BAB II
LANDASAN TEORI

2.1 Tinjauan Pustaka

Dalam penelitian ini, akan digunakan tinjauan studi yang nantinya dapat mendukung penelitian, daftar literatur yang diambil adalah sebagai berikut:

Tabel 2.1-1 Tinjauan Pustaka

No Literatur	Penulis	Tahun	Judul
Literatur 1	Gleison Brito, Marco Tulio Valente	2020	REST vs GraphQL: A Controller Experiment
Literatur 2	Camille Oggier	2020	How Fast GraphQL is Compared to REST APIs
Literatur 3	Armin Lawi, Benny L. E. Panggabean, Takaichi Yoshida	2021	Evaluating GraphQL and REST API Services Performance in a Massive and Intensive Accessible Information System
Literatur 4	Tobias Andersson, Hakan Reinholdsson	2021	REST API vs GraphQL – A Literature and Experimental Study
Literatur 5	Sri Lakshmi Vadlamani, Benjamin Emdon, Joshua Arts, Olga Baysal	2021	Can GraphQL Replace REST? A Study of Their Efficiency and Viability

Literatur 6	Mr.Sayan Guha, Mrs.Shreyasi Majumder	2020	A Comparative Study Between GraphQL & Restful Services in API Management of Stateless Architectures
Literatur 7	Gede Susrama Mas Diyasa, Gideon Setya Budiwitjaksono, Hafidz Amarul Ma'arufi	2020	Comparative Analysis of Rest and GraphQL Technology on Nodejs-Based Api Development
Literatur 8	Riku Ala-Laurinaho, Joel Mattila, Juuso Autiosalo, Jani Hietala, Heikki Laaki, Kari Tammi	2022	Comparison of REST and GraphQL Interfaces for OPC UA
Literatur 9	Gleison Brito, Thais Mombach, Marco Tulio Valente	2019	Migrating to GraphQL: A Practical Assessment
Literatur 10	Elias Frigard	2022	GraphQL vs REST – A Comparison of Runtime Performance

2.1.1 Tinjauan Pada Pustaka 1

Oleh Gleison Brito, Marco Tulio Valente (2020) dari Department of Computer Science, Federal University of Minas Gerais dengan judul REST vs GRAPHQL: A Controlled Experiment. Dimana dalam penelitian tersebut melakukan eksperimen yang bertujuan untuk melihat perbedaan antara model arsitektur berbasis *REST* dan *GRAPHQL* dalam mencari model arsitektur yang lebih efisien untuk diimplementasikan. Untuk menjawab berapa banyak waktu yang dibutuhkan developer untuk menerapkan kueri dalam REST dan GRAPHQL dan persepsi dari partisipan tentang REST dan GRAPHQL dengan menyontohkan kepada dua orang mahasiswa yang sudah lulus dan pembelajaran terkontrol kepada 22 subjek yang terdiri dari 10 mahasiswa dan 12 mahasiswa yang sudah lulus. Hasil yang didapatkan dari penelitian ini adalah GRAPHQL lebih efisien dalam hal pengembangan dibandingkan dengan REST, dengan kueri yang membutuhkan parameter dapat dibuat lebih cepat jika menggunakan GRAPHQL dan bahkan waktu pengembangan lebih sedikit walaupun oleh developer yang sebelumnya tidak memiliki pengalaman dengan GRAPHQL.

2.1.2 Tinjauan Pada Pustaka 2

Oleh Camille Oggier (2020) dari Haaga-Helia University of Applied Science dengan judul How fast GraphQL is compared to REST APIs. Dimana dalam penelitian tersebut penulis melakukan komparasi antara REST dan GraphQL yang mengukur keuntungan dari penggunaan GraphQL dibandingkan teknologi pendahulunya yaitu REST. Untuk mengetahui teknolgi yang lebih cepat dan lebih optimal antara REST dan GraphQL digunakan teknologi pembanding yang dibuat sendiri menggunakan JavaScript dan framework ReactJs sebagai client untuk API.

Metode yang digunakan untuk mengumpulkan hasil perbandingan adalah dengan menyalin dan menyimpan setiap metrik dari tiap hasil, membuat bagian dari bukti yang menampilkan tanggal dan waktu eksekusi percobaan dan menyimpan bagian bukti diantara hasil yang sudah disalin. Pengukuran atau percobaan dilakukan dengan beberapa variasi agar mendapat hasil yang lebih akurat. Hasil yang didapatkan dari penelitian ini adalah secara kecepatan GraphQL 35% - 46% lebih cepat dibanding REST dan secara ukuran GraphQL 21% - 71% lebih ringan dibanding dengan REST.

2.1.3 Tinjauan Pada Pustaka 3

Oleh Armin Lawi, Benny L.E. Panggabean, Takaichi Yoshida (2021) dari Universitas Hasanuddin, Universitas Universitas Pancasakti, Kyushu Institute of Technology jurusan Ilmu Komputer dengan judul Evaluating GraphQL and REST API Services Performance in a Massive and Intensive Accessible Information System. Dimana dalam penelitian tersebut penulis melakukan eksperimen dengan metode evaluasi arsitektur monolitik dari REST dan GraphQL dengan membuat service yang masing – masing servicenya dibuat dengan endpoint REST dan GraphQL. Dengan service yang telah dibuat dilakukan tes dengan melakukan request dari tiap endpoint. Hasil yang didapatkan dari dilakukannya testing terhadap endpoint REST dan GraphQL tersebut adalah REST lebih cepat sebesar 50.50% untuk response time dan 37.16% untuk throughput, sementara GraphQL lebih efisien dalam hal pemakaian sumber daya dengan CPU load sebesar 37.26% dan penggunaan memori sebesar 39.74%. Oleh karena itu, dapat disimpulkan GraphQL adalah pilihan terbaik ketika data sering berubah dan pertimbangan penggunaan

sumber daya adalah hal yang penting. REST digunakan ketika data sering diakses dan permintaan terhadap data tinggi.

2.1.4 Tinjauan Pada Pustaka 4

Oleh Tobias Andersson, Hakan Reinholdsson (2021) dari Kristianstad University jurusan Ilmu Komputer dengan judul REST API vs GraphQL A Literature and experimental study. Dimana dalam penelitian tersebut penulis melakukan studi literatur dan studi eksperimen terhadap REST API dan GraphQLuntuk menyimpulkan keuntungan ketika menggunakan REST API dan GraphQL. Dimana pada studi literatur menghasilkan kesimpulan bahwa GraphQL lebih menguntungkan dalam penggunaan kuerinya dan dapat beroperasi hanya menggunakan satu endpoint. Dalam studi literatur ini belum dapat ditentukan apakah REST akan tetap digunakan untuk membangun API di masa mendatang. Sedangkan pada studi eksperimen melakukan testing dengan membuat request pada REST API dan GraphQL dengan pembagian 100, 1000, 10000, 50000, 100000 request dan mengimplementasikan teknik caching, tingkat request dipilih untuk melihat apakah peningkatan request dapat mempengaruhi t eknik caching, karena jika hanya melakukan satu atau dua request tidak mempengaruhi cache. Perangkat yang digunakan untuk melakukan testing adalah laptop dengan spesifikasi prosesor Intel Core i5-7200 @2.50GHz, RAM 8GB dan sistem operasi Windows 10 Pro. Hasil yang didapatkan pada studi eksperimen yang dilakukan adalah jika cache dihidupkan maka jumlah request mempengaruhi response time dimana REST API lebih stabil sedangkan GraphQL semakin lama dalam memberikan response. Sedangkan jika cache dimatikan maka total response time pada GraphQL lebih lama dibandingkan pada REST API.

2.1.5 Tinjauan Pada Pustaka 5

Oleh Sri Lakshmi Vadlamani, Benjamin Emdon, Joshua Arts, Olga Baysal (2021) dari Carleton University jurusan Computer Science dengan judul Can GraphQL Replace REST? A Study of Their Efficiency and Viability. Dimana dalam literatur tersebut penulis membandingkan kedua arsitektur berdasarkan analisis kuantitatif dan analisis kualitatif untuk mengukur performa dari REST dan GraphQL dan mengetahui kelebihan dan kekurangannya. Pada analisis kuantitatif penulis membandingkan efisiensi API pada Github Public API yang dibuat dengan REST dan GraphQL lalu terdapat scenario testing yang dilakukan dengan request tunggal. Sedangkan pada analisis kualitatif penulis melakukan survey terhadap 38 software developer yang bekerja di Github Inc dan memiliki pengalaman kerja yang baik pada pengembangan REST API dan GraphQL. Hasil dari analisis kuantitatif yang dilakukan penulis adalah, dari beberapa scenario yang dilakukan yaitu melakukan GET dan POST dengan 50 request, hasilnya sangat mirip antara REST dan GraphQL dimana dibeberapa scenario REST lebih unggul tetapi di scenario lain GraphQL lebih unggul sehingga belum bisa dipastikan yang lebih baik antara REST dan GraphQL. Sedangkan pada analisis kualitatif, hasil yang didapatkan adalah, berdasarkan survey yang berfokus pada permintaan sumber daya, pemeliharaan, dan potensi keuntungan terhadap organisasi yang mengadopsi GraphQL dari 38 software developer, tidak mendapatkan pemenang yang jelas antara REST API dan GraphQL dikarenakan masing – masing memiliki kelemahan dan kelebihannya.

2.1.6 Tinjauan Pada Pustaka 6

Oleh Mr.Sayan Guha, Mrs.Shreyasi Majumder (2020) publikasi mandiri dari India dengan judul A Comparative Study between GraphQL & RESTful Services In API Management of Stateless Architectures. Dimana dalam literatur tersebut penulis melakukan eksperimen dengan melakukan komparasi pada REST API dan GraphQL dengan arsitektur stateless atau data yang didapatkan tidak disimpan ataupun diolah. Data yang digunakan sebagai eksperimen adalah public API postingan pada social media. Komparasi ini dilakukan untuk mengevaluasi performa dari REST dan GraphQL ketika melakukan fetching data atau pengambilan data, pengambilan data dilakukan dengan perulangan dan volume datanya dimulai dari 1000, 10000 dan 100000 data. Hasil yang didapatkan pada volume data 1000 adalah tidak ada perbedaan yang signifikan pada waktu respon dari REST dan GraphQL, sedangkan pada volume data 10000, waktu respon dari GraphQL lebih cepat sebanyak 35% dibandingkan pada REST, dan pada volume data 100000, waktu respon dari GraphQL lebih cepat sebanyak 40% dibandingkan pada REST. Kesimpulan yang didapatkan oleh penulis setelah melakukan eksperimen adalah GraphQL lebih disarankan sebagai API yang digunakan ketika performa dan waktu respon diutamakan.

2.1.7 Tinjauan Pada Pustaka 7

Oleh Gede Susrama Mas Diyasa, Hafidz Amarul Ma’rufi, Ilham Ade Widya Sampurno dari Universitas Pembangunan Nasional jurusan Informatika, Gideon Setya Budiwitjaksono dari Universitas Pembangunan Nasional jurusan Akuntansi (2020) dengan judul Comparative Analysis of Rest and GraphQL Technology on Nodejs-Based Api Development. Dimana dalam literatur tersebut penulis

melakukan komparasi REST dan GraphQL pada pengembangan API berbasis NodeJS. Penulis mengembangkan web service CRUD aplikasi ToDo List sebagai kandidat service untuk melakukan testing dimana endpoint dibuat dengan REST dan GraphQL. Testing dilakukan dengan melakukan pengulangan request sebanyak 100 request menggunakan Postman dengan total data adalah 1000 data. Hasil dari testing yang telah dilakukan adalah, ketika dilakukan request data, REST API bekerja lebih cepat memberikan respon dibandingkan GraphQL dengan rata – rata waktu respon sebesar 46.5ms pada REST dan 49ms pada GraphQL. Kesimpulan yang didapatkan oleh penulis adalah, REST API memiliki performa yang lebih baik dalam hal kecepatan dibandingkan dengan GraphQL, walaupun GraphQL lebih baik dalam hal memberikan data karena client bisa mengambil data sesuai keinginan sendiri sehingga mengurangi penggunaan bandwidth.

2.1.8 Tinjauan Pada Pustaka 8

Oleh Riku Ala-Laurinaho, Joel Mattila, Juuso Autiosalo, Jani Hietala, Heikki Laaki, Kari Tammi (2022) dari Aalto University jurusan Mechanical Engineering dan VIT Technical Research Centre of Finland Ltd dengan judul Comparison of REST and GraphQL Interfaces for OPC UA. Dimana dalam literatur tersebut penulis melakukan komparasi terhadap REST API dan GraphQL yang berfokus pada waktu respon dari service sederhana OPC UA. Menggunakan laptop sebagai client dan bantuan WireShark untuk merekam network traffic, laptop kedua digunakan sebagai hosting untuk REST dan GraphQL dengan spesifikasi laptop prosesor Intel i5 dan RAM 16GB. Eksperimen dilakukan dengan melakukan pembacaan data sebanyak 1 kali dan menulis data sebanyak 1 kali, dilanjutkan dengan pembacaan data sebanyak 50 kali dan menulis data sebanyak 50 kali. Proses

pembacaan data melibatkan teknologi caching. Hasil dari eksperimen yang dilakukan oleh penulis adalah, pada pembacaan dan penulisan sebanyak 1 kali, REST lebih cepat dibandingkan dengan GraphQL sedangkan pada pembacaan dan penulisan sebanyak 50 kali, GraphQL mengungguli REST dalam hal kecepatan.

2.1.9 Tinjauan Pada Pustaka 9

Oleh Gleison Brito, Thais Mombach, Marco Tulio Valente (2019) dari University of Minas Gerais jurusan Computer Science dengan judul Migrating to GraphQL: A Practical Assessment. Dimana pada literatur tersebut penulis melakukan migrasi atau perubahan arsitektur yang sebelumnya dibuat dalam REST diubah menjadi GraphQL dan melakukan komparasi ukuran dokumen JSON yang didapatkan ketika masih menggunakan REST dan ketika sudah diubah menjadi GraphQL. Ekperimen dilakukan setelah melakukan seleksi pada service yang akan diubah arsitekturnya dari REST menjadi GraphQL, selanjutnya pengujian dilakukan dengan melakukan request pada endpoint REST dan endpoint GraphQL. Hasil yang didapatkan setelah melakukan pengujian adalah dokumen JSON yang diberikan oleh GraphQL jauh lebih ringan dibandingkan yang diberikan oleh REST dimana terdapat dokumen JSON yang ketika diberikan oleh REST berukuran 400MB dan dengan dokumen JSON yang sama tetapi diberikan oleh GraphQL hanya berukuran 77KB.

2.1.10 Tinjauan Pada Pustaka 10

Oleh Elias Frigard (2022) dari Linnaeus University jurusan Computer Science dengan judul GraphQL vs REST - A Comparison of Runtime Performance. Dimana pada literatur tersebut penulis melakukan komparasi performa pada REST

dan GraphQL dengan melakukan studi eksperimen. Acuan yang digunakan oleh penulis untuk melakukan komparasi adalah konsumsi memori, penggunaan CPU dan waktu respon oleh API ke client. Dengan menggunakan public API dari Github yang bisa digunakan untuk mengambil data banyaknya repository, profil, dll. Penulis membuat API dengan REST dan GraphQL untuk mengambil data dari public Github API. Setelah mengembangkan API untuk mengambil data, selanjutnya API di deploy sebuah Cloud Server dengan spesifikasi CPU 1 core, Memory 2GB dan Operating System Ubuntu Live Server 20.04. Pengujian dilakukan dengan melakukan eksekusi endpoint yang dilakuka berulang secara dan penambahan pengulangan secara bertahap. Hasil yang didapatkan setelah dilakukannya pengujian adalah, dalam kecepatan waktu respon, perbedaan antara REST dan GraphQL tidak terlalu signifikan dengan detail REST lebih cepat sebesar 58.33% pada beberapa tes, dan jika semua tes dirata-rata maka menghasilkan REST lebih lambat sebesar 10.6% dibandingkan GraphQL. Sedangkan pada uji penggunaan CPU dan Memory menunjukkan bahwa GraphQL menggunakan sumber daya lebih besar sebanyak 5.3% dibandingkan dengan REST.

2.2 Keaslian Penelitian

Berdasarkan beberapa penelitian dari literatur diatas, adapun yang menjadi pembeda antara penelitian yang dilakukan penulis dan penelitian yang telah dilakukan sebelumnya adalah:

1. Penelitian yang dilakukan oleh penulis dibatasi hanya dengan studi eksperimental dengan penelitian kuantitatif terhadap dua metode komunikasi API berbasis REST dan GraphQL.

2. Pengembangan web service yang akan digunakan sebagai bahan uji menggunakan framework Java Springboot dan implementasinya menggunakan service dari aplikasi edukasi pernikahan bernama HaFy.
3. Penelitian menggunakan perangkat yang berbeda antara Server dan Client.
4. Penelitian yang dilakukan penulis menggunakan acuan berupa Response Time, Throughput dan Error.
5. Pengujian dilakukan menggunakan metode Load Testing/Stress Testing dengan aplikasi Apache JMeter.

2.3 Studi Eksperimen

Studi Eksperimental adalah rekaman pengamatan, kuantitatif atau kualitatif, dibuat dengan operasi yang didefinisikan dan direkam dalam kondisi yang sudah ditentukan, diikuti dengan pemeriksaan data, dengan aturan statistik dan matematika yang sesuai, untuk keberadaan hubungan yang signifikan. (Philip Cash, 2016). Penelitian eksperimen adalah metode penelitian yang melibatkan variabel di mana data awalnya belum ada, sehingga memerlukan manipulasi dengan memberikan treatment khusus kepada subjek penelitian. Dampak dari treatment tersebut kemudian diamati atau diukur untuk data yang akan tercipta. Penelitian eksperimen dilakukan dengan sengaja oleh peneliti, yang memberikan treatment tertentu kepada subjek penelitian untuk menginduksi kejadian atau keadaan tertentu yang kemudian akan diteliti untuk melihat akibatnya.(Amat Jadun, 2011).

Beberapa karakteristik penelitian eksperimen menurut Amat Jadun (2011), Ciri khas yang membedakan penelitian positivistik ini dari yang lain adalah:

1. Eksperimen adalah satu-satunya pendekatan penelitian yang dianggap paling efektif dalam menguji hipotesis sebab-akibat atau memastikan validitas internal.
2. Eksperimen adalah rancangan penelitian yang memberikan pengujian hipotesis yang paling ketat dibandingkan dengan pendekatan lainnya.
3. Eksperimen digunakan untuk mengidentifikasi pengaruh perlakuan tertentu dalam situasi yang terkendali.

2.4 Penelitian Kuantitatif

Penelitian kuantitatif merupakan upaya yang cermat dan komprehensif dalam menginvestigasi suatu fenomena atau masalah dengan menggunakan pengukuran yang obyektif, dengan tujuan mendapatkan fakta atau kebenaran serta menguji teori-teori yang berkaitan dengan fenomena atau masalah tersebut.(Bambang Prasetyo, 2008). Metode untuk memperoleh pemahaman atau menyelesaikan masalah secara teliti dan terstruktur, dengan data yang dikumpulkan berupa serangkaian atau kumpulan angka-angka.

Penelitian kuantitatif didefinisikan sebagai pendekatan penelitian yang didasarkan pada filosofi positivisme, digunakan untuk menginvestigasi populasi atau sampel tertentu, dengan pengumpulan data menggunakan instrumen penelitian, analisis data yang bersifat kuantitatif/statistik, dan tujuan untuk menguji hipotesis yang telah dirumuskan. (Sugiyono, 2019).

Penelitian kuantitatif adalah pendekatan penelitian yang secara konsisten mengandalkan penggunaan angka, dari tahap pengumpulan data hingga penafsiran hasil, sesuai dengan sifatnya yang mengedepankan aspek kuantitatif.. (Arikunto,

2019). Jadi penelitian kuantitatif merupakan suatu metode penelitian yang meneliti populasi atau sampel yang bersifat kualitatif atau menggunakan angka.

2.5 Web Server

Web server adalah perangkat lunak yang bertugas menyediakan layanan data dengan menerima permintaan HTTP (Hypertext Transfer Protocol) atau HTTPS (HTTP Secure) dari klien melalui peramban web, kemudian mengirimkan kembali hasilnya dalam bentuk halaman web, yang umumnya terdiri dari dokumen HTML (Hypertext Markup Language). Web server berperan sebagai tempat di mana aplikasi web dijalankan dan sebagai penerima permintaan dari klien. (Indra Warman, 2013). Menurut Abdulloh (2018:4) “Web server adalah perangkat lunak yang berperan dalam menerima permintaan (request) melalui protokol HTTP atau HTTPS dari klien, lalu mengirimkannya kembali dalam bentuk halaman web.”. Web server, dalam konteks fungsinya, adalah salah satu jenis server yang bertugas menyediakan layanan protokol HTTP. Contoh aplikasi web server meliputi Apache, Microsoft IIS, Tomcat, Nginx, dan sebagainya. (Oya Suryana, 2013).

Perangkat lunak web server dikenal karena kemampuannya untuk menerima permintaan HTTP dari klien yang terhubung dalam jaringan, serta memberikan layanan kepada pengguna yang meminta informasi terkait dengan situs web, dengan menyajikan halaman web yang ditampilkan dalam peramban. Web server terdiri dari dua komponen utama: komputer dan perangkat lunak web server yang digunakan. Inilah tempat situs web ditempatkan untuk memberikan atau bertukar informasi. Aplikasi web server dapat diperoleh dengan mudah, baik secara berbayar maupun gratis. Ketika memilih perangkat lunak aplikasi web server, seorang administrator web harus mempertimbangkan server mana yang akan digunakan

untuk melayani pengguna situs web institusi mereka. Keputusan ini dibuat ketika akan melakukan pengaturan hosting server untuk situs web institusi. (Abdul Aziz&Topan Tampati, 2015).

2.6 Web Service

Web service adalah sistem perangkat lunak yang dibuat untuk mendukung interoperabilitas antarmuka antar mesin melalui jaringan. Interaksi antar mesin tersebut dilakukan menggunakan mekanisme atau protokol tertentu. Web service juga menyediakan antarmuka yang menggambarkan semua layanan yang tersedia dalam format yang dapat diproses oleh mesin, yang disebut Web Service Description Language (WSDL) (Putra, 2021). Web service adalah sistem pertukaran informasi berbasis XML yang menggunakan internet sebagai medium untuk interaksi antar aplikasi. Teknologi ini telah menjadi standar yang diadopsi oleh banyak vendor perangkat lunak karena sifatnya yang terbuka (Erl, Service oriented architecture, 2005).

Web service merupakan layanan yang tersedia di Internet yang menggunakan format standar XML untuk pengiriman pesan. Web service tidak terikat pada bahasa pemrograman atau sistem operasi tertentu (Ethan Cerami, 2002). Web service adalah antarmuka yang menggambarkan kumpulan layanan yang dapat diakses dalam jaringan menggunakan format standar XML untuk pertukaran pesan. Web service bertanggung jawab atas tugas-tugas yang spesifik dan dideskripsikan menggunakan format notasi XML standar yang disebut deskripsi layanan. (Gottschalk, 2002).

2.7 Application Programming Interface (API)

API (Application Programming Interface) adalah antarmuka yang memfasilitasi transfer data antara aplikasi satu dengan yang lain. Salah satu jenis arsitektur API yang umum digunakan adalah REST dalam Rizqi Kartika & Hanson Prihantoro (2021). API (Application Programming Interface) adalah sebuah dokumentasi yang mencakup antarmuka, fungsi, kelas, struktur, dan elemen lainnya yang diperlukan untuk membangun perangkat lunak. Keunggulan utama dari API adalah kemampuannya untuk memungkinkan interaksi dan koneksi antara aplikasi satu dengan yang lain, memfasilitasi kerja sama dan integrasi antar aplikasi (Ramadhani, 2016). API adalah antarmuka program dari sebuah sistem yang dapat diakses melalui metode dan header pada protokol HTTP yang merupakan standar.

Web API dapat diakses dari berbagai jenis klien HTTP, seperti peramban web dan perangkat mobile. Keuntungan lain dari Web API adalah penggunaan infrastruktur yang sama dengan web, terutama dalam hal penggunaan caching dan concurrency (Block, 2014). Penggunaan API bertujuan untuk memudahkan integrasi dan pengembangan selanjutnya tanpa harus mengganggu kode utama program. Dalam penerapannya, API dibangun dengan menggunakan framework React Native yang ditujukan untuk aplikasi berbasis mobile. Dengan menggunakan API, efisiensi dan efektivitas dalam pengembangan aplikasi dapat ditingkatkan, karena pengembang tidak perlu memulai dari awal dalam pembuatan fitur yang sudah ada, melainkan dapat memanfaatkan layanan yang telah tersedia. Selain itu, penggunaan API juga meningkatkan fleksibilitas dalam pengembangan selanjutnya dalam Rizqi Kartika & Hanson Prihantoro (2021).

2.8 REST

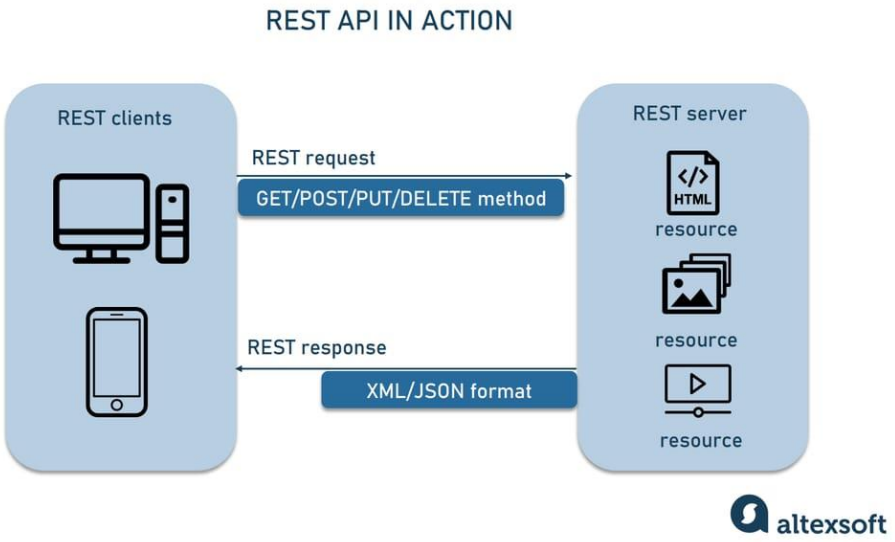
REST (Representational State Transfer) adalah kumpulan prinsip arsitektur yang digunakan untuk mentransfer data melalui antarmuka yang telah distandarisasi, seperti HTTP. (Dhingra, 2013).

REST API beroperasi mirip dengan aplikasi web konvensional. Klien dapat mengirimkan permintaan ke server melalui protokol HTTP, dan server akan memberikan respons kembali kepada klien. REST adalah sebuah filosofi desain yang mendorong penggunaan protokol dan fitur yang sudah ada di web untuk memetakan permintaan terhadap sumber daya dalam berbagai representasi dan memanipulasi data di internet (Scribner, 2009).

Representational State Transfer (REST), yang disingkat sebagai REST, merupakan salah satu jenis arsitektur untuk implementasi layanan web yang mengadopsi konsep perpindahan antara status (state). Konsep state di sini dapat diartikan sebagai permintaan dan respons antara klien dan server. Sebagai contoh, ketika peramban meminta halaman situs, server akan mengirimkan status (state) dari halaman situs saat itu kepada peramban. Navigasi melalui URL yang tersedia serupa dengan mengganti status (state) dari halaman situs. Cara kerja REST mirip dengan navigasi melalui tautan HTTP untuk melakukan berbagai aktivitas tertentu, sehingga terjadi perpindahan status antara satu ke yang lain. Metode HTTP yang umum digunakan dalam REST adalah GET, POST, PUT, atau DELETE. REST lebih umum digunakan dalam implementasi layanan web yang berfokus pada sumber daya data. Web service yang menerapkan arsitektur REST sering disebut sebagai RESTful web service (Rahman, dkk, 2013).

2.9 Arsitektur REST

Struktur komunikasi dalam REST API dapat digambarkan dalam Gambar 2.9-1 sebagai berikut ini.



Gambar 2.9-1 Arsitektur Komunikasi REST API

Untuk mengakses Resource, client mengirimkan HTTP Request. Selanjutnya, server menghasilkan HTTP Response (Altexsoft, 2022).

2.10 GraphQL

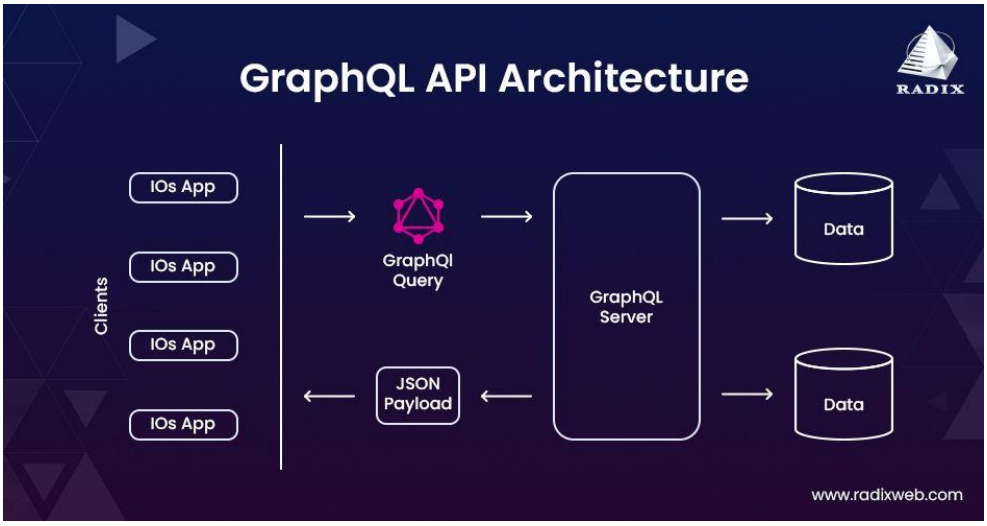
GraphQL adalah sebuah metode untuk mengakses API yang memungkinkan pengguna untuk menggambarkan data yang diinginkan sesuai dengan permintaan mereka dengan cara yang mudah dipahami (*https://graphql.org/*). GraphQL adalah sebuah bahasa yang dapat diajarkan kepada pengguna aplikasi. Aplikasi tersebut dapat berkomunikasi dengan layanan backend menggunakan GraphQL. Selain itu, GraphQL juga memungkinkan aplikasi untuk meminta data dengan mudah. GraphQL memiliki kemiripan dengan JSON dalam hal kedua-duanya memiliki operasi untuk membaca kueri dan menulis data. (dalam Arief Permana sastra, 2020). Selain itu, GraphQL memberikan fleksibilitas kepada pengelola API untuk menambahkan atau membatasi permintaan yang sudah ada. Pengembang dapat

membangun API dengan metode apa pun yang mereka inginkan, dan spesifikasi GraphQL akan memastikan bahwa API tersebut berfungsi dengan cara yang dapat diprediksi bagi klien (Antoni, Fikari & Akbar, 2018;Antoni & Akbar, 2019; Antoni, Jie & Abareshi, 2020; Antoni, Herdiansyah, Akbar & Sumitro, 2021).

GraphQL adalah sebuah bahasa kueri untuk API yang dikembangkan oleh Facebook dan digunakan dalam komunikasi antara klien dan server. Dalam GraphQL, klien hanya meminta data yang dibutuhkan, sehingga memungkinkan klien untuk menerima semua data yang dibutuhkan dalam satu kali permintaan ke server. GraphQL menyediakan runtime pada sisi server untuk mengeksekusi kueri. Setiap layanan GraphQL mendefinisikan tipe data pada GraphQL Schema. Selain itu, GraphQL memberikan deskripsi data yang lengkap dan dapat dipahami oleh API, memungkinkan klien untuk meminta hanya data yang mereka butuhkan, menghindari overfetching dan underfetching. GraphQL juga hanya memerlukan satu endpoint, yang lebih praktis dibandingkan dengan REST, dan dapat menerima respons dalam format JSON. Selain itu, GraphQL memiliki pustaka server yang tersedia dalam berbagai bahasa pemrograman, termasuk C#, Clojure, Elixir, Erlang, Go, Groovy, Java, JavaScript, .Net, PHP, Python, Scala, dan Ruby (Porcello & Banks, 2018).

2.11 Arsitektur GraphQL API

Struktur komunikasi dalam REST API dapat digambarkan dalam Gambar 2.11-1 sebagai berikut ini.



Gambar 2.11-1 Arsitektur GraphQL API

Seperti yang terlihat dalam gambar di atas, klien membuat permintaan, dan GraphQL menerima permintaan tersebut untuk diteruskan ke GraphQL Server guna diproses. Setelah proses selesai, GraphQL Server akan mengirimkan Response Data kembali kepada klien (Raval, 2023).

2.12 JSON Document

JSON (JavaScript Object Notation) adalah sebuah format pertukaran data yang ringan, mudah dibaca dan ditulis oleh manusia, serta mudah diterjemahkan dan dibuat (generate) oleh komputer. Format ini didasarkan pada bagian dari Bahasa Pemrograman JavaScript. Dokumen JSON merupakan salah satu format markup yang digunakan untuk pertukaran data, dimana JSON dibuat berdasarkan JavaScript dan memiliki sintaks yang mirip dengan JavaScript. Dengan membuat sebuah JSON, kita seolah-olah membuat sebuah objek dalam JavaScript itu sendiri (Sudirman, 2016).

Format JSON dikenal karena ringan (memiliki ukuran kecil), mudah dibaca, ditulis, dan dipahami oleh manusia, serta mudah untuk diurai dan dibuat oleh mesin. Format ini sangat cocok untuk digunakan pada layanan web REST, terutama dalam

lingkungan mobile, karena efisien dalam hal ukuran pesan dan waktu eksekusi. JSON dibuat berdasarkan bahasa pemrograman JavaScript, dengan standar ECMA-262 edisi ketiga yang diterbitkan pada Desember 1999. JSON bersifat independen dari bahasa pemrograman tetapi menggunakan aturan penulisan yang dikenal luas oleh para programmer dari keluarga bahasa C, seperti C, C++, C#, Java, JavaScript, Perl, Python, dan lainnya. Hal ini menjadikan JSON sangat ideal sebagai format dalam pertukaran data. (Sumber: Tim JSON, www.json.org) dalam Didiek&Ardhi (2012).

2.13 Postman

Postman adalah aplikasi yang berfungsi sebagai REST Client untuk menguji REST API. Dengan Postman, pengguna dapat menguji coba REST API yang telah dibuat. Aplikasi ini digunakan untuk melakukan pengecekan terhadap layanan web (web service) yang telah dibuat. Postman mampu menampilkan hasil dari permintaan HTTP yang kompleks dengan cepat dan mudah. (Dede Wintana dkk, 2022). Postman digunakan oleh developer pembuat API sebagai tools untuk menguji API yang telah dibuat dalam Kasyful Anwar&Tjahjanto (2021). Postman adalah sebuah alat atau tool yang digunakan untuk menguji endpoint atau Application Programming Interface (API). Alat ini dapat diinstal di berbagai sistem operasi (OS) atau diintegrasikan ke dalam browser sebagai Add-ons. Postman didukung oleh browser Google Chrome dan Mozilla Firefox. Fungsinya sangat penting dalam mencoba setiap endpoint atau API yang akan digunakan oleh sebuah sistem aplikasi. Postman memungkinkan pengguna untuk mendapatkan semua data yang dikembalikan oleh endpoint yang diuji, termasuk data dalam bentuk body, header, raw, dan lain-lain (Faerera Deka Sukma dkk, 2019).

2.14 Apache Jmeter

Apache JMeter adalah sebuah aplikasi perangkat lunak open source yang sepenuhnya berbasis Java, yang dirancang untuk melakukan uji beban (load testing), uji kinerja (performance testing), dan uji fungsional (functional testing). (Desy Intan Permatasari dkk, 2020). Apache JMeter pada awalnya dirancang untuk pengujian aplikasi web, tetapi sejak itu telah diperluas untuk fungsi tes lainnya. Ini adalah aplikasi open source berbasis Java yang digunakan untuk uji kinerja. Bagi seorang QA Engineer, JMeter dapat digunakan untuk melakukan uji beban (load/stress testing) pada aplikasi web, aplikasi FTP, dan pengujian server database. Apache JMeter dapat digunakan untuk menguji kinerja sumber daya baik yang statis maupun dinamis, termasuk layanan web (SOAP / REST), bahasa web dinamis seperti PHP, Java, ASP.NET, file, objek Java, pangkalan data dan kueri, server FTP, dan lainnya. Ini memungkinkan simulasi beban berat pada server, grup server, jaringan, atau objek untuk menguji kekuatan atau menganalisis kinerja secara keseluruhan di bawah berbagai jenis beban. Alat ini mengirimkan permintaan ke server target dengan mensimulasikan sekelompok pengguna, kemudian mengumpulkan data untuk menghitung statistik dan menampilkan metrik kinerja melalui berbagai format (Ni Luh Ayu Sonia dkk, 2021).