ANALISA PERBANDINGAN METODE GRAPHQL API DAN REST API DENGAN MENGGUNAKAN ASP.NET CORE WEB API FRAMEWORK

Fahri Hanif[1)], Imam Ahmad[2)], Dedi Darwis[3)], Ichtiar Lazuardi Putra[4)], Muhammad Fauzan Ramadhani[5)]

[1,4,5]Teknologi Informasi, Universitas Teknokrat Indonesia
[2]Sistem Informasi, Universitas Teknokrat Indonesia
[3]Sistem Informasi Akuntansi, Universitas Teknokrat Indonesia
[1,2,3,4,5]Jl. ZA. Pagar Alam No.9 -11, Labuhan Ratu, Kec. Kedaton, Kota Bandar Lampung
Email: [1]fahri_hanif.mhs@teknokrat.ac.id, [2]imam.ahmad@teknokrat.ac.id, [3]dedi.darwis@teknokrat.ac.id, [4]ichtiar_lazuardi_putra.mhs@teknokrat.ac.id, [5]m_fauzan_ramadhani.mhs@teknokrat.ac.id

Abstrak

REST (Representational State Transfer) API menjadi sebuah arsitektur yang standar untuk membangun sebuah web service. API (Application Programming Interface) digunakan untuk dapat berinteraksi dengan software lain. Permasalahan menggunakan REST API yaitu over-fetching dan under-fetching yang menyebabkan kinerja dari REST API menurun dikarenakan data yang diminta teralu lama untuk di kembalikan ke client. GraphQL API menjadi salah satu alternatif dalam membangun sebuah API karena dapat mengatasi permasalahan yang terdapat di REST API dengan teknologi ASP.NET Core Web API Framework.Oleh sebab itu, penulis melakukan perbandingan dari kinerja metode GraphQL API dan REST API dengan menggunakan ASP.NET Core Framework agar dapat menentukan mana yang lebih baik dari segi response time, latency, dan processing time. Hasil dari komparasi nilai kinerja kedua API menunjukan GraphQL API lebih unggul di HTTP Request menggunakan method GET dengan response time rata-rata 58,9ms sedangkan REST API mendapatkan response time rata-rata 4167,13ms. Namun untuk HTTP Request dengan methods PUT, POST, DELETE hanya selisih sedikit namun GraphQL API dengan menggunakan ASP.NET Core masih unggul.Kesimpulan dari penelitian yang telah dilakukan berdasarkan semua pengujian setiap arsitektur API tersebut dapat menunjukkan bahwa GraphQL API lebih unggul dibandingkan REST API dengan menggunakan ASP.NET Core dalam segi performa khususnya saat menggunakan method GET.

Kata Kunci: : *Kinerja API (Application Programming Interface), REST API, GraphQL API, ASP.NET Core Framework.*

1. Pendahuluan

API (Application Programming Interface) merupakan sebuah antarmuka yang diimplementasikan dengan menggunakan software sehingga dapat berinteraksi dengan software lain [1]–[5]. REST (Representational State Transfer) API merupakan sebuah metode yang menjadi standar untuk membangun sebuah web service. REST API mudah diimplementasikan karena tidak terikat dengan protocol transfer apa pun, kemudian memiliki desain utama yaitu addressability, uniform interface, dan statelessness [4].

Permasalahan yang ada di REST API membuat permintaan data menjadi sangat lambat apalagi di saat membutuhkan data yang ingin ditampilkan dengan beberapa tabel karena over-fetching dan under-fetching yang menjadi permasalahan utama pada REST API. Namun sebenarnya REST API bukanlah satu-satunya arsitektur API yang ada banyak pendahulu hingga pada tahun 2015 muncullah arsitektur API yang dikenalkan oleh Facebook yaitu GraphQL API [6]. GraphQL mampu memecahkan keterbatasan yang ada di REST API yaitu over-fetching dan under-fetching karena arsitektur GraphQL API memerlukan query untuk proses permintaan data, oleh karena itu client dapat menentukan data apa saja yang diperlukan.

Hingga saat ini penelitian terkait perbandingan antara REST API dan GraphQL API sudah beberapa dilakukan, namun teknologi yang dikembangkan rata-rata menggunakan NodeJS, Golang, ataupun Java Spring. Pada penelitian sebelumnya rata-rata REST API masih mengungguli dari segi performa, namun untuk fleksibilitas permintaan data GraphQL API bisa menjadi alternatif saat ini. Pada penelitian yang ingin dilakukan penulis, penulis mencoba untuk menggunakan teknologi lain yaitu ASP.NET Core yang dikembangkan oleh Microsoft. Merujuk pada penelitian sebelumnya sudah ada yang pernah membahas dengan teknologi ASP.NET Core namun kekurangannya adalah penelitian sebelumnya hanya menganalisis HTTP Request dengan method GET dan POST adapun kekurangan dari penelitian sebelumnya dalam penelitian masih melakukan permintaan data

dengan jumlah yang relatif kecil hanya sekitar 25-61 kb.

Oleh karena itu berdasarkan fakta dan permasalahan di atas, penulis bertujuan untuk menganalisis terhadap kinerja REST API dan GraphQL API yang diharapkan dapat membantu untuk menentukan arsitektur API yang terbaik dalam membangun sebuah menggunakan teknologi ASP.NET Core 6.0 Framework. Response time, latency, dan processing time menjadi tolak ukur dalam penelitian ini semakin cepat proses permintaan data yang kembalikan ke client maka semakin cepat juga informasi yang tersampaikan dan ini juga berpengaruh terhadap kepuasan pengguna web service.

2. Metode Penelitian

a. Analisis Kebutuhan

Adapun spesifiikasi dari perangkat keras dan perangkat lunak yang dibutuhkan dan digunakan [7]–[10] pada penelitian ini bisa dilhat di Tabel 1–3

Tabel 1 Daftar Perangkat Lunak

Nama	Keterangan
Visual Studio 2022	Software yang akan digunakan untuk membuat sistem
MySQL	Dibutuhkan sebagai sistem untuk database.
Apache JMeter	Tools yang digunakan untuk menjalankan pengujian
Ubuntu 22.04	Sistem Operasi Sistem Operasi yang akan digunakan sebagai server API.
Apache2	Digunakan sebagai web server yang dapat berjalan di virtual machine
ASP.NET Core 6.0	Framework yang digunakan untuk implementasi API.
C#	Dibutuhkan untuk implementasi ASP.NET Core 6.0 sebagai Bahasa pemrograman
Oracle VM VirtualBox 6.1	Software yang akan digunakan untuk virtualisasi sistem

Tabel 2 Spesifikasi Perangkat Keras

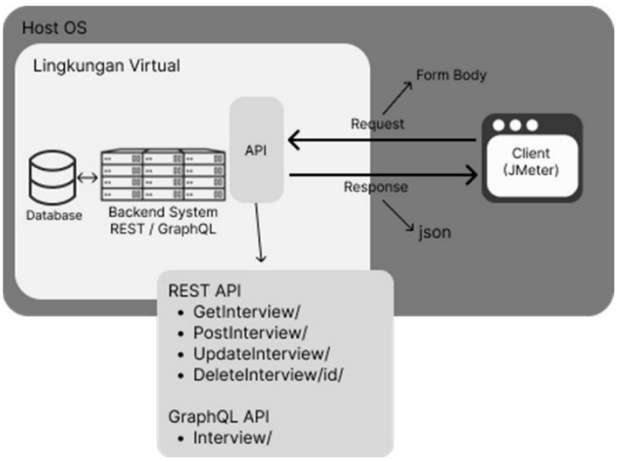
Keterangan	Spesifikasi
Processor	AMD Ryzen 5 3550H
Graphics	AMD Radeon™ Graphics 2GB & NVIDIA GTX 1650 4GB
RAM	8GB DDR 4
SSD	512 GB
Operating System	Windows 11 Pro 21H2

Tabel 3 Spesifikasi Sistem Virtual

Keterangan	Spesifikasi
Nama System	REST API / GraphQL API
Processor	2 CPU
Base Memory	2 GB
Storage	25 GB
Operating System	Ubuntu (64-bit)
Fungsi	Akan digunakan sebagai web server yang akan menjalankan REST API/GraphQL API

b. Desain Perancangan Sistem

Pada tahap ini dilakukan perancangan dan desain terhadap lingkungan sistem, baik dari database, struktur REST API dan struktur GraphQL API serta pendukung lainnya yang akan dilakukan dalam penelitian ini, berikut rancangan dari sistem yang dibuat bisa dilihat di Gambar 1.



Gambar 1. Desain Rancangan Sistem

b. Pengujian

Pengujian pada arsitektur REST API dan GraphQL API yang bertujuan untuk mengetahui performa dari masing-masing Web API. Pengujian akan dilakukan dalam empat skenario yaitu GET, POST, PUT, DELETE. Adapaun penjelasan dari setiap skenario pengujian adalam sebagai berikut:

1. Skenario Pengujian GET; Pengujian ini dilakukan untuk menerima data dari web service yang telah dibuat dan dikembalikan ke client berdasarkan data yang diminta untuk output yang diharapkan
2. Skenario Pengujian POST; Pengujian ini dilakukan untuk mengirim data ke server sesuai dengan kueri yang telah ditentukan kemudian server akan menyimpan sumber data ke dalam web service yang telah dibuat dan client akan menerima response apakah data berhasil disimpan atau gagal.
3. Skenario Pengujian PUT; Pengujian ini dilakukan untuk mengirim data ke server sesuai dengan kueri yang telah ditentukan kemudian server akan memperbaharui data sesuai dengan sumber data ke dalam web server yang telah dibuat dan client akan menerima response apakah data berhasil diubah atau gagal.
4. Skenario Pengujian Delete; Pengujian ini dilakukan untuk menghapus sumber data yang ada di web server dengan mengirim primary key (id)

kemudian client menerima response apakah data berhasil dihapus atau gagal.

Adapun detail konfigurasi pengujian di aplikasi Apache JMeter dapat dilihat di Tabel 4.

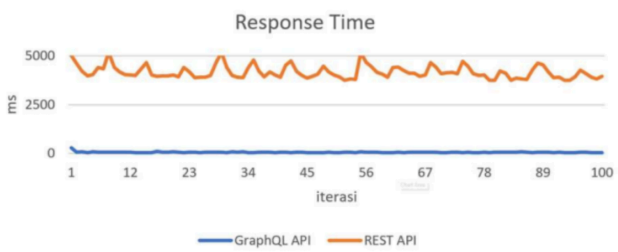
Tabel 4 Konfigurasi Aplikasi Apache Master

Number Of Thread	1	Banyaknya user yang akan melakukan permintaan
Loop Count	100	Pengulangan yang akan dilakukan per user
Ramp-Up Period	100	Total waktu untuk melakukan jumlah dari Number of Threads, jadi setiap user yang melakukan permintaan akan membutuhkan per satu detik.
HTTP Cookie Manager	-	Menghaus cookies setiap iterasi dilakukan
HTTP Cache Manager	5000	Menghapus setiap iterasi dilakukan dengan maksimal yang dijumlah
Constant Timer	300	Sebagai jeda pada thread yang ditentukan pada jumlah dalam bentuk milliseconds
HTTP Header Manager	-	Menambahkan HTTP header application/json pada setiap request
CSV Data Set Config	-	Menyisipkan data dummy yang diperlukan sebagai data pengujian

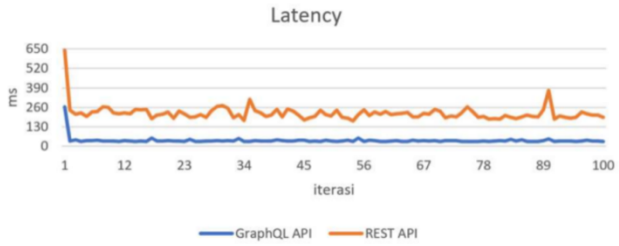
3. Hasil dan Pembahasan

A. Hasil Pengujian Skenario GET

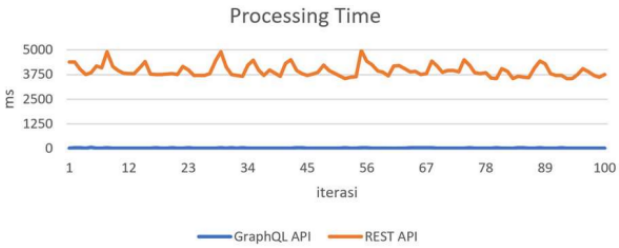
Berikut adalah data dari pengujian skenario GET pada arsitektur GraphQL API dan REST API dalam bentuk grafik dapat dilihat di Gambar 2-4.



Gambar 2. Hasil Pengujian Skenario GET (response time)



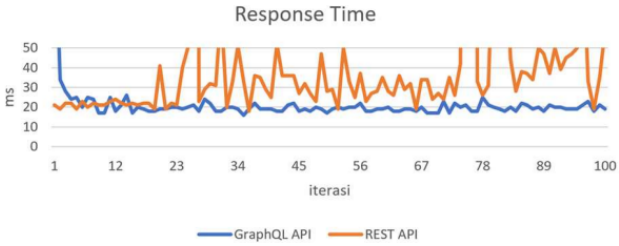
Gambar 3. Hasil Pengujian Skenario GET (latancy)



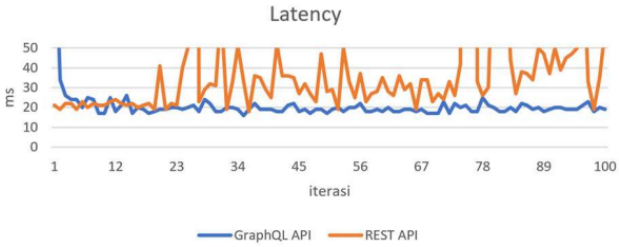
Gambar 4. Hasil Pengujian Skenario GET (processing time)

B. Hasil Pengujian Skenario POST

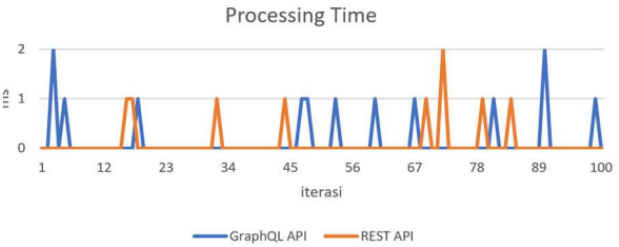
Berikut adalah data dari pengujian skenario GET pada arsitektur GraphQL API dan REST API dalam bentuk grafik dapat dilihat di Gambar 5-7.



Gambar 5. Hasil Pengujian Skenario POST (response time)



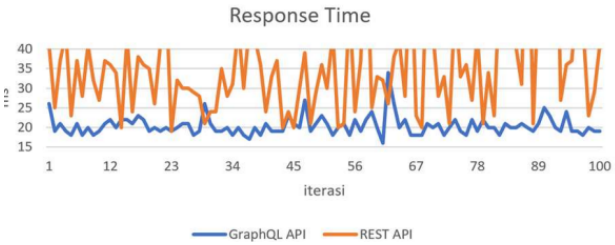
Gambar 6. Hasil Pengujian Skenario POST (latancy)



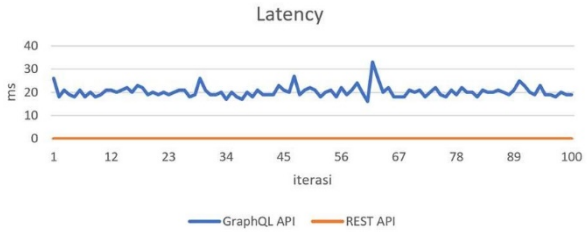
Gambar 7. Hasil Pengujian Skenario POST (processing time)

C. Hasil Pengujian Skenario PUT

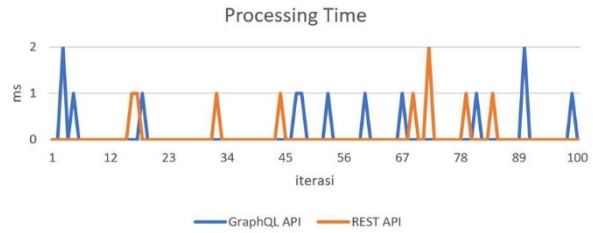
Berikut adalah data dari pengujian skenario GET pada arsitektur GraphQL API dan REST API dalam bentuk grafik dapat dilihat di Gambar 8-10.



Gambar 8. Hasil Pengujian Skenario PUT (response time)



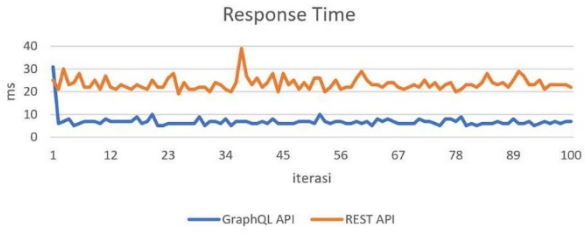
Gambar 9. Hasil Pengujian Skenario PUT (latency)



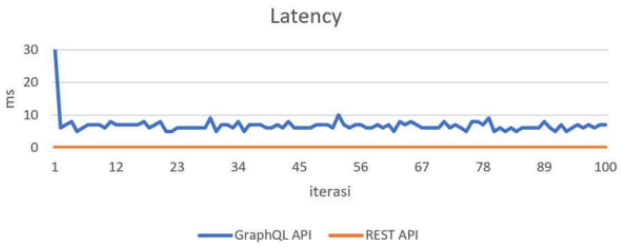
Gambar 10. Hasil Pengujian Skenario PUT (processing time)

D. Hasil Pengujian Skenario DELETE

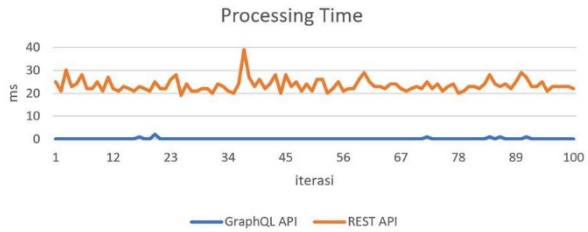
Berikut adalah data dari pengujian skenario GET pada arsitektur GraphQL API dan REST API dalam bentuk grafik dapat dilihat di Gambar 11-13.



Gambar 11. Hasil Pengujian Skenario DELETE (response time)

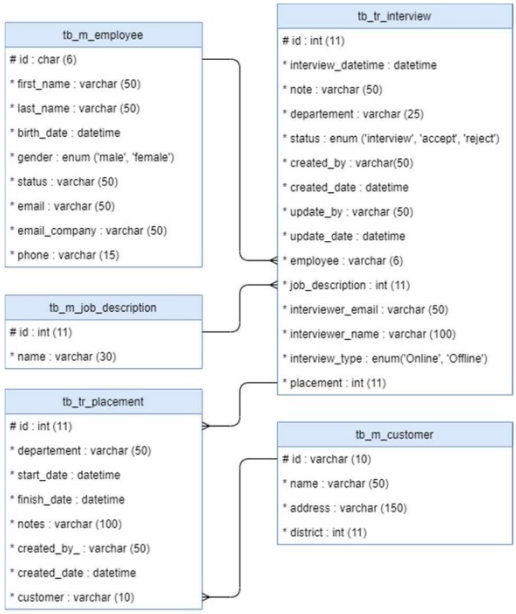


Gambar 12. Hasil Pengujian Skenario DELETE (latency)



Gambar 13. Hasil Pengujian Skenario DELETE (processing time)

Adapun perancangan database dilakukan untuk mengetahui bagaimana data akan disajikan yang nantinya berperan dalam pengujian API, Perancangan akan dibuat dalam bentuk diagram atau biasa disebut ERD (Entity Relationship Diagram). Berikut ERD yang diterapkan dapat dilihat di Gambar 14.



Gambar 14. Rancangan Database

4. Kesimpulan

Hasil dari pengujian yang dilakukan dengan skenario yang dibuat memiliki beberapa kesimpulan yaitu:

1. Pada skenario pengujian GET dengan arsitektur GraphQL API memiliki perbandingan yang cukup signifikan dibandingkan REST API jika

menggunakan ASP.NET Core 6.0, ini membuktikan bahwa GraphQL API dapat implementasi dalam project khususnya method GET karena selain dapat mengatasi kelemahan pada REST API, GraphQL API juga mampu menandingi REST API dalam segi performa. Pada skenario pengujian POST, PUT, dan DELETE hasil pengujian dari kedua arsitektur memang tidak terlalu signifikan namun GraphQL API masih menandingi dari segi performa REST API. Kestabilan dari kinerja setiap arsitektur API untuk seluruh HTTP Request juga berbeda-beda. Di mana GraphQL API lebih stabil dalam setiap permintaan data walaupun pada permintaan awal GraphQL API mendapatkan response time yang cukup tinggi namun pada iterasi berikutnya GraphQL API mulai stabil dibandingkan REST API yang sangat tidak stabil bahkan peningkatan atau penurunan cukup berbeda jauh pada setiap iterasi dengan parameter uji yang sama.

Masih terdapat kekurangan dalam penelitian ini penulis memberikan saran untuk dikembangkan dalam penelitian yang akan datang adalah sebagai berikut:

1. Penelitian selanjutnya diharapkan dapat menambahkan parameter uji lain seperti throughput atau standar deviasi, dan bisa juga menggunakan sistem operasi Ubuntu server.
2. Penelitian selanjutnya diharapkan dapat menguji setiap arsitektur API di dalam lingkungan cloud sehingga hasil dari pengujian tersebut dapat menunjukkan apakah lingkungan infrastruktur berpengaruh atau tidak terhadap kinerja arsitektur GraphQL API dan REST API.

Daftar Pustaka

[1] N. Shakhovska, O. Basystiuk, and K. Shakhovska, "Development of the Speech-to-Text Chatbot Interface Based on Google API.," in *MoMLeT*, 2019, pp. 212–221.

[2] A. Nurkholis, R. Bimantara, and N. Neneng, "Interactive English E-Learning Based on Cloud Speech-to-Text API," *Jurnal Ilmiah Edutic: Pendidikan dan Informatika*, vol. 9, no. 1, pp. 1–9, 2022.

[3] N. Anggraini, A. Kurniawan, L. K. Wardhani, and N. Hakiem, "Speech recognition application for the speech impaired using the android-based google cloud speech API," *TELKOMNIKA (Telecommunication Computing Electronics and Control)*, vol. 16, no. 6, pp. 2733–2739, 2018.

[4] D. I. S. Saputra, S. W. Handani, and G. A. Diniary, "Pemanfaatan Cloud Speech Api Untuk Pengembangan Media Pembelajaran Bahasa Inggris Menggunakan Teknologi Speech Recognition," *Jurnal Telematika*, vol. 10, no. 2, pp. 92–105, 2017.

[5] B. Iancu, "Evaluating Google Speech-to-Text API's Performance for Romanian e-Learning Resources.," *Informatica Economica*, vol. 23, no. 1, 2019.

[6] A. Quiña-Mera, J. M. García, P. Fernández, P. Vega-Molina, and A. Ruiz-Cortés, "GraphQL or REST for Mobile Applications?," in *Advanced Research in Technologies, Information, Innovation and Sustainability: Second International Conference, ARTIIS 2022, Santiago de Compostela, Spain, September 12–15, 2022, Revised Selected Papers, Part I*, Springer, 2022, pp. 16–30.

[7] A. Cetageti, A. Surahman, and A. Sucipto, "PENERAPAN TEKNOLOGI POINT OF SALES (POS) SEBAGAI MEDIA INFORMASI PENJUALAN IKAN HIAS BERBASIS WEB STUDI KASUS: KING KOI GROUB," *TELEFORTECH: Journal of Telematics and Information Technology*, vol. 2, no. 2, pp. 33–39, 2022.

[8] M. Ronaldo and D. Pasha, "Sistem Informasi Pengelolaan Data Santri Pondok Pesantren an-Ahl Berbasis Website," *TELEFORTECH: Journal of Telematics and Information Technology*, vol. 2, no. 1, pp. 17–20, 2021.

[9] S. Wulandari, J. Jupriyadi, and M. Fadly, "RANCANG BANGUN APLIKASI PEMASARAN PENGGALANGAN INFAQ BERAS (STUDI KASUS: GERAKAN INFAQ)," *TELEFORTECH: Journal of Telematics and Information Technology*, vol. 2, no. 1, pp. 11–16, 2021.

[10] F. R. A. Pratama, S. Styawati, and A. R. Isnain, "RANCANG BANGUN APLIKASI PENERIMAAN SISWA BARU MENGGUNAKAN METODE WEB ENGINEERING," *TELEFORTECH: Journal of Telematics and Information Technology*, vol. 1, no. 2, pp. 61–66, 2021.