

Penggunaan *GraphQL* Sebagai Alternatif *Rest API*: Studi Kasus Pada Pengembangan *Website*

Jepri Yandi¹, Muchlis²

Universitas Prabumulih

Penulis Korespondensi : yhandijefry@gmail.com

najwamuchlis@gmail.com

Informasi Artikel:

Received: 23 November 2024, Accepted: 02 Desember 2024, Published: 05 Februari 2025

Abstract

GraphQL memungkinkan klien untuk secara spesifik meminta data yang mereka butuhkan, mengurangi *over-fetching* dan *under-fetching*. Selain itu, dengan menggunakan *GraphQL*, pengembang dapat menggabungkan beberapa permintaan menjadi satu *query*, yang dapat meningkatkan efisiensi dan performa aplikasi. Namun, adopsi *GraphQL* juga menghadirkan tantangan baru, termasuk kompleksitas dalam implementasi dan potensi masalah keamanan.

Keywords: GraphQL, REST API, Website

1. Pendahuluan

Dalam era digital yang serba cepat ini, pengembangan aplikasi *web* menjadi semakin kompleks dan dinamis. Salah satu aspek krusial dalam pengembangan *web* adalah komunikasi antara klien dan *server*. Selama bertahun-tahun, *REST API* (*Representational State Transfer Application Programming Interface*) telah menjadi metode standar untuk mengelola interaksi ini [1]. *REST API* menawarkan cara yang sederhana dan terstruktur untuk pertukaran data, namun, seiring dengan perkembangan teknologi dan kebutuhan aplikasi yang semakin kompleks, beberapa keterbatasan *REST API* mulai muncul ke permukaan. [2].

REST API sering kali mengalami masalah *over-fetching*, di mana klien menerima lebih banyak data daripada yang dibutuhkan, atau *under-fetching*, di mana klien harus membuat beberapa permintaan untuk mendapatkan semua data yang diperlukan. Masalah-masalah ini dapat menyebabkan inefisiensi dalam penggunaan bandwidth dan penurunan kinerja aplikasi. Untuk mengatasi keterbatasan tersebut, *GraphQL* diperkenalkan oleh Facebook pada tahun 2015 sebagai sebuah *query language* yang memberikan fleksibilitas lebih besar dalam pengambilan data. [3]

GraphQL memungkinkan klien untuk secara spesifik meminta data yang mereka butuhkan, mengurangi *over-fetching* dan *under-fetching*. Selain itu,

Published by:

dengan menggunakan *GraphQL*, pengembang dapat menggabungkan beberapa permintaan menjadi satu *query*, yang dapat meningkatkan efisiensi dan performa aplikasi. Namun, adopsi *GraphQL* juga menghadirkan tantangan baru, termasuk kompleksitas dalam implementasi dan potensi masalah keamanan.

Berdasarkan uraian singkat diatas maka penulis tertarik untuk melakukan penelitian dengan judul : “Penggunaan *GraphQL* sebagai Alternatif *REST API*: Studi Kasus pada Pengembangan *Web*”.

2. Tinjauan Literatur

2.1 *GraphQL*

Dalam buku "*Learning GraphQL: Declarative Data Fetching for Modern Web Apps*," Jonathan Freeman, Samer Buna, dan Eve Porcello mendefinisikan *GraphQL* sebagai "*a query language for APIs and a runtime for executing those queries by using a type system that you define for your data.*" Mereka menjelaskan bahwa *GraphQL* memungkinkan pengembang untuk menentukan secara eksplisit struktur data yang dibutuhkan dan mengurangi redundansi data dalam aplikasi. [4]

2.2 *REST API*

Dalam buku "*Service Technology Magazine*," Thomas Erl menjelaskan *REST* sebagai "*an architectural style that leverages the ubiquity of HTTP protocols to create web services that are lightweight, scalable, and easy to maintain.*" Erl menekankan bahwa *RESTful webservices* menggunakan pendekatan yang sangat modular dan dapat diskalakan untuk mengelola sumber daya di *web*. [5]

2.3 *Pengembangan Website*

Menurut Ethan Marcotte, seorang pionir dalam desain web responsif, mendefinisikan pengembangan website sebagai "*the practice of creating web experiences that adapt to the constraints of various devices and screen sizes, using flexible grids, flexible images, and media queries.*" Dia menekankan pentingnya desain yang responsif untuk memastikan situs web dapat diakses dan digunakan dengan baik di berbagai perangkat. [6]

3. Metode Penelitian

3.1 Metode Pengumpulan

Pada penelitian kali ini, penulis melakukan cara untuk mengumpulkan data yang terkait dengan penelitian terdahulu seperti latar belakang permasalahan, landasan teori, metode penelitian, hasil dan pembahasan, dan pengujian pada proses implementasi. Metode pengumpulan data yang digunakan pada penelitian kali ini menggunakan studi literatur dan studi pustaka.

3.1.1 Studi Literatur dan Studi Pustaka

Metode studi literatur dan studi pustaka digunakan penulis untuk mempelajari dan memahami terkait dengan penelitian implementasi arsitektur *GraphQL* dari berbagai sumber terpercaya, seperti jurnal, *ebook*, dan situs – situs internet.

3.1.2 API Data Scraping

Penulis mendapatkan *database* dari arsitektur *ReST* ke arsitektur *GraphQL* melalui API

Published by:

documenter sebagai jembatan antara keduanya. Proses dilakukan dengan memanggil API dari metode HTTP yaitu Post, Get, Put, dan Delete untuk mengambil data dari *database* arsitektur ReST. Selanjutnya, penulis membuat *schema*, *resolver*, dan *query* untuk memanggil data dari API *documenter* dan menyimpannya di dalam *database* pada arsitektur GraphQL.

3.2 Metode Research dan Development

Metode *research and development* atau penelitian dan pengembangan digunakan pada penelitian kali ini. Menurut Sugiyono penelitian dan pengembangan merupakan metode penelitian yang menghasilkan produk tertentu dan menguji keefektifan produk tersebut [7]. metode penelitian dan pengembangan adalah suatu proses untuk mengembangkan suatu produk baru atau menyempurnakan terhadap produk yang sudah ada dan memvalidasi produk tersebut [8]. Produk yang dimaksud tidak selalu mengarah kepada produk perangkat keras (*hardware*) seperti modul, buku, komputer tetapi dapat melainkan perangkat lunak (*software*) seperti sistem pengolahan data, aplikasi pembelajaran, arsitektur pengembangan, dan lain sebagainya. Oleh karena itu, penelitian kali ini akan mengembangkan penelitian sebelumnya tentang arsitektur ReST menjadi arsitektur GraphQL pada rancangan arsitektur aplikasi.

Metode penelitian dan pengembangan memiliki beberapa model pengembangan untuk membuat suatu produk yang bisa dimanfaatkan. Menurut Puslitjaknov model pengembangan merupakan hal dasar untuk mengembangkan suatu produk yang ingin dihasilkan. Model pengembangan dapat berupa model prosedural, model konseptual, dan model teoritik [9]. Dalam penelitian ini menggunakan model prosedural karena dianggap sesuai dengan tujuan penelitian yang ingin dicapai yaitu mengimplementasikan arsitektur GraphQL pada pengembangan *website*, hal tersebut sesuai dengan tujuan penelitian yang harus melalui langkah – langkah tertentu untuk diikuti dalam mencapai arsitektur yang dimaksud [10].

4. Hasil dan Pembahasan

4.1 Response Time

Tabel 1. Response Time

Arsitektur	Requests 1 (ms)	Requests 2 (ms)	Requests 3 (ms)	Requests 4 (ms)
GraphQL	167	166	157	141
ReST	1683	1677	1672	1670

Waktu respon yang dihasilkan menunjukkan arsitektur GraphQL lebih cepat dalam menanggapi *request* dari pengguna dengan nilai rata – rata 157,75ms pada rentang 141 – 167 ms. Sedangkan arsitektur ReST memiliki rata – rata 1675 ms dalam rentang waktu 1670 – 1685 ms.

Perbandingan waktu respon antara arsitektur GraphQL dan ReST pada pengembangan website didapatkan bahwa arsitektur GraphQL lebih cepat menerima waktu respon pengguna dibandingkan dengan arsitektur ReST.

4.2 Throughput

Tabel 2. *Throughput*

Arsitektur	<i>Requests 1</i> (bps)	<i>Requests 2</i> (bps)	<i>Requests 3</i> (bps)	<i>Requests 4</i> (bps)
GraphQL	130	247	366	484
ReST	117	223	330	436

Throughput yang digunakan pada pengembangan *website* menggunakan arsitektur GraphQL dapat menangani rata – rata 306 bps dengan kisaran 130– 484 bps. Sedangkan untuk arsitektur ReST dapat menangani rata – rata 276 bps dengan kisaran 117– 436 bps.

Nilai perbandingan *throughput* antara arsitektur GraphQL dan ReST didapatkan bahwa arsitektur ReST lebih cepat memproses jumlah permintaan yang diproses oleh *server* per satuan waktu (*bit per seconds*).

4.3 CPU Load

Tabel 3. CPU Load

Arsitektur	<i>Requests 1</i> (%)	<i>Requests 2</i> (%)	<i>Requests 3</i> (%)	<i>Requests 4</i> (%)
GraphQL	8,67	9,51	10,28	13,97
ReST	29,98	33,11	37,53	25,39

Pengujian CPU *load* pada pengembangan *website* dilakukan sebanyak 4 kali percobaan. Dalam percobaan pertama Arsitektur GraphQL memiliki rata – rata penggunaan CPU sebesar 10% dalam kisaran 9– 1%. Sementara itu, arsitektur ReST menggunakan konsumsi CPU yang lebih besar dengan nilai rata- rata sebesar 31% dalam kisaran 29– 37%.

Pada tabel 3 didapatkan bahwa penggunaan arsitektur GraphQL lebih unggul dari ReST dalam hal kinerja terkait konsumsi dan pemanfaatan sumber daya. Kinerja beban CPU pada arsitektur GraphQL lebih efisien dalam memanfaatkan sumber daya pemrosesan dengan nilai grafik selalu berada dibawah ReST.

5. Kesimpulan

Dapat disimpulkan bahwa implementasi arsitektur GraphQL berhasil dan cocok diterapkan pada pengembangan *website*, dapat dilihat dari pengujian dengan menggunakan metode *quality of service* pada parameter *response time*, CPU *load*, dan *memory utilization* yang dapat menerima respon dengan cepat, efisien dalam memanfaatkan sumber daya pemrosesan, dan penggunaan memori yang lebih sedikit. Hal ini sesuai dengan penelitian yang dilakukan oleh Lawi et Al. tahun 2021 bahwa arsitektur GraphQL sangat baik dalam menghasilkan respon yang cepat, memanfaatkan sumber daya dan menggunakan memori yang lebih sedikit dibandingkan dengan arsitektur ReST.

Dalam hal kecepatan waktu respon, arsitektur GraphQL menunjukkan bahwa semakin banyak jumlah pengguna dengan waktu konstan 100 detik, maka waktu yang diperlukan *server* untuk memproses permintaan semakin cepat. Alasan lain yang mendukung bahwa arsitektur GraphQL dalam memanfaatkan sumber daya CPU dan penggunaan memori jauh lebih efisien, hal itu terjadi karena pengembang hanya memerlukan satu *endpoint* untuk menentukan kebutuhan data yang ingin didapatkan pada *output* didalam pendefinisian *query* permintaan.

Referensi

- [1] Ehsan, A., Abuhaliqa, M. A. M., Catal, C., & Mishra, D. (2022). RESTful API testing methodologies: Rationale, challenges, and solution directions. *Applied Sciences*, 12(9), 4369.
- [2] Mikula, M. and Dzieńkowski, M., 2020. Comparison of REST and GraphQL web technology performance. *Journal of Computer Sciences Institute*, 16, pp.309-316.
- [3] Biehl, M. (2016). *RESTful Api Design* (Vol. 3). API-University Press.
- [4] Porcello, E., & Banks, A. (2018). *Learning GraphQL: declarative data fetching for modern web apps*. " O'Reilly Media, Inc."
- [5] Li, L., Chou, W., Zhou, W., & Luo, M. (2016). Design patterns and extensibility of REST API for networking applications. *IEEE Transactions on Network and Service Management*, 13(1), 154-167.
- [6] Rivani, R. A., Hartiwati, E. N., & Dwitama, F. (2022). Pembuatan Website Arsyla Catering. *Jurnal Ilmiah Teknik*, 1(3), 14-21.
- [7] Prasetyo, I. (2012). Teknik analisis data dalam research and development. *Jurusan PLS FIP Universitas Negeri Yogyakarta*.
- [8] Violadini, R., & Mustika, D. (2021). Pengembangan e-modul berbasis metode inkuiri pada pembelajaran tematik di sekolah dasar. *Jurnal basicedu*, 5(3), 1210-1222.
- [9] Ramdhan, Muhammad. *Metode penelitian*. Cipta Media Nusantara, 2021.
- [10] Abdussamad, H. Z., & Sik, M. S. (2021). *Metode penelitian kualitatif*. CV. Syakir Media Press.

Copyrights

Copyright for this article is retained by the author(s), with first publication rights granted to the journal.

This is an open-access article distributed under the terms and conditions of the Creative Commons Attribution license (<http://creativecommons.org/licenses/by/4.0/>)

