**IMPLEMENTASI GRAPHQL UNTUK MENGATASI *UNDER-FETCHING* PADA PENGEMBANGAN SISTEM INFORMASI PELACAKAN ALUMNI POLITEKNIK NEGERI MALANG**

**Fany Ervansyah1, Putra Prima Arhandi2, ST., M.Kom.**

1program studi teknik informatika, Politeknik Negeri Malang

email: 1fanyervansyah.9c@gmail.com, 2putraprima@gmail.com

**Abstract -** Data distribution using REST API has several problems. Under-fetching is one of those problems, where the frontend side have to do more than 1 request to get the data they needed. This problem can be solved by implementing GraphQL. The main point of this research is to compare the performance REST API and GraphQL when under-fetching occured. The case study of the research is the development of alumni tracking information system. GraphQL showed a good performance on case where the data is complex and quite large, also when there are many users accessing the data at the same time. On the other hand, REST API showed good performance if the data is simple and not really big. But, if there are many users trying to access data at the same time, REST API can handle less users than GraphQL.

**Intisari -** Pendistribusian data dengan menggunakan REST API memiliki beberapa kekurangan. Salahsatunya adalah masalah *under-fetching*, dimana bagian *frontend* harus melakukan lebih dari 1 kali *request*. Masalah ini dapat diselesaikan dengan menerapkan GraphQL. Tujuan dari penelitian kali ini adalah untuk membandingkan performa antara sistem informasi dengan REST API dan sistem informasi dengan GraphQL ketika *under-fetching* terjadi. Studi kasus pada penelitian ini adalah pada pengembangan sistem informasi pelacakan alumni Politeknik Negeri Malang. GraphQL menunjukkan performa yang baik pada jumlah data yang besar dan kompleks, serta ketika terdapat banyak pengguna yang mengakses data dalam waktu yang bersamaan. Sedangkan untuk data yang sederhana dan sistem informasi yang tidak memiliki banyak pengguna yang akan mengakses data secara bersamaan, maka REST API masih lebih unggul.

**Kata Kunci :** *Sistem Informasi, GraphQL, under-fetching*

1. PENDAHULUAN

Distribusi data merupakan hal yang sangat penting dalam pengembangan sebuah sistem informasi. Dalam pendistribusian data, beberapa *website* menerapkan metode yang berbeda-beda. Mulai dari menggabungkan antara bagian yang bertugas menampilkan informasi ke pengguna (*Frontend*) dan bagian yang mengatur bagaimana data - data diolah (*Backend*) hingga menyediakan layanan distribusi data seperti REST API, untuk sistem dengan struktur *frontend* dan *backend* yang terpisah.

Pada tahun 2017, metode yang paling banyak digunakan dalam pendistribusian data adalah dengan menggunakan metode REST API (Motroc, 2017). Namun, dalam metode REST API, terdapat suatu masalah yang disebut *under-fetching*, dimana bagian *frontend* perlu untuk melakukan permintaan data lebih dari 1 kali ke bagian *backend* (Porcello, Banks, 2018). Hal itu akan meningkatkan latensi, yang membuat pengakses website harus menunggu lebih lama sebelum data dikirim pada bagian *frontend* dengan sempurna. Selain itu, kompleksitas program juga semakin bertambah karena bagian *frontend* harus menambahkan 1 permintaan data lagi ke bagian *backend*.

Salah satu cara mengatasi *under-fetching* pada REST API adalah dengan membuat *endpoint* baru yang melakukan pengambilan data sesuai dengan apa yang diminta oleh *frontend*. Namun, jika terdapat banyak data yang mirip dan memiliki endpoint masing-masing, maka bentuk kode pada bagian *backend* akan menjadi kompleks dan kurang baik diakibatkan oleh adanya kode yang memiliki fungsi mirip namun ditulis lebih dari 1 kali.

Pada tahun 2015, secara publik, Facebook meluncurkan sebuah *query language* yang menjadi metode baru dalam mengatur pendistribusian data. Nama *query language* tersebut adalah GraphQL. Salah satu masalah yang dapat diatasi oleh GraphQL adalah masalah *under-fetching*.

Oleh karena itu, kali ini penulis mencoba mengimplementasikan GraphQL untuk mengatasi *under-fetching* pada Pengembangan Sistem Informasi Pelacakan Alumni Politeknik Negeri Malang yang akan penulis kembangkan. Dengan adanya GraphQL, diharapkan dapat meningkatkan performa situs web Sistem Informasi Pelacakan Alumni Politeknik Negeri Malang.

1. PENELITIAN TERKAIT

Sejauh ini, penulis menemukan 5 jurnal yang membahas mengenai graphQL. Jurnal - jurnal tersebut adalah:

1. “*An Initial Analysis of Facebook’s GraphQL Language*” bertujuan untuk memahami bahasa Graph milik GraphQL dan menunjukkan bahwa bahasa tersebut memiliki kompleksitas yang rendah (Hartig, Pérez, 2017).
2. “*API Design in Distributed Systems: A Comparison between GraphQL and REST*” bertujuan untuk membandingkan REST API dan GraphQL (Eizinger, 2017).
3. “*Improving the OEEU’s data-driven technological ecosystem’s interoperability with GraphQL*” yaitu menerapkan GraphQL pada Observatory of Employment and Employability, sebuah grup riset untuk lulusan universitas di Spanyol. Hasil dari penerapan GraphQL menunjukkan peningkatan performa, *flexibility*, dan *maintainability* (Vazquez, Cruz, García, 2017).
4. “*Performance of frameworks for declarative data fetching: An evaluation of Falcor and Relay+GraphQL*” membahas tentang perbandingan performa pengambilan data menggunakan Falcor, dan Relay+GraphQL. Falcor merupakan produk buatan Netflix yang juga dapat membantu pengambilan data dari *backend* ke *frontend*. Sedangkan Relay adalah sebuah *framework* untuk mempermudah pengambilan data dari sisi *client* (Cederlund, 2016).
5. “*Using GraphQL for Content Delivery in Kentico Cloud*” bertujuan untuk meriset GraphQL sebagai alternatif untuk mengantar konten pada Kentico Cloud, selain menggunakan API yang sudah disediakan oleh Kentico Cloud, yaitu sebuah CMS online yang menyediakan konten sebagai *service*. Kentico menyediakan REST API untuk melakukan pengambilan data, namun di sini, dicoba diterapkan GraphQL (Čechák, 2017).
6. METODE PENELITIAN
7. *Metode Pengambilan Data*

Pengambilan data dilakukan dengan cara *scraping* pada situs Linkedin dan input secara manual melalui sistem informasi yang akan dibuat.

Pada pengambilan data di Linkedin, data yang diambil adalah data mahasiswa yang sudah mendaftarkan Politeknik Negeri Malang sebagai riwayat pendidikan mereka. Mekanisme pengambilan datanya adalah dengan menggunakan bot untuk melakukan login terotomatisasi, kemudian masuk ke halaman Politeknik Negeri Malang, lalu membuka halaman detail setiap *card* dari mahasiswa yang muncul. Dari halaman detail yang sudah dibuka, akan diambil data-data yang diperlukan.

Pada pengambilan data dari input manual, pengambilan data dilakukan dengan mengisikan data alumni melalui *form* yang sudah disediakan di sistem. *Form* tersebut memiliki *field* diantaranya berupa nama, tahun masuk dan lulus dari Politeknik Negeri Malang, jurusan yang diambil, pekerjaan saat ini, jabatan/posisi yang dipegang pada pekerjaan saat ini, dan email yang bisa dihubungi.

1. Metode Pengujian

Pengujian pertama akan dilakukan dengan cara memuat suatu halaman yang sama dan memiliki masalah *under-fetching*, pada Sistem Informasi yang menggunakan GraphQL maupun yang menggunakan REST sebagai metode pendistribusian datanya. Kemudian, halaman akan di-*refresh* sebanyak 20 kali. Pada masing-masing sesi *refresh,* akan ada beberapa data yang dicatat pada masing-masing sistem informasi. Diantaranya adalah jumlah *request* ke *backend*, waktu muat *request* hingga selesai, dan waktu total memuat halaman. Dari data-data tersebut, akan dirata-rata pada masing-masing sistem informasi dan akan dibandingkan. Sehingga, dapat diketahui kelebihan/kekurangan dari graphQL dan REST jika digunakan dalam pendistribusian data.

Uji coba kedua adalah dengan menguji kemampuan *backend* dari masing-masing metode pendistribusian data dalam menangani jumlah *user* yang melakukan permintaan data secara bersamaan. Jumlah *user* yang disimulasikan sejumlah 50 *users* dan batas durasi yang diberikan adalah 1 menit. Sehingga, 50 *users* akan melakukan *request* secara bersamaan dan selama 1 menit, akan dicatat berapa *user* yang mendapatkan respon dari *backend*.

GraphQL dikatakan berhasil menyelesaikan masalah *under-fetching* jika sistem yang dikembangkan dengan mengimplementasikan GraphQL dapat melakukan pengaksesan *endpoint* lebih sedikit daripada sistem dengan metode REST API dalam memenuhi kebutuhan datanya atau jika server dengan GraphQL dapat merespon sejumlah *user* yang lebih banyak dibandingkan dengan REST API.

1. HASIL DAN PEMBAHASAN