diwakili dengan warna biru dan MVP dengan warna merah. Waktu respons terendah berada di titik paling bawah; 116,30 ms untuk MVVM dan 107,70 ms untuk MVP. Waktu respons tertinggi berada di titik paling atas; 407,50 ms untuk MVVM dan 390,80 ms untuk MVP. Waktu respons median ditunjukkan oleh garis di dalam kotak, pada 256,50 ms untuk MVVM dan 231,65 ms untuk MVP. Secara visual, kedua pola desain memiliki kotak dengan ukuran yang sama, tetapi MVP sedikit lebih optimal daripada MVVM.

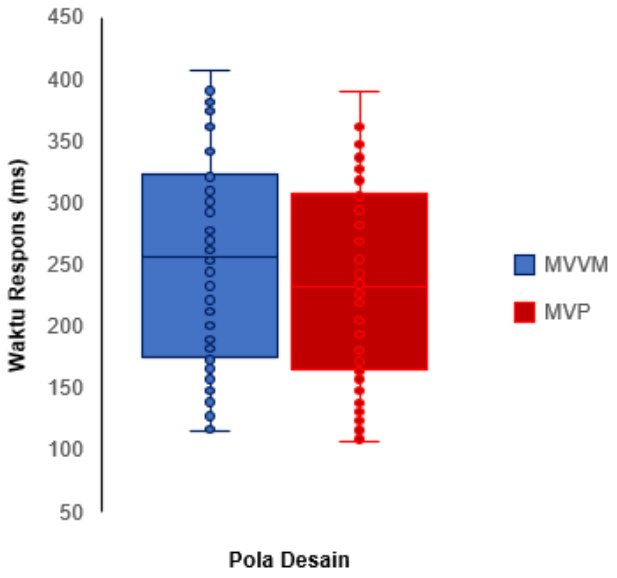
Setelah data eksperimen dikumpulkan, langkah selanjutnya adalah menganalisis data tersebut. Analisis pertama melibatkan pengujian kenormalan data, yang menentukan jenis uji perbedaan yang akan digunakan. Jika data terdistribusi normal, uji parametrik diterapkan. Jika tidak, uji nonparametrik yang digunakan.

Untuk penggunaan CPU dengan MVVM, nilai-*p* adalah 0,986, lebih besar dari 0,05, yang menunjukkan distribusi normal. Demikian pula, penggunaan CPU untuk MVP memiliki nilai-*p* 0,8933, juga lebih besar dari 0,05, yang mengonfirmasi distribusi normal. Untuk penggunaan memori, MVVM memiliki nilai-*p* 0,674 dan MVP memiliki nilai-*p* 0,1883; keduanya menunjukkan distribusi normal. Untuk waktu respons, MVVM dan MVP memiliki nilai-*p* masing-masing sebesar 0,1768 dan 0,1809; keduanya lebih besar dari 0,05, yang mengonfirmasi kenormalan.

Dengan demikian, semua kategori data ditemukan terdistribusi secara normal, yang memungkinkan dilakukannya pengujian parametrik. Tabel III merangkum hasil uji kenormalan Shapiro-Wilk untuk setiap kategori kinerja. Pengujian homogenitas dilakukan untuk menentukan sama tidaknya varians antara masing-masing kelompok, yang diperlukan untuk uji-*t* independen. Untuk CPU, nilai-*p* adalah 0,5931, lebih besar dari 0,05, yang menunjukkan varians homogen. Untuk memori, nilai-*p* adalah 0,9689, juga lebih besar dari 0,05, yang mengonfirmasi homogenitas. Untuk waktu respons, nilai-*p* adalah 0,756, lebih besar dari 0,05, yang menunjukkan bahwa varians homogen. Karena semua kategori data memiliki varians yang homogen, uji-*t* independen dapat dilakukan.

Dengan prasyarat terpenuhi, pengujian akhir dilakukan untuk menentukan ada tidaknya perbedaan signifikan antara pola desain MVVM dan MVP di setiap kategori kinerja. Hipotesis untuk uji-*t* independen adalah sebagai berikut.

1. Hipotesis nol (H0): Jika $p > 0,05$, tidak ada perbedaan signifikan.



Gambar 8. Diagram *boxplot* waktu respons.

TABEL III
UJI NORMALITAS

Pola Desain	CPU	Memori	Waktu Respons
MVVM	0,986	0,674	0,1768
MVP	0,8933	0,1883	0,1809

2. Hipotesis alternatif (H1): Jika $p \leqslant 0{,}05$, ada perbedaan signifikan.

Untuk penggunaan CPU, nilai-*p* adalah $1{,}866\times10^{-10}$, kurang dari 0,05, yang menunjukkan adanya perbedaan signifikan antara pola desain. Untuk penggunaan memori, nilai-*p* adalah 0,9608, lebih besar dari 0,05, yang menunjukkan bahwa tidak ada perbedaan signifikan. Untuk waktu respons, nilai-*p* adalah 0,756, juga lebih besar dari 0,05, yang menunjukkan tidak adanya perbedaan signifikan. Dengan demikian, perbedaan signifikan diamati dalam penggunaan CPU antara pola desain. Sementara itu, tidak ditemukan perbedaan signifikan untuk penggunaan memori dan waktu respons. Selain itu, hasil analisis menunjukkan bahwa dalam hal penggunaan CPU, MVVM lebih efisien daripada MVP, dengan *Cohen's d* = -1,64, yang menunjukkan efek yang sangat besar dan signifikan. Sebaliknya, dalam waktu respons, MVP menunjukkan kinerja yang lebih baik daripada MVVM, dengan *Cohen's d* = 1,07, yang juga merupakan efek yang besar. Sementara itu, dalam hal penggunaan memori, perbedaan antara kedua pola desain tersebut tidak signifikan (*Cohen's d* = -0,03) dengan nilai-*p* yang tinggi, yang menunjukkan bahwa penggunaan memori antara MVVM dan MVP relatif sama. Mengenai potensi bias, hasil uji normalitas (uji Shapiro-Wilk) dan uji homogenitas varians (uji Levene) mengonfirmasi bahwa asumsi statistik terpenuhi, yang memastikan bahwa hasil uji-*t* dapat diandalkan dan bebas dari bias yang signifikan.

B. DISKUSI

Di antara tiga kategori kinerja—CPU, memori, dan waktu respons—hanya penggunaan CPU yang menunjukkan perbedaan signifikan antara pola desain MVVM dan MVP. MVVM unggul dalam penggunaan CPU, dengan penggunaan rata-rata 8,02%, dibandingkan dengan 11,15% untuk MVP. Untuk penggunaan memori, MVVM sedikit mengungguli MVP, dengan rata-rata 121,48 MB versus 121,55 MB. Namun, MVP unggul dalam waktu respons, dengan rata-rata 236,88 ms dibandingkan dengan 252,68 ms untuk MVVM. Temuan ini berbeda dari penelitian sebelumnya, yang melaporkan bahwa MVVM unggul dalam penggunaan CPU dan waktu respons, sedangkan MVP unggul dalam penggunaan memori [17]. Perbedaan ini dapat dikaitkan dengan perbedaan dalam lingkungan pengujian, kompleksitas aplikasi, atau penggunaan Kotlin khusus dalam penelitian ini, yang dapat memengaruhi metrik kinerja. Selain itu, variasi dalam cara mengelola alur kerja *rendering* UI antara berbagai versi Android juga dapat berkontribusi terhadap perbedaan ini karena pengoptimalan tertentu dalam eksekusi utas UI dapat lebih mengutamakan satu arsitektur daripada yang lain.

Berdasarkan hasil uji-*t* sampel independen, hanya kategori CPU yang menunjukkan perbedaan signifikan dalam kinerja aplikasi antara pola desain untuk aplikasi berbasis Android yang dikembangkan menggunakan Kotlin. Perbedaan ini diasumsikan muncul dari cara berbeda setiap pola desain mengelola aliran data dan menyajikan data kepada pengguna. Perbedaan utamanya terletak pada komponen perantara yang digunakan untuk menangani permintaan ke *model* dan mengirimkan data ke *view*: *viewmodel* dalam MVVM dan *presenter* dalam MVP. *Viewmodel* memfasilitasi mekanisme pengikatan data reaktif yang mengurangi *overhead* pembaruan yang sering, sehingga mengoptimalkan penggunaan CPU. Temuan ini konsisten dengan penelitian sebelumnya [16], yang menyoroti efisiensi paradigma pemrograman reaktif dalam mengurangi beban CPU. Lebih jauh lagi, karena MVVM memanfaatkan LiveData dan alur dalam Kotlin, MVVM memindahkan operasi yang banyak menggunakan komputasi ke *thread* latar belakang secara lebih efisien daripada MVP, yang *presenter*-nya secara aktif mengendalikan pembaruan UI dan mungkin menyimpan komputasi yang terikat UI yang tidak perlu di dalam *thread* utama.

Sebaliknya, MVP menunjukkan waktu respons yang lebih unggul, kemungkinan karena arsitektur alur datanya yang lebih sederhana, yaitu *presenter* bertindak sebagai saluran langsung antara *model* dan *view*. Ini sejalan dengan penelitian sebelumnya [28], yang mengamati bahwa meminimalkan jumlah lapisan perantara dalam alur data dapat menghasilkan waktu respons yang lebih cepat. Namun, keuntungan ini mengorbankan penggunaan CPU yang lebih tinggi, karena *presenter* memerlukan interaksi yang lebih sering dengan komponen *view* dan *model*, terutama dalam aplikasi yang kompleks.

Menariknya, sedikit keuntungan MVVM dalam penggunaan memori dibandingkan dengan MVP bertentangan dengan temuan sebelumnya [17], yaitu MVP dilaporkan unggul dalam kategori ini. Penjelasan yang mungkin adalah penanganan *lifecycle-aware component* yang lebih efisien oleh MVVM, yang mengurangi kemungkinan kebocoran memori—masalah umum dalam MVP saat mengelola *presenter* yang berumur panjang. Hal ini sejalan dengan penelitian sebelumnya [20], yang menekankan pentingnya komponen yang sadar siklus hidup dalam mengelola konsumsi memori secara efektif.

Secara keseluruhan, temuan ini menggarisbawahi *trade-off* yang bernuansa antara MVVM dan MVP, khususnya ketika diterapkan pada pengembangan Android menggunakan Kotlin. MVVM menunjukkan efisiensi CPU yang lebih unggul dan penggunaan memori yang sedikit lebih baik, sedangkan MVP menawarkan pengalaman pengguna yang lebih responsif. Pilihan antara keduanya harus mempertimbangkan prioritas dan kompleksitas kinerja aplikasi serta keakraban tim pengembangan dengan setiap pola desain. Penelitian di masa

mendatang dapat lebih jauh mengeksplorasi *trade-off* ini dengan memasukkan skenario yang lebih kompleks atau mengintegrasikan metrik kinerja tambahan, seperti konsumsi energi atau kemudahan perawatan.

Studi ini memberikan wawasan berharga tentang kinerja komparatif pola desain MVVM dan MVP dalam aplikasi Android yang dikembangkan dengan Kotlin. Salah satu keuntungan utama dari penelitian ini adalah evaluasi empirisnya terhadap skenario aplikasi dunia nyata, yang memastikan bahwa temuan tersebut relevan dan berlaku untuk pengembangan Android modern. Selain itu, penggunaan analisis statistik memperkuat validitas kesimpulan, menawarkan data konkret kepada pengembang untuk mendukung pemilihan pola desain berdasarkan kebutuhan kinerja tertentu. Namun, penelitian ini juga memiliki keterbatasan. Pengujian dilakukan pada satu model perangkat, yang mungkin tidak sepenuhnya menangkap variasi kinerja perangkat keras di berbagai perangkat Android. Lebih jauh, penelitian ini terutama berfokus pada penggunaan CPU, konsumsi memori, dan waktu respons, sedangkan faktor penting lainnya, seperti efisiensi energi, kemudahan perawatan, dan skalabilitas, tidak dieksplorasi secara mendalam. Penelitian di masa mendatang dapat mengatasi keterbatasan ini dengan memperluas cakupan dari sisi jenis perangkat dan juga metrik kinerja tambahan untuk memberikan evaluasi yang lebih komprehensif terhadap pola desain ini.

V. KESIMPULAN

Studi ini membandingkan kinerja pola desain MVVM dan MVP dalam aplikasi Android yang dibuat dengan Kotlin, dengan fokus pada penggunaan CPU, konsumsi memori, dan waktu respons. Hasilnya menunjukkan bahwa MVVM mengungguli MVP dalam hal efisiensi CPU, dengan penggunaan rata-rata 8,92% dibandingkan dengan 11,15%. MVVM juga menunjukkan efisiensi memori yang sedikit lebih baik (121,48 MB versus 121,55 MB), meskipun perbedaannya tidak signifikan secara statistik. Sebaliknya, MVP menunjukkan waktu respons yang lebih cepat, rata-rata 236,88 ms dibandingkan dengan 252,68 ms untuk MVVM.

Temuan ini menunjukkan bahwa pilihan pola desain harus didasarkan pada prioritas kinerja aplikasi. MVVM lebih cocok untuk aplikasi yang memerlukan manajemen CPU dan memori yang dioptimalkan, khususnya yang memiliki skenario pengikatan data yang kompleks atau kendala sumber daya. Di sisi lain, MVP lebih disukai untuk aplikasi yang menuntut respons waktu nyata dengan latensi minimal. Penelitian di masa mendatang dapat memperluas temuan ini dengan memasukkan metrik tambahan seperti konsumsi energi, kemudahan perawatan, dan skalabilitas. Lebih jauh lagi, pengujian pada perangkat yang lebih beragam dan skenario dunia nyata dapat memberikan wawasan yang lebih mendalam tentang kinerja pola desain ini dalam berbagai kondisi, memastikan penerapan yang lebih luas dari kesimpulan yang diambil dari penelitian ini.

KONFLIK KEPENTINGAN

Penulis menyatakan bahwa tidak ada konflik kepentingan dalam penelitian dan penyusunan makalah ini.

KONTRIBUSI PENULIS

Konseptualisasi, Fajar Pradana dan Raziqa Izza Langundi; metodologi, Fajar Pradana; perangkat lunak, Raziqa Izza Langundi; validasi, Djoko Pramono; analisis formal, Nur Ida Iriani; investigasi, Fajar Pradana; sumber daya, Raziqa Izza Langundi; kurasi data, Raziqa Izza Langundi; penulisan—persiapan draf asli, Fajar Pradana; penulisan—resensi dan penyuntingan, Fajar Pradana; visualisasi, Raziqa Izza Langundi; pengawasan, Fajar Pradana; administrasi proyek, Nur Ida Iriani; perolehan pendanaan, Nur Ida Iriani.

UCAPAN TERIMA KASIH

Penelitian ini didanai oleh Fakultas Ilmu Komputer, Universitas Brawijaya, Malang.

REFERENSI

[1] International Data Corporation (IDC) “Worldwide smartphone market forecast to grow 6.2% in 2024, fueled by Robust growth for Android in emerging markets and China, according to IDC.” Tanggal akses: 23-Jan-2025. [Online]. Tersedia: https://www.idc.com/getdoc.jsp?containerId=prUS52757624

[2] A. Karapantelakis dkk., “Generative AI in mobile networks: A survey,” *Ann. Telecommun.*, vol. 79, no. 1–2, hal. 15–33, Feb. 2024, doi: 10.1007/s12243-023-00980-9.

[3] D. Rimawi dan S. Zein, “A static analysis of Android source code for design patterns usage,” *Int. J. Adv. Trends Comput. Sci. Eng.*, vol. 9, no. 2, hal. 2178–2186, Mar./Apr. 2020, doi: 10.30534/ijatcse/2020/194922020.

[4] S. Papadakis, M. Kalogiannakis, dan N. Zaranis, “Educational apps from the Android Google Play for Greek preschoolers: A systematic review,” *Comput. Educ.*, vol. 116, hal. 139–160, Jan. 2018, doi: 10.1016/j.compedu.2017.09.007.

[5] J.B. Jorgensen dkk., “Variability handling for mobile banking apps on iOS and Android,” dalam *2016 13th Work. IEEE/IFIP Conf. Softw. Archit. (WICSA)*, 2016, hal. 283–286, doi: 10.1109/WICSA.2016.29.

[6] F.M. Kundi, A. Habib, A. Habib, dan M.Z. Asghar, “Android-based health care management system,” *Int. J. Comput. Sci. Inf. Secur. (IJCSIS)*, vol. 14, no. 7, hal. 77–87, Jul. 2016.

[7] M. Prakash, U. Gowshika, dan T. Ravichandran, “A smart device integrated with an Android for alerting a person’s health condition: Internet of things,” *Indian J. Sci. Technol.*, vol. 9, no. 6, hal. 1–6, Feb. 2016, doi: 10.17485/ijst/2016/v9i6/69545.

[8] G.H. Prakash dkk., “Development and validation of Android mobile application in the management of mental health,” *Clin. Epidemiol. Glob. Health*, vol. 31, hal. 1–7, Jan./Feb. 2025, doi: 10.1016/j.cegh.2024.101894.

[9] W. Li, Y. Zhou, S. Luo, dan Y. Dong, “Design factors to improve the consistency and sustainable user experience of responsive interface design,” *Sustainability*, vol. 14, no. 15, hal. 1–26, Agu. 2022, doi: 10.3390/su14159131.

[10] D. Amalfitano, M. Júnior, A.R. Fasolino, dan M. Delamaro, “A GUI-based metamorphic testing technique for detecting authentication vulnerabilities in Android mobile apps,” *J. Syst. Softw.*, vol. 224, hal. 1–17, Jun. 2025, doi: 10.1016/j.jss.2025.112364.

[11] N. Hoshieah, S. Zein, N. Salleh, dan J. Grundy, “A static analysis of Android source code for lifecycle development usage patterns,” *J. Comput. Sci.*, vol. 15, no. 1, hal. 92–107, Jan. 2019, doi: 10.3844/jcssp.2019.92.107.

[12] B.S. Panca, S. Mardiyanto, dan B. Hendradjaya, “Evaluation of software design pattern on mobile application based service development related to the value of maintainability and modularity,” dalam *2016 Int. Conf. Data Softw. Eng. (ICoDSE)*, 2016, hal. 1–5, doi: 10.1109/ICODSE.2016.7936132.

[13] B.B. Mayvan, A. Rasoolzadegan, dan Z.G. Yazdi, “The state of the art on design patterns: A systematic mapping of the literature,” *J. Syst. Softw.*, vol. 125, hal. 93–118, Mar. 2017, doi: 10.1016/j.jss.2016.11.030.

[14] A. Naghdipour, S.M.H. Hasheminejad, dan M.R. Keyvanpour, “DPSA: A brief review for design pattern selection approaches,” dalam *2021 26th Int. Comput. Conf. Comput. Soc. Iran (CSICC)*, 2021, hal. 1–6, doi: 10.1109/CSICC52343.2021.9420629.

[15] D. Panchal, “Comparative study on Android design patterns,” *Int. Res. J. Eng. Technol.*, vol. 7, no. 9, hal. 833–840, Sep. 2020.

[16] R.L.B. Baptista, “Framedrop-Mobile Client,” Tesis, University of Coimbra, Coimbra, Portugal, 2023.

[17] B. Wisnuadhi, G. Munawar, dan U. Wahyu, “Performance comparison of native Android application on MVP and MVVM,” dalam *Proc. Int. Semin.*

Sci. Appl. Technol. (ISSAT 2020), 2020, hal. 276–282, doi: 10.2991/aer.k.201221.047.

[18] M. Willocx, J. Vossaert, dan V. Naessens, "Comparing performance parameters of mobile app development strategies," dalam *MOBILESoft '16, Proc. Int. Conf. Mob. Softw. Eng. Syst.*, 2016, hal. 38–47, doi: 10.1145/2897073.2897092.

[19] R.A. Doherty dan P. Sorenson, "Keeping users in the flow: Mapping system responsiveness with user experience," *Procedia Manuf.*, vol. 3, hal. 4384–4391, 2015, doi: 10.1016/j.promfg.2015.07.436.

[20] F. Rösler, A. Nitze, dan A. Schmietendorf, "Towards a mobile application performance benchmark," dalam *ICIW 2014, 9th Int. Conf. Internet Web Appl. Serv.*, 2014, hal. 55–59.

[21] G. Lim, C. Min, dan Y.I. Eom, "Enhancing application performance by memory partitioning in Android platforms," dalam *2013 IEEE Int. Conf. Consum. Electron. (ICCE)*, 2021, hal. 649–650, doi: 10.1109/ICCE.2013.6487055.

[22] W. Ngaogate, "Applying the Flyweight design pattern to Android application development," *ASEAN J. Sci. Technol. Rep. (AJSTR)*, vol. 26, no. 2, hal. 49–57, Apr.-Jun. 2023, doi: 10.55164/ajstr.v26i2.247607.

[23] D. Rimawi dan S. Zein, "A model based approach for Android design patterns detection," dalam *2019 3rd Int. Symp. Multidiscip. Stud. Innov. Technol. (ISMSIT)*, 2019, hal. 1–10, doi: 10.1109/ISMSIT.2019.8932921.

[24] R.F. García, "MVVM: Model–view–viewmodel," dalam *iOS Architecture Patterns*. Berkeley, CA, AS: Apress, 2023, hal. 145–224.

[25] X. Li, S. Wang, Z. Liu, dan G. Wu, "Design and implementation of enterprise web application common framework based on model-view-viewmodel architecture," dalam *5th Int. Conf. Mechatron. Comput. Technol. Eng. (MCTE 2022)*, 2022, hal. 1-4, doi: 10.1117/12.2661040.

[26] M.I. Alfathar dkk., "Penerapan MVVM (model view viewmodel) pada pengembangan aplikasi bank sampah digital," *J. Ris. Apl. Mhs. Inform. (JRAMI)*, vol. 5, no. 2, hal. 406–414, Apr. 2024, doi: 10.30998/jrami.v5i2.11071.

[27] C.J. Sampayo-Rodríguez, R. González-Ambriz, B.A. Gonzalez-Martinez, dan J. Aldana-Herrera, "Processor and memory performance with design patterns in a native Android application," *J. Appl. Comput.*, vol. 6, no. 18, hal. 53–61, Jun. 2022, doi: 10.35429/JCA.2022.18.6.53.61.

[28] F.F. Anhar, M.H.P. Swari, dan F.P. Aditiawan, "Analisis perbandingan implementasi clean architecture menggunakan MVP, MVI, dan MVVM pada pengembangan aplikasi Android native," *Jupiter, Publ. Ilmu Keteknikan Ind. Tek. Elekt. Inform.*, vol. 2, no. 2, hal. 181–191, Mar. 2024, doi: 10.61132/jupiter.v2i2.155.

[29] A. Moreno-Azze, D. López-Plaza, F. Alacid, dan D. Falcón-Miguel, "Validity and reliability of an iOS mobile application for measuring change of direction across health, performance, and school sports contexts," *Appl. Sci.*, vol. 15, no. 4, hal. 1–11, Feb. 2025, doi: 10.3390/app15041891.

[30] L. Corral, A. Sillitti, dan G. Succi, "Mobile multiplatform development: An experiment for performance analysis," *Procedia Comput. Sci.*, vol. 10, hal. 736–743, 2012, doi: 10.1016/j.procs.2012.06.094.

[31] F. Pradana, P. Setyosari, S. Ulfa, dan T. Hirashima, "Development of gamification-based e-learning on web design topic," *Int. J. Interact. Mob. Technol. (iJIM)*, vol. 17, no. 3, hal. 21–38, Feb. 2023, doi: 10.3991/ijim.v17i03.36957.

[32] S. Pargaonkar, "A comprehensive review of performance testing methodologies and best practices: Software quality engineering," *Int. J. Sci. Res. (IJSR)*, vol. 12, no. 8, hal. 2008–2014, Agu. 2023, doi: 10.21275/SR23822111402.