Jurnal Teknologi Informasi dan Ilmu Komputer (JTIIK) DOI: 10.25126/jtiik.2025128498
Vol. 12, No. 1, Februari 2025, hlm. 87-92 p-ISSN: 2355-7699
Akreditasi KEMENRISTEKDIKTI, No. 36/E/KPT/2019 e-ISSN: 2528-6579

ANALISIS PERBEDAAN PERFOMA PENGGUNAAN LAZYGRID DAN RECYCLERVIEW DALAM MENAMPILKAN KOLEKSI DATA

Raditya Muhammad Salman[1], Muhammad Aminul Akbar*[2] , Tri Afirianto[3]

[1,2,3]Universitas Brawijaya, Malang
Email: [1]dityaasalman@gmail.com, [2]muhammad.aminul@ub.ac.id, tri.afirianto@ub.ac.id
*Penulis Korespondensi

(Naskah masuk: 12 Januari 2024, diterima untuk diterbitkan: 10 Februari 2025)

Abstrak

Jetpack Compose adalah alat modern untuk membuat sebuah antarmuka pengguna aplikasi Android. Jetpack Compose dapat menyederhanakan dan mempercepat pengembangan antarmuka pengguna pada aplikasi Android. Namun, pada dokumentasi resmi developer.android.com belum dijelaskan apakah Jetpack Compose memiliki performa lebih baik dari XML. Penelitian ini dilakukan untuk mengetahui begaimana performa aplikasi yang menggunakan Jetpack Compose dan XML. Penelitian dilakukan pada kemampuan Jetpack Compose dan XML dalam menampilkan koleksi data, yaitu LazyGrid dan RecyclerView. Parameter yang digunakan untuk mengukur performa aplikasi tersebut adalah CPU utilization, memory usage, dan rendering time. Parameter CPU utilization dan memory usage digunakan karena perbedaan kode untuk membangun antarmuka pengguna. Perbedaan kode tersebut dapat mempengaruhi kinerja suatu aplikasi. Sedangkan parameter rendering time digunakan untuk mengetahui kecepatan aplikasi dalam menampilkan sebuah frame. Metode dari penelitian dimulai dengan merancang skenario pengujian, implementasi RecyclerView dan LazyGrid, pengambilan data, pengolahan data dan analisis hasil, kemudian menarik kesimpulan dari hasil penelitian. Dari penelitian ini didapatkan bahwa terdapat perbedaan signifikan pada performa aplikasi yang menggunakan RecyclerView dan LazyGrid kecuali pada memory usage dengan jumlah koleksi data 50. Aplikasi yang menggunakan RecyclerView unggul pada CPU utilization dan rendering time dengan jumlah koleksi data 20, 50, dan 100. Aplikasi yang menggunakan RecyclerView juga unggul pada memory usage dengan jumlah koleksi data 20 namun aplikasi yang menggunakan LazyGrid unggul dalam menampilkan jumlah koleksi data 50 dan 100.

Kata kunci: *Jetpack Compose, XML, LazyGrid, RecyclerView, perbandingan, performa aplikasi, CPU utilization, memory usage, rendering time, koleksi data.*

DIFFERENCE ANALYSIS OF LAZYGRID AND RECYCLERVIEW PERFORMANCE IN SHOWING DATA COLLECTION

Abstract

Jetpack Compose is a modern tool dedicated for building Android apps user interface. Jetpack Compose simplifies and accelerate user interface development in Android apps. But in official documentation developer.android.com have not been told how Jetpack Compse and XML perfomances are. The research was done to the ability of Jetpack Compose and XML in showing data collection which are LazyGrid and RecyclerView. The parameters that had been used to measure the apps performances are CPU utilization, memory usage, and rendering time. CPU utilization and memory usage are used as parameters because of the code that are used for developing user interface are different. The code difference can affect how the Application works. Rendering time is used as a parameter to know how much time does it takes to complete a single frame. The methods of doing this research started by designing test scenarios, RecyclerView and LazyGrid implementation, collecting data, processing data and result analysis, and comclusion. From this research was achieved that a significant differences of performance on apps with RecyclerView and LazyGrid have found except in memory usage with 50 total of data collection. Apps wity RcyclerView was superior in CPU utilization and rendering time with 20, 50, and 100 total of data collection. Apps performance that using RecyclerView was also superior in memory usage with 20 total of data collection but apps that using LazyGrid was superior with 50 and 100 total of data collection.

Keywords: *Jetpack Compose, XML, LazyGrid, RecyclerView, comparison, apps performances, CPU utilization, memory usage, rendering time, data collection*

1. PENDAHULUAN

Sejak pertama kali Android diluncurkan pada tahun 2008, jumlah pengguna Android meningkat pesat (Horton & Safari, 2019). Pada saat itu pengembang aplikasi Android hanya mengenal satu cara yaitu View-based untuk membuat tampilan antarmuka pengguna aplikasi. Cara ini mengharuskan pengembang untuk membuat sebuah *layout* pada file XML. *Layout* ini kemudian dihubungkan ke class *Activity* pada file Java/Kotlin. Namun, sekarang telah ada cara baru untuk membuat antarmuka pengguna yaitu menggunakan Jetpack Compose.

Jetpack Compose adalah alat modern untuk membuat sebuah antarmuka pengguna aplikasi Android (Android Developers, 2023). Jetpack Compose dapat menyederhanakan dan mempercepat pengembangan antarmuka pengguna pada aplikasi Android. Pengembangan antarmuka pengguna juga membutuhkan sedikit kode dan lebih intuitif sehingga pengembang hanya perlu mendeskripsikan antarmuka. Namun, pada dokumentasi resmi developer.android.com belum dijelaskan apakah Jetpack Compose memiliki performa lebih baik dari XML terutama pada kemampuan aplikasi dalam menampilkan koleksi data. Performa aplikasi dalam menampilkan koleksi data dianggap penting karena koleksi data dapat ditemukan pada hampir setiap aplikasi Android.

Performa aplikasi yang dimaksud adalah efektifitas dari sebuah aplikasi dalam menggunakan sumber daya (Microsoft Learn, 2023). Aspek yang dicakup dalam performa aplikasi antara lain penggunaan CPU, konsumsi memori, konsumsi daya, pemanfaatan jaringan dan penyimpanan, performa animasi, dan lainnya. Baik atau buruknya performa aplikasi dapat mempengaruhi pengalaman pengguna secara langsung (AppMaster, 2023).Aplikasi yang memiliki performa buruk, seperti lamban dan kurang responsif, dapat menyebabkan pengguna merasa frustasi dan akhirnya meninggalkan aplikasi tersebut.

Terdapat penelitian yang dilakukan oleh Jakub Szczukin (2023). Penelitian ini membandingkan dua cara untuk mengimplementasikan antarmuka pengguna pada aplikasi Android. Dibandingkan dengan XML yang digunakan untuk membuat antarmuka pengguna untuk aplikasi Android, cara baru dalam mengimplementasikan antarmuka pengguna ini masih jarang digunakan oleh pengembang. Penelitian ini juga membandingkan performa aplikasi ketika menggunakan kedua metode tersebut secara bersamaan. Pengujian kinerja aplikasi pada penelitian ini berfokus pada waktu yang dibutuhkan untuk melakukan operasi pada antarmuka pengguna. Waktu yang diperlukan untuk menyelesaikan suatu operasi diukur menggunakan alat Macrobenchmark yang ada di Android Studio. Hasil penelitian ini menunjukkan bahwa kinerja aplikasi yang menggunakan Jetpack Compose sedikit lebih buruk dibandingkan XML, dan aplikasi yang menggunakan kombinasi kedua metode tersebut memiliki kinerja yang lebih buruk. (Szczukin, 2023).

Berdasarkan permasalahan yang telah ditemukan maka dapat dilakukan sebuah analisis terhadap performa aplikasi Android yang dibangun menggunakan Jetpack Compose dan XML. Analisis ini dilakukan dengan cara mengimplementasikan Jetpack Compose dan XML pada aplikasi *dummy*. Aplikasi dummy yang digunakan adalah aplikasi yang dapat menampilkan koleksi data. Aplikasi diambil dari GitHub karena kemudahan akses dan Aplikasi yang tersedia bersifat *open source*. Implementasi yang dilakukan dibatasi pada *activity* yang menggunakan RecyclerView dan LazyGrid untuk menampilkan data.

RecyclerView adalah ViewGroup yang berisikan views berkaitan dengan data (Android Developers, n.d.). RecyclerView sering kali digunakan untuk menampilkan koleksi data. Koleksi data merupakan sekumpulan data yang dapat ditampilkan pada Recyclerview. Data yang ditampilkan bisa berupa tulisan, angka, atau gambar dengan format daftar atau grid.

LazyGrids adalah composable yang dapat menyajikan koleksi data dalam bentuk grid (Android Developers, n.d.). Dalam penggunaanya, LazyGrids dapat digunakan secara vertikal atau horizontal. LazyVerticalGrid akan menampilkan data yang dapat digeser secara vertikal dan data yang dimuat membentang secara horizontal begitu juga sebaliknya pada LazyHorizontalGrid.

Analisis dilakukan dengan cara mengukur performa aplikasi pada *activity* yang mengimplementasikan Recyclerview dan LazyGrid. Performa yang diuji adalah *CPU utilization, memory usage,* dan *rendering time*. Parameter *CPU utilization* dan *memory usage* digunakan karena perbedaan kode untuk membangun antarmuka pengguna. Perbedaan kode tersebut dapat mempengaruhi kinerja suatu aplikasi. Sedangkan parameter *rendering time* digunakan untuk mengetahui kecepatan aplikasi dalam menampilkan sebuah *frame*. Setelah pengujian dilakukan, hasil pengujian dapat dianalisis apakah ada perbedaan performa pada aplikasi Jetpack Compose dan XML.

2. METODE PENELITIAN

Metode yang digunakan pada penelitian ini dimulai dengan perancangan skenario pengujian dan diikuti implementasi LazyGrid dan RecyclerView. Kemudian dilakukan pengambilan dan pengolahan data. Data yang telah diolah dianalisis hasilnya dan ditarik kesimpulan dari penelitian yang dilakukan.

Skenario pengujian dirancang dengan perameter *CPU utilization, memory usage,* dan *rendering time.* Aplikasi yang dilakukan pengujian adalah aplikasi yang menggunakan RecyclerView dan LazyGrid dalam menampilkan koleksi data. Pengujian dilakukan pada kedua aplikasi dengan jumlah koleksi data 20, 50, dan 100.

Aplikasi yang digunakan untuk penelitian ini dilakukan implementasi LazyGrid dan

RecyclerView. Implementasi ini dilakukan secara teliti dengan memperhatikan parameter pada tiap *item*-nya seperti lebar, tinggi, padding, margin, serta parameter lainnya yang dapat mempengaruhi tampilan. Hal ini dilakukan untuk mengurangi kemungkinan bias pada data uji hasil pengujian.

Pengambilan data uji dilakukan sebanyak 30 kali pada 3 parameter pengujian. Pada parameter pengujian *CPU utilization* dan *memory usage*, data uji diambil secara *real-time* pada saat aplikasi digeser dari halaman paling atas ke halaman paling bawah. Pada parameter pengujian *rendering time*, data uji diambil saat aplikasi telah digeser ke halaman paling bawah karena pengambilan data uji ini hanya mengambil 120 *frame* terbaru.

Data yang telah didapat kemudian diolah. Pengolahan data ini dilakukan dengan cara data uji dilakukan uji normalitas untuk mengetahui apakah data uji terdistribusi secara normal. Kemudian data uji dilakukan uji beda untuk menemukan apakah terdapat perbedaan yang signifikan pada kedua aplikasi.

Analisis hasil dilakukan berdasarkan hasil dari pengolahan data. Data uji dianalisis apakah terdapat perbedaan yang signifikan berdasarkan hasil pengolahan data uji. Perbedaan dari data uji kemudian dihitung persentase perbandingannya.

Dari analisis hasil dapat ditarik kesimpulan apakah ada perbedaan performa aplikasi yang menggunakan RecyclerView dan LazyGrid. Kesimpulan aplikasi mana yang lebih unggul juga dapat diambil.

3. HASIL DAN PEMBAHASAN

3.1. Perancangan Skenario Pengujian

Pengujian dilakukan dengan cara mengambil 30 data uji pada setiap parameter dan jumlah koleksi data. Pada pengambilan data, halaman digeser ke bawah untuk menampilkan koleksi data dari paling atas hingga paling bawah. Untuk menjaga konsistensi dalam menggeser halaman digunakan aplikasi AutoScroll. Tiga alat yang berbeda digunakan untuk mengukur setiap performa aplikasi. SnapDragon Profiler digunakan untuk merekam CPU utilization, Android Studio Profiler digunakan untuk mengukur memory usage, dan Profile HWUI Rendering digunakan untuk mengukur rendering time. Langkah pengujian aplikasi dapat dilihat pada Tabel 1.

Tabel 1 Langkah pengujian performa aplikasi

Langkah ke:	Langkah pengujian
1	Menghubungkan perangkat Android ke perangkat komputer menggunakan kabel USB.
2	Membuka aplikasi Automatic Scroll dan mengaktifkan fitur auto-scroll.
3	Membuka aplikasi yang diuji.
4	Menjalankan auto-scroll bersamaan dengan memulai pengujian.
5	Menghentikan pengujian ketika halaman sudah ditampilkan.
6	Menyimpan data hasil pengujian pada tabel.

Perangkat keras yang digunakan untuk penelitian ini adalah sebuah ponsel Android dan laptop. Ponsel digunakan untuk menjalankan aplikasi yang diuji. Laptop digunakan untuk menjalankan aplikasi penguji yaitu SnapDragon Profiler, Android Studio Profiler, dan Profile HWUI Rendering. Spesifikasi dari kedua perangkat keras dapat dilihat pada

Tabel 2 dan Tabel 3.

Tabel 2 Spesifikasi Laptop

Tipe	ASUS VivoBook 14
CPU	Intel i5-10210U
RAM	8 GB
Disk	512 GB

Tabel 3 Spesifikasi Ponsel

Tipe	Redmi Note 10 5G
CPU	MediaTek Dimensity 700
RAM	8 GB
Storage	128 GB

3.2. Implementasi

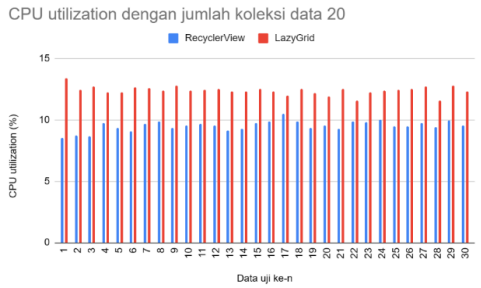
Implementasi yang dilakukan adalah menambahkan koleksi data pada aplikasi dan menerapkan LazyGrid dan RecyclerView.

Koleksi data yang ditampilkan pada aplikasi disimpan dalam bentuk arrayList. Hal tersebut dilakukan untuk mencegah hasil pengujian bias karena perbedaan library untuk memuat gambar. Koleksi data yang disimpan pada arrayList memungkinkan aplikasi untuk menyimpan gambar secara langsung tanpa harus mengakses internet.

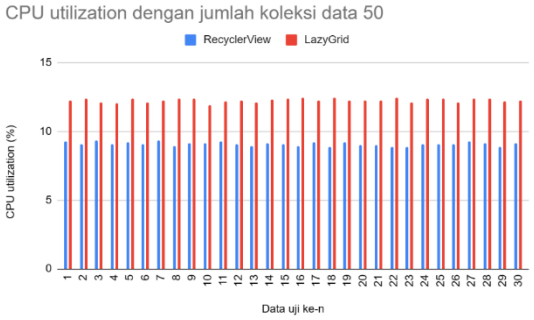
Implementasi LazyGrid dan RecyclerView dilakukan pada *activity* untuk menampilkan koleksi data pada aplikasi Android. Implementasi ini dilakukan secara teliti dengan memperhatikan parameter pada tiap item-nya seperti lebar, tinggi, padding, margin, serta parameter lainnya yang dapat mempengaruhi tampilan. Hal ini dilakukan untuk mengurangi kemungkinan bias pada data uji hasil pengujian.

3.3 Pengambilan Data

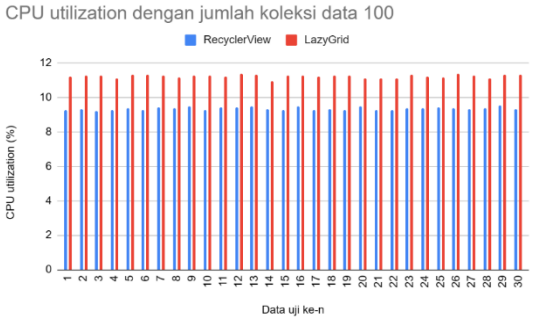
Pengambilan data performa CPU utilization pada setiap Aplikasi dilakukan dengan SnapDragon Profiler. SnapDragon Profiler akan menampilan persentase real-time dan rata-rata dari CPU utilization. Data yang diambil untuk penelitian ini dalah rata-rata dari CPU utilization. Data yang diambil ditampilkan dengan bagan pada Gambar 1, Gambar 2, dan Gambar 3.



Gambar 1 *CPU utilization* dengan jumlah koleksi data 20

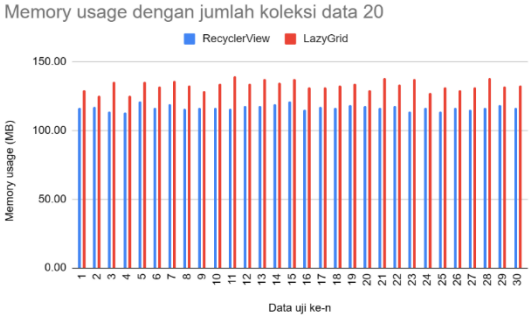


Gambar 2 *CPU utilization* dengan jumlah koleksi data 50

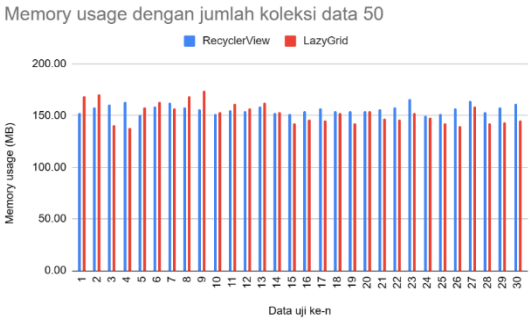


Gambar 3 *CPU utilization* dengan jumlah koleksi data 100

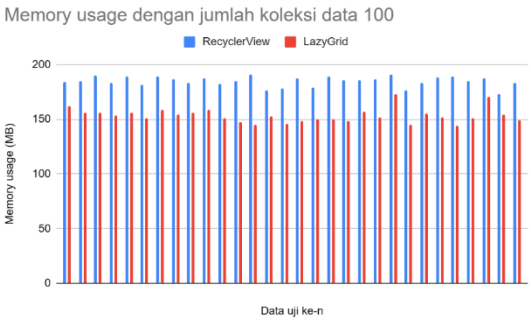
Pengambilan data performa memory usage pada setiap Aplikasi dilakukan dengan Android Studio Profiler. Android Studio Profiler akan menampilan jumlah real-time dari memory usage. Data yang diambil untuk penelitian ini adalah rata-rata nilai terbesar dari memory usage. Data yang diambil ditampilkan dengan bagan pada Gambar 4, Gambar 5, dan Gambar 6.



Gambar 4 *Memory usage* dengan jumlah koleksi data 20

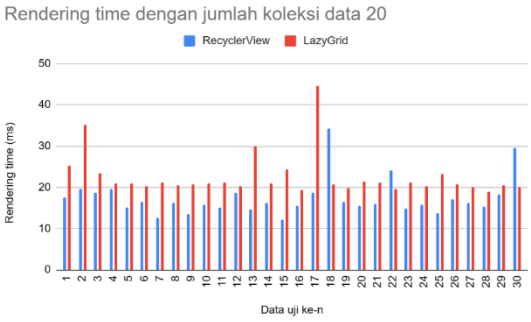


Gambar 5 *Memory usage* dengan jumlah koleksi data 50

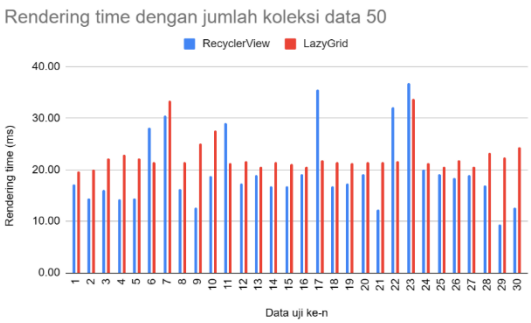


Gambar 6 *Memory usage* dengan jumlah koleksi data 100

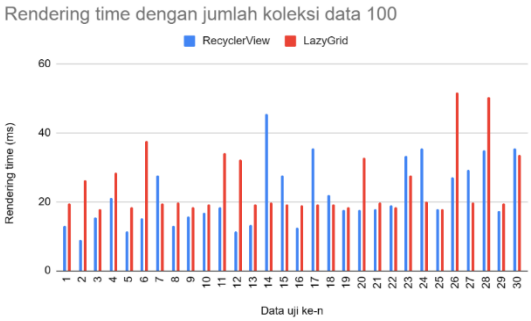
Pengambilan data performa rendering time pada setiap Aplikasi dilakukan dengan Profile HWUI Rendering. Profile HWUI Rendering akan menampilan data rendering time 120 frame terakhir. Data yang diambil untuk penelitian ini dalah rata-rata dari rendering time 120 frame terakhir. Data yang diambil ditampilkan dengan bagan pada



Gambar 7 *Rendering time* dengan jumlah koleksi data 20



Gambar 8 *Rendering time* dengan jumlah koleksi data 50



Gambar 9 *Rendering time* dengan jumlah koleksi data 100

3.4 Pengolahan Data

Data uji yang telah terkumpul dilakukan uji normalitas untuk mengetahui normalitas distribusi data. Uji normalitas digunakan dengan fungsi yang tersedia pada bahasa pemrograman R. Dari uji normalitas didapatkan p-value. Jika p-value yang didapatkan bernilai lebih besar dari 0.05 maka distribusi data uji normal. Langkah pengujian selanjutnya ditentukan dari hasil uji normalitas. Jika salah satu dari data uji yang dibandingkan bersifat tidak normal maka data tersebut dilakukan uji Mann-Whitney dan jika kedua data uji normal maka data tersebut dilakukan uji T-test. Hasil uji normalitas dapat dilihat pada Tabel 4.

Tabel 4 Tabel hasil uji normalitas

Parameter uji	Jumlah koleksi data	*p-value*		Uji beda yang akan dilakukan
		RecyclerView	LazyGrid	
CPU utilization	20	0,372*	0,043	Mann-Whitney
	50	0,760*	0,167*	T-test
	100	0,338*	0,023	Mann-Whitney
Memory usage	20	0,403*	0,604*	T-test
	50	0,276*	0,106*	T-test
	100	0,047	0,006	Mann-Whitney
Rendering time	20	< 0,001	< 0,001	Mann-Whitney
	50	< 0,001	< 0,001	Mann-Whitney
	100	0,008	< 0,001	Mann-Whitney

Nilai *p-value* data uji yang memiliki tanda bintang (*) adalah data uji yang terdistribusi secara normal. Data uji yang akan dilakukan uji beda parametrik T-test adalah data uji pada parameter *CPU utilization* dengan jumlah koleksi data 50 dan parameter *Memory usage* dengan jumlah koleksi data 20 dan 50.

Pada hasil uji normalitas ditemukan tiga pasang data uji yang dibandingkan menggunakan T-test. Ketiga pasang data tersebut adalah data dengan parameter uji CPU utilization dengan jumlah koleksi data 50 serta data dengan parameter Memory usage dengan jumlah koleksi data 20 dan 50. Ketiga pasang data uji tersebut dilakukan uji homogenitas sebelum kedua pasang data uji dilakukan uji beda. Hasil dari uji homogenitas menunjukkan bahwa ketiga pasang data uji tidak bersifat homogen.Semua pasangan data uji sesuai dengan parameter uji dan jumlah data, kecuali pasangan data uji yang telah dilakukan uji beda T-test, dilakukan uji beda Mann-Whitney. Hasil dari uji beda yang telah dilakukan dapat dilihat pada Tabel 5.

Tabel 5 Tebel hasil uji beda

Parameter uji	Jumlah koleksi data	Uji beda yang dilakukan	*p-value*
CPU utilization	20	Mann-Whitney	< 0,001
	50	T-test	< 0,001
	100	Mann-Whitney	< 0,001
Memory usage	20	T-test	< 0,001
	50	T-test	0,053
	100	Mann-Whitney	< 0,001
Rendering time	20	Mann-Whitney	< 0,001
	50	Mann-Whitney	< 0,001
	100	Mann-Whitney	0,027

3.5 Analisis Hasil

Uji beda pada parameter memory usage dengan jumlah koleksi data 50 didapatkan p-value lebih dari 0.05. Sehingga tidak ada perbedaan yang signifikan pada aplikasi yang menggunakan RecyclerView dan LazyGrid dengan parameter pengujian memory usage dan jumlah koleksi data 50. Sedangkan uji beda pada parameter dan jumlah koleksi data yang lain didapatkan p-value kurang dari 0.05. Sehingga terdapat perbedaan yang signifikan pada aplikasi yang menggunakan RecyclerView dan LazyGrid dengan parameter pengujian dan jumlah koleksi data yang lain.

Meskipun perbedaan pada parameter memory usage dengan jumlah koleksi data 50 tidak signifikan. Perbedaan performa aplikasi dengan RecyclerView dan LazyGrid dapat dihitung.

Hasil dari penghitungan persentase perbandingan RecyclerView dan LazyGrid dapat dilihat pada Tabel 6. Persentase perbandingan yang bernilai negatif berarti RecyclerView memiliki rata-rata yang lebih kecil dari LazyGrid.

Nilai negatif dari hasil penghitungan persentase perbandingan menunjukkan bahwa rata-rata performa RecyclerView lebih sedikit dibandingkan dengan LazyGrid dan sebaliknya jika nilai persentase perbandingan positif.

CPU utilization aplikasi yang menggunakan RecyclerView lebih sedikit 26,44% pada jumlah koleksi data 20, 29,95% pada jumlah koleksi data 50, dan 18,35% pada jumlah koleksi data 100 dibandingkan dengan aplikasi yang menggunakan LazyGrid.

Tabel 6 Persentase Perbandingan Performa

Parameter pengujian	Jumlah koleksi data	Rata-rata RecyclerView	Rata-rata LazyGrid	Persentase perbandingan RecyclerView dan LazyGrid (%)
CPU utilization	20	9.52	12.40	-26.44
	50	9.09	12.26	-29.95
	100	9.34	11.22	-18.35
Memory usage	20	116.6	132.8	-13.0
	50	156.5	152.5	2.6
	100	184.9	153.5	18.6
rendering time	20	17.5	22.6	-25.6
	50	19.5	22.7	-14.8
	100	21.5	24.6	-13.1

Memory usage aplikasi yang menggunakan RecyclerView lebih sedikit 12,99% pada jumlah koleksi data 20, lebih banyak 2,5% pada jumlah koleksi data 50, dan lebih banyak 18,62% pada jumlah koleksi data 100 dibandingkan aplikasi yang menggunakan LazyGrid.

Rendering time aplikasi yang menggunakan RecyclerView lebih sedikit 25,64% pada jumlah koleksi data 20, 14,8% pada jumlah koleksi data 50, dan 13,14% pada jumlah koleksi data 100 dibandingkan dengan aplikasi yang menggunakan LazyGrid.

Performa RecyclerView yang lebih unggul dipengaruhi oleh LazyGrid masih dalam pengembangan dan belum berada di versi paling optimal. Pada *memory usage* RecyclerView lebih unggul pada jumlah koleksi data yang lebih sedikit karena RecyclerView menggunakan kembali *views* yang berada di *cache* sedangkan LazyGrid membuat *views* baru untuk ditampilkan. Namun untuk menampilkan jumlah data koleksi yang lebih banyak, LazyGrid lebih unggul karena hanya memuat *items* yang dibutuhkan sedangkan RecyclerView memuat semua *items* yang akan ditampilkan untuk disimpan di *cache* (Google Developers Training Team, 2018).

4. KESIMPULAN

Kesimpulan yang dapat diambil dari penelitian yang telah dilakukan adalah performa aplikasi yang menggunakan RecyclerView untuk menampilkan koleksi data lebih unggul dalam CPU utilization dan rendering time dibandingkan aplikasi yang menggunakan LazyGrid. Memory usage aplikasi yang menggunakan RecyclerView juga unggul saat menampilkan 20 koleksi data. Namun untuk menampilkan koleksi data berjumlah 50 dan 100, memory usage aplikasi yang menggunakan LazyGrid lebih unggul.

Saat penelitian ini dilakukan, LazyGrid masih terus dikembangkan. Besar kemungkinan bahwa performa aplikasi yang menggunakan LazyGrid akan meningkat. Saran yang dapat penulis berikan untuk penelitian selanjutnya adalah menambahkan jumlah koleksi data yang akan diuji. Penelitian selanjutnya juga dapat dilakukan menggunakan koleksi data yang didapat dari API. Orientasi LazyGrid dan RecyclerView horizontal juga dapat dilakukan pada penelitian selanjutnya.

DAFTAR PUSTAKA

ANDROID DEVELOPERS. 2023. *Why Compose | Jetpack Compose*. [Online] Tersedia di: https://developer.android.com/jetpack/compose/why-adopt [Diakses pada 21 September 2023].

Android Developers, n.d. *Create dynamic lists with RecyclerView.* [Online] Tersedia di: https://developer.android.com/develop/ui/views/layout/recyclerview [Diakses pada 13 September 2023].

Android Developers, n.d. *Lists and grids | Jetpack Compose.* [Online] Tersedia di: https://developer.android.com/jetpack/compose/lists [Diakses pada 21 September 2023].

APPMASTER. 2023. *Pengalaman Pengguna (UX).* [Online] Tersedia di: https://appmaster.io/id/glossary/pengalaman-pengguna-ux [Diakses pada 21 September 2023].

GOOGLE DEVELOPERS TRAINING TEAM, 2018. *Android Developer Fundamentals.* [Online] Available at: https://google-developer-training.github.io/android-developer-fundamentals-course-concepts-v2/unit-2-user-experience/lesson-4-user-interaction/4-5-c-recyclerview/4-5-c-recyclerview.html [Accessed 9 Januari 2024].

MICROSOFT LEARN. 2023. *Pengantar Performa Aplikasi Windows - Windows apps.* [Online] Tersedia di: https://learn.microsoft.com/id-id/windows/apps/performance/introduction [Diakses pada 21 September 2023].

HORTON, J., & SAFARI, AN O. M. COMPANY. 2019. *Android Programming with Kotlin for Beginners*.

SZCZUKIN, J. 2023. Performance analysis of user interface implementation methods in mobile applications Analiza wydajności metod implementacji interfejsów użytkownika w aplikacjach mobilnych. JCSI,.