Volume 5 Nomor 3 September 2020

INFORMASI INTERAKTIF

JURNAL INFORMATIKA DAN TEKNOLOGI INFORMASI

PROGRAM STUDI TEKNIK INFORMATIKA – FAKULTAS TEKNIK -UNIVERSITAS JANABADRA

INFORMASI INTERAKTIF	Vol. 5	No. 3	Hal. 92-147	Yogyakarta September 2020	ISSN 2527-5240

INFORMASI INTERAKTIF

ISSN 2527-5240

DEWAN EDITORIAL

Penerbit : Program Studi Teknik Informatika Fakultas Teknik Universitas Janabadra

Ketua Penyunting (*Editor in Chief*) : Fatsyahrina Fitriastuti, S.Si., M.T. (Universitas Janabadra)

Penyunting (*Editor*) :
1. Prof. Selo, S.T., M.T., M.Sc., Ph.D. (Universitas Gajah Mada)
2. Dr. Kusrini, S.Kom., M.Kom. (Universitas Amikom Yogyakarta)
3. Jemmy Edwin B, S.Kom., M.Eng. (Universitas Janabadra)
4. Ryan Ari Setyawan, S.Kom., M.Eng. (Universitas Janabadra)
5. Yumarlin MZ, S.Kom., M.Pd., M.Kom. (Universitas Janabadra)

Alamat Redaksi : Program Studi Teknik Informatika Fakultas Teknik
Universitas Janabadra
Jl. Tentara Rakyat Mataram No. 55-57
Yogyakarta 55231
Telp./Fax : (0274) 543676
E-mail: informasi.interaktif@janabadra.ac.id
Website : http://e-journal.janabadra.ac.id/

Frekuensi Terbit : 3 kali setahun

JURNAL INFORMASI INTERAKTIF merupakan media komunikasi hasil penelitian, studi kasus, dan ulasan ilmiah bagi ilmuwan dan praktisi dibidang Teknik Iinformatika. Diterbitkan oleh Program Studi Teknik Informatika Fakultas Teknik Universitas Janabadra di Yogyakarta, tiga kali setahun pada bulan Januari, Mei dan September.

ISSN 2527-5240

DAFTAR ISI

ISSN 2527-5240

PENGANTAR REDAKSI

Puji syukur kami panjatkan kehadirat Allah Tuhan Yang Maha Kuasa atas terbitnya JURNAL INFORMASI INTERAKTIF Volume 5, Nomor 3, Edisi September 2020. Pada edisi kali ini memuat 8 (delapan) tulisan hasil penelitian dalam bidang teknik informatika.

Harapan kami semoga naskah yang tersaji dalam JURNAL INFORMASI INTERAKTIF edisi September tahun 2020 dapat menambah pengetahuan dan wawasan di bidangnya masing-masing dan bagi penulis, jurnal ini diharapkan menjadi salah satu wadah untuk berbagi hasil-hasil penelitian yang telah dilakukan kepada seluruh akademisi maupun masyarakat pada umumnya.

Redaksi

INTEROPERABILITAS APLIKASI BERBASIS *WEB SERVICE*

Bernard Renaldy Suteja[1], Rusdy Agustaf[2]

[1]Fakultas Teknologi Informasi, Universitas Kristen Maranatha
[2]Fakultas Teknik, Universitas Janabadra

Email : [1]myjournalid@gmail.com, [2]rustaf52@gmail.com

ABSTRACT

Today, the need to fill information as data is increasing rapidly. At the past, fill many forms in the format of papers. But now, the forms are changing to paperless format. This can be achieve by making an application capable of data input, edit the data, and post the data across all computing/technological platform available these days. The need for this application can communicate with each other. Related to this, there are many things needed, to achieve unlimited data transfer in various applications. One of the most important aspect that can make seamless communication/integration across applications is web service.

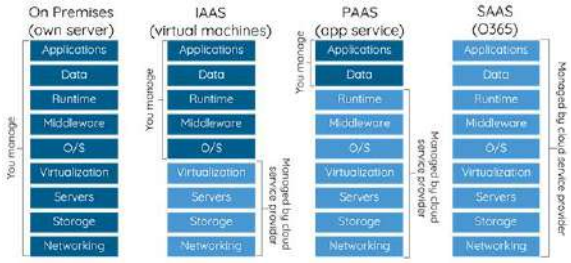
Keywords: *REST API, SOAP, Semantic*

1. PENDAHULUAN

Kebutuhan akan pengisian informasi saat ini dalam bentuk formulir untuk memperoleh data semakin meningkat. Namun untuk era teknologi seperti sekarang ini, sepertinya sudah tidak relevan lagi jika proses pengisian formulir masih dalam bentuk kertas. Tentu hal ini akan menyulitkan karena selain resiko kehilangan informasi pada formulir kertas, data yang ada tidak bisa diakses atau ditautkan ke data lain dengan cepat.

Informasi harus tersimpan di dalam aplikasi dan dapat diakses melalui mana saja atau melalui aplikasi lain. Kemudahan yang tersebut dapat dicapai dengan menggunakan layanan web (*web service*). Layanan web tersedia di komputasi awan. Komponen pembentuk dibangun dalam bentuk layanan menggunakan *Service Oriented Architecture* (SOA) dengan standar *web services*. Model dalam layanan awan (*cloud*) umumnya dibagi menjadi tiga [1]:

- IAAS (*Infrastructure as a Service*): kategori ini menyediakan layanan server virtual untuk dikelola secara mandiri termasuk jaringan dan database-nya. Contoh: AWS EC2, Rackspace, Google Compute Engine.
- Paas (*Platform as a Service*): kategori ini menyediakan layanan untuk ketersediaan kerangka kerja yang akan digunakan dalam pengembangan aplikasi. Contoh: Heroku, Windows Azure, OpenShift.
- Saas (*Software as a Service*): kategori ini menyediakan layanan untuk penyediaan aplikasi. Contoh: BigCommerce, Google Apps, Dropbox, Zendesk.



http://opentechconsultants.com/iaas-paas-saas

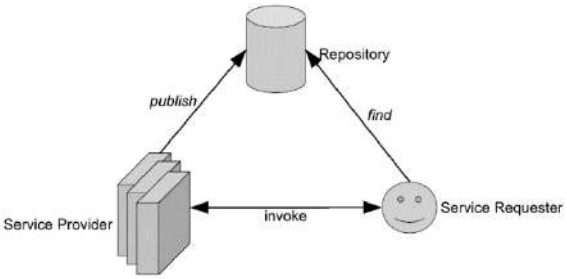
Gambar 1. Perbandingan model layanan awan

Area aplikasi yang paling banyak memanfaatkan layanan web hingga saat ini adalah *Enterprise Application Integration* (EAI) dan *e-commerce*. Pada EAI sistem yang sudah berjalan dengan melibatkan semua aplikasi yang telah dibuat dan kemampuan yang terjamin tidak perlu dibuat ulang, hanya membutuhkan integrasi yang fleksibel sehingga dapat menekan biaya implementasi ulang. Lain halnya dengan *e-commerce*. Dalam *e-commerce*, antarmuka web yang sudah ada dan familiar yang diakses oleh pengguna dapat dipertahankan hanya pada *backend* dapat diintegrasikan layanan yang dapat memberikan nilai tambah dari keberadaan *e-commerce* itu sendiri.

Service Oriented Architecture (SOA) adalah bentuk arsitektur yang mengimplementasikan konsep berorientasi layanan. Pendekatan yang

dilakukan oleh arsitektur ini merupakan pendekatan pemecahan masalah yang besar dengan membaginya menjadi beberapa layanan [2]. Layanan ini dapat berdiri sendiri atau ditautkan ke layanan lain. Arsitektur berbasis SOA memiliki tingkat fleksibilitas yang tinggi dan tidak bergantung pada teknologi tertentu untuk memungkinkan peningkatan kinerja suatu sistem. Dalam implementasi SOA, konsep berorientasi layanan diimplementasikan pada lapisan antara lapisan proses bisnis dan lapisan aplikasi, yang keduanya merupakan bagian dari logika perusahaan yang dinamai lapisan antarmuka layanan.

Teknologi komputasi awan dapat dipadukan dengan baik dengan SOA sehingga menghasilkan solusi alternatif untuk membangun sistem dan teknologi informasi di suatu perusahaan. Komputasi awan mampu memberikan dukungan kepada SOA dalam bentuk desain layanan dan kemampuan untuk memperluas layanan. Sebaliknya, SOA mampu memberikan dukungan untuk komputasi awan dalam bentuk arsitektur layanan.



Gambar 2. Service Oriented Architecture (SOA)

Pada gambar 2 terlihat bahwa SOA terdiri dari:

- *Service Provider*, berfungsi untuk memberikan layanan dan memproses resume agar layanan tersebut dapat tersedia.
- *Service Requester*, adalah pemohon layanan yang mencari dan menemukan layanan yang dibutuhkan dan menggunakan layanan tersebut.
- *Service Repository*, adalah lokasi sentral yang menjelaskan semua layanan yang telah didaftarkan.

2. WEB SERVICE

Layanan web (*web service*) dapat diartikan sebagai: suatu sistem perangkat lunak yang dirancang untuk mendukung interoperabilitas dan interaksi antar sistem dalam suatu jaringan. Layanan web memiliki antarmuka yang dijelaskan dalam format yang dapat diproses mesin (secara khusus dikenal sebagai *Web Services Description Language* atau WSDL) [3]. Sistem lain berinteraksi dengan layanan web dengan cara yang ditentukan oleh deskripsi menggunakan pesan SOAP, biasanya dikirimkan menggunakan HTTP dengan serialisasi XML dalam hubungannya dengan standar terkait web lainnya. Layanan web digunakan sebagai fasilitas yang disediakan oleh suatu website untuk memberikan layanan (berupa informasi) kepada sistem lain, sehingga sistem lain dapat berinteraksi dengan sistem melalui layanan yang disediakan oleh suatu sistem yang menyediakan layanan web. Layanan web menyimpan data informasi dalam format XML atau JSON, sehingga data ini dapat diakses oleh sistem lain meskipun berbeda platform, sistem operasi, atau bahasa pemrograman komputer.

Definisi lain: layanan web adalah aplikasi perangkat lunak yang tidak terpengaruh oleh platform tertentu, yang akan menyediakan metode yang dapat diakses oleh jaringan. Layanan web bertujuan untuk meningkatkan kolaborasi antara programmer dan perusahaan, yang memungkinkan suatu fungsi dalam layanan web dapat dipinjam oleh aplikasi lain tanpa perlu mengetahui detail pemrograman yang terdapat di dalamnya. Manfaat layanan web, antara lain: dapat digunakan sebagai alternatif dalam pengembangan aplikasi N-Tier, yang dipisahkan antara database server, aplikasi, dan client.

Beberapa keuntungan dalam mengimplementasikan layanan web adalah:

- Dengan format XML atau JSON yang telah menjadi salah satu standar pertukaran data, pengguna web service akan lebih mudah melakukan pertukaran data di berbagai sistem dengan platform yang berbeda. Saat membuat layanan web dengan teknologi Java, fungsi-fungsi yang terdapat pada layanan web tersebut dapat dibaca menggunakan sistem lain yang sama sekali berbeda dari Java, misalnya menggunakan .Net atau PHP.
- Layanan Web didukung oleh Microsoft (NET), SUN (Open Net Environment - ONE), IBM (Web Service Conceptual Architecture - WSCA), W3C (Web Service

Workshop), Oracle (Web Perantara Layanan), Hewlett-Packard (Platform Layanan Web).

- Dalam penerapan N-Tier, untuk business layer atau logika aplikasi dapat diaplikasikan dengan web service, sehingga sisi client tidak direpotkan dengan pemasangan business layer seperti cobra, atau jenis lainnya. Dengan layanan web, metode atau fungsi yang telah dibuat dapat digunakan berulang kali bahkan untuk kebutuhan aplikasi yang berbeda (fungsi yang dapat digunakan kembali). Implementasi lebih lanjut dari layanan web adalah Service Oriented Architecture (SOA) yang telah disebutkan dalam pendahuluan di atas. Dengan layanan web sebagai dasarnya.
- Layanan web dibangun berdasarkan dokumen berbasis teks dalam format XML atau JSON, sehingga komunikasi data relatif lebih ringan dibandingkan dengan aplikasi yang mengakses basis data secara langsung melalui jaringan. Saat mengimplementasikan layanan web untuk aplikasi yang menggunakan aplikasi berbasis desktop, tidak perlu menginstal konektor database seperti menggunakan ODBC, OLEDB, atau jenis penyedia data lainnya. Dengan jumlah client yang besar, akan sangat merepotkan jika harus menginstal satu per satu untuk konektor database. Dengan menggunakan web service, cukup menambahkan referensi web service pada client, sedangkan koneksi database hanya perlu dilakukan pada *web server service*.
- Komunikasi data melalui layanan web dilakukan melalui HTTP atau protokol internet terbuka lainnya, hal ini sangat mudah karena protokol tersebut merupakan protokol yang umum digunakan.

Secara umum, layanan web memiliki tiga operasi yang terlibat:

- Publish / Unpublish, publish / delete services layanan ke atau dari pendaftaran layanan.
- Find: pemohon layanan mencari dan menemukan layanan yang dibutuhkan.
- Bind: pemohon layanan setelah menemukan layanan yang dicari, kemudian mengikat ke penyedia layanan untuk berinteraksi dan mengakses layanan yang disediakan oleh penyedia layanan.

3. TEKNOLOGI WEB SERVICE

Beberapa teknologi yang dapat ditemukan di layanan web adalah: XML, SOAP, WSDL, dan UDDI. SOAP (Simple Object Access Protocol) adalah bahasa mark-up berbasis XML untuk pengalihan pesan antar aplikasi. SOAP dapat diibaratkan sebagai amplop yang digunakan untuk bertukar objek data di jaringan. SOAP mendefinisikan empat aspek komunikasi: Message envelope, Encoding, konvensi panggilan RPC, dan cara menyatukan pesan dalam protokol transport. Pesan SOAP terdiri dari amplop SOAP dan dapat terdiri dari lampiran atau tidak memiliki lampiran. Amplop SOAP terdiri dari header SOAP dan badan SOAP, sementara lampiran SOAP memungkinkan data non-XML dimasukkan ke dalam pesan SOAP, dikodekan, dan ditempatkan ke dalam pesan SOAP menggunakan MIME-multipart [3].

WSDL (Web Service Description Language) adalah bahasa berbasis XML untuk mendeskripsikan XML. WSDL menyediakan layanan yang menjelaskan permintaan layanan menggunakan protokol dan pengkodean yang berbeda. WSDL memfasilitasi komunikasi antar aplikasi. WSDL akan menjelaskan apa yang akan dilakukan layanan web, bagaimana menemukannya dan bagaimana mengoperasikannya. Spesifikasi WSDL mendefinisikan tujuh jenis elemen:

- *Types*, elemen untuk mendefinisikan tipe data. Elemen tersebut akan mendefinisikan tipe data seperti string atau integer elemen dalam sebuah pesan.
- *Message*, abstrak, mendefinisikan jenis data yang akan dikomunikasikan.
- *Operation*, deskripsi abstrak dari tindakan yang didukung oleh layanan.
- *Port Type*, kumpulan abstrak operasi yang didukung oleh lebih dari satu titik akhir.
- *Binding*, mendefinisikan integrasi jenis port (kumpulan operasi yang tersedia) ke dalam protokol transport dan format data (misalkan SOAP 1.1 pada HTTP). Ini adalah protokol konkret dan spesifikasi format data dalam tipe port tertentu.
- *Port*, mendefinisikan komunikasi titik akhir sebagai kombinasi alamat jaringan dan pengikatan. Untuk protokol HTTP bentuknya URL sedangkan untuk protokol SMTP adalah bentuk alamat email.
- *Service*, sekumpulan port terkait atau *endpoints*.

WSDL mendefinisikan layanan sebagai sambungan dari titik akhir jaringan. Definisi abstrak dari titik akhir dan pesan adalah bahwa bagian-bagian tersebut terpisah dari konstruksi jaringan atau format data terpadu. Pembagian ini menyebabkan penggunaan kembali deskripsi abstrak dari data yang akan dipertukarkan (pertukaran pesan) dan pengumpulan abstrak dari operasi protokol (port) konkret dan spesifikasi format data untuk jenis port tertentu menentukan binding mana yang dapat digunakan kembali. Port adalah alamat jaringan yang digabungkan dengan binding yang dapat digunakan kembali; layanan adalah kumpulan port.

Sedangkan UDDI (Deskripsi Universal, Penemuan, dan Integrasi) adalah registrasi layanan untuk mengalokasikan layanan web. UDDI menggabungkan SOAP dan WSDL untuk membentuk API registrasi untuk pendaftaran dan pengenalan layanan. Ini menyediakan area umum di mana organisasi dapat mengiklankan keberadaan dan layanan yang disediakan (layanan web). Semantik pada layanan web adalah harapan bersama tentang perilaku layanan, terutama dalam menanggapi pesan yang dikirim ke tujuan. Akibatnya, ini adalah "kontrak" antara entitas yang meminta dan badan penyedia terkait tujuan dan konsekuensi dari interaksi tersebut. Arsitektur layanan web melibatkan beberapa teknologi berlapis yang saling terkait satu sama lain. Ada banyak cara untuk memvisualisasikan teknologi ini, seperti banyak cara untuk membangun dan menggunakan layanan web.

Teknologi berlapis pada layanan web juga memberikan tingkat keamanan yang terjamin. Keamanan pada layanan web bersifat unik karena interaksi yang terjadi pada layanan web bukanlah interaksi antara manusia dan program, melainkan interaksi antara suatu program dengan program lain. Jadi keamanan disini adalah keamanan seperti kontrol akses, otentikasi, keamanan data dan privasi. Skema keamanan layanan web yang paling umum belakangan ini adalah SSL. Oleh karena itu, teknologi layanan web telah mulai bergerak menuju skema keamanan berbasis XML. Seperti enkripsi XML, XKMS (XML Key Management Specification), SAML (Secure Assertion Markup Language), WS-Security (Web Service Security), atau ebXML Message Service. Karena layanan web dibuat menggunakan protokol HTTP, layanan web memiliki kerentanan yang sama seperti situs web biasa. Hal tersebut dapat diatasi dengan memperhatikan aspek keamanan pada saat membuat layanan web, seperti:

- Otentikasi, gunakan *Public Key Infrastructure*, atau direktori aktif.
- Otorisasi, batasi hak kontrol akses ke data.
- Kerahasiaan, mengenkripsi konten pesan
- Integritas data, menerapkan Secure Security Layer / SSL selama proses komunikasi data di jaringan.
- *Non-repudiation*, menggunakan tanda tangan digital dan teknologi timestamping dan menerapkan log audit di setiap transaksi.

4. RESTful Web Service or GraphQL

REST (*Representational State Transfer*) merupakan standar arsitektur komunikasi berbasis web yang sering digunakan dalam pengembangan layanan berbasis web. Umumnya menggunakan HTTP sebagai protokol untuk komunikasi data. REST pertama kali disetujui oleh Roy Fielding pada 2000 silam. Pada arsitektur REST, server REST menyediakan sumber daya dan *client* REST mengakses dan menampilkan sumber daya ini untuk digunakan di masa mendatang. Setiap sumber daya diidentifikasi oleh URI (Universal Resource Identifiers) atau ID global. Sumber daya ini direpresentasikan dalam format teks, JSON, atau XML [4]. Secara umum format yang sering digunakan adalah XML dan JSON. Kelebihan REST adalah:

- Platform bahasa dan agnostik
- Lebih sederhana untuk dikembangkan dibandingkan dengan SOAP
- Mudah dipelajari, tidak bergantung pada alat
- Ringkas, tidak membutuhkan lapisan pertukaran pesan tambahan
- Dengan desain dan filosofi yang lebih dekat ke web

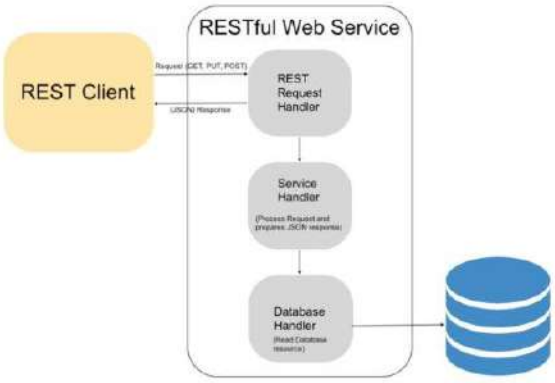
Kerugian dari REST adalah:

- Mengasumsikan model komunikasi titik-ke-titik - tidak dapat digunakan untuk lingkungan komputasi terdistribusi di mana pesan akan melalui satu atau lebih perantara
- Kurangnya dukungan standar untuk keamanan, kebijakan, keandalan pesan. Sehingga layanan yang memiliki

persyaratan yang lebih canggih akan lebih sulit untuk dikembangkan
- Berkaitan dengan model transport HTTP

Metode HTTP berikut biasanya digunakan dalam arsitektur berbasis REST:
- GET, hanya menyediakan akses baca ke sumber daya
- PUT, digunakan untuk membuat sumber daya baru
- DELETE, digunakan untuk menghapus sumber daya
- POST, digunakan untuk memperbarui sumber daya yang ada atau membuat sumber daya baru
- OPTIONS, digunakan untuk mendapatkan operasi yang didukung pada sumber daya

Layanan web adalah standar yang digunakan untuk bertukar data antar aplikasi atau sistem, karena aplikasi yang bertukar data dapat ditulis dalam bahasa pemrograman yang berbeda atau berjalan pada platform yang berbeda. Contoh implementasi layanan web adalah SOAP dan REST [5]. Layanan web berdasarkan arsitektur REST kemudian dikenal sebagai layanan web *RESTful*. Layanan web ini menggunakan metode HTTP untuk mengimplementasikan konsep arsitektur REST. Cara kerja layanan web RESTful seperti pada gambar 3.



https://phppot.com/php/php-restful-web-service

Gambar 3. Layanan web *RESTful*

Client mengirimkan data atau permintaan, melalui Permintaan HTTP dan server merespons melalui Respons HTTP. Komponen permintaan HTTP meliputi:
- *Verb*, metode HTTP yang digunakan, misalnya GET, POST, DELETE, PUT, dan lain lain.
- URI (*Uniform Resource Identifier*), untuk mengidentifikasi lokasi sumber daya di server.
- Versi HTTP, menunjukkan versi HTTP yang digunakan.
- *Request Header*, berisi metadata untuk permintaan HTTP. Contoh: jenis *client /browser*, format yang didukung oleh client, format pesan, pengaturan *cache*, dan sebagainya.
- *Request body*, isi data.

Sedangkan komponen respon HTTP meliputi:
- Kode status / respons, menunjukkan status server dari sumber daya yang diminta.
- Versi HTTP, menunjukkan versi HTTP yang digunakan.
- Respons Header, berisi metadata untuk tanggapan HTTP.
- Response Body, isi data yang diberikan

GraphQL adalah kueri data sumber terbuka dan manipulasi data sumber terbuka untuk API, serta waktu proses untuk mengisi kueri dengan data yang ada. GraphQL merupakan standar terbaru untuk API dengan karakteristik khusus, yaitu: lebih efisien dan juga lebih bertenaga daripada REST API. GraphQL pertama kali diperkenalkan dan dipopulerkan oleh Facebook. Karena sifatnya yang terbuka, GraphQL sekarang didukung secara luas oleh komunitas dan perusahaan di seluruh dunia. GraphQL memungkinkan client untuk mengambil data yang dibutuhkan dari server secara tepat sesuai dengan permintaan.

Sebelum munculnya GraphQL, REST adalah konsep paling populer yang digunakan sebagai standar komunikasi antar API terkait pertukaran data antar aplikasi. Saat REST API dikembangkan, aplikasi sisi client masih terbilang sederhana dengan kebutuhan kecepatan dan akses yang tidak seperti saat ini. Namun, dalam beberapa tahun terakhir, ada beberapa hal yang menyebabkan perlunya kecepatan transfer data yang lebih tinggi:
- Meningkatnya penggunaan aplikasi mobile mendorong Facebook untuk mempermudah jumlah data yang dikirim melalui jaringan sehingga menjadi lebih efisien.
- Berbagai macam kerangka kerja yang digunakan untuk sisi depan / sisi client menciptakan kesulitan untuk memelihara API yang dapat digunakan dengan semua kebutuhan kerangka kerja.

- Kebutuhan akan kecepatan dalam mengembangkan atau memodifikasi perangkat lunak. REST API yang dulunya dianggap cepat, kini sudah tidak relevan lagi dengan kebutuhan saat ini, malah dianggap sulit / sebagai penghambat [6]. Client tidak memiliki kendali penuh atas informasi yang dikirim oleh server.

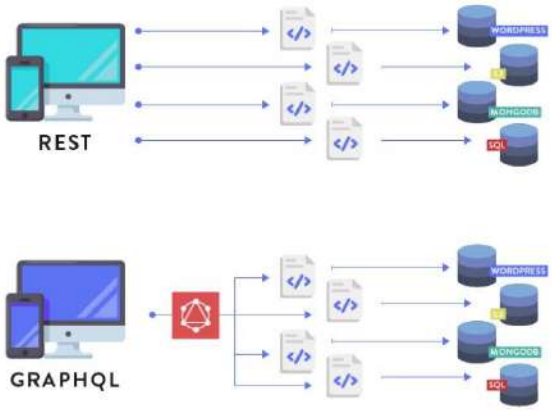
Perbedaan yang paling terlihat antara REST API dan GraphQL adalah pada struktur keluaran data yang lebih fleksibel untuk GraphQL. Di REST API, ketika client menginginkan data nama pengguna dengan nilai = "Batman" yang terdapat dalam database, client membuat permintaan dengan format seperti ini: GET / users / name / Batman /

Format di atas dapat dibaca / diartikan seperti ini: tampilan user dengan nama "Batman". Ketika client melakukan request ke server (endpoint), server akan menampilkan data dengan nama Batman ke client. Mekanisme yang demikian, pihak yang mendefinisikan data adalah sisi server. Artinya: pihak yang menentukan data pengguna apa yang akan ditampilkan kepada client seperti id, nama, jenis kelamin. Email dan telepon adalah server.

Sedangkan jika client menggunakan GraphQL, maka pihak yang menentukan data yang dibutuhkan adalah dari sisi client (frontend). Client dapat dengan leluasa menentukan data apa saja yang dibutuhkan dan sesuai dengan komponen aplikasi yang sedang dikerjakan. Dengan GraphQL, server hanya akan mengirim data sesuai permintaan client. Penggunaan GraphQL memberi arti:

- Hanya ada 1 titik akhir untuk berkomunikasi dengan server dalam proses mendapatkan data.
- Client (*frontend*) dapat mendefinisikan data yang dibutuhkan sesuai keinginan sehingga akan meningkatkan efisiensi dalam penggunaan API dan jumlah pertukaran data.

Keuntungan menggunakan GraphQL, yaitu mengatasi *overfetching*. Ini lebih dikenal sebagai pengambilan data yang melebihi permintaan client. Ada juga keuntungan yang didapat dalam menggunakan GraphQL adalah perkembangan pesat di frontend. Hal tersebut dapat diatasi karena client dapat menentukan sendiri data yang diinginkan tanpa harus meminta server untuk memberikan data baru yang akan menyebabkan proses pengembangan pada sisi *frontend* terhambat karena harus menunggu server melakukan penambahan data dan membuat titik akhir baru.



https://www.nan-labs.com/blog/headless-wordpress-with-react-and-graphql

Gambar 4. REST vs GraphQL

Dengan adanya kelebihan-kelebihan yang diberikan GraphQL kepada client jika dibandingkan dengan REST API, maka proses pertukaran data menjadi lebih akurat dan efisien. GraphQL juga dapat digunakan untuk melakukan perubahan pada data dengan menggunakan perintah mutation. Terdapat tiga macam mutation pada GraphQL :

- CREATE data baru
- UPDATE data yang sudah ada
- DELETE data

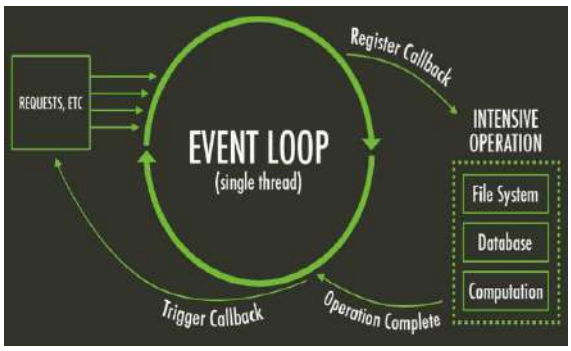
5. RESTful Web Service or GraphQL

Pada uji coba program kali ini, teknologi yang digunakan adalah Node.Js, MongoDB, dan Postman. Node.Js adalah platform yang dibuat menggunakan Chrome Javascript Runtime yang sangat cepat dalam membangun aplikasi jaringan. Sangat ringan dan efisien, sangat cocok untuk aplikasi dengan database intensif dan juga membutuhkan kecepatan waktu nyata pada mesin yang berbeda [7]. Node.Js adalah perangkat lunak open source dan dapat dijalankan di berbagai platform di sisi server. Node.Js ditulis dalam Javascript dan menawarkan kumpulan modul yang terkait dengan Javascript yang memfasilitasi pengembangan aplikasi berbasis web. Berikut adalah fitur-fitur penting yang ditawarkan oleh Node. JS [8]:

- Sinkronisasi dan berbasis kejadian - semua API dari Node.Js terhubung satu sama lain, yang berarti bahwa server di Node.Js tidak

pernah menunggu API untuk menyediakan data.
- Sangat cepat - Dibuat dengan Mesin Javascript Google Chrome V8, Node.Js sangat cepat dalam menjalankan kode.
- No Pause (bbfering) - Aplikasi Node.Js tidak pernah buffer
- Lisensi - Node.Js tersedia dengan MIT Lisence.

Node.Js banyak digunakan dalam aplikasi Input Output (I /O), aplikasi streaming data, aplikasi Data Intensive Real-time (DIRT), aplikasi berbasis JSON API, dan aplikasi dengan satu halaman. Banyak perusahaan besar telah menggunakan Node.Js seperti Ebay, Microsoft, PayPal, Uber, dan Yahoo. Konsep dari Node.Js sendiri adalah sebagai berikut:



https://www.oodlestechnologies.com/blogs/Internal-Architecture-Of-NodeJS

Gambar 5. Node.Js Diagram

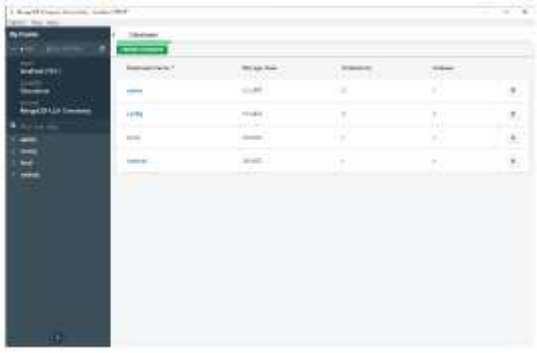
Tidak disarankan untuk menggunakan Node.Js pada jenis aplikasi yang bergantung pada penggunaan daya CPU. Setelah Node.Js untuk API digunakan dalam aplikasi, dalam sistem penyimpanan data, MongoDB akan digunakan. MongoDB adalah database NoSQL berorientasi objek, sederhana, dinamis, dan terukur. Objek data disimpan sebagai dokumen terpisah dalam sebuah koleksi. MongoDB menyimpan dokumen ini dengan menyediakan fitur penyimpanan data berkinerja tinggi. MongoDB sangat mudah dipasang dan digunakan. MongoDB menggunakan format penyimpanan file JSON yang didukung oleh platform seperti Windows, Linux, Mac, dan Solaris. Konsep umum MongoDB adalah sebagai berikut:

SQL Server	MongoDB
Database	Database
Table	Collection
Index	Index
Row	Document
Column	Field
Joining	Linking & Embedding
Partition	Sharding (Range Partition)
Replication	ReplSet

Gambar 6. Konsep MongoDB

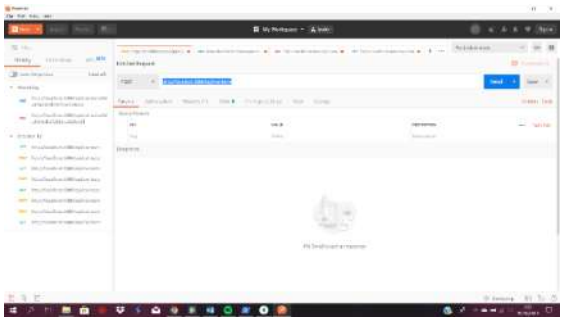
Terdapat keunggulan dalam pemanfaatan MongoDB, karena:
- Berorientasi pada dokumen
- Performa tinggi
- Sangat tersedia (replikasi)
- Bisa diukur
- Dinamis, tidak ada skema khusus / standar
- Fleksibel - kolom penambahan / pengurangan memiliki pengaruh yang sangat kecil pada aplikasi
- Data dalam format JSON
- Dapat dengan mudah diintegrasikan dengan Hadoop
- Bahasa query berbasis dokumen yang hampir setara dengan SQL.
- Distribusi cloud

MongoDB sangat cocok diimplementasikan untuk:
- Katalog produk e-commerce
- Blog dan juga manajemen konten
- Analisis waktu nyata dan logging kecepatan tinggi, caching, dan skalabilitas tinggi
- Manajemen konfigurasi
- Data geospasial
- Situs seluler dan jejaring sosial
- Kebutuhan data yang terus bertambah



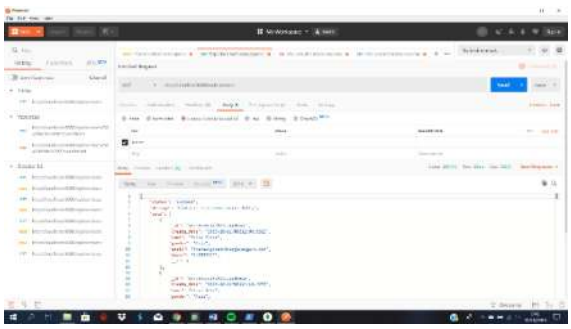
Gambar 7. MongoDB dengan MongoDB Compass

Setelah menentukan penggunaan API menggunakan Node.Js dan penggunaan database menggunakan MongoDB, kini saatnya menentukan penggunaan program yang berfungsi untuk menguji API yang telah dibuat dapat berfungsi dengan baik. Program yang digunakan adalah Postman. Postman adalah alat yang digunakan saat menguji RESTful API yang membuat permintaan dalam HTML tanpa memberikan masalah kepada pengguna (tidak perlu khawatir harus menulis kode program yang sulit) hanya untuk mengetahui apakah API sedang berjalan atau tidak.



Gambar 8. Pengujian dengan Postman

Misalnya untuk menampilkan request GET dari database, di Postman hanya perlu memasukkan URL aplikasi yang diinginkan, lalu pilih tab GET untuk menampilkan datanya. Jika ingin menambah data baru, pilih saja Postman dan klik tab POST setelah mengisi informasi yang ingin disimpan ke database. Data yang tersimpan dapat dilihat secara keseluruhan dengan mengetikkan URL aplikasinya dan data yang tersedia akan ditampilkan dalam format JSON. Selain dua fungsi GET dan POST di atas, Postman juga menyediakan pilihan fungsi PUT, PATCH, dan DELETE untuk mengubah dokumen yang disimpan di MongoDB.



Gambar 9. GET pada Postman untuk menampilkan dokument / data

6. KESIMPULAN

Dengan adanya web service, proses pertukaran data dapat terjadi dengan mudah. Bahasa antar platform yang awalnya berbeda dapat diseragamkan sehingga memungkinkan terjadinya pertukaran data antar platform. Keseragaman ini menggunakan format bahasa standar yang telah ditentukan sebelumnya seperti XML atau JSON. Tidak terbatas pada ruang lingkup pertukaran data antar perusahaan, layanan web juga memungkinkan pertukaran data antara aplikasi dan model bisnis. Ada banyak alat yang dapat digunakan untuk penggunaan layanan web seperti MongoDB, Node.Js, dan Postman seperti yang dibahas di atas. Perkembangan web services juga tidak lepas dari peran API di baliknya. Tanpa API yang bertindak sebagai standar pertukaran data antar aplikasi, tentunya dunia digital yang ada saat ini tidak akan tercapai.

Salah satu standar yang sering digunakan untuk komunikasi antar aplikasi adalah REST API. Namun perkembangan zaman saat ini membutuhkan kecepatan pertukaran data yang jauh lebih cepat dari sebelumnya. Belum lagi prosesnya juga harus efisien dan mudah dari sebelumnya. Oleh karena itu, Facebook meluncurkan standar API baru yang disebut GraphQL yang memungkinkan semua kebutuhan ini terpenuhi. Dengan GraphQL, proses pertukaran data antar aplikasi menjadi lebih efisien, lebih mudah, dan lebih cepat. Hal tersebut dikarenakan data yang dikirim hanya berupa data yang sesuai dengan permintaan dari client tanpa harus mengirimkan semua data yang terdapat di sisi server.

DAFTAR PUSTAKA

[1] Khurana S, Verma A G; Comparison of Cloud Computing Service Models: SaaS, PaaS, IaaS; IJECT Vol. 4 ; ISSN : 2230-7109, 2013

[2] Daconta M C, Obsrt LC, Smith K T; The Semantic Web : A Guide to the Future of XML, Web Services, and Knowledge Management; Wiley Publishing, 2003

[3] Pennington C, Cardoso J, Miller J A, Patterson R S, Vasques I; Introduction to Web Services; IGI Global, 2007

[4] Seabra M, Nazário M F, Pinto G; REST or GraphQL?: A Performance Comparative Study; Proceedings of the XIII Brazilian Symposium

on Software Components, Architectures, and Reuse Pages 123-132, 2019
[5] Adina Ploscar;XML-RPC vs. SOAP vs. REST web services in Java – uniform using WSWrapper; INTERNATIONAL JOURNAL OF COMPUTERS Issue 4, Volume 6, 2012
[6] Vibha Kumari;Web Services Protocol: SOAP vs REST; International Journal of Advanced Research in Computer Engineering & Technology (IJARCET) Volume 4 Issue 5, May 2015
[7] Agustaf R, Suteja B R,;Progressive Web Apps Untuk Rekayasa Hybrid Application Berbasis Teknologi Mean Stack; Journal Informasi Interaktif; Volume 4 No 3, May 2019
[8] Nirgudkar N, Singh P,;The Mean Stack; International Research Journal of Engineering and Technology, May 2017