Jurnal Pengembangan Teknologi Informasi dan Ilmu Komputer e-ISSN: 2548-964X
Vol. 9, No. 9, September 2025, hlm. x-x http://j-ptiik.ub.ac.id

Analisis Perbandingan Kinerja Komunikasi API Menggunakan Retrofit dan Ktor di Aplikasi Android (Studi Kasus: Aplikasi Katalog Film)

Muhammad Syafi Islam[1], Agi Putra Kharisma[2], Putra Pandu Adikara[3]

Program Studi Teknik Informatika, Fakultas Ilmu Komputer, Universitas Brawijaya
Email: [1]muhammadsyafi@student.ub.ac.id, [2]agi@ub.ac.id, , [3]adikara.putra@ub.ac.id

Abstrak

Pertumbuhan signifikan pengguna *smartphone* di Indonesia turut meningkatkan kebutuhan terhadap aplikasi Android yang efisien dan responsif. Salah satu aspek penting dalam pengembangan aplikasi Android adalah efisiensi komunikasi *Application Programming Interface* (API). Penelitian ini bertujuan untuk melakukan analisis perbandingan performa komunikasi API menggunakan dua *networking library* populer, yaitu Retrofit dan Ktor, dengan studi kasus pada aplikasi katalog film. Aspek yang dibandingkan meliputi waktu respons, penggunaan CPU dan penggunaan memori. Metode penelitian yang digunakan mencakup studi literatur, analisis kebutuhan, perancangan dan implementasi aplikasi, pengujian performa komunikasi API dengan 30 sampel, serta analisis data menggunakan uji-t. Hasil pengujian menunjukkan bahwa tidak terdapat perbedaan secara statistik pada aspek waktu respons ($p = 0,52$) dan penggunaan CPU ($p = 0,84$). Namun, terdapat perbedaan secara statistik pada aspek penggunaan memori ($p = 0,02$), sedangkan Retrofit menunjukkan penggunaan memori yang lebih rendah dibandingkan Ktor ($85,99 \pm 22,36$ vs. $106,32 \pm 12,74$). Dengan demikian, meskipun keduanya memiliki performa serupa dalam hal waktu respons dan penggunaan CPU, Retrofit lebih unggul dalam efisiensi penggunaan memori. Hasil ini diharapkan dapat menjadi acuan bagi pengembang dalam memilih *library* komunikasi API yang sesuai dengan kebutuhan aplikasi Android mereka.

Kata kunci: *API*, *Retrofit*, *Ktor*, *Android*, Performa Komunikasi

Abstract

The significant growth of smartphone users in Indonesia has increased the demand for efficient and responsive Android applications. One critical aspect of Android app development is the efficiency of Application Programming Interface (API) communication. This study aims to analyze the performance comparison of API communication using two popular networking libraries, namely Retrofit and Ktor, with a case study on a movie catalog application. The aspects compared include response time, CPU usage, and memory usage. The research method includes literature review, requirements analysis, application design and implementation, API communication performance testing with 30 samples, and data analysis using the t-test. The test results show no statistically significant difference in response time ($p = 0.52$) and CPU usage ($p = 0.84$). However, a statistically significant difference was found in memory usage ($p = 0.02$), where Retrofit demonstrated lower memory consumption compared to Ktor (85.99 ± 22.36 vs. 106.32 ± 12.74). Thus, although both libraries offer similar performance in terms of response time and CPU usage, Retrofit is superior in memory efficiency. These findings are expected to serve as a reference for developers in selecting the appropriate API communication library for their Android applications

Keywords: API, Retrofit, Ktor, Android, Communication Performance

1. PENDAHULUAN

Pada zaman ini teknologi merupakan sebuah kebutuhan pokok yang diperlukan oleh semua manusia terutama pada penggunaan *smartphone*. Pada tahun 2023, pengguna aktif *smartphone* di Indonesia mencapai 209,3 juta orang sedangkan pada tahun 2015 penggunanya hanya sebanyak 54 juta orang (Syafa Fadhilah Tegar Andalas, 2024). Data tersebut membuktikan bahwa penggunaan *smartphone* akan terus bertambah seiring berjalannya waktu. Menurut Akraman et al. (2018), pada awal tahun 2016 *smartphone* yang populer digunakan adalah Android dengan 1,8 miliar pengguna yang kemudian diikuti oleh iOS dengan 463 juta pengguna.

Seiring meningkatnya jumlah pengguna *smartphone*, permintaan terhadap pengembangan sebuah aplikasi juga semakin meningkat. Salah satu faktor utama yang menentukan performa aplikasi adalah efisiensi komunikasi API dalam mengelola data dari server ke pengguna. *Application Programming Interfaces* (API) merupakan hal esensial pada pengembangan perangkat lunak modern yang memberikan koleksi dari layanan protokol dan alat yang memungkinkan terjadinya komunikasi antara perangkat lunak lainnya (Mohammed Mudassir & Mohammed Mushtaq, 2024). Menurut Choirudin & Adil (2019) API memungkinkan berbagai sistem untuk berkomunikasi, mengambil dan mengirim data, sehingga dapat menyediakan konten dinamis dan pengalaman pengguna yang lebih baik. Seiring dengan perkembangan di era digital, banyak beberapa jenis *networking library* yang sering digunakan oleh banyak pengembang aplikasi Android (Vernanda et al., 2023).

Meskipun komunikasi API merupakan komponen krusial dalam pengembangan aplikasi Android, pemilihan *library* yang tepat masih menjadi tantangan bagi pengembang. Retrofit dan Ktor adalah dua *library* populer yang sering digunakan, namun masing-masing memiliki perbedaan karakteristik yang unik dalam penggunaannya dalam mengembangkan aplikasi Android (Santoso et al., 2025). Retrofit dikenal dengan kemudahannya saat memanggil layanan api dan konversi data Javascript Object Notation (JSON) yang sudah ditangani (Iqbal & Wali, 2022). Ktor menawarkan pendekatan berbasis coroutine yang menyederhanakan *asynchronous programming* (Lang & Prähofer, 2024). Namun, penelitian yang secara langsung membandingkan performa kedua *library* ini masih tergolong minim. Kurangnya informasi mengenai keunggulan dan kelemahan dari masing-masing *library* dapat menyebabkan pengembang kesulitan dalam menentukan pilihan terbaik sesuai dengan kebutuhan aplikasi mereka. Oleh karena itu, diperlukan analisis perbandingan performa antara Retrofit dan Ktor untuk memahami sejauh mana perbedaan keduanya dalam meningkatkan efisiensi komunikasi API pada aplikasi Android.

Retrofit merupakan *networking library* oleh Square yang populer sangat karena kemudahan penggunaannya dan mengubah HTTP API menjadi sebuah *interface* (Sufyan & Banerjee, 2019). Retrofit menggunakan konsep *Representational State Transfer* (REST) dan mendukung berbagai jenis permintaan *Hypertext Transfer Protocol* (HTTP) seperti GET, POST, PUT, dan DELETE (Christhover et al., 2022). Selain itu, Retrofit memungkinkan penggunaan converter untuk mengubah data JSON menjadi objek Java atau Kotlin secara otomatis dengan bantuan dari *library* gson (Iqbal & Wali, 2022).

Di sisi lain, Ktor adalah *framework* yang dikembangkan oleh JetBrains, yang menawarkan fleksibilitas tinggi dan performa yang cepat (Lang & Prähofer, 2024). Ktor mulai mendapatkan perhatian karena kemampuannya untuk mendukung baik klien maupun server dalam satu *framework*, serta fleksibilitas dalam konfigurasi dan pengelolaan permintaan HTTP (Lang & Prähofer, 2024). Kedua *library* ini memiliki karakteristik, kelebihan, dan kekurangan masing-masing dalam hal performa, kemudahan penggunaan, serta fleksibilitas. Retrofit dengan kemudahan yang diberikan dan dukungan saat melakukan penanganan API (Widyaningtyas & Wahyono, 2024). Namun, Ktor menawarkan pendekatan yang lebih fleksibel dan modern, dengan dukungan penuh untuk operasi *asynchronous*, yang memungkinkan pengelolaan komunikasi API yang lebih efisien dan responsif (Lang & Prähofer, 2024). Mengingat pentingnya komunikasi API dalam aplikasi Android dan adanya beberapa pilihan *library* yang dapat digunakan, analisis perbandingan performa antara Retrofit dan Ktor menjadi sangat relevan. Performa dalam hal ini dapat mencakup beberapa aspek, seperti waktu respons, penggunaan CPU dan memori.

2. LANDASAN KEPUSTAKAAN

2.1 Kajian Pustaka

Penelitian mengenai perbandingan *library networking* oleh Ferryansyah et al. (2018) dengan "Analisis Performansi HTTP Networking Library pada Android (Studi Kasus:

Portal Berita)" Penelitian ini akan melakukan perbandingan antara library networking yang umum digunakan, seperti HttpUrlConnection, Asynchronus Http Client, Retrofit, dan OkHttp. Pada penelitian tersebut metrics yang digunakan adalah waktu respons, penggunaan memori dan penggunaan internet sebagai pembanding antara library yang digunakan. Penguji membandingkan pengambilan data yang terdiri dari teks dan teks bersamaan dengan gambar yang dilakukan sebanyak dua puluh kali untuk setiap library yang dibandingkan. Hasil penelitian ini menunjukkan bahwa penggunaan network pada library Asyncronous Http sangat kecil dibandingkan dengan library lainnya. Namun library OkHttp menunjukkan waktu respons yang paling cepat dari semua library yang dibandingkan.

Terdapat penelitian mengenai "Analysis on the Comparison of Retrofit and Volley Libraries on Android-Based Mosque Application" oleh (Saputra et al., 2018) yang membuat aplikasi bernama MasjidPro. Aplikasi ini dapat digunakan untuk melihat informasi detail dari masjid terdeakat, aktivitas dari tiap masjid dan lain-lain. Tujuan dari penelitian ini adalah membandingkan dua library networking, yaitu Retrofit dan volley. Setelah dibandingkan, library terbaik yang akan digunakan dalam pengembangan aplikasi MasjidPro. Percobaan akan dilakukan kepada masing-masing library yang dibandingkan dengan mengambil volume data masjid dari jumlah 10 hingga 10.000 sebanyak lima kali percobaan. Hasil dari penelitian tersebut peneliti mendapatkan hasil library Retrofit lebih unggul dalam segi parsing data dan waktu yang singkat pada saat mengambil data. Oleh karena itu, peneliti menggunakan Retrofit sebagai networking library yang digunakan dalam mengembangkan aplikasi MasjidPro.

Terdapat sebuah penelitian dengan judul "Studi Performa Android Networking Library antara Fast Android Network Library, Retrofit, dan OkHttp" oleh (Vernanda et al., 2023) yang membanding networking library android yang umum digunakan pada pengembangan aplikasi berbasis Android, yaitu Fast Android Network, Retrofit dan OkHttp. Pada penelitian ini akan dinilai berdasarkan tiga kriteria, yaitu waktu respons, penggunaan memori dan penggunaan internet. Pengujian akan dilakukan dengan menjalankan delapan kasus uji untuk mendapatkan data dari tiap library yang nantinya data dari hasil kasus uji akan digunakan sebagai analisis menurut tiga kriteria yang sudah ditentukan sebelumnya. Data hasil pengujian akan dianalisis dengan rata-rata perbandingan dan menggunakan Wilcoxon Test atau uji-t. Hasil dari penelitian ini menunjukkan OkHttp unggul pada waktu respons yang disusul oleh Retrofit lalu Fast Android Network. Untuk penggunaan internet tidak perbedaan yang signifikan pada ketiga library namun OkHttp merupakan yang terbaik. Dari segi penggunaan memori dari Retrofit terhitung rendah dibandingkan ketiga library. Berdasarkan hasil penelitian yang sudah dilakukan, OkHttp menjadi pilihan terbaik untuk digunakan sebagai networking library dalam pengembangan aplikasi Android.

2.2 Android

Android adalah sebuah sistem operasi atau *software*yang dikembang oleh Google yang bersifat *opensource* berbasis pada Linux yang dapat berjalan pada perangkat seperti telepon seluler, *smartphone*, komputer dan juga tablet (Niesa & Nasution, 2022). Android dikembangkan langsung oleh Android Inc. yang mendapatkan dukungan penuh dari Google *Finance* namun dibeli pada tahun 2005. Setelah itu, pada 7 November 2007 Android secara resmi rilis bersamaan dengan Open Handset Allience (Setiawan, 2019). Dikarenakan Android bersifat *opensource*, banyak para pengembang android yang bermunculan sehingga banyak aplikasi android yang dapat memenuhi kebutuhan masyarakat. Hal ini merupakan salah satu faktor mengapa *smartphone* berbasis Android diterima oleh banyak masyarakat (Panji Rachmat Setiawan et al., 2020).

2.3 Application Programming Interface (API)

Application Programming Interface atau biasa dikenal dengan singkatan API, merupakan sebuah kumpulan prosedur, fungsi dan peraturan yang menentukan bagaimana tiap komponen atau perangkat lunak yang berbeda berkomunikasi dengan API. API dapat diumpamakan perantara antara dua orang yang sedang berinteraksi menggunakan dua bahasa yang berbeda (Heryandi, 2018). Manfaat dari menggunakan API adalah memungkinkan seorang pengembang untuk mengintegrasikan dua bagian suatu aplikasi atau aplikasi yang berbeda. Dengan menggunakan API, proses dari pengembangan lebih singkat karena

pengembang tidak perlu membuat fitur yang sama. API juga dapat diartikan sebagai sebuah *class* yang dirancang khusus untuk menghubungkan aplikasi yang dikembangkan dengan basis data yang digunakan (Riadi et al., 2019). API mempunyai aturan standar untuk melayani permintaan, oleh karena itu tidak perlu untuk mengubah *platform* yang sudah ada jika *platform* baru ditambahkan. Selain itu fitur keamanan pada API memungkin pengembang mengubah izin akses bagi pengembang lain (Hindriyanto Dwi Purnomo et al., 2016).

2.4 Kotlin

Kotlin merupakan bahasa pemrograman yang menjadi alternatif dari bahasa pemrograman Java. Kotlin hadir pada tahun 2011 yang bertujuan untuk menutupi banyak kekurangan yang dimiliki Java dalam pemeliharaan sebuah aplikasi. Pada tahun 2017, Kotlin menjadi bahasa pemrograman yang populer dalam pengembangan aplikasi Android dengan dukungan alat yang lengkap (Coppola et al., 2024). Kotlin yang merupakan bahasa pemrograman baru ini dapat menjalankan Java *Virtual Machine* (JVM). Tidak hanya sebagai alternatif dari Java, Kotlin dapat bekerja berdampingan dengan Java tanpa menimbulkan adanya masalah (Coppola et al., 2019). Berdasarkan riset yang dilakukan oleh Coppola et al. (2024), dari 1232 proyek yang diperbarui pada Oktober tahun 2017, sekitar 20% dari proyek tersebut sudah melakukan transisi ke bahasa pemrograman Kotlin dan 12% memiliki kode dengan bahasa Kotlin daripada bahasa Java. Berdasarkan hasil eksperimen dari Peters et al. (2021), dengan melakukan migrasi ke bahasa Kotlin memiliki pengaruh besar kepada penggunaan CPU, penggunaan memori dan durasi *rendering frame.*

2.5 Retrofit

Retrofit merupakan sebuah *library client* HTTP khusus untuk pengembangan Android dengan desain terdepan yang menyediakan teknologi terbaru. Penggunaan Retrofit memudahkan pengembang dalam melakukan mekanisme untuk mengirim *request*, mengatur respons, dan *date parsing* yang memungkinkan pengembang untuk membuat aplikasi yang kokoh (Kaura et al., 2024).

2.6 Ktor

Ktor merupakan *framework* fleksibel dan ringan yang dikembang oleh Jetbrains yang didesain untuk membuat *web application* secara asinkron dan ringkas dalam Kotlin. Dengan sifat asinkron dari Ktor memungkinkan penanganan permintaan konkuren secara efisien, membuatnya cocok untuk membuat aplikasi yang memiliki performa tinggi. Di dalamnya tersedia fitur *routing*, negosiasi konten, *authentication* untuk menyingkat proses pengembangan (Gregor Lang, 2024). Selain berfungsi sebagai *web application*, ktor mampu menjadi klien yang mampu berkomunikasi dengan internet. Ktor termasuk klien HTTP asinkron *multiplatform* yang memungkinkan pengembang untuk membuat *request*, menangani respons, menambahkan fungsionalitasnya dengan *plugins*, seperti autentikasi, JSON *Serialization* dan lainnya (Jetbrains, 2024).

2.7 Rest API

Representational State Transfer (REST) merupakan arsitektur yang digunakan dalam pengembangan layanan berbasis *web* untuk mendukung interoperabilitas dan aksesibilitas lintas platform dalam ekosistem *World Wide Web* (WWW). REST API memainkan peran penting dalam desain arsitektur *microservices*, yang sering disebut sebagai *web* API. Arsitektur ini terdiri dari berbagai *endpoint*, dengan setiap *endpoint* merepresentasikan implementasi konkret dari suatu fungsi dalam proses bisnis. Secara umum, REST API dapat diakses melalui protokol *Hypertext Transfer Protocol* (HTTP) dengan menerapkan metode standar seperti *GET*, *POST*, *PUT*, dan *DELETE*. Pemanggilan API dapat dilakukan melalui alamat yang dikenal sebagai *Uniform Resource Identifier* (URI), yang berfungsi sebagai identitas unik untuk setiap sumber daya yang tersedia dalam sistem (Ehsan et al., 2022).

2.9 Android Studio Profiler

Android Studio Profiler merupakan sebuah alat yang disediakan oleh Android Studio untuk memantau performa aplikasi yang sedang dikembangkan. Profiler di Android Studio dapat mendeteksi bagian aplikasi yang menggunakan *resource* secara tidak efisien, seperti CPU, memori, grafik, atau baterai perangkat. Jenis *profiling* yang disediakan oleh Android Studio adalah *record system trace, captute heap dump,*

sample callstack, record Java/Kotliin *allocations, , record* Java/Kotliin *methods,* dan *record natice allocations* (Google, 2023).

2.9.1 Record System Trace

Fitur *record system trace* akan menampilkan aktivitas dan penggunaan *resource* oleh sistem, sehingga pengguna dapat melihat proses yang berjalan pada saat aplikasi dijalankan. Fitur ini biasanya digunakan untuk bagaimana proses aplikasi dan sistem didistribusikan ke berbagai *core* dan *thread* perangkat, Seberapa lancar antarmuka pengguna ditampilkan, dan penggunaan baterai pada level aplikasi dan perangkat. Setelah melakukan perekaman jejak pada aplikasi, pengguna dapat melihat beberapa grafik yang ditampilkan berdasarkan *timeline.* Grafik yang ditampilkan setelah melakukan perekaman adalah CPU yang digunakan oleh aplikasi, interaksi pengguna dengan aplikasi, seberapa lancar antarmuka ditampilkan, *thread* sistem dan aplikasi dijalankan, aktivitas dari inti CPU yang digunakan oleh perangkan, total memori yang digunakan oleh aplikasi, dan penggunaan baterai.

2.10 Android Studio *Network Inspector*

Fitur *network inspector* pada android studio akan menampilkan aktivitas penggunaan internet secara *real-time*. Fitur ini juga memungkinkan pengguna untuk bagaimana dan kapan aplikasi melakukan transfer data. Fitur ini menyediakan beberapa informasi mengenai *request* yang dilakukan oleh aplikasi, mulai dari grafik penggunaan internet, waktu yang dibutuhkan untuk melakukan *request,* ukuran data yang diminta dan melihat *request* dan juga *response (Google, n.d.)*.

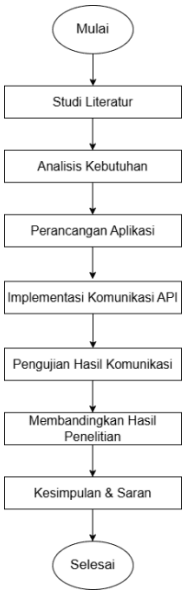
2.11 *Confidence Interval*

Confidence Interval merupakan sebuah estimasi dari rentang sebuah nilai yang sudah mencakup nilai populasi yang sesungguhnya untuk sebuah statistik, salah satu contohnya adalah rata-rata. *Confidence Interval* dapat dibuat untuk menunjukkan rentang nilai rata-rata dengan tingkat keyakinan tertentu yang sudah mencakup nilai rata-rata dari populasi sesungguhnya. Terdapat tiga variabel yang dapat mempengaruhi *Confidence Interval*, yaitu jumlah dari ukuran sampel yang digunakan, standar deviasi dari sampel yang digunakan, dan nilai *alpha* yang digunakan (Albert & Tullis Tom, 2022). Tingkat kepercayaan yang biasanya digunakan adalah 95% atau 90%, diikuti dengan nilai *alpha* yang digunakan, jika tingkat kepercayaan yang digunakan sebesar 95% maka nilai *alpha*-nya adalah 5%. Nilai *Confidence Interval* dengan tingkat kepercayaan 95% dapat dihitung dengan

2.12 Uji-T

Uji-t merupakan sebuah metode statistika untuk membanding rata-rata dari dua kelompok sebuah data. Metode ini merupakan metode yang paling sering digunakan untuk melakukan uji hipotesis. Tujuan utama dari uji-t adalah untuk menentukan apakah terdapat yang signifikan di antara rata-rata dari dua kelompok data yang dibandingkan. uji-t dibagi menjadi dua tipe, yang pertama adalah uji-t secara independen yang dapat digunakan pada saat dua kelompok data yang digunakan tidak saling berkaitan. Untuk tipe kedua adalah Uji-T *Paired* yang digunakan apabila dua kelompok data yang digunakan saling berkaitan (Tae Kyun Kim, 2015).

3. METODOLOGI



Gambar 1. Alur Penelitian

Gambar 1 merupakan gambar dari alur penelitian. Pada langkah pertama peneliti akan mencari jurnal yang relevan sebagai referensi yang akan mendukung penelitian dan memperkuat landasan teori penelitian yang sedang dilakukan. Selanjutnya adalah tahap Analisis Kebutuhan, tahap ini akan menentukan fitur apa saja yang akan digunakan pada aplikasi.

Fitur yang sudah ditentukan akan menjadi kasus uji untuk membandingkan *library* Retrofit dan Ktor. Fitur dari aplikasi akan ditentukan berdasarkan hasil wawancara yang akan dilakukan oleh peneliti kepada beberapa responden. Tahap berikutnya adalah Perancangan Aplikasi, pada tahap ini peneliti akan merancang desain antarmuka aplikasi yang akan menjadi media dilakukannya pengujian dan analisis perbandingan *librar* Retrofit dan Ktor. Tahap ini sangat diperlukan sebelum mengimplementasikan *library networking* kepada aplikasi. Setelah selesai merancang aplikasi saatnya lanjut ke tahap Implementasi Komunikasi API. Tahap ini adalah melibatkan peneliti untuk mengimplementasikan *library networking* kepada aplikasi yang sudah dirancang sebelumnya. Data yang didapatkan dari komunikasi dengan API akan disambungkan dengan antarmuka yang sudah dibuat sebelumnya dan data tersebut akan ditampilkan.

Langkah berikutnya adalah melakukan pengujian berdasarkan hasil komunikasi yang dilakukan. Pengujian ini akan dilakukan berdasarkan fitur-fitur dari aplikasi katalog film. Pada saat melakukan komunikasi dengan API, peneliti akan mencatat data waktu respons saat melakukan komunikasi, berapa persen penggunaan CPU yang dibutuhkan, dan berapa banyak memori yang dibutuhkan saat melakukan komunikasi. Setelah mendapatkan data yang dibutuhkan, penelitian ini akan lanjut ke tahap membanding hasil penelitian berdasarkan data yang sudah didapatkan. Data yang akan dibandingkan dan dilakukan analisis adalah waktu respons, penggunaan CPU, dan penggunaan memori. Jumlah sampel dari ketiga aspek tersebut adalah 30 untuk dan akan dianalisis dengan metode Uji-T.

4. PENGEMBANGAN APLIKASI

4.1 Analisis Kebutuhan

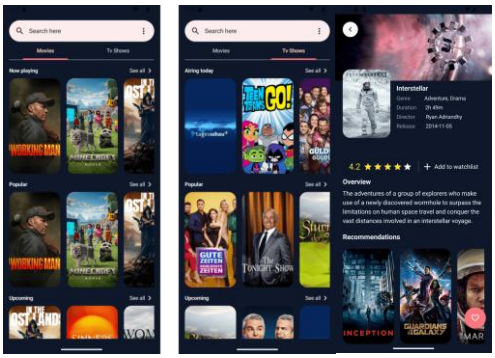
Bagian ini merupakan langkah awal untuk memulai pengembangan aplikasi katalog film. Peneliti akan melakukan wawancara kepada 12 responden. Jawaban dari para responden akan diolah menjadi fitur pada aplikasi sekaligus kasus uji untuk pengujian dua *library networking,* Retrofit dan Ktor. Fitur aplikasi dapat dilihat pada Tabel 1.

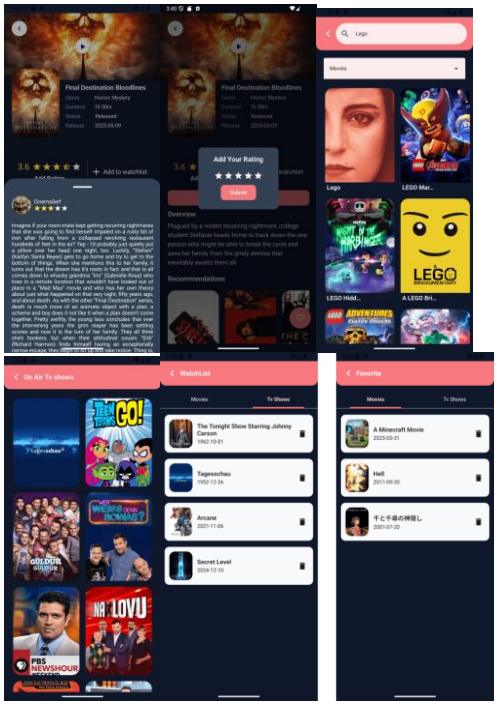
Tabel 1. Fitur Aplikasi

No.	Fitur
1	Melihat daftar film pada halaman utama (sedang tayang, sedang popular, yang akan datang, dengan peringkat tinggi).
2	Melihat daftar berdasarkan kategori (yang sedang tayang/popular/yang akan datang/dengan peringkat tinggi).
3	Melakukan pencarian film berdasarkan nama.
4	Melihat detail dari film yang dipilih.
5	Menambah dan menghapus film favorit.
6	Menambah dan menghapus film yang akan ditonton.
7	Melihat daftar film favorit.
8	Melihat daftar tontonan film.
9	Melihat ulasan dari film yang sedang dilihat
10	Memberikan penilaian kepada film yang sedang dilihat

4.2 Perancangan Aplikasi

Setelah mendapatkan daftar fitur berdasarkan hasil wawancara dengan para responden, peneliti dapat melanjutkan ke tahap berikutnya, yaitu membuat desain antarmuka dari kedua aplikasi. Berdasarkan daftar fitur yang sudah didapatkan sebelumnya, peneliti akan membuat halaman berdasarkan daftar fitur. Halaman yang terdapat pada aplikasi ini adalah, *home, detail, search, see all, favorite, dan watchlist*. Desain antarmuka aplikasi dapat dilihat pada Gambar 2.





Gambar 2. Desain Antarmuka Aplikasi

4.2 Implementasi API

Setelah membuat desain antarmuka dari kedua aplikasi adalah untuk lanjut ke tahap berikutnya, mengimplementasikan *library networking* pada aplikasi yang sudah dirancang. Untuk Retrofit akan diimplementasikan pada aplikasi pertama dengan nama Retrolog, Ktor akan diimplementasikan pada aplikasi kedua dengan nama KtorLog.

5. PENGUJIAN DAN ANALISIS

5.1 Prosedur Pengujian

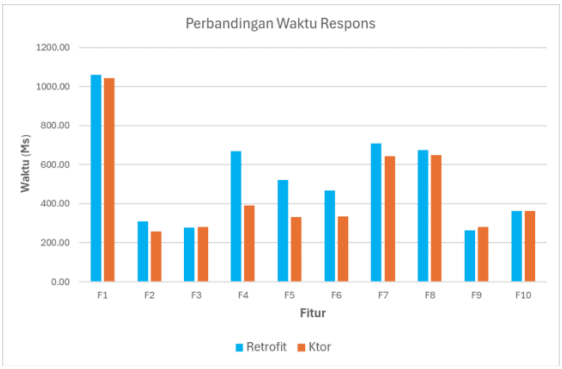
Bagian ini akan menjelaskan tata cara peneliti mendapatkan data pada saat aplikasi melakukan komunikasi dengan API menggunakan *library* yang sudah ditentukan. Aspek yang akan dibandingkan adalah waktu respons dalam satuan *milisecond* (ms), penggunaan CPU dalam satuan persentase penggunaan (%), dan penggunaan memori dengan banyaknya memori yang digunakan (MB). Pengujian akan dilakukan pada dua aplikasi yang sudah diintegrasikan dengan *library networking* yang akan dibandingkan, terdapat 10 uji kasus yang merupakan fitur dari aplikasi dan setiap uji kasus akan di uji sebanyak 30 kali. Untuk mendapatkan data waktu respons dapat menggunakan fitur yang disediakan oleh Android Studio, yaitu *network inspection*. Sedangkan untuk penggunaan CPU dan memori menggunakan fitur Android Studio Profiler dengan mode *capture system trace*.

5.2 Hasil Pengujian

Hasil pengujian waktu respons dapat dilihat pada Tabel 2 dan Gambar 3.

Tabel 2. Rata-rata Waktu Respons

No.	Retrofit	Ktor
1	1060,80	1044,00
2	309,67	258,33
3	277,10	280,73
4	669,27	391,73
5	520,87	331,63
6	466,63	335,93
7	709,67	643,93
8	674,20	647,90
9	263,00	281,53
10	362,60	363,97

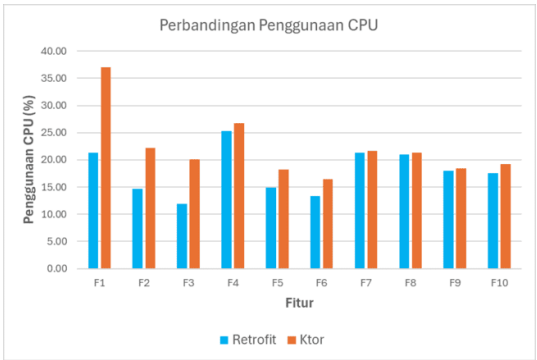


Gambar 3. Hasil Pengujian Waktu Respons

Hasil pengujian penggunaan CPU dapat dilihat pada Tabel 3 dan Gambar 4.

Tabel 3. Rata-rata Penggunaan CPU

No.	Retrofit	Ktor
1	21,32	37,07
2	14,71	22,24
3	11,86	20,13
4	25,31	26,76
5	14,86	18,26
6	13,38	16,48
7	21,32	21,60
8	20,98	21,33
9	18,02	18,48
10	17,59	19,23

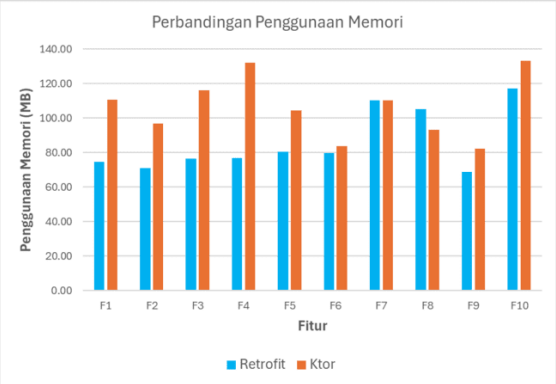


Gambar 4. Hasil Pengujian Penggunaan CPU

Hasil pengujian penggunaan memori dapat dilihat pada Tabel 4 dan Gambar 5.

Tabel 4. Rata-rata Penggunaan Memori

No.	Retrofit	Ktor
1	74,63	110,63
2	70,85	96,92
3	76,43	115,90
4	76,74	132,00
5	80,28	104,40
6	79,53	83,71
7	110,42	110,32
8	105,24	93,08
9	68,67	82,31
10	117,10	133,07



Gambar 5. Hasil Pengujian Pengguaaan Memori

5.3 Analisis Data

Hasil analisis data untuk waktu respons mendapatkan hasil Retrofit dengan rata-rata 531,38 ms dan Ktor dengan rata-rata 457,97 ms. Sehingga dapat dikatakan berdasarkan rata-rata Ktor lebih unggul dibandingkan dengan Retrofit. Namun berdasarkan hasil Uji-T, tidak terdapat perbedaan secara statistik karena nilai *p* yang didapatkan adalah 0,52. Hasil analisis data pada aspek penggunaan CPU, Retrofit mendapatkan rata-rata 17,94% dan Ktor mendapatkan rata-rata 22,16%. Berdasarkan hasil rata-rata yang didapatkan Retrofit lebih unggul dibandingkan Ktor. Tetapi, berdasarkan hasil dari Uji-T nilai *p* yang didapatkan adalah 0.08. Sehingga berdasarkan hasil Uji-T, tidak terdapat perbedaan secara statistik. Pada aspek penggunaan memori, Retrofit mendapatkan rata-rata 85,99 MB dan Ktor mendapatkan 106,32 MB. Berdasarkan nilai rata-rata, Retrofit lebih unggul dibandingkan dengan Ktor. Berdasarkan hasil dari Uji-T, nilai *p* yang didapatkan adalah 0.02. Sehingga dikatakan untuk aspek penggunaan memori terdapat perbedaan secara statistik.

6. KESIMPULAN DAN SARAN

6.1 Kesimpulan

Pada aspek waktu respons hasil pengujian dengan uji-t memperoleh nilai *p* yang lebih besar dibandingkan nilai *alpha* (nilai $p = 0{,}52$). Oleh karena itu, dapat disimpulkan bahwa tidak terdapat perbedaan secara statistik pada aspek waktu respons antara kedua *library*. Pada aspek penggunaan CPU, proses analisis dengan uji-t memperoleh nilai *p* yang lebih besar dibandingkan nilai *alpha* (nilai $p = 0.84$). Sehingga tidak terdapat perbedaan secara statistik untuk aspek penggunaan CPU. Pada aspek penggunaan memori nilai *p* yang diperolah dari analisis data menggunakan uji-t lebih kecil daripada nilai *alpha* (nilai $p = 0.02$). Sehingga dapat disimpulkan bahwa terdapat perbedaan secara statistik pada aspek penggunaan memori antara kedua *library*.

6.2 Saran

Pada penelitian ini, peneliti menggunakan *Public* API oleh TMDB API sebagai sumber data yang digunakan. Oleh karena itu untuk penelitian selanjutnya dapat mengembangkan API sendiri agar sesuai dengan kebutuhan. Pada penelitian ini menggunakan versi *library* yang disarankan oleh Google, untuk penelitian berikutnya dapat menggunakan versi terbaru *library*.

DAFTAR PUSTAKA

Choirudin, R., & Adil, A. (2019). Implementasi Rest Api Web Service dalam Membangun Aplikasi Multiplatform untuk Usaha Jasa. *MATRIK : Jurnal Manajemen, Teknik Informatika Dan Rekayasa Komputer*, *18*(2), 284–293. https://doi.org/10.30812/matrik.v18i2.407

Christhover, R., Sugiyatno, & Herlawati. (2022). PENERAPAN REST API MENGGUNAKAN RETROFIT UNTUK SISTEM INFORMASI FILM BERBASIS ANDROID (STUDI KASUS: SINOPSIS FILM). *Journal of Students' Research in Computer Science*, *3*(2), 159–170. https://doi.org/10.31599/jsrcs.v3i2.1393

Coppola, R., Ardito, L., & Torchiano, M. (2019). Characterizing the transition to kotlin of android apps: A study on F-Droid, Play Store, and GitHub. *WAMA 2019 - Proceedings of the 3rd ACM SIGSOFT International Workshop on App Market Analytics, Co-Located with ESEC/FSE 2019*, 8–14. https://doi.org/10.1145/3340496.3342759

Coppola, R., Fulcini, T., & Torchiano, M. (2024). Poster: Kotlin Assimilating the Android Ecosystem - An Appraisal of Diffusion and Impact on Maintainability. *Proceedings - International Conference on Software Engineering*, 266–267. https://doi.org/10.1145/3639478.3643071

Ehsan, A., Abuhaliqa, M. A. M. E., Catal, C., & Mishra, D. (2022). RESTful API Testing Methodologies: Rationale, Challenges, and Solution Directions. In *Applied Sciences (Switzerland)* (Vol. 12, Issue 9). MDPI. https://doi.org/10.3390/app12094369

Ferryansyah, M. S., Tri Ananta, M., & Fanani, L. (2018). *Analisis Performansi HTTP Networking Library pada Android (Studi Kasus: Portal Berita)* (Vol. 2, Issue 5). http://j-ptiik.ub.ac.id

Google. (n.d.). *Inspect network traffic with the Network Inspector*. Retrieved May 23, 2025, from https://developer.android.com/studio/debug/network-profiler

Google. (2023). *Profile your app performance*. https://developer.android.com/studio/profile#start-profiling

Gregor Lang. (2024). Ktor Reactive Server Applications. In *Institut für Systemsoftware*. JOHANNES KEPLER UNIVERSITY LINZ.

Heryandi, A. (2018). Developing Application ProgramMing Interface (API) for Student Academic Activity Monitoring using Firebase Cloud Messaging (FCM). *IOP Conference Series: Materials Science and Engineering*, *407*(1). https://doi.org/10.1088/1757-899X/407/1/012149

Hindriyanto Dwi Purnomo, Dody Agung Saputro, Ramos Somya, & Charitas Fibriani. (2016). The Application of Restful Web Service and Json for Poultry Farm Monitoring System. *JEECS (Journal of Electrical Engineering and Computer Sciences)*, *1*(1), 25–30. https://doi.org/10.54732/jeecs.v1i1.183

Iqbal, T., & Wali, M. (2022). IDOL: Retrofit-Kotlin Service-Based Online Digital Library Application and College Open Data Repository. *International Journal Software Engineering and Computer Science (IJSECS)*, *2*(1), 1–8. https://doi.org/10.35870/ijsecs.v2i1.760

Jetbrains. (2024). *Create Client Application*. Https://Ktor.Io/Docs/Client-Create-New-Application.Html. https://ktor.io/docs/client-create-new-application.html

Kaura, S., Arora, J., & School of Business Jhanjeri, C. (2024). Redesigning Android Development: Using Reactive Programming to Retrofit REST APIs and Concurrency. *International Journal of Scientific Research in Engineering and Management*. https://doi.org/10.55041/IJSREM30623

Lang, G., & Prähofer, H. (2024). *Ktor Reactive Server Applications* [JOHANNES KEPLER UNIVERSITY LINZ]. https://ssw.jku.at/Teaching/BachelorTheses/2024/Lang_Gregor.pdf

Mohammed Mudassir, & Mohammed Mushtaq. (2024). The Role of APIs in Modern Software Development. *World Journal of Advanced Engineering Technology and Sciences*, *13*(1), 1045–1047. https://doi.org/10.30574/wjaets.2024.1

3.1.0515

Niesa, C., & Nasution, N. R. (2022). Perbandingan Sistem Operasi Android Kitkat dengan Android Lollipop. *Jurnal Nasional Informatika Dan Teknologi Jaringan*, *7*(1). http://www.tabloidaplikasiandroid.com/2014/11/android-

Panji Rachmat Setiawan, Muhammad Syaifullah, & Pandu Pratama Putra. (2020). Sistem Pemesanan Menu Pada Restoran Berbasis Android. *IT Journal Research and Development*, *5*(2), 193–203. https://doi.org/10.25299/itjrd.2021.vol5(2).5866

Peters, M., Scoccia, G. L., & Malavolta, I. (2021). How does Migrating to Kotlin Impact the Run-Time Efficiency of Android Apps? *Proceedings - IEEE 21st International Working Conference on Source Code Analysis and Manipulation, SCAM 2021*, 36–46. https://doi.org/10.1109/SCAM52516.2021.00014

Riadi, I., Ananda Raharja, P., & Dahlan Yogyakarta, A. (2019). ANALISIS APPLICATION PROGRAMMING INTERFACE PADA MOBILE E-VOTING MENGGUNAKAN METODE TEST-DRIVEN DEVELOPMENT. *Hal*, *20*(2). http://jurnalnasional.ump.ac.id/index.php/Techno

Santoso, D. A., Pramono, D., & Pinandito, A. (2025). *Analisis Performa Networking Library Retrofit, Ktor Client, Okhttp Dan Volley Pada Pengembangan Aplikasi Perangkat Bergerak Berbasis Android* (Vol. 9, Issue 5). http://j-ptiik.ub.ac.id

Saputra, K., Farhan, K., & Irvanizam, I. (2018). Analysis on the Comparison of Retrofit and Volley Libraries on Android-Based Mosque Application. *2018 International Conference on Electrical Engineering and Informatics (ICELTICs)*, 117–121. https://doi.org/10.1109/ICELTICS.2018.8548881

Setiawan, N. (2019). *KASUS KEJAHATAN SIBER PADA TELEPON SELULER ANDROID* (Vol. 2, Issue 1).

Sufyan, F., & Banerjee, A. (2019). Comparative Analysis of Network Libraries for Offloading Efficiency in Mobile Cloud Environment. In *IJACSA) International Journal of Advanced Computer Science and Applications* (Vol. 10, Issue 2). www.ijacsa.thesai.org

Vernanda, B., Prayoga, A., Pinandito, A., & Kharisma, A. P. (2023). *Studi Performa Android Networking Library antara Fast Android Network Library, Retrofit dan OkHttp* (Vol. 7, Issue 6). http://j-ptiik.ub.ac.id

Widyaningtyas, S., & Wahyono, T. (2024). IMPLEMENTASI REST API MENGGUNAKAN RETROFIT PADA APLIKASI MONITORING GROOMING BERBASIS ANDROID. *IT-Explore: Jurnal Penerapan Teknologi Informasi Dan Komunikasi*, *3*(2), 147–161. https://doi.org/https://doi.org/10.24246/itexplore.v3i2.2024.pp147-161