No.

Date

Nama : Sulis Tryah
No BP : 2001081002

RESTFUL API

A. Integrasi Aplikasi

Saat membuat aplikasi, sering kali kita akan melakukan integrasi dengan aplikasi lain. Baik itu aplikasi yang kita buat sendiri, ataupun aplikasi yang dibuat oleh pihak lain.

Misal :

=> Saat kita membuat toko online, sistem akan terintegrasi dengan sistem logistic untuk mengelola pengiriman barangnya.

=> Saat kita membuat aplikasi belajar online, sistem akan terintegrasi dengan payment gateway untuk menyediakan layanan pembayaran kelas online.

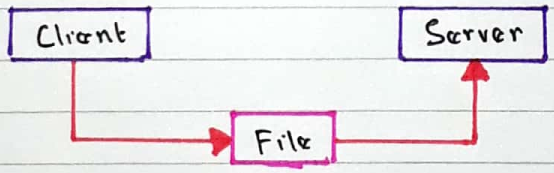
=> Saat kita membuat aplikasi mobile, aplikasi kita akan terintegrasi dengan sistem kita yang terdapat diserver untuk mengirim atau mengambil data.

B. Cara Integrasi Aplikasi

1. File Sharing

File sharing merupakan integrasi aplikasi dengan cara berbagi file. Integrasi menggunakan file sharing adalah integrasi yang paling mudah dilakukan dan masih banyak dilakukan sampai sekarang. Biasanya aplikasi yang memiliki data akan membuat file (misal excel, csv, text, json), dan aplikasi yang membutuhkan data akan membaca data tersebut dari file.

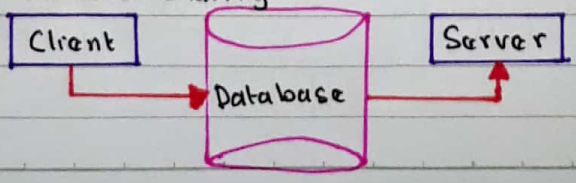
Diagram file sharing



2. Database Sharing

Database sharing merupakan integrasi antar aplikasi yang memanfaatkan database untuk berbagai data. Database sharing sangat mudah dilakukan ketika aplikasi berada ditempat yang sama dan bisa mengakses database yang sama. Aplikasi hanya perlu menyimpan data ke database, dan secara otomatis aplikasi lain bisa membaca data tersebut dari database secara langsung.

Diagram database sharing



No.

Date

3. Remote Procedure Invocation

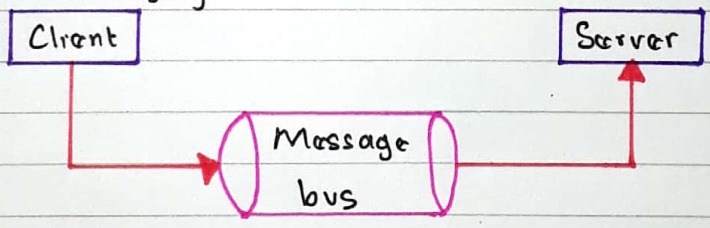
Remote procedure invocation merupakan mekanisme integrasi antar aplikasi dengan cara membuat API yang bisa digunakan oleh aplikasi lain. Aplikasi yang memiliki data akan membuat API, dan aplikasi yang membutuhkan akan menggunakan API tersebut untuk mendapatkan data dari aplikasi tersebut. Remote procedure Invocation merupakan cara sulit, namun sangat populer dilakukan saat ini. Hal ini karena menggunakan remote procedure invocation, integrasi bisa dilakukan dengan cara real time, dan kompleksitas internal data aplikasi tidak perlu diexpose ke aplikasi lain.

Diagram Remote Procedure Invocation

4. Messaging

Messaging merupakan cara integrasi aplikasi yang memanfaatkan aplikasi message broker atau message bus. Aplikasi yang memiliki data akan mengirim data ke aplikasi message broker, dan aplikasi yang membutuhkan data akan mengambil data dari message broker. Messaging tidak real time, kadang butuh waktu sampai data sampai ke aplikasi yang menerima data, sederhanaya proses di messaging adalah Asynchronous, sedangkan proses di remote procedure invocation adalah Synchronous.

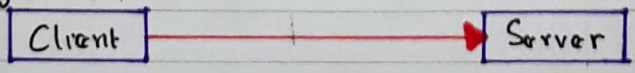
Diagram Messaging



C. Pengenalan API

API singkatan dari Application Programming Interface. API adalah perantara yang menghubungkan satu pihak dengan pihak lain agar bisa saling berkomunikasi. API berisi kumpulan prosedur, fungsi, cara berkomunikasi atau peralatan untuk komunikasi. Pihak yang terlibat dalam API bisa dalam bentuk perangkat lunak ataupun perangkat keras. API sebenarnya sama dengan Remote Procedure Invocation, hanya saja sekarang lebih populer istilah API dibilang API.

Diagram API



No.

Date

Contoh penggunaan API

=> Saat kita menggunakan sistem operasi, sistem operasi tidak bisa langsung berkomunikasi dengan perangkat keras. sistem operasi membutuhkan API berupa driver yang perlu dipasang terlebih dahulu.

=> Saat kita ingin berkomunikasi dengan aplikasi facebook, kita membutuhkan API dari facebook agar aplikasi kita bisa berinteraksi dengan aplikasi facebook.

Contoh Implementasi API

=> Driver Perangkat keras, sebagai API untuk sistem operasi

=> SOAP (Simple Object Access Protocol)

=> CORBA (Common Object Request Broker Architecture)

=> RESTful API

=> Apache Thrif

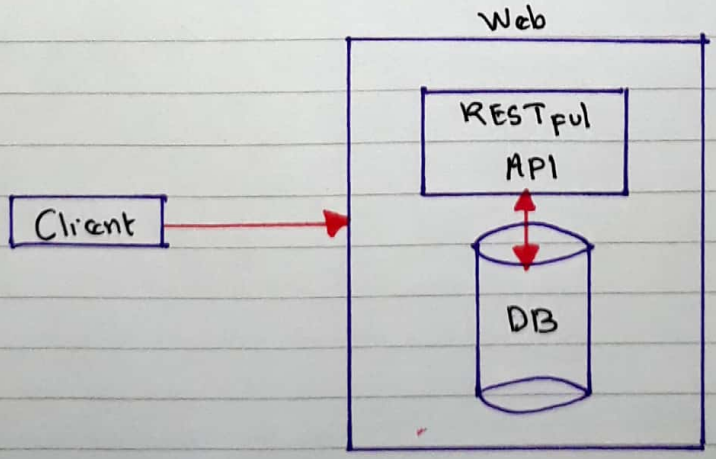
=> Protocol Buffer

=> GRPC

D. Pengenalan RESTful API

REST singkatan dari REpresentasional State Transfer. REST dikenalkan tahun 2000 oleh Roy Fielding dalam sertasinya. RESTful API merupakan salah satu implementasi API yang memanfaatkan HTTP sebagai protokol komunikasinya. Walaupun SOAP juga berjalan diatas HTTP, namun RESTful API sangat sederhana dibandingkan SOAP. RESTful API sangat mudah digunakan, dan bisa diadaptasi disemua bahasa pemograman secara mudah. Saat ini RESTful API sudah menjadi standard API yang banyak digunakan ketika kita membuat sistem yang butuh menyediakan API untuk pihak lain.

Diagram RESTful API



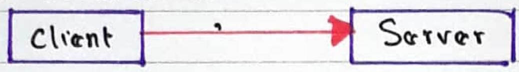
E. Architectural Constraints

REST (REpresentation State Transfer) merupakan architectural pattern yang dikenalkan oleh Roy Fielding tahun 2000. REST didasari berjalan menggunakan HTTP, dan sering digunakan sebagai Web Services. Roy Field memperkenalkan beberapa dasar principal ketika kita akan membuat REST, yaitu:

1. Client Server

Design principal pertama adalah client server. RESTFUL API haruslah memisahkan antara kompleksitivitas data internal dengan yang akan diekspose ke client. Oleh karena itu, Restful API haruslah menggunakan arsitektur Client Server. Sehingga Client tidak perlu tahu kompleksita logic yang terjadi diserver.

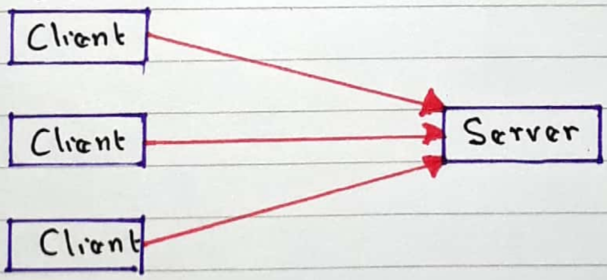
Diagram client server



2. Stateless

Interaksi antar client dan server dalam restful haruslah stateless. Artinya tiap interaksi harus tidak tergantung dengan interaksi sebelumnya atau setelahnya, dan setiap interaksi harus mengirim seluruh informasi yang dibutuhkan. Salah satu kegunaan stateless adalah sehingga mudah untuk discalling, baik itu jumlah client juga server karena server atau client tidak perlu peduli harus berinteraksi dengan client atau server manapun

Diagram stateless



3. Cachable

Untuk menghemat komunikasi, RESTful API bisa mengimplementasikan Cache. Mirip seperti Cache di HTTP, diclient pada RESTful API juga bisa melakukan cache data dilocal, sehingga tidak perlu selalu meminta data terbaru dari server. Cara Implementasi Cache di RESTful API tidak sesederhana seperti di HTTP.

No.

Date

4. Uniform Interface

Salah satu yang membedakan RESTful API dengan teknologi RPC lain adalah, penggunaan antarmuka komunikasi yang seragam untuk semua pihak. Hal ini dikarenakan salah satunya karena RESTful API menggunakan teknologi HTTP yang sudah standard sehingga seragam disemua teknologi atau bahasa pemograman. Data yang diekspose di RESTful API juga haruslah generatel, tidak melihatkan kompleksitas internal dari pemilik data, hal ini membuat perubahan apapun yang terjadi pada internal aplikasi, tidak akan berpengaruh dengan data yang direkspose di API.

5. Layered System

Untuk melakukan improvement pada sistem RESTful API, sistem RESTful API juga dapat menggunakan layered system. Layered system menjadikan sistem bisa disusun sesuai dengan datanya, dan agar kompleksitas pada RESTful API tidak harus diketahui oleh client. Layered juga bisa digunakan untuk melakukan enkapsulasi aplikasi lama yang tidak memiliki kemampuan RESTful API, atau menjadi load balancer untuk RESTful API lain.

6. Code On Demand

RESTful API juga diperbolehkan mengembalikan script yang bisa dieksekusi oleh client jika diperlukan. Hal ini bisa mempermudah dari sisi client sehingga tidak perlu mengimplementasikan kode terlalu banyak, karena kode bisa dikirim oleh server. Code on Demand adalah design principal yang tidak wajib diimplementasikan ketika kita membuat RESTful API

F. Resource Naming

Resource dalam RESTful API adalah data yang sifatnya bisa satu atau banyak. Misal, "Customers" adalah kumpulan dari "Customer", dimana "customer" adalah satu data "customer"

Gunakan kata benda, Bukan kata kerja.

=> {B} http://api.example.com/products

=> {S} http://api.example.com/get-all-products

Gunakan Hirarki

=> {B} http://api.example.com/products/{productId}/images

=> {S} http://api.example.com/product-images/{productId}

Gunakan Action Pada Resource

=> {B} http://api.example.com/users/login

=> {S} http://api.example.com/login-user

Gunakan - dan lowercase

=> {B} http://api.example.com/products/{productId}/warehouse-locations

=> {S} http://api.example.com/products/{productId}/warehouse_locations

Gunakan CRUD pada HTTP Method

=> {B} GET http://api.example.com/products/{productId}

=> {B} POST http://api.example.com/products

=> {S} GET http://api.example.com/get-products-by-id/{productId}

=> {S} POST http://api.example.com/create-product

Gunakan Query untuk Filter

=> {B} http://api.example.com/products?name=Indomie

=> {S} http://api.example.com/products/filter-by-name/{name}

G. Content Negotiation

Saat membuat Web menggunakan HTTP, maka biasanya content (Body) yang akan kita gunakan akan menggunakan HTML. Pada RESTful API pun, untuk berkomunikasi antara client dan server, biasanya menggunakan body. Body Content yang biasa digunakan di RESTful API, seperti JSON (JavaScript Object Notation), XML, dan lain-lain. Namun yang paling populer digunakan saat ini adalah JSON.

H. Cache

Secara sederhana cache adalah data bersifat sementara yang disimpan pada srstem penyimpanan. Dalam Restful API, data cache biasanya disimpan diclient. Cache biasa digunakan untuk menurunkan jumlah data transfer antara client dan server sehingga proses komunikasi lebih cepat.

I. Idempotence

Dalam RESTful API, ketika membuat multiple request yang identik, harus memiliki efek yang sama seperti membuat satu request. Dalam hal ini, maka RESTful API kita bisa saling idempotent. Idempotent itu sangat penting, karena saat membuat RESTful API, kita akan melakukan komunikasi antara client dan server via network. sehingga error bisa terjadi. Belum lagi, banyak framework atau library client yang bisa secara otomatis melakukan request ulang ketika terjadi error pada network.

No.

Date

Idempotent di HTTP Method POST

Umumnya, pada POST, kita tidak perlu membuat API nya menjadi idempotent. Reqvest berkali-kali menggunakan POST dengan data yang sama akan selalu membuat record baru. Namun kadang hal ini berbahaya, misal ketika ~~dess~~ terjadi kesalahan client mengirim dua data yang sama, maka akan menjadi 2 record diserver, padahal awalnya hanya ingin membuat 1 record. Namun perlu diingat, ini optional, pada POST implementasi idempotent tidak diwajibkan.

Idempotent di HTTP Method GET

Method GET tidak pernah mengubah data diserver. Method GET hanya digunakan untuk mengambil data yang ada diserver. Jadi mengirim reqvest GET berkali-kali keserver tidak akan mengubah data apapun diserver, sehingga GET secara default sudah idempotent.

Idempotent di HTTP Method PUT dan PATCH

Method PUT dan PATCH digunakan untuk mengubah data yang sudah ada. Jika kita mengirim reqvest PUT dan PATCH berkali-kali dengan data yang sama, maka server akan melakukan proses update data berkali-kali dengan data yang sama. Request pertama akan mengubah data didatabase, request selanjutnya hanya akan data request pertama, sehingga hasil akhirnya sebenarnya tetap sama.

Idempotent di HTTP Method DELETE

Method DELETE digunakan untuk menghapus data diserver. Ketika kita mengirim permintah DELETE berkali-kali diserver, response nya mungkin akan berbeda. Walaupun secara response mungkin berbeda, namun sebenarnya DELETE sudah secara default idempotent, karena mengirim request DELETE berkali-kali, hasilnya tetap sama, data yang ada di server terhapus.

J. Security

Ada kalanya kita butuh mengamankan RESTful API yang kita buat atau ingin membatasi pihak yang boleh mengakses RESTful API yang kita buat. Salah satunya adalah dengan menggunakan Authentication dan Authorization

Authentication

Memvalidasi kredensial untuk memverifikasi pemilik identitas. Contoh prosesnya adalah proses login ~~seperti~~ menggunakan username dan password, dan banyak yg lainnya

Authorization

Adalah proses yg dilakukan setelah proses Authentication. Memvalidasi apakah pemilik identitas memiliki hak akses untuk mengakses resource yg diminta. Contoh: Access-Control List. dan banyak yang lainnya.

No.

Date

Contoh Authentication dan Autorization

1. Basic Auth

↳ Authentication sederhana menggunakan username dan password

↳ Cukup menggunakan header Authorization

2. API-Key

↳ Authentication sederhana menggunakan API-Key atau Secret Key

↳ Cukup Menggunakan header sesuai dgn yg dinginkan dan value berisi API-key atau secret key.

3. OAuth 2

↳ Mekanisme Authentication dan Autorization yang saat ini sangat populer

↳ Banyak digunakan untuk integrasi antara aplikasi Mobile dan Server

K. Versioning

Ada banyak cara melakukan versioning pada RESTful API. Versioning adalah memberi tahu bahwa bahwa kita memiliki banyak versi terhadap aplikasi RESTful API kita. Client bisa memilih versi yang mana yang akan digunakan.

L. HATEOAS

HATEOAS Singkatan dari Hypermedia as the Engine of Application State. Hypermedia artinya content yang memiliki link menuju resource yang ada.

Keuntungan:

=> Biasanya URL API pada RESTful API sudah di hardcode di client.

=> Dengan menggunakan HATEOAS, client bisa secara dinamis mendapatkan URL lokasi resource dari response data server

M. Documentation

Dokumentasi RESTful API bisa dibuat menggunakan apapun, dari dokumen sederhana menggunakan microsoft word / google doc, sampai menggunakan tool khusus untuk dokumentasi API. Sangat disarankan untuk menggunakan tool khusus dokumentasi API.

Contoh:

=> Swagger

=> Stoplight

=> Redoc

=> Dan lain-lain

No.

Date

N. Development

Kesalahan ketika membuat tabel RESTful API

=> Selalu membuat CRUD API untuk table di database

=> Membuat Response data sama dengan table di database

=> Membuat API yang tidak dibutuhkan client

=> Mengembalikan data selengkap-lengkapnya di API

=> Membuat API terlebih dahulu, baru mengerjakan Web atau Mobile menggunakan API yang sudah dibuat.

O. Maintenance

Dalam membuat product, fitur pasti akan selalu bertambah. Tak jarang kita mungkin melakukan perubahan di screen Web atau Mobile yang sama. Oleh karena itu, Maintenance RESTful API sangatlah penting, agar RESTful API kita tidak menjadi masalah di kemudian hari

Maintenance yg boleh dilakukan

=> Menambah data baru di API yang sudah ada

=> Menambah API baru di endpoint URL berbeda

=> Mempercepat proses di API yang sudah ada

=> Menggabungkan beberapa API menjadi satu, tanpa menghilangkan API Lama.

Maintenance yang tidak boleh dilakukan

=> Mengubah total response data di API yang sudah ada

=> Merubah field Response data di API yang sudah ada

=> Menghilangkan API yang sudah ada

=> Non-split API yang sudah ada menjadi dua atau lebih

=> Menggabungkan beberapa API menjadi satu dengan menghapus API lama

P. Maturity Model

Leonard Richardson telah melakukan analisa banyak sekali Web Service, dan akhirnya menyimpulkan dalam 4 kategori. Seberapa sempurna RESTful API.

=> Level Zero

=> Level One

=> Level Two

=> Level Three