REPOSITOR, Vol. 7, No. 3, Agustus 2025, Pp. 325-330
ISSN : 2714-7975
E-ISSN : 2716-1382

Analisis Perbandingan Implementasi Arsitektur Rest dengan GRAPHQL (Studi Kasus Perusahaan Hospitality Service di Bali)

Robin Nusantara Haming*[1], Wildan Suharso[1], Didih Rizki Chandranegara[1]
Universitas Muhammadiyah Malang
robinnusantara@webmail.umm.ac.id *[1], wsuharso@umm.ac.id[1], didihrizki@umm.ac.id[1]

Abstrak

Pengembangan sistem informasi memerlukan metode distribusi data yang efisien untuk mendukung pembuatan keputusan. Pada umumnya, sistem informasi terdiri dari frontend yang menampilkan informasi dan backend yang memproses data, dengan REST API sebagai standar arsitektur untuk menghubungkan keduanya. Namun, kemajuan teknologi telah memperkenalkan GraphQL, yang menawarkan fleksibilitas dan efisiensi lebih dalam pendistribusian data. REST API memiliki batasan seperti kompleksitas sistem, tuntutan kualitas layanan tinggi, dan pengambilan data dinamis yang membatasi fleksibilitasnya. Sebaliknya, GraphQL, yang dirilis pada 2015, memungkinkan query data yang lebih fleksibel dan responsif, serta efisiensi dalam pengembangan API. Membandingkan dua arsitektur API, REST dan GraphQL, untuk menentukan mana yang lebih efisien dalam hal kecepatan waktu respon, ukuran respon, kompleksitas bahasa query, dan skalabilitas. Studi ini melibatkan analisis terhadap aplikasi perusahaan hospitality di Bali yang menggunakan kedua arsitektur. Evaluasi dilakukan dengan mengukur waktu respon, throughput, beban CPU, dan penggunaan memori. Hasil menunjukkan REST lebih cepat dalam waktu respon dan throughput, sedangkan GraphQL lebih efisien dalam penggunaan sumber daya. Penelitian ini bertujuan membantu pengembang memilih arsitektur API yang optimal berdasarkan efisiensi kegunaan, sumber daya, waktu, biaya, dan skalabilitas. Diharapkan temuan ini dapat menjadi panduan untuk pengembang dalam memilih arsitektur yang sesuai untuk pengembangan sistem informasi di masa depan.

Kata Kunci: *Sistem Informasi, Teknologi, Arsitektur Rest, GraphQL*

Abstract

The development of information systems requires efficient data distribution methods to support decision-making. Generally, information systems consist of a frontend that displays information and a backend that processes data, with REST APIs serving as the architectural standard to connect the two. However, advancements in technology have introduced GraphQL, which offers greater flexibility and efficiency in data distribution. REST APIs have limitations such as system complexity, high service quality demands, and dynamic data retrieval that restrict their flexibility. On the other hand, GraphQL, released in 2015, allows for more flexible and responsive data queries, as well as efficiency in API development. This study compares the two API architectures, REST and GraphQL, to determine which is more efficient in terms of response time, response size, query language complexity, and scalability. The study involves analyzing a hospitality application in Bali that uses both architectures. Evaluation is carried out by measuring response time, throughput, CPU load, and memory usage. Results show that REST is faster in response time and throughput, while GraphQL is more efficient in resource usage. This research aims to help developers choose the optimal API architecture based on usability efficiency, resource usage, time, cost, and scalability. It is hoped that these findings will serve as a guide for developers in selecting the appropriate architecture for future information system development.

Keywords: *Information System, Technology, Rest Architecture, GraphQL*

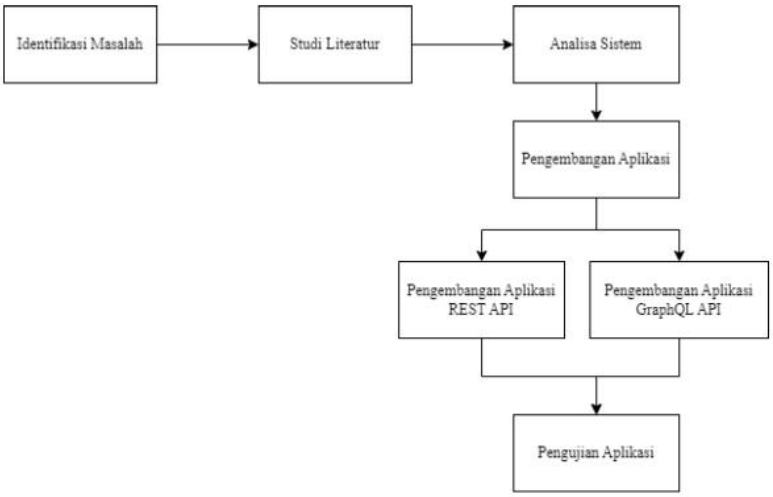
1. Pendahuluan

Pengembangan sistem informasi memerlukan metode distribusi data yang efisien untuk mendukung pembuatan keputusan. Pada umumnya, sistem informasi terdiri dari frontend yang menampilkan informasi dan backend yang memproses data, dengan REST API sebagai standar arsitektur untuk menghubungkan keduanya. Namun, kemajuan teknologi telah memperkenalkan GraphQL, yang menawarkan fleksibilitas dan efisiensi lebih dalam pendistribusian data. REST

API memiliki batasan seperti kompleksitas sistem, tuntutan kualitas layanan tinggi, dan pengambilan data dinamis yang membatasi fleksibilitasnya. Sebaliknya, GraphQL, yang dirilis pada 2015, memungkinkan query data yang lebih fleksibel dan responsif, serta efisiensi dalam pengembangan API. Membandingkan dua arsitektur API, REST dan GraphQL, untuk menentukan mana yang lebih efisien dalam hal kecepatan waktu respon, ukuran respon, kompleksitas bahasa query, dan skalabilitas. Studi ini melibatkan analisis terhadap aplikasi perusahaan hospitality di Bali yang menggunakan kedua arsitektur. Evaluasi dilakukan dengan mengukur waktu respon, throughput, beban CPU, dan penggunaan memori. Hasil menunjukkan REST lebih cepat dalam waktu respon dan throughput, sedangkan GraphQL lebih efisien dalam penggunaan sumber daya. Penelitian ini bertujuan membantu pengembang memilih arsitektur API yang optimal berdasarkan efisiensi kegunaan, sumber daya, waktu, biaya, dan skalabilitas. Diharapkan temuan ini dapat menjadi panduan untuk pengembang dalam memilih arsitektur yang sesuai untuk pengembangan sistem informasi di masa depan.

2. Metode Penelitian

Metode penelitian menjelaskan kronologi penelitian, termasuk desain penelitian, prosedur penelitian (termasuk algoritma, *pseudocode* atau hal lain yang terkait), bagaimana menguji dan proses akuisisi data [3], [4]. Setiap deskripsi terkait metode penelitian sebaiknya ditunjang dengan referensi.



Gambar 1. Alur Penelitian

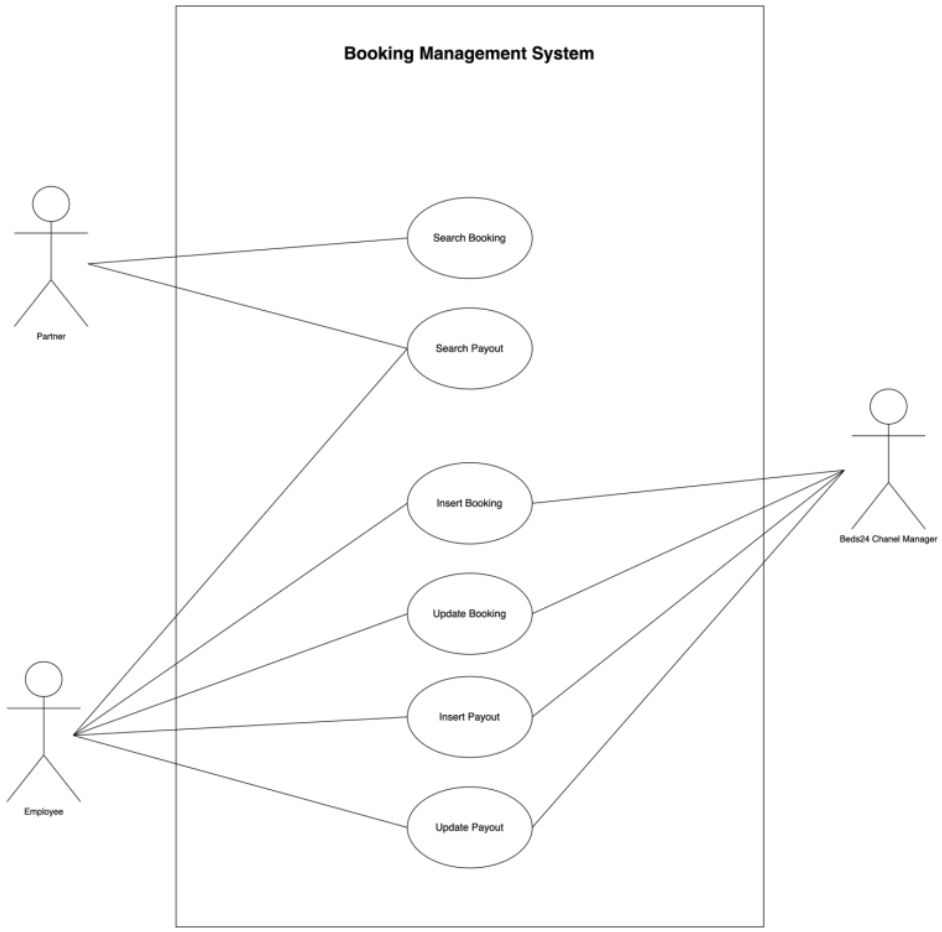
Proses penelitian ini diawali dengan identifikasi masalah, di mana peneliti menemukan permasalahan yang akan diatasi melalui pengembangan sistem atau aplikasi. Setelah masalah teridentifikasi, langkah berikutnya adalah melakukan studi literatur. Pada tahap ini, peneliti mencari informasi dan solusi yang sudah ada terkait permasalahan yang dihadapi untuk mendapatkan pemahaman yang lebih mendalam dan menentukan pendekatan yang paling sesuai. Setelah memperoleh landasan teori yang cukup, peneliti melanjutkan ke tahap analisa sistem, di mana dilakukan analisis menyeluruh terhadap kebutuhan pengguna serta bagaimana sistem yang akan dikembangkan mampu memenuhi kebutuhan tersebut. Tahap ini sangat penting untuk memastikan bahwa desain sistem yang direncanakan sesuai dengan ekspektasi.

Berdasarkan hasil analisa, peneliti kemudian melakukan pengembangan aplikasi. Pada proses pengembangan ini, terdapat dua pendekatan API yang digunakan, yaitu REST API dan GraphQL API. Penggunaan REST API memungkinkan komunikasi antara aplikasi secara efisien, sementara GraphQL API digunakan untuk pengambilan data yang lebih optimal dari server, bergantung pada kebutuhan spesifik yang dimiliki aplikasi. Tahap akhir dari alur penelitian ini adalah pengujian aplikasi. Pada tahap ini, dilakukan pengujian menyeluruh untuk memastikan bahwa aplikasi yang telah dikembangkan berjalan dengan baik dan mampu menyelesaikan masalah yang telah diidentifikasi pada awal penelitian. Hasil pengujian ini akan menentukan apakah aplikasi sudah siap digunakan atau memerlukan perbaikan lebih lanjut.

3. Hasil Penelitian dan Pembahasan

3.1 Pengembangan Aplikasi

Sistem Manajemen Booking memiliki dua aktor aktif dan satu sistem yang bekerja sama. Karyawan memiliki akses langsung ke seluruh sistem, menangani tugas-tugas seperti memasukkan, memperbarui, dan mengelola pemesanan dan pembayaran. Mitra hanya dapat mencari pemesanan dan pembayaran. Sinkronisasi dengan Beds24 Channel Manager terjadi secara otomatis. Gambar 2 berikut adalah diagram use case dalam pengembangan aplikasi ini.



Gambar 2 Usecase Diagram

Diagram usecase yang ditampilkan menggambarkan interaksi antara tiga aktor utama dalam Sistem Manajemen Pemesanan: Karyawan, Mitra, dan Beds24 Channel Manager. Karyawan memiliki akses penuh untuk melakukan berbagai tugas terkait pemesanan dan pembayaran, sementara Mitra hanya memiliki akses terbatas untuk pencarian data. Beds24 Channel Manager berperan dalam sinkronisasi otomatis data pemesanan dan pembayaran, memastikan bahwa semua informasi selalu terupdate. Diagram ini penting untuk memahami alur kerja dan peran setiap aktor, serta menunjukkan pentingnya sinkronisasi otomatis untuk menjaga efisiensi dan keandalan sistem.

3.2 Desain dan Implementasi REST API

REST API digunakan untuk mendukung komunikasi antara klien (Karyawan dan Mitra) dengan server (Sistem Manajemen Booking). Desain API ini dirancang untuk memastikan bahwa setiap fungsi utama dapat diakses secara mudah dan efisien melalui HTTP methods seperti GET, POST, dan PUT. Berikut adalah beberapa endpoint utama beserta penjelasannya:

3.2.1 GET /bookings

Endpoint ini digunakan oleh karyawan untuk mengambil data pemesanan yang ada di sistem. Dengan menggunakan metode GET, karyawan dapat meminta informasi lengkap tentang pemesanan tertentu. Dalam responsnya, sistem akan mengembalikan berbagai informasi terkait pemesanan, seperti ID pemesanan, detail tamu (ID, nama, email, nomor telepon, dan asal), tanggal check-in dan check-out, status pemesanan, nilai pemesanan, serta detail unit dan properti yang dipesan.

3.2.2 POST /bookings

Endpoint ini digunakan oleh karyawan untuk membuat data pemesanan baru dalam sistem. Dengan menggunakan metode POST, karyawan dapat memasukkan detail pemesanan yang diperlukan melalui request body dalam format JSON.

3.2.3 PUT /bookings/{id}

Endpoint ini digunakan oleh karyawan untuk memperbarui data pemesanan berdasarkan ID pemesanan. Pembaruan dapat mencakup perubahan informasi tamu, tanggal check-in dan check-out, jumlah tamu, serta detail lainnya yang perlu diperbarui.

3.2.4 GET /payouts

Endpoint ini digunakan untuk mengambil data pembayaran. Karyawan dan Mitra dapat menggunakan endpoint ini untuk melihat detail pembayaran. Seperti pada endpoint pemesanan, Karyawan memiliki hak akses penuh, sedangkan Mitra hanya dapat melihat data pembayaran yang relevan.

3.2.5 POST /payouts

Endpoint ini memungkinkan Karyawan untuk memasukkan data pembayaran baru. Informasi yang dimasukkan meliputi jumlah pembayaran, tanggal pembayaran, dan detail penerima pembayaran.

3.2.6 PUT /payouts/{id}

Endpoint ini digunakan oleh Karyawan untuk memperbarui data pembayaran berdasarkan ID pembayaran. Pembaruan ini mencakup perubahan jumlah, tanggal, atau detail lain yang perlu diperbarui. Pembaruan ini juga akan memicu sinkronisasi otomatis untuk menjaga konsistensi data.

3.3 Desain dan Implementasi GraphQL

GraphQL digunakan untuk mendukung komunikasi antara klien (Karyawan dan Mitra) dengan server (Sistem Manajemen Pemesanan). Skema GraphQL dirancang untuk memastikan bahwa setiap fungsi utama dapat diakses dengan permintaan yang fleksibel, memungkinkan klien untuk menentukan data yang mereka butuhkan secara spesifik. Beberapa fitur utama dalam implementasi GraphQL adalah sebagai berikut:

3.3.1 Query getBookings

Mengambil data pemesanan. Fitur ini memungkinkan Karyawan dan Mitra untuk meminta hanya data yang mereka perlukan, seperti informasi tamu, detail properti, atau status pemesanan. Fleksibilitas ini mengurangi beban jaringan dan meningkatkan efisiensi pengambilan data.

3.3.2 Mutation createBooking

Memasukkan data pemesanan baru. Fitur ini digunakan oleh Karyawan untuk menambah pemesanan baru dengan memberikan semua detail yang diperlukan dalam satu permintaan. Hal ini mempermudah proses input data dan memastikan bahwa semua informasi yang relevan disertakan dalam satu operasi.

3.3.3 Mutation updateBooking

Memperbarui data pemesanan berdasarkan ID pemesanan. Fitur ini memungkinkan Karyawan untuk memperbarui informasi tertentu dalam pemesanan yang ada. Pembaruan ini mencakup perubahan informasi tamu, perubahan tanggal, atau detail lain yang perlu diperbarui.

3.3.4 Query getPayouts

Mengambil data pembayaran. Fitur ini memungkinkan Karyawan dan Mitra untuk mengakses detail pembayaran yang relevan, seperti jumlah pembayaran, tanggal pembayaran, dan penerima pembayaran.

3.3.5 Mutation createPayout

Memasukkan data pembayaran baru. Fitur ini memungkinkan karyawan untuk menambah pembayaran baru dengan detail yang diperlukan, seperti jumlah pembayaran, metode pembayaran, tanggal pembayaran, dan identitas pihak yang melakukan pembayaran. Ini memastikan semua informasi penting tercatat dalam satu operasi input data, membantu dalam pemantauan dan pelacakan pembayaran secara efisien.

3.3.6 Mutation updatePayout

Memperbarui data pembayaran berdasarkan ID pembayaran. Fitur ini memungkinkan Karyawan untuk memperbarui informasi tertentu dalam pembayaran yang ada, seperti jumlah pembayaran atau tanggal pembayaran. Setiap pembaruan data akan dicatat dalam log sistem untuk keperluan audit dan pemantauan.

4. Kesimpulan

Berdasarkan pengujian performa fitur booking dan payout pada arsitektur GraphQL dan REST, disimpulkan bahwa GraphQL memiliki waktu respons rata-rata yang sedikit lebih tinggi, meskipun ukuran responsnya lebih kecil dibandingkan dengan REST. GraphQL juga menunjukkan variasi waktu respons yang lebih besar, menunjukkan kinerja yang lebih bervariasi. Dalam hal skalabilitas, GraphQL menawarkan fleksibilitas dalam menangani permintaan kompleks dan mengurangi jumlah permintaan, mendukung skalabilitas yang lebih besar. Namun, ini dapat meningkatkan beban server dengan meningkatnya jumlah klien. REST memberikan kinerja yang lebih stabil dan efisien dalam menangani banyak permintaan, meskipun kurang efisien untuk permintaan data yang sangat spesifik. Pemilihan antara GraphQL dan REST harus mempertimbangkan kebutuhan aplikasi, kompleksitas permintaan, dan kapasitas server.

Referensi

[1] A. Lawi, B. L. E. Panggabean, and T. Yoshida, “Evaluating graphql and rest api services performance in a massive and intensive accessible information system,” Computers, vol. 10, no. 11, 2021, doi: 10.3390/computers10110138.

[2] S. K. Mukhiya, F. Rabbiab, V. K. I. Punax, A. Rutle, and Y. Lamo, “A graphql approach to healthcare information exchange with hl7 fhir,” Procedia Comput. Sci., vol. 160, pp. 338–345, 2019, doi: 10.1016/j.procs.2019.11.082.

[3] I. G. S. Masdiyasa, G. S. Budiwitjaksono, H. A. M, I. A. W. Sampurno, and N. M. I. M. Mandenni, “Graph-QL Responsibility Analysis at Integrated Competency Certification Test System Base on Web Service,” Lontar Komput. J. Ilm. Teknol. Inf., vol. 11, no. 2, p. 114, 2020, doi: 10.24843/lkjiti.2020.v11.i02.p05.

[4] R. Houghton, N. Balfe, and J. R. Wilson, “Systems Analysis and Design,” Eval. Hum. Work Fourth Ed., vol. 11, no. 4, pp. 221–248, 2015, doi: 10.1201/b18362-20.

[5] J. Cutler and M. Dickenson, Introduction to Machine Learning with Python. In: Computational Frameworks for Political and Social Research with Python. 2020.

[6] F. Halili and E. Ramadani, “Web Services: A Comparison of Soap and Rest Services,” Mod. Appl. Sci., vol. 12, no. 3, p. 175, 2018, doi: 10.5539/mas.v12n3p175.

[7] C. R. Necco, C. L. Gordon, and N. W. Tsai, “Systems Analysis and Design: Current Practices,” MIS Q., vol. 11, no. 4, p. 461, Dec. 1987, doi: 10.2307/248975.

[8] A. K. Chandrasekhar and D. A. S. Chandran, “Comparative Analysis of Load Testing Tools,” vol. 9, no. 6, 2021.

[9] A. Sharma, R. Kumar, and V. Mansotra, “Application of TypeScript Language: A Brief Overview,” Int. J. Innov. Res. Comput. Commun. Eng. ISO Certif. Organ., vol. 3297, no. 6, pp. 11449–11455, 2016, doi: 10.15680/IJIRCCE.2016.

[10] H. Shah and T. R. Soomro, “Node. Js Challenges in Implementation,” Glob. J. Comput. Sci. Technol. E Netw. Web Secur., vol. 17, no. 2, pp. 73–83, 2017.

[11] A. Mardan, “Starting with Express.js,” Expressjs, pp. 3–14, 2014, doi: 10.1007/978-1-4842-0037-7_1.

[12] A. K. Chandrasekhar and Dr. A. S. ; Chandran, “Comparative Analysis of Load Testing Tools Sahi And Selenium,” Int. J. Creat. Res. Thoughts IJCRT, vol. 5, no. 7, pp. 55–60, 2016.
[13] Isha, A. S. (2019). Automated API Testing. International Journal of Engineering Science and Computing.
[14] Kumari, B., Chauhan, N., & Vedpal. (2018). A Comparison Between Manual Testing and Automated Testing. Journal of Emerging Technologies and Innovative Research.
[15] Neumann, A., Laranjeiro, N., & Bernardino, J. (2021). An Analysis of Public REST Web Service APIs. IEEE Transactions on Services Computing.
[16] Brito, G., Mombach, T., & Valente, M. T. (2019). Migrating to GraphQL: A Practical Assessment. SANER 2019 - Proceedings of the 2019 IEEE 26th International Conference on Software Analysis, Evolution, and Reengineering, 140–150. https://doi.org/10.1109/SANER.2019.8667986.
[17] Eizinger, T. (2017). API Design in Distributed Systems: A Comparison between GraphQL and REST. 1–64. https://eizinger.io/assets/Master-Thesis.pdf
[18] F. Halili and E. Ramadani, “Web Services: A Comparison of Soap and Rest Services,” Mod. Appl.Sci., vol. 12, no. 3, p. 175, 2018, doi: 10.5539/mas.v12n3p175.
[19] J. Kopecký, P. Fremantle, and R. Boakes, “A history and future of Web APIs,” It - Inf. Technol., vol. 56, 2014.
[20] B. Costa, P. F. Pires, F. C. Delicato, and P. Merson, “Evaluating a Representational State Transfer (REST) Architecture: What is the Impact of REST in My Architecture?,” in 2014 IEEE/IFIP Conference on Software Architecture, 2014, pp. 105–114.