Jurnal Infosecure Vol.3 No.1, April 2022
p-ISSN: 2087-1023
MTI-UNLA

# PERBANDINGAN GRAPHQL DAN RESTFUL API DARI PERSPEKTIF PENGGUNAAN SUMBER DAYA PADA APLIKASI PARKIR PINTAR

Ferdian Chendra

*Program Studi Magister Teknik Informatika, Universitas Langlangbuana*

chendra.f@gmail.com

**Abstrak—** Penelitian ini membandingkan secara kuantitatif performa arsitektur API RESTful dan GraphQL dari perspektif penggunaan sumber daya server dalam konteks aplikasi parkir pintar. Dengan mengimplementasikan kedua API menggunakan Ruby on Rails dan database MySQL, pengujian dilakukan melalui tiga skenario simulasi, under-fetching, over-fetching, dan kondisi ideal, dengan beban pengguna simultan yang bervariasi (45, 90, 180 dan 361). Metrik utama yang diukur adalah penggunaan CPU, konsumsi memori, dan waktu eksekusi (elapse time) untuk mengevaluasi efisiensi dan skalabilitas masing-masing arsitektur. Hasil pengujian menunjukkan bahwa tidak ada satu arsitektur yang unggul mutlak, performa sangat bergantung pada pola permintaan data. GraphQL terbukti jauh lebih efisien dalam skenario under-fetching dengan mengurangi elapse time secara signifikan di bawah beban tinggi (360 concurrent user). Sebaliknya, RESTful menunjukkan keunggulan dalam hal penggunaan sumber daya yang lebih rendah dan waktu eksekusi yang lebih cepat pada skenario over-fetching dan kondisi ideal, di mana struktur permintaan datanya sederhana dan terdefinisi dengan baik. Analisis statistik menggunakan uji-t mengonfirmasi bahwa perbedaan kinerja ini signifikan ($p < 0.05$) di sebagian besar skenario.

Hasil dari penelitian ini menegaskan bahwa pemilihan arsitektur API secara langsung memengaruhi skalabilitas dan efisiensi sistem parkir pintar. Untuk aplikasi berskala besar dengan kebutuhan data yang fleksibel, GraphQL menawarkan skalabilitas yang lebih baik. Sebaliknya, untuk sistem dengan kebutuhan data yang lebih terprediksi, RESTful menjadi pilihan yang lebih stabil, asalkan desain endpoint dirancang secara cermat untuk menghindari inefisiensi.

Untuk sistem parkir pintar RESTful menjadi pilihan yang lebih baik karena data pada sistem parkir pintar umumnya tidak terlalu fleksibel. Penelitian ini juga memperkaya literatur teknologi API dengan menyajikan analisis perbandingan dari sisi penggunaan sumber daya server, sebuah aspek yang masih jarang dibahas.

**Kata kunci**— GraphQL, RESTful API, penggunaan sumber daya, parkir pintar, *under-fetching, over-fetching,* elapse time

## I. PENDAHULUAN

### *A. Latar Belakang*

Perkembangan kota menuju konsep smart city mendorong perlunya layanan publik yang lebih cepat dan efisien dengan bantuan teknologi. Salah satu masalah yang umum dijumpai di wilayah perkotaan adalah kemacetan akibat pengelolaan parkir yang kurang efektif. Untuk mengatasinya, sistem parkir berbasis reservasi mulai banyak digunakan, menurut laporan pasar, pada 2024 sekitar 75% area parkir komersial di San Francisco dan New York telah mengimplementasikan sistem reservasi berbasis aplikasi (Zion Market Research, 2024), agar pengemudi tidak perlu lagi berkeliling mencari tempat kosong.

Hal ini diperkuat oleh penelitian yang dilakukan oleh E. Polycarpou dan kawan-kawan, yang melakukan survei kepada pengendara di kota Nicosia, Cyprus dan Chania, Yunani. Ketika pengendara tiba di lokasi parkir yang mereka pilih namun ternyata tidak tersedia tempat kosong, mereka menunjukkan respons negatif, sekitar 34% dari mereka memilih parkir secara ilegal di sekitar lokasi tersebut, yang berisiko terkena denda dan bahkan dapat menyebabkan gangguan lalu lintas (E. Polycarpou, dkk., 2014). Kondisi ini menunjukkan bahwa pencarian tempat parkir secara manual dapat memperparah kemacetan, sehingga sistem parkir berbasis reservasi menjadi solusi yang relevan untuk mengurangi kepadatan lalu lintas.

Keberhasilan sistem parkir pintar sangat bergantung pada kecepatan dan efisiensi pertukaran data antara aplikasi dan server. API, sebagai protokol komunikasi antara aplikasi dan server, menjadi komponen yang sangat penting dalam arsitektur sistem digital. Hal ini sejalan dengan pandangan T. Schellenbach (2022) yang menyatakan bahwa API merupakan fondasi utama dalam pengembangan berbagai layanan pada smart city. Ketika sistem dihadapkan pada lonjakan jumlah pengguna, arsitektur API yang dipilih akan menentukan apakah layanan tetap responsif atau justru mengalami bottleneck di sisi server.

Dua arsitektur API yang saat ini paling banyak digunakan adalah RESTful dan GraphQL (Postman, 2023). RESTful dikenal luas dan telah lama menjadi standar dalam pengembangan web, sementara GraphQL menawarkan pendekatan yang lebih fleksibel dengan memungkinkan klien menentukan data yang diinginkan.

Banyak studi sebelumnya telah membandingkan kedua API, seperti penelitian G. Brito and M. T. Valente, 2020, meneliti waktu yang dihabiskan untuk mengimplementasi RESTful dan GraphQL. Beliau juga meneliti perbedaan

bandwidth (G. Brito, dkk., 2020). D.A Hartina, dkk., 2018 meneliti aspek response time dan througput, sedangkan E.Lee, dkk., 2020 meneliti aspek response time dan response size. Dari beberapa peneliatian tersebut, belum ada studi yang secara spesifik membandingkan keduanya dari sisi penggunaan sumber daya server, seperti CPU dan memori terutama dalam konteks aplikasi sistem parkir pintar.

Kondisi di Indonesia memperkuat urgensi penelitian ini. Berbagai studi menunjukkan bahwa sistem digital publik seperti e-government masih menghadapi tantangan keterbatasan server, bandwidth, dan efisiensi pemrosesan data (A.Tasyah, 2020 dan D. Dwi Maharani, 2021). Oleh karena itu, sistem seperti parkir pintar yang akan digunakan secara luas harus mempertimbangkan aspek efisiensi arsitektur sejak tahap pengembangan, agar tidak menimbulkan beban berlebih di sisi backend.

Berdasarkan urgensi tersebut, penelitian ini dilakukan untuk mengisi celah pada studi-studi terdahulu yang umumnya membandingkan RESTful dan GraphQL dari sisi response time, throughput, atau bandwidth. Penelitian ini menawarkan pendekatan baru dengan membandingkan keduanya dari perspektif penggunaan sumber daya server yaitu CPU usage, memory consumption, dan elapse time pada berbagai skenario beban pengguna, khususnya dalam konteks aplikasi parkir pintar. Hasil penelitian ini diharapkan dapat memberikan kontribusi praktis dalam pemilihan arsitektur API yang paling efisien untuk sistem layanan publik berbasis Smart City.

### *B. Rumusan Masalah*

Dengan perbedaan yang dimiliki oleh GraphQL dan RESTful API, maka perbandingan antara keduanya telah menjadi topik menarik dalam komunitas pengembangan perangkat lunak. Berbagai penelitian dan diskusi telah dilakukan untuk membandingkan kelebihan dan kekurangan masing-masing protokol. Namun, masih sedikit penelitian yang secara khusus fokus pada aspek penggunaan sumber daya.

Dalam konteks inilah penelitian ini ingin memberikan kontribusinya. Dengan demikian rumusan masalah dari penelitian ini adalah:

Q1. Bagaimana perbandingan performa kedua API dari segi penggunaan CPU, memory dan elapse time pada aplikasi parkir pintar dengan skenario jumlah pengguna yang berbeda-beda?

Q2. Bagaimana perbandingan performa kedua API dari segi penggunaan CPU, memory dan elapse time dalam berbagai pola pengambilan data, seperti under-fetching, over-fetching, dan kondisi ideal?

Q3. Q3. Arsitektur API mana yang lebih baik (penggunaan CPU dan memori yang lebih rendah serta elapse time yang lebih cepat dalam kondisi beban pengguna yang tinggi) digunakan dalam pengembangan sistem parkir pintar, khususnya pada lingkungan dengan sumber daya server yang terbatas?

### *C. Batasan Masalah*

Batasan masalah dari penelitian ini adalah:

1. Pengujian hanya dilakukan pada satu bahasa pemrograman dan satu jenis database, yaitu menggunakan Ruby on Rails (RoR) sebagai backend framework dan MySQL sebagai database.
2. Sistem parkir pintar yang dikembangkan hanya mencakup perancangan dan pengujian dari sisi API (backend), tanpa melibatkan pembuatan antarmuka pengguna (frontend).
3. Jenis query yang diuji hanya sebatas query baca (read), tidak termasuk query penulisan, penambahan, pembaruan, atau penghapusan data.
4. Dataset yang digunakan berasal dari penelitian Yulianto, Marlieza dalam studinya berjudul Evaluasi Kapasitas Parkir di Mall Paris Van Java Bandung, yang digunakan untuk mensimulasikan data lahan parkir.
5. Pengujian dilakukan secara lokal (localhost) menggunakan satu perangkat laptop sebagai lingkungan server dan klien, sehingga hasil pengujian tidak mempertimbangkan kondisi jaringan eksternal, latency internet, API Gateway, atau distribusi beban (load balancing).
6. Pengujian dilakukan tanpa membangun aplikasi mobile secara nyata, namun perancangan API disesuaikan dengan kebutuhan aplikasi mobile.
7. Pengujian dilakukan pada tiga skenario pengambilan data, yaitu under-fetching, over-fetching, dan kondisi ideal, dengan variasi jumlah concurrent user sebanyak 45, 90, 180, dan 361.
8. Pengukuran performa terbatas pada tiga metrik utama, yaitu: CPU usage (%), memory usage (MB), dan elapse time (detik).
9. Pengujian dilakukan melalui simulasi load menggunakan Apache JMeter tanpa melibatkan antarmuka pengguna (frontend).

### *D. Tujuan*

Berdasarkan rumusan masalah yang telah disusun, penelitian ini bertujuan untuk menjawab dan menganalisis aspek-aspek performa yang relevan terhadap penggunaan API dalam sistem parkir pintar. Adapun tujuan dari penelitian ini adalah sebagai berikut:

1. Untuk mengetahui perbandingan performa GraphQL dan RESTful API dari segi penggunaan CPU, memori, dan elapse time pada aplikasi parkir pintar dengan variasi jumlah concurrent user.

2. Untuk menganalisis performa kedua API dari sisi penggunaan sumber daya dalam berbagai pola pengambilan data, seperti under-fetching, over-fetching, dan kondisi ideal.
3. Untuk menentukan arsitektur API yang paling efisien untuk diterapkan dalam sistem parkir pintar, khususnya pada kondisi server dengan keterbatasan sumber daya

## II. METODE.

Penelitian ini menggunakan metode eksperimen kuantitatif dengan pendekatan komparatif. Metode ini dipilih untuk menguji dan membandingkan performa dua arsitektur API, yaitu RESTful dan GraphQL, berdasarkan penggunaan sumber daya server dalam konteks aplikasi sistem parkir pintar.

Dalam penelitian ini, sampel data lingkungan pengujian didapat dengan menggunakan data sekunder, yang berarti data sudah dikumpulkan dari penelitian sebelumnya. Data diambil dari penelitan yang dilakukan oleh Yulianto, Marlieza, pada penelitiannya yang berjudul Evaluasi Kapasitas Parkir Di Mall Paris Van Java Bandung yang dilakukan pada tanggal 2009 dengan total data parkit sebanyak 12.919.

Data juga akan diperoleh melalui penggunaan alat pemantau performa. Pengukuran penggunaan CPU, memori dan *elapse time* akan dilakukan menggunakan *software* monitor performa, yaitu Apache JMeter. Data diambil pada saat seluruh *concurrent user* telah aktif mengakses API, sehingga sistem berada dalam kondisi beban maksimum (*peak usage*). Hasil pengamatan CPU (%) dan penggunaan memori (MB) dicatat menggunakan PerfMon (Server Performance Monitoring), yaitu plugin tambahan dari Apache JMeter yang memungkinkan pemantauan performa server secara *real-time*.

Setiap skenario pengujian diulang sebanyak lima kali untuk memperoleh data yang lebih stabil. Apabila terdapat nilai ekstrim (*outlier*) yang tidak konsisten atau disebabkan oleh proses eksternal, maka data tersebut disaring dan pengujian diulang untuk menjaga validitas hasil. Nilai yang dicatat adalah penggunaan CPU dan memori tertinggi (*peak usage*) selama proses pengujian, karena nilai tersebut mencerminkan beban maksimum yang harus ditangani oleh sistem. Hasil akhir berupa rata-rata dari lima kali pengujian bersih digunakan sebagai dasar dalam analisis statistik dan visualisasi performa.

Uji-t dua sampel independen digunakan untuk membandingkan nilai rata-rata dari dua kelompok data yang tidak saling bergantung, yaitu hasil pengujian RESTful dan GraphQL. Rumus uji-t yang digunakan dapat dilihat pada gambar 1.

$$t = \frac{\bar{X}_1 - \bar{X}_2}{\sqrt{\frac{S_1^2}{n_1} + \frac{S_2^2}{n_2}}}$$

Gambar. 1 rumus uji-t dua sampel independen

Keterangan:

- $\bar{X}1$, $\bar{X}2$ = rata-rata sampel kelompok 1 (RESTful) dan kelompok 2 (GraphQL)
- $S_1^2$, $S_2^2$ = varians dari masing-masing kelompok
- $n_1$, $n_2$ = jumlah sampel dari masing-masing kelompok

Nilai p-value yang dihasilkan akan dibandingkan dengan tingkat signifikansi $\alpha=0.05$. Jika p-value < 0.05, maka perbedaan antara dua kelompok dianggap signifikan secara statistik.

Tahapan penelitian pada tesis ini adalah sebagai berikut:

1. Studi Literatur dan Pengumpulan Data Sekunder
   Tahap awal melibatkan penelusuran pustaka terkait konsep Smart City, sistem parkir pintar, arsitektur API (RESTful dan GraphQL), serta pendekatan pengujian performa sistem. Selain itu, data parkir sekunder diperoleh dari penelitian sebelumnya oleh Yulianto, Marlieza, untuk digunakan sebagai dasar skenario pengujian.
2. Perancangan Arsitektur Logis Sistem
   Peneliti menyusun arsitektur logis sistem parkir pintar yang menggambarkan alur komunikasi antar komponen, termasuk aplikasi mobile, API server, dan database. Desain ini menjadi kerangka dasar dalam pengembangan dan pengujian API.
3. Perancangan Struktur API dan Format Respons (JSON)
   API dirancang menggunakan dua pendekatan, yaitu RESTful dan GraphQL. Untuk memastikan kesetaraan pengujian, masing-masing API dirancang untuk menyediakan data yang sama dalam struktur JSON yang disesuaikan dengan kebutuhan aplikasi mobile.
4. Penyusunan Skenario Pengujian
   Tiga pola pengambilan data diterapkan dalam pengujian, yaitu:
   - Under-fetching: data tersebar di beberapa endpoint (REST) atau nested secara parsial (GraphQL).
   - Over-fetching: client menerima lebih banyak data dari yang diperlukan.
   - Kondisi Ideal: struktur query tepat sesuai kebutuhan.

   Setiap skenario diuji dengan variasi jumlah concurrent user.
5. Pelaksanaan Pengujian Performansi
   Pengujian dilakukan menggunakan Apache JMeter dengan plugin PerfMon untuk mencatat penggunaan CPU, memory dan elapse time saat API diakses secara bersamaan. Pengujian diulang sebanyak lima kali per skenario, dan nilai tertinggi (peak usage) dari CPU dan memori dicatat untuk dianalisis.
6. Analisis Data

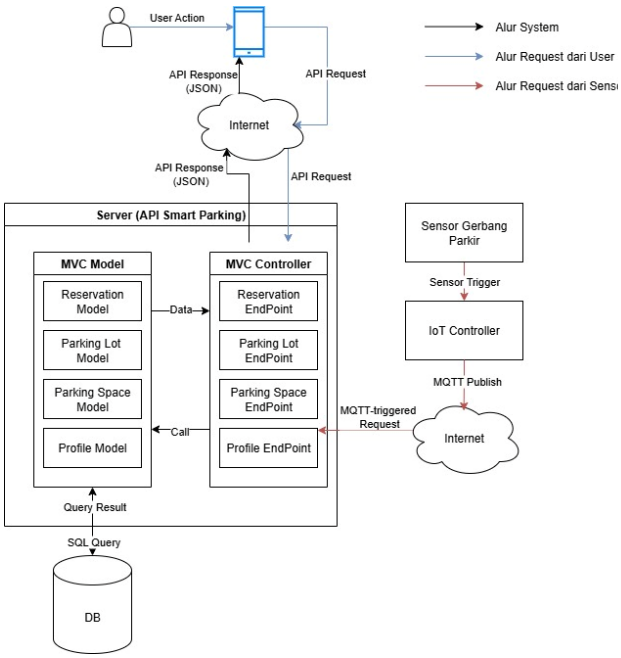
Hasil pengujian kemudian dianalisis menggunakan pendekatan statistik deskriptif. Statistik deskriptif meliputi rata-rata, standar deviasi, dan visualisasi data. Uji-t dua sampel independen digunakan untuk mengetahui apakah perbedaan performa antara RESTful dan GraphQL signifikan secara statistik.

7. Kesimpulan kinerja
   Berdasarkan hasil analisis, disimpulkan arsitektur API mana yang lebih efisien dari sisi penggunaan sumber daya (CPU, memori, dan elapse time), serta rekomendasi implementasi pada sistem parkir pintar.

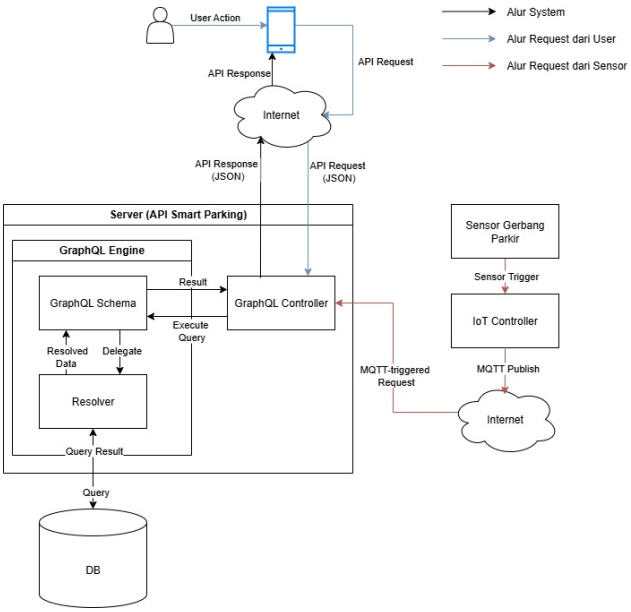
## III. HASIL DAN PEMBAHASAN

### A. *Perancangan Arsitektur Logis Sistem*

Penelitian dalam tesis ini difokuskan pada aspek komunikasi antara aplikasi *mobile* dengan *server* sebagai bagian dari sistem parkir pintar. Penggunaan aplikasi *mobile* dinilai penting karena memungkinkan pengguna untuk mengakses sistem dari mana saja, serta mendapatkan informasi secara *real-time* mengenai ketersediaan dan status pemesanan tempat parkir. Oleh karena itu, komunikasi antara aplikasi dan server database dalam sistem ini dilakukan melalui API. Gambar 2 dan 3 menggambarkan arsitektur logis sistem parkir pintar yang digunakan dalam penelitian ini, masing-masing untuk pendekatan RESTful dan GraphQL.




Gambar. 2 arsitektur logis RESTful API dalam sistem parkir pintar




Gambar. 3 Arsitektur logis GraphQL API dalam sistem parkir pintar

### B. *Arsitektur API dan Mode Pengembangan*

Sistem *backend* dikembangkan menggunakan framework Ruby on Rails. RoR dipilih karena mendukung pengembangan API yang cepat dan terstruktur melalui prinsip convention over configuration, serta memiliki dukungan native terhadap RESTful dan integrasi GraphQL melalui library bawaan.

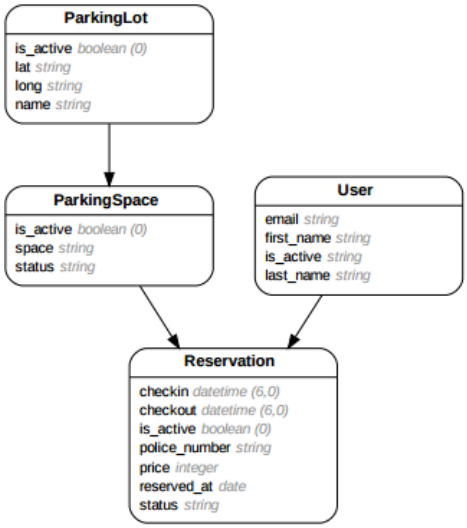
Pada penelitian ini RoR dibangun dengan konfigurasi --api, yang dirancang khusus untuk aplikasi API-only. Dengan mode ini, Rails hanya memuat middleware yang diperlukan untuk komunikasi API, dan menonaktifkan fitur-fitur seperti *cookies*, *view rendering*, dan *asset pipeline*.

Selain lebih ringan, pendekatan ini juga membuat struktur aplikasi lebih modular dan fokus pada penyediaan data dalam format JSON. Semua controller mewarisi dari ActionController::API dan tidak menghasilkan tampilan HTML.

Untuk RESTful API, digunakan konvensi controller dan routing standar. Sementara itu, implementasi GraphQL menggunakan gem graphql versi 2.4.7, yang menyediakan infrastruktur untuk schema, query, mutation, dan resolver.

### C. *ERD*

Database yang digunakan adalah MySQL. Perancangan ERD didasarkan pada simulasi kebutuhan minimum dari aplikasi mobile sistem parkir pintar. ERD yang dirancang mewakili permintaan data yang secara logis diperlukan oleh pengguna, seperti melihat daftar tempat parkir, slot parkir, reservasi, dan melihat profil.




Gambar. 4 ERD

Dengan demikian, table yang akan digunakan pada aplikasi ini terdiri dari empat table, yaitu:

- Tabel User: Menyimpan data dari pengguna yang mendaftar pada aplikasi.
- Tabel Parking Lot: Menyimpan data dari tempat parkir yang dapat dipesan oleh pengguna. Table ini mencakup column latitude dan longitude.
- Tabel Parking Space: Menyimpan data dari slot tempat parkir yang dimiliki oleh sebuah tempat parkir. Table ini mencakup column status, yang berfungsi untuk menandakan apakah slot ini sudah terisi atau belum.
- Tabel Reservation: Menyimpan data dari kegiatan reservasi yang dilakukan pengguna, baik itu reservasi yang sedang berjalan, maupun yang sudah tidak berjalan. Tabel ini memiliki column status, yang berfungsi untuk menandakan apakah reservasi tersebut sedang berjalan, sudah selesai atau dibatalkan oleh pengguna.

Perancangan database dengan menggunakan ERD dapat dilihat pada Gambar 4.

### *D. RESTful API*

RESTful API pada sistem ini dirancang untuk mendukung proses pertukaran data antara aplikasi mobile dan server berdasarkan struktur relasi antar entitas yang telah ditentukan sebelumnya. Implementasi REST dilakukan dengan pendekatan berbasis resource, di mana setiap entitas utama seperti User, ParkingLot, ParkingSpace, dan Reservation direpresentasikan sebagai endpoint tersendiri. Perancangan API dapat dilihat pada TABEL I.

TABEL I
PERANCANGAN API

| Code | Method | URL |
|---|---|---|
| EP01 | GET | /users/:id |
| EP02 | GET | /parking_lots?page={page} |
| EP03 | GET | /parking_lots/:id |
| EP04 | GET | /parking_spaces/:id |
| EP05 | GET | /parking_spaces/status?parking_lot_id={id} |
| EP06 | GET | /parking_spaces/available?parking_lot_id={id} |
| EP07 | GET | /reservations/reserved?user_id={id} |
| EP08 | GET | /reservations/history?user_id={id}&page={page} |

Perancangan *response* dari API pada TABLE 1 dapat dilihat pada TABEL II.

TABEL II
PERANCANGAN JSON API

| Code | Response |
|---|---|
| EP01 | {<br>"id": 1,<br>"email": "ferdian.chendra@wgs.co.id",<br>"first_name": "Ferdian",<br>"last_name": "Chendra",<br>"is_active": "1",<br>"created_at": "2024-11-28T08:50:04.599Z",<br>"updated_at": "2024-11-28T08:50:04.599Z"<br>} |
| EP02 | [<br>{<br>"id": 27,<br>"name": "Tempat Parkir PVJ",<br>"lat": "-6.887253124504945",<br>"long": "107.59574959220828",<br>"is_active": true,<br>"created_at": "2024-10-31T02:47:08.735Z",<br>"updated_at": "2024-10-31T02:47:08.735Z"<br>},<br>// Data dipersingkat: total 10 item per halaman<br>{<br>"id": 29,<br>"name": "Tempat Parkir Riau Junction",<br>"lat": "-6.906326083750841",<br>"long": "107.61239145520965",<br>"is_active": true,<br>"created_at": "2024-10-31T02:47:08.801Z",<br>"updated_at": "2024-10-31T02:47:08.801Z"<br>}<br>] |
| EP03 | {<br>"id": 27,<br>"name": "Tempat Parkir PVJ",<br>"lat": "-6.887253124504945",<br>"long": "107.59574959220828",<br>"is_active": true,<br>"created_at": "2024-10-31T02:47:08.735Z",<br>"updated_at": "2024-10-31T02:47:08.735Z"<br>} |
| EP04 | {<br>"id": 1,<br>"parking_lot_id": 27,<br>"space": "1",<br>"status": "Full",<br>"is_active": true,<br>"created_at": "2024-10-31T02:47:09.038Z",<br>"updated_at": "2024-11-19T04:12:20.562Z"<br>} |

| | |
|---|---|
| EP05 | {<br>"status": "1638/942"<br>} |
| EP06 | [<br>{<br>"id": 279,<br>"parking_lot_id": 27,<br>"space": "279",<br>"status": "Empty",<br>"is_active": true,<br>"created_at": "2024-10-31T02:47:11.993Z",<br>"updated_at": "2024-11-19T04:18:11.387Z"<br>},<br>// Data dipersingkat: total 10 item per halaman<br>{<br>"id": 298,<br>"parking_lot_id": 27,<br>"space": "298",<br>"status": "Empty",<br>"is_active": true,<br>"created_at": "2024-10-31T02:47:12.172Z",<br>"updated_at": "2024-11-19T04:18:10.136Z"<br>}<br>] |
| EP07 | [<br>{<br>"id": 12881,<br>"user_id": 1,<br>"parking_space_id": 266,<br>"police_number": "D 1721 JW",<br>"reserved_at": "2024-12-04",<br>"checkin": null,<br>"checkout": null,<br>"status": "RESERVED",<br>"price": null,<br>"is_active": true,<br>"created_at": "2024-12-04T07:05:49.726Z",<br>"updated_at": "2024-12-04T07:05:49.726Z"<br>}<br>] |
| EP08 | [<br>{<br>"id": 639,<br>"user_id": 1,<br>"parking_space_id": 114,<br>"police_number": "D 1721 JW",<br>"reserved_at": null,<br>"checkin": "2009-04-17T14:43:00.000Z",<br>"checkout": "2009-04-17T17:41:00.000Z",<br>"status": "OUT",<br>"price": null,<br>"is_active": true,<br>"created_at": "2024-11-19T03:49:56.717Z",<br>"updated_at": "2024-11-19T03:52:50.462Z",<br>"space": "114"<br>},<br>// Data dipersingkat: total 10 item per halaman<br>{<br>"id": 1716,<br>"user_id": 1,<br>"parking_space_id": 67,<br>"police_number": "D 1721 JW",<br>"reserved_at": null,<br>"checkin": "2009-04-17T21:32:00.000Z",<br>"checkout": null,<br>"status": "IN",<br>"price": null,<br>"is_active": true,<br>"created_at": "2024-11-19T03:53:44.547Z",<br>"updated_at": "2024-11-19T03:53:44.547Z",<br>"space": "67"<br>}<br>] |

### *E. GraphQL API*

Karena pada dasarnya GraphQL akan mengembalikan semua data yang diperlukan oleh aplikasi, maka perancangan pada GraphQL mencerminkan struktur database dengan bentuk seperti dibawah ini:

```
# User entity
type User {
  id: ID!
  email: String!
  firstName: String!
  lastName: String!
  is_active: String!
  createdAt: String!
  updatedAt: String!
  reservations:   [Reservation!]!       #   One-to-Many relationships to Reservation
}

# Parking Lot entity
type ParkingLot {
  id: ID!
  name: String!
  lat: String!
  long: String!
  is_active: String!
  createdAt: String!
  updatedAt: String!
  parkingSpaces:   [ParkingSpace!]!      #   One-to-Many relationships to Parking Space
}

# Parking Space entity
type ParkingSpace {
  id: ID!
  parking_lot_id: String!
  space:String!
  status: String!
  is_active: String!
  createdAt: String!
  updatedAt: String!
  parkingLot: ParkingLot!  # Many-to-One relationship to Parking Lot
}

# Reservation entity
type Reservation {
  id: ID!
  user_id: ID!
  parking_space_id: String!
  police_number: String!
  reserved_at: String!
  checkin: Date!
  checkout: Date!
  status: String!
  price: Int!
  createdAt: String!
  updatedAt: String!
  user: User!  # Many-to-One relationship to User
  parkingSpace: ParkingSpace!  # Many-to-One relationship to Parking Space
}
```

### *F. Penyusunan Skenario*

Untuk mengevaluasi performa RESTful dan GraphQL API dalam konteks sistem parkir pintar, dilakukan

pengujian berdasarkan tiga pola pengambilan data, yaitu *under-fetching, over-fetching*, dan kondisi ideal.

Pada skenario under-fetching, pengujian dilakukan dengan mensimulasikan aktivitas pengguna saat membuka halaman daftar tempat parkir pada aplikasi. Dalam halaman tersebut, sistem perlu menampilkan tiga jenis data sekaligus, yaitu:

- Nama pengguna yang sedang login
- Daftar tempat parkir, dan
- Ketersediaan tempat parkir di masing-masing lokasi.

Berdasarkan desain *endpoint* pada Tabel IV.1, maka endpoint yang digunakan dalam skenario ini adalah EP01, EP02, dan EP05. Dalam satu halaman, sistem melakukan permintaan sebanyak 12 kali, terdiri dari 1 kali pemanggilan data pengguna, 1 kali pemanggilan daftar tempat parkir, dan 10 kali permintaan status ketersediaan tempat parkir berdasarkan jumlah lokasi. Hasil JSON dari skenario ini akan berbentuk seperti:

```
{
  "data": {
    "completeDataByUser": {
      "id": "1",
      "fullName": "Ferdian Chendra",
      "email": "ferdian.chendra@wgs.co.id"
    },
    "parkingLots": {
      "collection": [
        {
          "id": "27",
          "name": "Tempat Parkir PVJ",
          "availability": "1638/942"
        },
        // Data dipersingkat: total 10 item per halaman
        {
          "id": "36",
          "name": "Tempat Parkir Miko Mall 1",
          "availability": "0/0"
        }
      ]
    }
  }
}
```

Pada skenario over-fetching, pengujian dilakukan dengan mensimulasikan aktivitas pengguna saat mengakses halaman histori reservasi. Pada halaman ini, sistem hanya perlu menampilkan beberapa informasi penting seperti waktu reservasi dan nama slot parkir. Namun, *endpoint* yang digunakan (EP08) dirancang untuk mengembalikan seluruh data dari tabel reservation beserta seluruh atribut dari tabel parking_space yang berelasi dengannya. Tabel III menampilkan perbandingan dari JSON yang diperlukan dengan JSON yang dikembalikan oleh EP08.

TABEL III
STRUKTUR JSON YANG DIPERLUKAN VS RESPON API ENDPOINT EP08

| JSON yang diperlukan | EP08 |
|---|---|
| <pre>{<br>"id": "639",<br>"reservedAt": null,<br>"checkin": "2009-04-17",<br>"checkout": "2009-04-17",<br>"parkingSpace": {<br>"space": "114"<br>},<br>"price": null<br>}</pre> | <pre>{<br>"id": 639,<br>"user_id": 1,<br>"parking_space_id": 114,<br>"police_number": "D 1721 JW",<br>"reserved_at": null,<br>"checkin": "2009-04-17T14:43:00.000Z",<br>"checkout": "2009-04-17T17:41:00.000Z",<br>"status": "OUT",<br>"price": null,<br>"is_active": true,<br>"created_at": "2024-11-19T03:49:56.717Z",<br>"updated_at": "2024-11-19T03:52:50.462Z",<br>"space": "114"<br>}</pre> |

Pada skenario kondisi ideal, endpoint EP01 akan digunakan. EP01 secara langsung mengembalikan data sesuai kebutuhan tampilan tanpa kelebihan atau kekurangan atribut.

### *G. Skenario Konkurensi Pengguna*

Penentuan jumlah *concurrent user* dalam pengujian didasarkan pada data historis penggunaan tempat parker dari dataset Mall Paris Van Java (PVJ). Data diolah dengan menggunakan query dan didapatkan pada dua waktu tertentu, 18 April 2009 pukul 19:15:00 dan pukul 16:36:00, terjadi aktivitas reservasi terbanyak, yaitu sebanyak 19 transaksi yang dilakukan secara bersamaan, data ini digabungkan dengan data dari Badan Pusat Statistik Kota Bandung (2019) pada tabel "Jumlah Pasar Modern", diketahui bahwa terdapat 19 mall besar di kota Bandung, sehingga total maksimum concurrent user yang akan disimulasikan dalam pengujian adalah 19 user × 19 mall = 361 *concurrent user*. Pendekatan *doubling* digunakan untuk jumlah user, sehingga *concurrent user* yang akan digunakan dapat dilihat pada TABEL IV.

TABEL IV
JUMLAH CONCURRENT USER

| **Jumlah *Concurrency User*** |
|---|
| 45 |
| 90 |
| 180 |
| 361 |

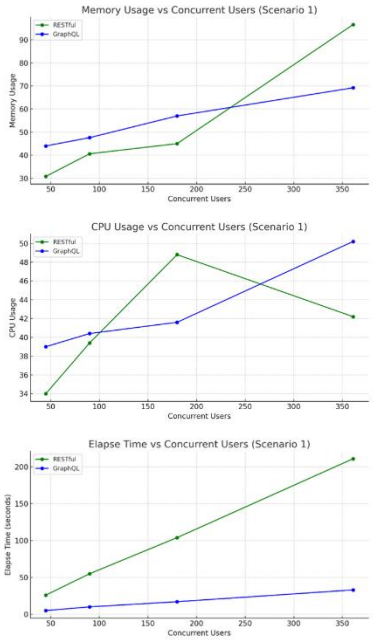
*H. Pelaksanaan Pengujian Performansi*

Spesifikasi komputer yang digunakan untuk pengujian adalah:

1. CPU: Inter Core i3-1005G1 CPU@1.20GHz
2. Memory: 6GiB
3. Web Server menggunakan Puma dengan 10 threads

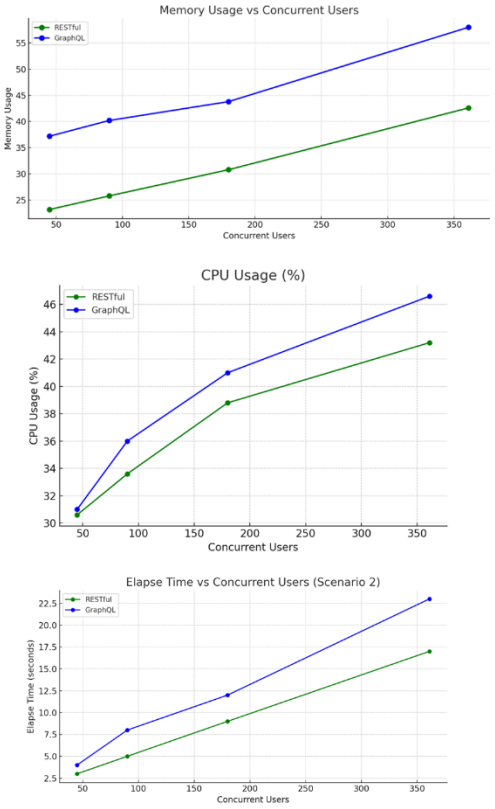
Setiap kali pengujian dilakukan, web server akan dimulai ulang (restart) untuk memastikan tidak ada cache atau data sementara yang tersimpan dari pengujian sebelumnya.

Hasil percobaan dari skenario 1 dapat dilihat pada Gambar 5.



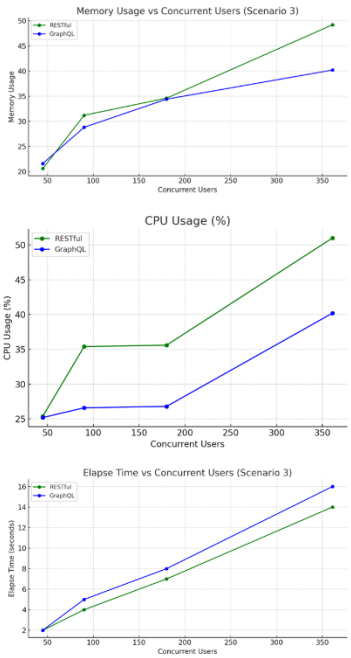

Gambar. 5 Grafik Hasil Skenario 1

Hasil percobaan dari skenario 2 dapat dilihat pada Gambar 6.




Gambar. 6 Grafik Hasil Skenario 2

Hasil percobaan dari skenario 3 dapat dilihat pada Gambar 7.




Gambar. 7 Grafik Hasil Skenario 3

Hasil tersebut lalu dilakukan pengujian uji-t, hasil dari pengujian dapat dilihat pada TABEL V, TABEL VI dan

TABEL VII yang berturut-turut mejabarkan hasil pengujian pada skenario 1, 2 dan 3.

TABEL V
HASIL UJI-T SKENARIO 1

| Metrik | Kesimpulan |
|---|---|
| **Memory** | Tidak terdapat perbedaan signifikan antara RESTful dan GraphQL pada semua jumlah user, kecuali pada 361 user, di mana penggunaan memori RESTful jauh lebih besar dan signifikan. |
| **CPU** | Penggunaan CPU cenderung tidak berbeda signifikan di semua level user, kecuali pada 361 user, di mana GraphQL menggunakan CPU lebih tinggi secara signifikan. |
| **Elapse Time** | GraphQL secara signifikan lebih cepat dari RESTful pada semua level concurrent user. Perbedaan waktu eksekusi sangat signifikan. |

TABEL VI
HASIL UJI-T SKENARIO 2

| Metrik | Kesimpulan |
|---|---|
| **Memory** | Tidak ada perbedaan signifikan pada semua jumlah user. RESTful dan GraphQL memiliki performa Memory yang sebanding. |
| **CPU** | Tidak ada perbedaan signifikan pada semua jumlah user. RESTful dan GraphQL memiliki performa CPU yang sebanding. |
| **Elapse Time** | GraphQL cenderung lebih lambat, namun hanya pada user 90, 180, dan 361 perbedaannya signifikan. RESTful lebih cepat. |

TABEL VII
HASIL UJI-T SKENARIO 1

| Metrik | Kesimpulan |
|---|---|
| **Memory** | Tidak ada perbedaan signifikan pada seluruh jumlah user antara RESTful dan GraphQL. |
| **CPU** | Terdapat perbedaan signifikan pada 90, 180 dan 361 user, di mana RESTful cenderung menggunakan CPU lebih tinggi. |
| **Elapse Time** | Pada 180 dan 361 user perbedaan waktu eksekusi signifikan (RESTful lebih cepat). Di jumlah user lain, perbedaan tidak signifikan. |

## IV. SIMPULAN

Penelitian ini telah berhasil membandingkan performa GraphQL dan RESTful API dari perspektif penggunaan sumber daya pada aplikasi parkir pintar. Berdasarkan analisis hasil pengujian yang komprehensif, berikut adalah kesimpulan yang menjawab rumusan masalah yang diajukan:

1. Peningkatan jumlah *concurrent user* secara langsung dan signifikan memengaruhi penggunaan sumber daya (CPU dan memori) serta waktu eksekusi (*elapse time*) pada kedua arsitektur API (RESTful dan GraphQL).
2. Menjawab Rumusan Masalah Q2: Pengujian pada tiga skenario (under-fetching, over-fetching, dan kondisi ideal) menunjukkan bahwa performa GraphQL dan RESTful API bervariasi tergantung pola permintaan data. Pada skenario *under-fetching*, GraphQL lebih unggul karena mampu mengambil seluruh data dalam satu permintaan, menghasilkan *elapse time* yang lebih cepat, meskipun penggunaan CPU meningkat. Sebaliknya, pada skenario *over-fetching* dan kondisi ideal, RESTful menunjukkan efisiensi lebih baik dalam *elapse time*, terutama karena desain endpoint yang sesuai dan query yang tidak kompleks. Hasil uji-t statistik menunjukkan perbedaan signifikan pada hampir seluruh parameter, sehingga pemilihan arsitektur API dapat disesuaikan dengan kebutuhan: GraphQL untuk fleksibilitas dan efisiensi CPU, RESTful untuk waktu respons yang cepat.
3. Menjawab Rumusan Masalah Q3: Meskipun RESTful cenderung kurang efisien pada skenario *under-fetching*, kondisi ini dapat dihindari jika developer merancang *endpoint* dengan baik. Sebaliknya, GraphQL lebih cocok untuk aplikasi dengan kebutuhan data yang fleksibel. Karena sistem parkir pintar memiliki struktur data yang relatif tetap, RESTful menjadi pilihan yang lebih efisien selama *under-fetching* dapat diminimalkan melalui desain endpoint yang tepat.

## REFERENSI

[1] E. Polycarpou, L. Lambrinos, and E. Protopapadakis, “Smart parking solutions for urban areas,” in 2013 IEEE 14th International Symposium on A World of Wireless, Mobile and Multimedia Networks (WoWMoM), Madrid: IEEE, Jun. 2013, pp. 1–6, doi: 10.1109/WoWMoM.2013.6583499.

[2] T. Schellenbach, “How APIs Are Powering Smart Cities,” [Online]. Available: https://www.forbes.com/sites/forbestechcouncil/2021/05/28/how-apis-are-powering-smart-cities/?sh=21fe76a22fdb. [Diakses: 12 Okt. 2022].

[3] Postman, 2023 State of the API Report, 2023.

[4] A. Tasyah, P. A. Lestari, A. Syofira, C. A. Rahmayani, R. D. Cahyani, and N. Tresiana, “Inovasi Pelayanan Publik Berbasis Digital (E-Government) di Era Pandemi Covid-19,” JIA, vol. 18, no. 2, pp. 212–224, Des. 2021, doi: 10.31113/jia.v18i2.808.

[5] D. Dwi Maharani and D. Ghulam Manar, “Penerapan E-Planning sebagai bentuk Sistem Perencanaan Daerah Berbasis Teknologi di Kabupaten Demak,” 2021.

[6] D. D. Nocera, C. D. Napoli, and S. Rossi, “A Social-Aware Smart Parking Application,” 2014, doi: 10.13140/2.1.4114.3040..

[7] C. Ashhwath, V. Rohitram, and G. Sumathi, “Smart Parking System using MQTT Communication Protocol and IBM Cloud,” J. Phys.: Conf. Ser., vol. 2115, no. 1, p. 012013, Nov. 2021, doi: 10.1088/1742-6596/2115/1/012013.

[8] A. Sularsa, A. Tri Laksono, and Periyadi, “Parkir Pintar Sistem Nirkabel,” 2021.

[9] A. Somani, S. Periwal, K. Patel, and P. Gaikwad, "Cross Platform Smart Reservation Based Parking System," in 2018 Int. Conf. Smart City and Emerging Technology (ICSCET), Mumbai: IEEE, Jan. 2018, pp. 1–5, doi: 10.1109/ICSCET.2018.8537282.

[10] M. A. Dzulqarnain, M. L. Aziz, M. F. Rachman, and A. R. Atmadja, "Aplikasi Pencarian Parkir Jakarta Berbasis Android Menggunakan RESTful API," 2019.

[11] U. Lestari, E. Fatkhiyah, and M. R. Rinaldi, "Sistem Pemesanan Parkir Berbasis Mobile," SMARTICS, vol. 5, no. 2, pp. 75–80, Okt. 2019, doi: 10.21067/smartics.v5i2.3741.

[12] S. Nijim and B. Pagano, APIs For Dummies, Apigee Special Edition, 2014.

[13] D. A. Hartina, A. Lawi, and B. L. E. Panggabean, "Performance Analysis of GraphQL and RESTful in SIM LP2M of the Hasanuddin University," in 2018 2nd East Indonesia Conf. on Computer and Information Technology (EIConCIT), Makassar: IEEE, Nov. 2018, pp. 237–240, doi: 10.1109/EIConCIT.2018.8878524.

[14] G. Brito and M. T. Valente, "REST vs GraphQL: A Controlled Experiment," in 2020 IEEE Int. Conf. Software Architecture (ICSA), Salvador: IEEE, Mar. 2020, pp. 81–91, doi: 10.1109/ICSA47634.2020.00016.

[15] B. Cooksey, "An Introduction to APIs," Zapier Inc., Apr. 23, 2014.

[16] S. Buna, GraphQL in Action, 2021.

[17] G. Brito, T. Mombach, and M. T. Valente, "Migrating to GraphQL: A Practical Assessment," in 2019 IEEE 26th Int. Conf. Software Analysis, Evolution and Reengineering (SANER), Hangzhou: IEEE, Feb. 2019, pp. 140–150, doi: 10.1109/SANER.2019.8667986.

[18] E. Lee, K. Kwon, and J. Yun, "Performance Measurement of GraphQL API in Home ESS Data Server," in 2020 Int. Conf. Information and Communication Technology Convergence (ICTC), Jeju: IEEE, Oct. 2020, pp. 1929–1931, doi: 10.1109/ICTC49870.2020.9289569.

[19] D. K. Abdullah et al., Metodologi Penelitian Kuantitatif, Yayasan Penerbit Muhammad Zaini, Jul. 2022.

[20] M. Yulianto, "Evaluasi Kapasitas Parkir di Mall Paris Van Java Bandung," 2013.

[21] Ruby on Rails Guides, "Using Rails for API-only Applications." [Online]. Available: https://guides.rubyonrails.org/api_app.html. [Diakses: 22 Jan. 2024].

[22] Badan Pusat Statistik Kota Bandung, "Jumlah Pasar Modern - Tabel Statistik." [Online]. Available: https://bandungkota.bps.go.id/id/statistics-table/2/MjU3IzI=/jumlah-pasar-modern.html. [Diakses: 22 Apr. 2025].

[23] M. Atli and E. Karakoc, "Performance Comparison of RESTful and GraphQL APIs," in 2019 1st Int. Informatics and Software Engineering Conf. (UBMYK), Ankara: IEEE, 2019, pp. 1–5, doi: 10.1109/UBMYK48245.2019.8965462.

[24] N. Sieck, M. Almalag, and C. Calpin, "Machine Vision Smart Parking Using Internet of Things (IoTs) in a Smart University," in PerCom Workshops, Austin: IEEE, 2020, pp. 1–6, doi: 10.1109/PerComWorkshop488775.2020.9156121.

[25] DB-Engines, "DB-Engines Ranking - Popularity Ranking of Database Management Systems." [Online]. Available: https://db-engines.com/en/ranking. [Diakses: 17 Jul. 2025].

[26] Zion Market Research, "Parking Reservation System Market Size, Share, Growth Analysis 2032," Zion Market Research, 2024. [Online]. Available: https://www.zionmarketresearch.com/report/parking-reservation-system-market. [Accessed: Jul. 19, 2025]..