Analisis Pengaruh Arsitektur MVVM dan MVP pada Performa *Database* GreenDao

Mumuh Kustino Muharram[1], Zikri Ariachandra[2], Bambang Wisnuadhi[3], Ghifari Munawar[4]

[1234]*Jurusan Teknik Komputer dan Informatika - Politeknik Negeri Bandung, Bandung 40012*
E-mail : [1]mumuh.kustino.tif417@polban.ac.id, [2]zikri.ariachandra.tif417@polban.ac.id, [3]bwisnu@jtk.polban.ac.id, [4]ghifari.munawar@polban.ac.id

ABSTRAK

Arsitektur dan *database* adalah dua komponen yang perlu diterapkan dalam aplikasi *mobile*, karena keduanya dapat mempengaruhi performa dan *user experience*, yang merupakan faktor utama bagi pengguna aplikasi untuk meng-*install* dan *uninstall* aplikasi. Keragaman model arsitektur dan *database* menjadi tantangan bagi *developer* sehingga dibutuhkan penelitian untuk mengetahui arsitektur mana yang paling optimal diterapkan, karena dapat menghasilkan performa yang berbeda. Dalam penelitian ini model arsitektur yang digunakan adalah *Model View View-model* (MVVM) dan *Model View Presenter* (MVP), kemudian *database* yang dijadikan bahan penelitian adalah greenDao. Tujuan penelitian ini adalah untuk menganalisis pengaruh arsitektur MVVM dan MVP terhadap performa *database* greenDao. Performa diukur dari waktu eksekusi. Penelitian dilakukan dengan mengungkap pengaruh arsitektur *mobile* terhadap *database* dan mencari kombinasi arsitektur dan *database* yang memiliki performa paling optimal terhadap data bertipe *string* yang berjumlah 10.000, 100.000, 500.000, dan 1.000.000. Untuk melakukan eksperimen dibangun dua buah aplikasi untuk pengelolaan data obat-obatan pada rumah sakit. Tiap aplikasi memiliki fungsionalitas dan *database* yang sama namun menerapkan arsitektur yang berbeda. Hasil eksperimen menunjukkan bahwa secara umum performa aplikasi dengan arsitektur MVVM lebih baik dari arsitektur MVP.

Kata Kunci

Database greenDao, MVVM, MVP, Performa, Waktu eksekusi.

1. PENDAHULUAN

Menurut situs gs.statcounter.com, Android menempati peringkat pertama pada *Mobile Operating System Market Share Worldwide* – Desember 2020 dengan persentase sebesar 72,48% [1]. Hal ini menunjukkan bahwa Android merupakan teknologi *mobile* yang paling populer saat ini.

Corral dan Fronza menyatakan bahwa kualitas dari sebuah *source code* hanya memiliki dampak yang minimal terhadap kesuksesan sebuah aplikasi. Faktor terpenting yang mempengaruhi kesuksesan aplikasi adalah "*responsiveness*, *easiness*, *functionality*, dan *performance*" [2]. Selain itu, menurut Selim, Kai, dan Javier, penyebab *user* meng-*uninstall* sebuah aplikasi disebabkan oleh *crash*. *Crash* dalam studi literatur tersebut dikaitkan dengan *performance*, *sluggish behavior*, *freeze*, *slow*, *laggy*, dan *force quit* [3].

Melihat fakta–fakta di atas dapat disebutkan bahwa studi terkait teknologi *mobile* menjadi penting terutama terkait performa. Selain itu juga, menurut studi literatur yang dibuat oleh B. Longho dinyatakan bahwa studi terkait performa masih terbatas [4]. Penelitian yang dilakukan oleh Obradovic, Kelec dan Dujlovic berfokus pada pengujian performa *database*. Menurutnya, *database* merupakan elemen paling penting dalam sebuah aplikasi, karena aplikasi harus *fast feedback* dan dapat memproses data banyak. Kedua hal tersebut tidak dapat diwujudkan tanpa bantuan *database* [5].

Belum ditemukan pada studi literatur sebelumnya mengenai pengaruh arsitektur terhadap performa *database*. Penambahan variasi arsitektur *mobile* dalam penelitian ini untuk menambah unsur kebaruan. *Model view view-model* (MVVM) dan *Model view presenter* (MVP) dipilih sebagai bahan penelitian karena keunggulan dari kedua arsitektur tersebut [6]. Dengan meneliti dua arsitektur sekaligus diharapkan dapat mengetahui secara langsung perbedaan performa dari kedua arsitektur. Dengan memilih arsitektur yang paling unggul yaitu MVVM dan MVP, hasil penelitian akan lebih bermanfaat bagi masyarakat. *Database* greenDao yang dijadikan bahan penelitian dan eksperimen adalah berdasarkan popularitas *database* tersebut. Kendati demikian, hasil penelitian dapat memberikan informasi yang lebih relevan dan bermanfaat bagi masyarakat luas [7].

Kumpulan data yang mengandung informasi yang relevan untuk *enterprise* disebut dengan *database*. *Database management system* (DBMS) adalah kumpulan dari data yang saling berhubungan dan rangkaian program untuk mengakses data tersebut. Tujuan utama dari *database* adalah menyediakan solusi untuk menyimpan dan mengambil informasi dari *database* secara mudah dan efisien. Sistem *database* dirancang untuk menangani informasi dalam skala besar [8].

2. PENELITIAN TERKAIT

Nikola Obradovic, Alexander Kelec, dan Igor Dujlovic pada tahun 2019 membuat penelitian terkait analisis performa pada Android SQLite Database [5]. Penelitian

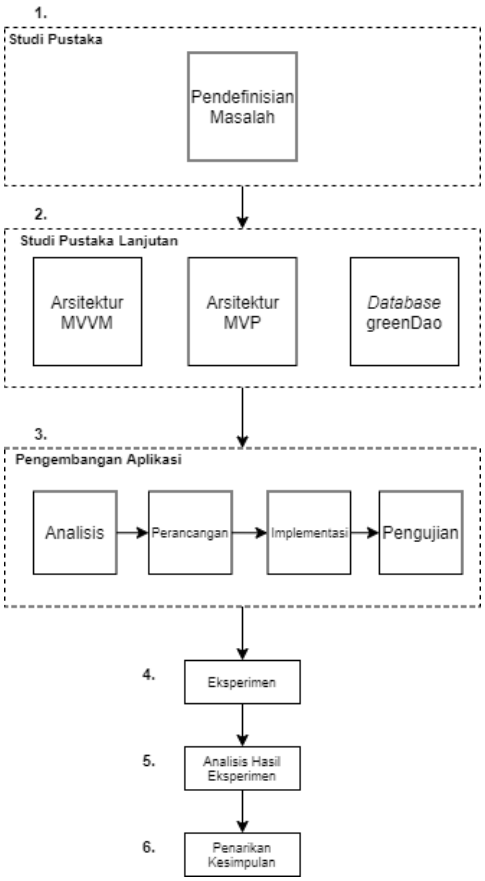
ini dilatarbelakangi oleh pentingnya studi terkait performa *database* dan juga perkembangan pengguna *mobile*. Studi literatur berfokus pada penelitian terhadap SQLite Android *relational database management system* (RDBMS). Untuk melihat performanya SQLite Android RDBMS diujikan dengan beberapa variasi skenario yang mewakili penggunaan aplikasi pada umumnya. Skenario tersebut adalah salah satu *basic data operations* yaitu *create read update delete* (CRUD). Selanjutnya, untuk variasi skenario yang digunakan adalah operasi *database* untuk *encrypted* data, *unencrypted* data, dan akses *concurrent* ke *database*. Penelitian ini dilakukan dengan cara membuat aplikasi untuk mengotomasi proses pengujian. Pengujian tersebut melakukan *test* performa operasi CRUD terhadap *datasets* yang besar. Hasil dari pengujian menunjukkan bahwa operasi terhadap *encrypted* data membutuhkan waktu yang lebih banyak ketimbang data *unencrypted*. Selain itu juga, hasil pengujian menunjukkan bahwa pengujian lebih dari lima puluh ribu data mulai menunjukkan perbedaan hasil yang signifikan, dibawah itu hasil tidak menunjukkan perbedaan yang signifikan.

Arsitektur *software* menjadi hal yang sangat penting dalam pengembangan aplikasi baik dalam skala kecil maupun besar, lebih spesifiknya pada pengembangan aplikasi Android. Keunggulan dari arsitektur MVVM dan MVP bisa menjadi pertimbangan bagi Android *developer* untuk migrasi dari arsitektur MVC ke arsitektur MVP atau MVVM [9].

Pada arsitektur MVVM dan MVP memiliki kelebihan dari aspek *memory usage* dibandingkan dengan arsitektur MVC [6]. Selain itu, arsitektur MVVM dan MVP memiliki kelebihan pada aspek *testability*, *modifiability*, dan *performance* serta penggunaan *memory* dibandingkan arsitektur MVC. Arsitektur MVVM memiliki kelebihan pada aspek *testability*, sedangkan arsitektur MVP memiliki kelebihan pada aspek *modifiability* [6]. Melihat dari performa dan kelebihan yang dimiliki oleh MVVM dan MVP maka dipilihlah kedua arsitektur tersebut untuk dijadikan bahan eksperimen ketimbang MVC.

3. METODOLOGI

Mengacu pada sebagian metodologi pada penelitian [10], Gambar 1 di bawah ini merupakan diagram dari tahapan penelitian



Gambar 1. Diagram dari Tahapan Penelitian

3.1 Pendefinisian Masalah dan Solusi

Tahapan awal pada penelitian ini yaitu mengumpulkan dan mempelajari literatur – literatur yang dibutuhkan untuk memenuhi konsep terkait Android, arsitektur, pengukuran performa *database*, *library database* greenDao, dan *tools* yang digunakan pada penelitian, sehingga ditemukan masalah yang perlu diselesaikan serta solusi yang diperlukan untuk masalah tersebut.

3.2 Eksplorasi *Project* Android MVP-greenDao

Library greenDao *database* merupakan *library database* aplikasi Android *open-source* [11]. greenDao memungkinkan proses pengkodean tanpa *mapping* Java *object* ke *table database*, sehingga dapat melakukan *store*, *update*, *delete* dan *query* untuk *object* Java menggunakan *object-oriented* API sederhana. Eksplorasi greenDao *database* dilakukan dengan cara membedah *source code* yang berasal dari *repository* github mindorks dengan menggunakan bahasa pemrograman Java [12]. Bedah *source code* tersebut dilakukan dengan menentukan komponen – komponen mana yang termasuk ke dalam arsitektur MVP dan *library* greenDao *database*.

3.3 Penentuan Aplikasi Objek Penelitian

Pada tahap ini berisi pertimbangan dalam pemilihan atau penentuan aplikasi yang dikembangkan sebagai objek penelitian. Hal ini menjadi penting karena terdapat beberapa faktor yang perlu diperhatikan dalam pembuatan aplikasi, diantaranya adalah *graphic user interface* yang diimplementasi pada aplikasi seperti bagaimana *button* pada aplikasi, *field* inputan yang tersedia untuk pengguna, dan tampilan lainnya. Kemudian fitur-fitur teknis yang terdapat pada aplikasi seperti fitur CRUD pada tiap *tab* dan *button* yang bernilai *query* statis menyesuaikan dengan masukkan pengguna dan *tab*. Pengguna dapat mengganti jumlah data berupa masukan angka pada *field* untuk *query* yaitu jumlah data 10.000, 100.000, 500.000 dan 1.000.000.

3.4 Pengumpulan Data

Pada tahap ini data dikumpulkan dan diolah agar sudah siap digunakan sebagai bahan eksperimen. Data yang digunakan oleh aplikasi adalah data medis yang didapatkan dari *java-faker* [13]. Data ini berupa objek yang memiliki empat *field* yaitu *medicine_name, disease_name, hospital_name,* dan *symptoms*. Data ini disajikan dalam bentuk .yml.

3.5 Pengembangan Aplikasi Objek Penelitian

Pada tahapan ini terdapat beberapa bagian terdiri dari analisis, perancangan, implementasi, dan pengujian untuk aplikasi yang dilakukan pada eksperimen kedepan.

Dibangun dua aplikasi Android untuk eksperimen pada penelitian. Aplikasi yang dibangun berupa aplikasi yang dapat melakukan operasi CRUD termasuk menampilkan hasil waktu eksekusi dari hasil CRUD. Kedua aplikasi dibangun dengan mengkombinasikan arsitektur MVVM dan MVP dengan *database* greenDao. Sehingga ada dua kombinasi, yaitu MVVM dan greenDao; MVP dan greenDao.

Pengembangan perangkat lunak menerapkan SDLC *model waterfall*. Sehingga terdapat empat fase yang diterapkan, yaitu:

- Analisis dan penetapan *requirement*;
- Desain sistem dan perangkat lunak;
- Implementasi dan pengujian unit;
- Integrasi dan pengujian sistem.

Operasi dan pemeliharaan tidak dilakukan karena, aplikasi yang dibuat hanya untuk eksperimen pada penelitian ini.

3.5.1 Analisis

Tahap ini merupakan tahapan yang bertujuan untuk menganalisis aplikasi yang dibangun. Analisis yang dilakukan yaitu terdiri dari menentukan fitur – fitur yang diimplementasikan atau diterapkan pada aplikasi, serta data seperti apa yang dibutuhkan. Selain itu, analisis dilakukan untuk mengidentifikasi komponen apa saja yang ada dalam memenuhi arsitektur dan komponen apa saja yang termasuk ke dalam bagian *database*. Pada bagian ini ditentukan fitur untuk aplikasi yang dibangun sebagai pendukung penelitian. Fitur utama dari aplikasi tersebut adalah pengelolaan data obat-obatan pada rumah sakit. Aplikasi ini memungkinkan pengguna untuk dapat menambahkan, menghapus, mengubah, dan melihat data obat-obatan.

3.5.2 Perancangan

Tahap perancangan merupakan tahapan yang dilakukan untuk merancang dan menentukan tampilan atau *user interface* (UI) dari aplikasi. Tampilan atau UI ini dirancang berdasarkan kebutuhan dari fitur – fitur dan data yang telah ditentukan. Dengan demikian, data tersebut dapat digunakan untuk memenuhi fitur – fitur yang dibutuhkan. Selain merancang dan menentukan tampilan, tahap ini juga merancang struktur data, diagram – diagram, dan model data.

3.5.3 Implementasi

Tahap implementasi merupakan tahapan yang dilakukan setelah tahapan analisis selesai dan tahapan perancangan telah dilakukan. Tahapan implementasi yang dilakukan yaitu membuat dua aplikasi dari kombinasi arsitektur dan *database*. Dengan demikian, semua kandidat dari arsitektur dan *database* dapat terwakilkan yaitu sebagai berikut:

- Aplikasi yang mengimplementasikan arsitektur MVVM dan *database* greenDao;
- Aplikasi yang mengimplementasikan arsitektur MVP dan *database* greenDao.

3.5.4 Pengujian

Tahap pengujian merupakan tahapan terakhir pada pengembangan aplikasi objek penelitian. Tahapan ini dilakukan untuk menguji aplikasi yang telah diimplementasi seperti menguji fitur – fitur yang ada pada aplikasi, pengecekan *input* yang telah ditentukan dan *output* yang diharapkan. Pengujian dilakukan dengan metode *blackbox*.

Tabel 1 Hasil Pengujian *Requirement*

No	Fitur	MVVM greenDao	MVP greenDao
1	Menambahkan data nama rumah sakit	Berhasil	Berhasil
2	Menambahkan data nama obat – obatan pada rumah sakit	Berhasil	Berhasil
3	Menghapus data nama obat – obatan pada rumah sakit	Berhasil	Berhasil
4	Mengubah data nama obat – obatan pada rumah sakit	Berhasil	Berhasil
5	Melihat data nama obat – obatan pada rumah sakit	Berhasil	Berhasil

3.6 Perancangan Eksperimen

Perancangan eksperimen merupakan tahapan setelah aplikasi objek penelitian selesai dikembangkan.

Tahapan ini dilakukan untuk menentukan tujuan eksperimen, alat eksperimen, persiapan data eksperimen, metode eksperimen, dan skenario eksperimen.

3.6.1 Tujuan Eksperimen

Tujuan dari eksperimen adalah membuktikan adanya pengaruh arsitektur terhadap performa *database* serta menemukan kombinasi arsitektur dan *database* yang memiliki performa paling optimal terhadap data eksperimen yang sudah ditentukan.

3.6.2 Alat Eksperimen

Alat yang digunakan pada eksperimen penelitian ini adalah sebagai berikut:

1. Perangkat *laptop* dan *desktop* yang digunakan untuk mengembangkan dan menjalankan aplikasi ditunjukkan pada Tabel 2 dan Tabel 3.

Tabel 2. Spesifikasi Perangkat *Laptop*

Spesifikasi	Rincian
OS	Windows 10 Pro 64-bit
CPU *Cores*	4 *Core*
Processor	Intel i7-7700HQ CPU @ 2.80GHz
RAM	16 GB

Tabel 3. Spesifikasi Perangkat *Desktop*

Spesifikasi	Rincian
OS	Windows 10 Pro 64-bit
CPU *Cores*	4 *Core*
Processor	AMD Ryzen 3 3200G @ 3.60GHz
RAM	16 GB

2. Perangkat Android yang digunakan untuk menjalankan aplikasi yang dijadikan objek penelitian ditunjukkan pada Tabel 4.

Tabel 4. Spesifikasi Perangkat Android

Spesifikasi	Rincian
Merk	Mi A2 Lite
OS	Android 9.0 (Pie)
CPU *Cores*	8 *Core*
Processor	Snapdragon 625 Octa-core 2.0 GHz
RAM	3 GB

3.6.3 Metode Eksperimen

Sebelum eksperimen dimulai terdapat beberapa hal yang perlu dipersiapkan yaitu aplikasi, skenario eksperimen, dan data untuk menjalankan eksperimen.

Dalam eksperimen ini terdapat tiga waktu eksekusi yang diperhatikan yaitu waktu eksekusi operasi *database*, waktu eksekusi pada proses menampilkan data (*view*), dan waktu eksekusi total. Waktu eksekusi ini didapatkan dari selisih waktu operasi dimulai dan waktu operasi selesai ditunjukkan pada Gambar 2, Gambar 3 dan Gambar 4.

```
…
selectTime.set(System.currentTimeMillis());
…
selectDbTime.set(selectDbTime.longValue() +
(System.currentTimeMillis() -
selectTime.longValue()));
…
```

startDbTime

endDbTime

Gambar 2. *Source Code* Penghitungan Waktu Eksekusi DB

```
…
   AtomicLong allSelectTime = new
AtomicLong(System.currentTimeMillis());
…
   AtomicLong endTime = new
AtomicLong(System.currentTimeMillis());
   AtomicLong timeElapsed = new
AtomicLong(endTime.longValue() -
allSelectTime.longValue());
…
```

startTime

endTime

Gambar 3. *Source Code* Penghitungan Waktu Eksekusi Total

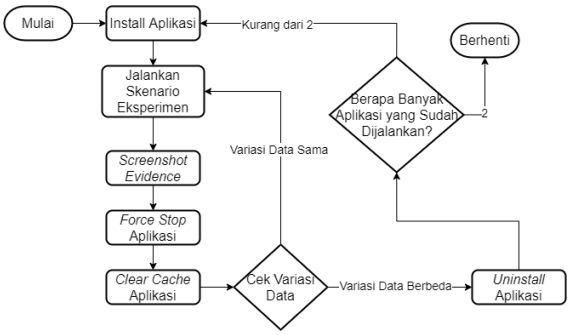
```
…
   AtomicLong allSelectTime = new
AtomicLong(System.currentTimeMillis());
…
   AtomicLong endTime = new
AtomicLong(System.currentTimeMillis());
   AtomicLong timeElapsed = new
AtomicLong(endTime.longValue() -
allSelectTime.longValue());
```

Selisih waktu total dan waktu *database*

```
viewSelectTime.set(timeElapsed.longValue()
- selectDbTime.longValue());
…
```

Gambar 4. *Source Code* Penghitungan Waktu Eksekusi *View*

Metode eksperimen yang diterapkan dilakukan pada tiap aplikasi eksperimen. Gambar 5 menunjukkan metode yang dilakukan untuk menjalankan eksperimen:

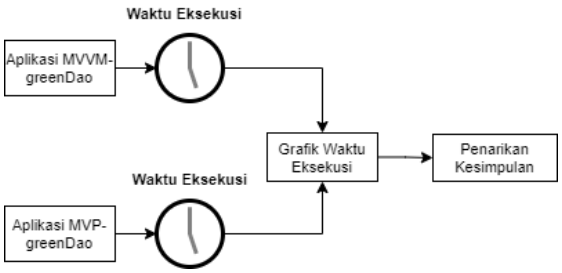


Gambar 5. *Flowchart* Metode Eksperimen

Skenario eksperimen dimulai dengan menjalankan operasi *insert*. Data yang dioperasikan dimulai dari jumlah 10.000 data, dilanjutkan dengan skenario operasi *select*, *update* dan *delete* dengan jumlah yang sama. Perulangan pada skenario eksperimen dilakukan sampai jumlah data yang dioperasikan mencapai 1.000.000.

3.6.4 Skenario Eksperimen

Skenario eksperimen yang dilakukan yaitu eksperimen setiap jenis *query* dilakukan dengan empat variasi data yaitu 10.000, 100.000, 500.000, dan 1.000.000. Setiap variasi data dilakukan sebanyak satu kali. Skenario eksperimen diilustrasikan pada Gambar 6.



Gambar 6. Ilustrasi Skenario Eksperimen

3.6.5 Persiapan Data Eksperimen

Data yang diperoleh dari *java-faker* adalah sebagai berikut :

- Nama rumah sakit didapatkan sebanyak 1021 data;
- Nama obat-obatan didapatkan sebanyak 1082 data;

Data tersebut sudah mencukupi dari segi kuantitas sebagai bahan eksperimen. Karena data yang dibutuhkan untuk eksperimen mulai dari rentang 10.000 hingga 1.000.000.

Agar data siap digunakan sebagai bahan eksperimen, data sebelumnya perlu diproses dalam beberapa tahap terlebih dahulu, yakni :

1. Data yang didapat dari *java-faker* berisi empat *field*, yaitu *medicine*, *hospital*, *disease*, dan *symptom*. Akan tetapi, hanya dua *field* saja yang digunakan yaitu *medicine* dan *hospital*. Alasan dipilihnya dua *field* saja karena sudah mencukupi kebutuhan untuk eksperimen. Kedua *field* yang dipilih juga sudah bisa merepresentasikan dunia nyata.
2. Data yang didapat awalnya berekstensi (.yml), data dimasukkan ke dalam *spreadsheet* untuk dilakukan proses *cartesian product*. *Cartesian product* dilakukan agar memenuhi kebutuhan kuantitas data untuk dilakukan eksperimen. Sehingga data akan berubah menjadi (.csv).
3. Karena data yang dibutuhkan oleh aplikasi adalah data berbentuk (.json) maka *file* (.csv) dikonversikan menjadi (.json).
4. Data yang sudah dikonversi menjadi (.json) selanjutnya di-*minify* sehingga *parser* yang digunakan didalam *project* dapat membaca *file*. Jika tidak di-*minify*, *parser* mengalami kesulitan untuk membaca *file* terutama jika data sudah melebihi 100.000.

Data yang sudah diproses selanjutnya dimasukkan ke dalam *folder assets* di dalam aplikasi. Aplikasi akan memilih *file* yang diperlukan berdasarkan masukan dari *user*.

Gambar 7. Tahapan Persiapan Data Eksperimen

3.7 Analisis Hasil Eksperimen

Data hasil eksperimen atau pengukuran dari aplikasi eksperimen adalah waktu eksekusi setiap skenario eksperimen (nilai waktu eksekusi berupa angka dalam satuan milidetik).

Data hasil eksperimen lalu disimpan ke dalam tabel yang telah dirancang. Data yang telah terkumpul di dalam tabel lalu dikonversikan ke dalam bentuk grafik garis agar perbandingan hasil kombinasi arsitektur dan *database* akan lebih mudah untuk dipahami. Dengan mengkonversikan data dari tabel ke dalam bentuk grafik memungkinkan pembaca dapat mengetahui informasi tanpa melihat keseluruhan tabel.

3.8 Penarikan Kesimpulan Eksperimen

Tahapan ini dilakukan untuk mendapatkan kesimpulan dari hasil eksperimen. Setelah data yang diperoleh dari tahap eksperimen, nilai – nilai waktu eksekusi setiap skenario dibandingkan.

Semakin kecil waktu eksekusi yang diperoleh maka semakin baik, sehingga dari hasil waktu eksekusi tersebut dapat diketahui kombinasi arsitektur dan *database* yang memiliki performa paling optimal.

4. HASIL DAN PEMBAHASAN

Pada bagian ini memperlihatkan hasil eksperimen sesuai dengan rancangan eksperimen yang sudah ditetapkan. Tujuan dari bab ini adalah untuk mendapatkan tujuan eksperimen, yaitu kombinasi arsitektur dan *database* yang memiliki performa paling baik dari aspek waktu eksekusi. Eksperimen dilakukan pada dua aplikasi eksperimen dengan tipe data dan jumlah data yang sama.

Nilai yang menjadi penentu performa dalam eksperimen ini adalah waktu eksekusi. Kendati demikian, waktu eksekusi yang dipertimbangkan ada tiga, yaitu:

- Waktu eksekusi *database*, waktu eksekusi yang diperlukan oleh *database* untuk memproses data.
- Waktu eksekusi *view*, waktu eksekusi yang dibutuhkan oleh aplikasi untuk menampilkan data dari *database* ke layar.
- Waktu eksekusi total, waktu eksekusi yang dibutuhkan oleh aplikasi untuk menjalankan satu eksperimen. Waktu eksekusi *database* ditambah waktu eksekusi *view*.

Hasil eksperimen disajikan dalam bentuk tabel dan grafik. Pada tabel, tiap *cell*-nya berisi waktu eksekusi. Pada sub-bab 4.1, 4.2, 4.3 dan 4.4 terdapat tiga tabel yang mewakilkan tiga jenis waktu eksekusi yakni *database*, *view*, dan total serta hasil eksperimen juga

ditampilkan pada sub-bab 4.1, 4.2, 4.3 dan 4.4. Data pada tabel dikonversi ke dalam bentuk grafik agar informasi pada tabel dapat diperoleh dengan mudah. Eksperimen yang dilakukan sesuai dengan metodologi yang telah ditulis pada bab 3.6.4.

Berikut adalah Tabel 5 yang berisikan data waktu eksekusi total tiap aplikasi eksperimen. Data waktu eksekusi yang dimasukkan ke dalam Tabel 5 adalah rata-rata waktu eksekusi yang dibutuhkan untuk menjalankan tiap skenario eksperimen.

Tabel 5. Nilai Rata – Rata Waktu Eksekusi Total Tiap Skenario

Aplikasi Eksperimen	*Avg. Insert* (Total) (*ms*)	*Avg. Select* (Total) (*ms*)	*Avg. Update* (Total) (*ms*)	*Avg. Delete* (Total) (*ms*)	*Avg.* Waktu Eksekusi (Total) (*ms*)
MVVM-greenDao	141.416,25	266.754,75	386.868,5	531.116,75	331.539,0625
MVP-greenDao	195.706,5	268.216	380.679,25	522.273,25	341.718,75

Mengacu pada Tabel 5, urutan performa tiap aplikasi eksperimen adalah sebagai berikut:

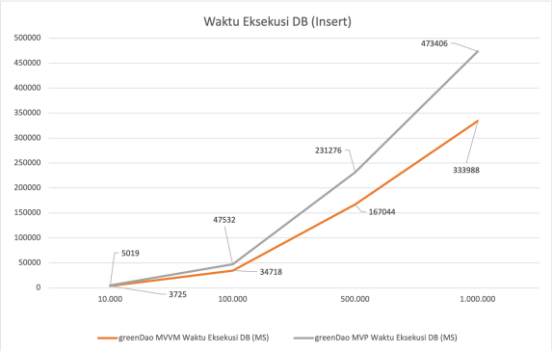
- MVVM-greenDao, dengan rata-rata waktu eksekusi 331.539 ms.
- MVP-greenDao, dengan rata-rata waktu eksekusi 341.719 ms.

4.1 Eksperimen *Insert* Data

Pada bagian ini memperlihatkan hasil eksperimen yang dilakukan pada tiap aplikasi eksperimen dalam melakukan operasi *insert* data. Sebagian besar skenario *insert* diungguli oleh MVVM dan greenDao. MVP dan greenDao mengungguli pada waktu eksekusi *view* untuk 10.000, 100.000, dan 500.000 data. Tabel 6, Tabel 7, Tabel 8 dan Gambar 8, Gambar 9 dan Gambar 10 menunjukkan hasil eksperimen *insert*.

Tabel 6. Hasil Waktu Eksekusi pada Operasi *Insert*

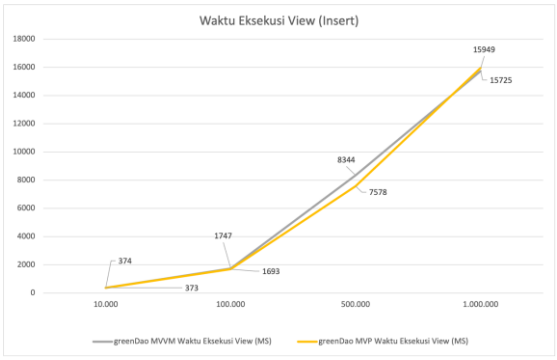
Skenario	Jumlah Data	Jenis Query	MVVM greenDao Waktu Eksekusi DB (MS)	MVP greenDao Waktu Eksekusi DB (MS)
1	10,000	INSERT	3725	5019
2	100,000	INSERT	34718	47532
3	500,000	INSERT	167044	231276
4	1,000,000	INSERT	333988	473406



Gambar 8. Grafik Hasil Waktu Eksekusi pada Operasi *Insert*

Tabel 7. Hasil Waktu Eksekusi pada Proses *View Insert*

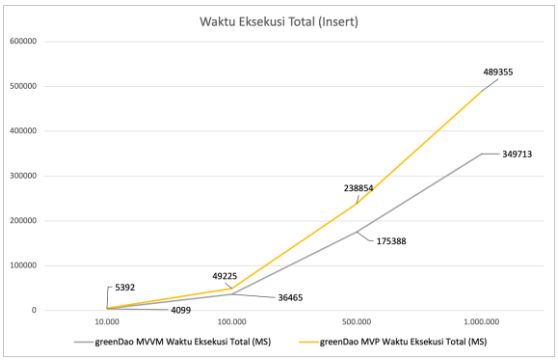
Skenario	Jumlah Data	Jenis Query	MVVM greenDao Waktu Eksekusi View (MS)	MVP greenDao Waktu Eksekusi View (MS)
1	10,000	INSERT	374	373
2	100,000	INSERT	1747	1693
3	500,000	INSERT	8344	7578
4	1,000,000	INSERT	15725	15949



Gambar 9. Grafik Hasil Waktu Eksekusi pada Proses *View Insert*

Tabel 8. Hasil Waktu Eksekusi Total Operasi *Insert*

Skenario	Jumlah Data	Jenis Query	MVVM greenDao Waktu Eksekusi Total (MS)	MVP greenDao Waktu Eksekusi Total (MS)
1	10,000	INSERT	4099	5392
2	100,000	INSERT	36465	49225
3	500,000	INSERT	175388	238854
4	1,000,000	INSERT	349713	489355



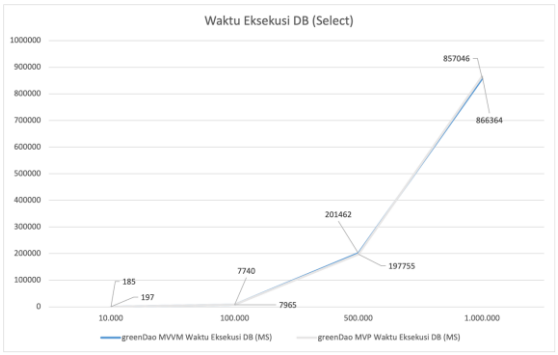
Gambar 10. Grafik Hasil Waktu Eksekusi Total Operasi *Insert*

4.2 Eksperimen *Select* Data

Pada skenario ini sebagian besar diungguli oleh MVVM dan greenDao, namun MVP dan greenDao mengungguli pada sebagian besar waktu eksekusi *view*. Tabel 9, Tabel 10, Tabel 11 dan Gambar 11, Gambar 12 dan Gambar 13 menunjukkan hasil eksperimen *select*.

Tabel 9. Hasil Waktu Eksekusi pada Operasi *Select*

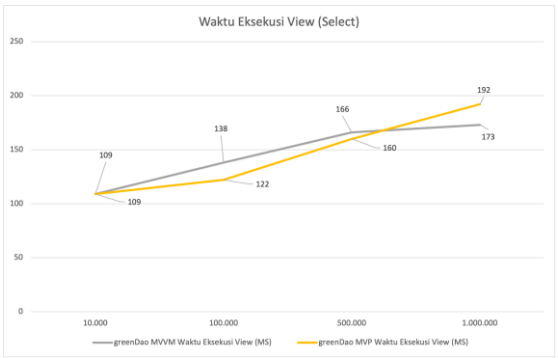
Skenario	Jumlah Data	Jenis Query	MVVM greenDao Waktu Eksekusi DB (MS)	MVP greenDao Waktu Eksekusi DB (MS)
1	10,000	SELECT	185	197
2	100,000	SELECT	7740	7965
3	500,000	SELECT	201462	197755
4	1,000,000	SELECT	857046	866364



Gambar 11. Grafik Hasil Waktu Eksekusi pada Operasi *Select*

Tabel 10. Hasil Waktu Eksekusi pada Proses *View Select*

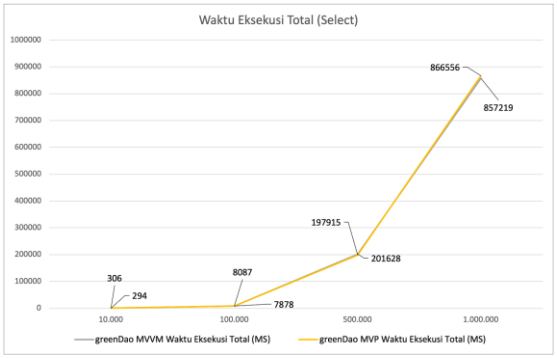
Skenario	Jumlah Data	Jenis Query	MVVM greenDao Waktu Eksekusi View (MS)	MVP greenDao Waktu Eksekusi View (MS)
1	10,000	SELECT	109	109
2	100,000	SELECT	138	122
3	500,000	SELECT	166	160
4	1,000,000	SELECT	173	192



Gambar 12. Grafik Hasil Waktu Eksekusi pada Proses *View Select*

Tabel 11. Hasil Waktu Eksekusi Total Operasi *Select*

Skenario	Jumlah Data	Jenis Query	MVVM greenDao Waktu Eksekusi Total (MS)	MVP greenDao Waktu Eksekusi Total (MS)
1	10,000	SELECT	294	306
2	100,000	SELECT	7878	8087
3	500,000	SELECT	201628	197915
4	1,000,000	SELECT	857219	866556



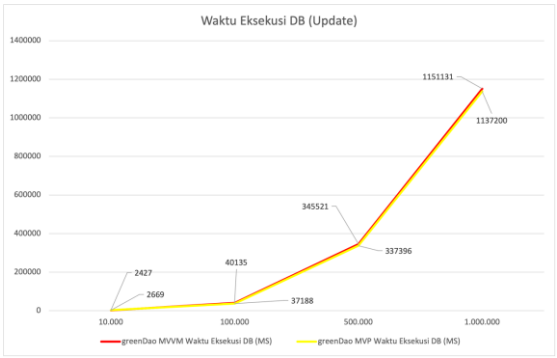
Gambar 13. Grafik Hasil Waktu Eksekusi Total Operasi *Select*

4.3 Eksperimen *Update* Data

Pada bagian ini, memperlihatkan hasil eksperimen yang dilakukan pada tiap aplikasi eksperimen dalam melakukan operasi *update* data. Pada skenario *update* data sebagian besar skenario eksperimen diungguli oleh arsitektur MVP. Tabel 12, Tabel 13, Tabel 14 dan Gambar 14, Gambar 15 dan Gambar 16 menunjukkan hasil eksperimen *update*.

Tabel 12. Hasil Waktu Eksekusi pada Operasi *Update*

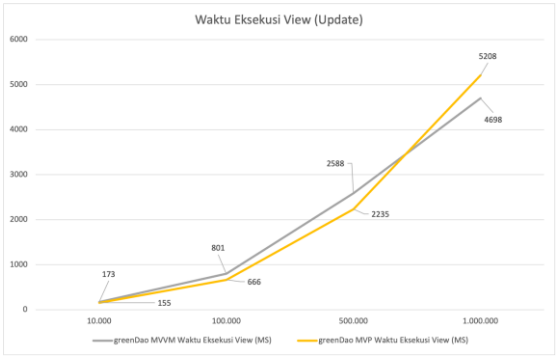
Skenario	Jumlah Data	Jenis Query	MVVM greenDao Waktu Eksekusi DB (MS)	MVP greenDao Waktu Eksekusi DB (MS)
1	10,000	UPDATE	2427	2669
2	100,000	UPDATE	40135	37188
3	500,000	UPDATE	345521	337396
4	1,000,000	UPDATE	1151131	1137200



Gambar 14. Grafik Hasil Waktu Eksekusi pada Operasi *Update*

Tabel 13. Hasil Waktu Eksekusi pada Proses *View Update*

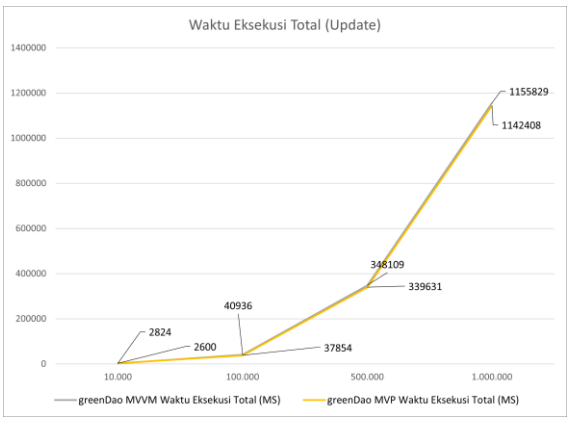
Skenario	Jumlah Data	Jenis Query	MVVM greenDao Waktu Eksekusi View (MS)	MVP greenDao Waktu Eksekusi View (MS)
1	10,000	UPDATE	173	155
2	100,000	UPDATE	801	666
3	500,000	UPDATE	2588	2235
4	1,000,000	UPDATE	4698	5208



Gambar 15. Grafik Hasil Waktu Eksekusi pada Proses *View Update*

Tabel 14. Hasil Waktu Eksekusi Total Operasi *Update*

Skenario	Jumlah Data	Jenis Query	MVVM greenDao Waktu Eksekusi Total (MS)	MVP greenDao Waktu Eksekusi Total (MS)
1	10,000	UPDATE	2600	2824
2	100,000	UPDATE	40936	37854
3	500,000	UPDATE	348109	339631
4	1,000,000	UPDATE	1155829	1142408



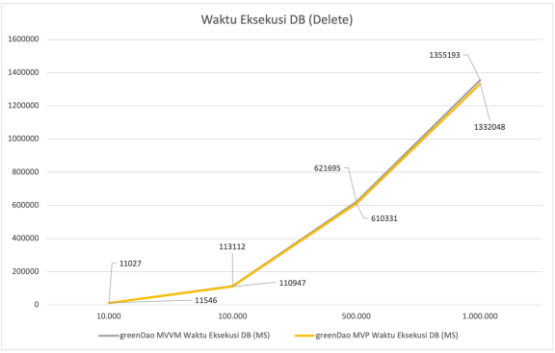
Gambar 16. Grafik Hasil Waktu Eksekusi Total Operasi *Update*

4.4 Eksperimen *Delete* Data

Hasil dari skenario *delete* sedikit berbeda dari skenario lainnya dimana MVP mengungguli banyak waktu eksekusi (*database* dan total), sedangkan MVVM dan greenDao unggul pada waktu eksekusi *view*. Tabel 15, Tabel 16, Tabel 17 dan Gambar 17, Gambar 18 dan Gambar 19 menunjukkan hasil eksperimen *delete*.

Tabel 15. Hasil Waktu Eksekusi pada Operasi *Delete*

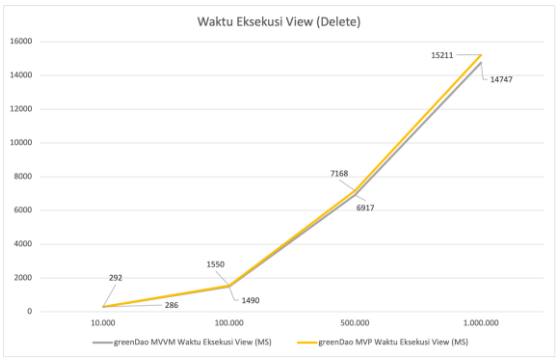
Skenario	Jumlah Data	Jenis Query	MVVM greenDao Waktu Eksekusi DB (MS)	MVP greenDao Waktu Eksekusi DB (MS)
1	10,000	DELETE	11027	11546
2	100,000	DELETE	113112	110947
3	500,000	DELETE	621695	610331
4	1,000,000	DELETE	1355193	1332048



Gambar 17. Grafik Hasil Waktu Eksekusi pada Operasi *Delete*

Tabel 16. Hasil Waktu Eksekusi pada Proses *View Delete*

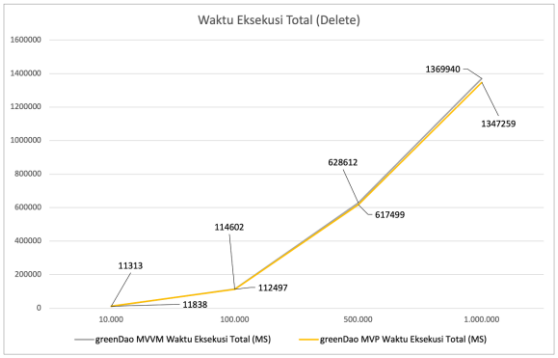
Skenario	Jumlah Data	Jenis Query	MVVM greenDao Waktu Eksekusi View (MS)	MVP greenDao Waktu Eksekusi View (MS)
1	10,000	DELETE	286	292
2	100,000	DELETE	1490	1550
3	500,000	DELETE	6917	7168
4	1,000,000	DELETE	14747	15211



Gambar 18. Grafik Hasil Waktu Eksekusi pada Proses *View Delete*

Tabel 17. Hasil Waktu Eksekusi Total Operasi *Delete*

Skenario	Jumlah Data	Jenis Query	MVVM greenDao Waktu Eksekusi Total (MS)	MVP greenDao Waktu Eksekusi Total (MS)
1	10,000	DELETE	11313	11838
2	100,000	DELETE	114602	112497
3	500,000	DELETE	628612	617499
4	1,000,000	DELETE	1369940	1347259



Gambar 19. Grafik Hasil Waktu Eksekusi Total Operasi *Delete*

4.5 Analisis Eksperimen

Mengacu pada hasil eksperimen yaitu Tabel 5, aplikasi dengan arsitektur MVVM dan *database* greenDao mengungguli sebagian besar skenario eksperimen. Kemudian, aplikasi yang menerapkan arsitektur MVVM lebih unggul dibanding MVP sebesar 10.180 ms (± 10 detik) pada rata–rata waktu eksekusi total.

Hasil eksperimen lebih detail tertera pada Tabel 6, Tabel 7, Tabel 8, Tabel 9, Tabel 10, Tabel 11, Tabel 12, Tabel 13, Tabel 14, Tabel 15 dan Tabel 16 yaitu sebagai berikut:

- MVVM mengungguli waktu eksekusi *database* pada 9 skenario, sedangkan MVP mengungguli waktu eksekusi *database* pada 4 skenario.
- MVVM mengungguli waktu eksekusi *view* pada 9 skenario, sedangkan MVP mengungguli waktu eksekusi *view* pada 8 skenario.
- MVVM mengungguli waktu eksekusi total pada 9 skenario, sedangkan MVP mengungguli waktu eksekusi total pada 4 skenario.

5. KESIMPULAN

Melihat dari hasil eksperimen di atas dapat diambil kesimpulan bahwa adanya pengaruh arsitektur terhadap performa *database* adalah benar. Aplikasi yang menerapkan arsitektur MVVM lebih unggul pada sebagian besar skenario, namun keunggulan MVVM dari aspek waktu eksekusi tidaklah signifikan bahkan pada waktu eksekusi *view* sebagian besar diungguli oleh MVP.

Eksperimen yang dilakukan masih menggunakan data dengan tipe *string*, disarankan untuk menggunakan data dengan tipe lain yang lebih bervariatif seperti *string* yang mengandung *base64* (untuk pengolahan data gambar). Disarankan pula untuk melakukan eksperimen dengan kompleksitas lain pada *query* tiap operasi CRUD.

DAFTAR PUSTAKA

[1] "Mobile Operating System Market Share Worldwide," *StatCounter*, Dec. 2020. https://gs.statcounter.com/os-market-share/mobile/worldwide (accessed Jan. 12, 2021).

[2] L. Corral and I. Fronza, "Better Code for Better Apps: A Study on Source Code Quality and Market Success of Android Applications," May 2015. doi: 10.1109/MobileSoft.2015.10.

[3] S. Ickin, K. Petersen, and J. Gonzalez-Huerta, "Why Do Users Install and Delete Apps? A Survey Study," 2017. doi: 10.1007/978-3-319-69191-6_13.

[4] B. C. Longho, "Room vs greenDAO for Android: A comparative analysis of performance of Room and greenDAO," Östersund, 2020.

[5] N. Obradovic, A. Kelec, and I. Dujlovic, "Performance analysis on Android SQLite database," Mar. 2019. doi: 10.1109/INFOTEH.2019.8717652.

[6] T. Lou, "A comparison of Android Native App Architecture MVC, MVP and MVVM," 2016.

[7] "Database Libraries," *AppBrain*, 2021. https://www.appbrain.com/stats/libraries/tag/database/database-libraries (accessed Apr. 11, 2021).

[8] A. Silberschatz, H. F. Korth, and S. Sudarshan, *Database system concepts*. McGraw-Hill, 2011.

[9] B. Wisnuadhi, G. Munawar, and U. Wahyu, "Performance Comparison of Native Android Application on MVP and MVVM," vol. 198, no. Issat, pp. 276–282, 2020, doi: 10.2991/aer.k.201221.047.

[10] N. S. Sibarani, G. Munawar, and B. Wisnuadhi, "Analisis Performa Aplikasi Native Android Menggunakan Bahasa Pemrograman Java dan Kotlin," *Researchget*, no. December, 2018, [Online]. Available: https://www.researchgate.net/publication/329525878_Analisis_Performa_Aplikasi_Android_Pada_Bahasa_Pemrograman_Java_dan_Kotlin

[11] "Core Classes," *Greenrobot.org*, 2020. https://greenrobot.org/greendao/documentation/introduction/ (accessed May 20, 2021).

[12] Mindorks, "Android MVP Architecture: Sample App," 2020. https://github.com/MindorksOpenSource/android-mvp-architecture (accessed Aug. 04, 2020).

[13] DiUS, "Java-Faker." https://github.com/DiUS/java-faker (accessed Aug. 04, 2020).