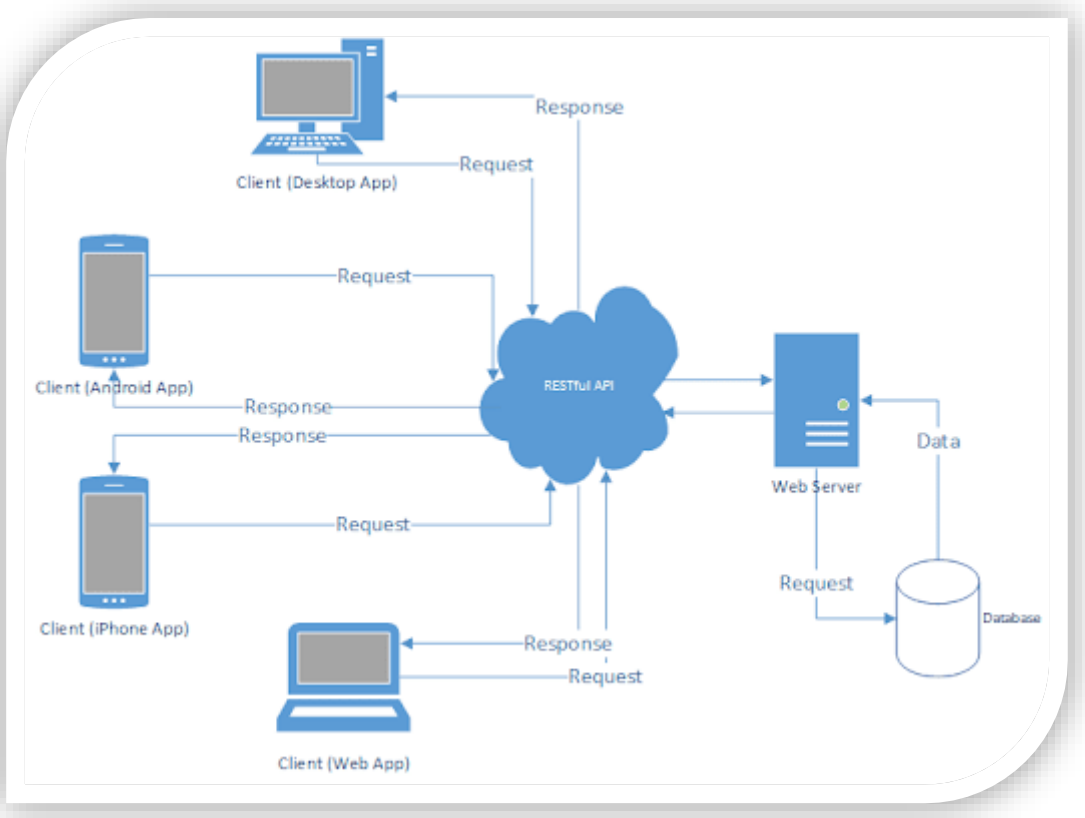
# RESTful Web Services

## Konsep




### API

API (Application Protocol Interface) adalah kumpulan dari subroutine definitions, protocol dan juga tools untuk berkomunikasi data antar aplikasi software.

### Web service

Web service adalah standar yang digunakan untuk melakukan pertukaran data antar aplikasi atau sistem, karena aplikasi yang melakukan pertukaran data bisa ditulis dengan bahasa pemrograman yang berbeda atau berjalan pada platform yang berbeda. Contoh implementasi dari web service antara lain adalah SOAP dan REST.

Web service yang berbasis arsitektur REST kemudian dikenal sebagai RESTful web services. Layanan web ini menggunakan metode HTTP untuk menerapkan konsep arsitektur REST.REST adalah salah satu implementasi dari web service sebagai sebuah standar yang digunakan untuk pertukaran data antar aplikasi atau sistem.

Biasanya aplikasi atau sistem ini menggunakan bahasa pemrogramman yang berbeda sehingga untuk bisa berkomunikasi satu sama lainnya bisa menggunakan web service ini.

**REST (REpresentational State Transfer)**

merupakan standar arsitektur berbasis web. Jadi bisa dibilang, REST itu adalah salah satu dari desain arsitektur di dalam API.

Meskipun REST dapat digunakan di hampir semua protokol, tapi biasanya memanfaatkan HTTP ketika digunakan untuk Web API. Hal ini membantu pengembang web tidak perlu menginstal library atau perangkat lunak tambahan untuk memanfaatkan desain REST API.

Design REST API pertama kali diperkenalkan oleh Dr. Roy Fielding dalam disertasi doktor tahun 2000-nya. REST API terkenal karena fleksibilitasnya yang luar biasa.

Data tidak terikat dengan metode dan sumber daya, REST memiliki kemampuan untuk menangani beberapa jenis panggilan, mengembalikan format data yang berbeda dan bahkan mengubah secara struktural tentunya dengan implementasi yang benar.

Rest API secara eksplisit memanfaatkan metodologi HTTP yang ditentukan oleh **protokol RFC 2616.**

REST yang digunakan oleh browser dapat dianggap sebagai bahasa internet. Dengan meningkatnya penggunaan cloud, API muncul untuk mengekspos layanan web.

REST adalah pilihan logis untuk membangun API yang memungkinkan pengguna untuk terhubung dan berinteraksi dengan layanan cloud. API telah banyak digunakan oleh situs-situs seperti Amazon, Google, LinkedIn dan Twitter.

Pada arsitektur REST, REST server menyediakan resources (sumber daya/data) dan REST client mengakses dan menampilkan resource tersebut untuk penggunaan selanjutnya.

Setiap resource diidentifikasi oleh URIs (Universal Resource Identifiers) atau global ID. Resource tersebut direpresentasikan dalam bentuk format teks, JSON atau XML. Pada umumnya formatnya menggunakan JSON dan XML.

**Keuntungan REST**

- bahasa dan platform agnostic
- lebih sederhana/simpel untuk dikembangkan ketimbang SOAP
- mudah dipelajari, tidak bergantung pada tools
- ringkas, tidak membutuhkan layer pertukaran pesan (messaging) tambahan
- secara desain dan filosofi lebih dekat dengan web

**Kelemahan REST**

- Mengasumsi model point-to-point komunikasi - tidak dapat digunakan untuk lingkungan komputasi terdistribusi di mana pesan akan melalui satu atau lebih perantara
- Kurangnya dukungan standar untuk keamanan, kebijakan, keandalan pesan, dll, sehingga layanan yang mempunyai persyaratan lebih canggih lebih sulit untuk dikembangkan ("dipecahkan sendiri")
- Berkaitan dengan model transport HTTP

**Berikut metode HTTP yang umum digunakan dalam arsitektur berbasis REST.**

- GET, menyediakan hanya akses baca pada resource
- PUT, digunakan untuk menciptakan resource baru
- DELETE, digunakan untuk menghapus resource
- POST, digunakan untuk memperbarui resource yang ada atau membuat resource baru
- OPTIONS, digunakan untuk mendapatkan operasi yang disupport pada resource

**Lalu bagaimana cara kerja REST API?**

Pertama harus ada sebuah REST server yang akan menyediakan resource/data. Sebuah REST client akan membuat HTTP request ke server melalui sebuah global ID atau URIs dan server akan merespon dengan mengirimkan balik sebuah HTTP response sesuai yang diminta client.

**komponen dari HTTP request:**

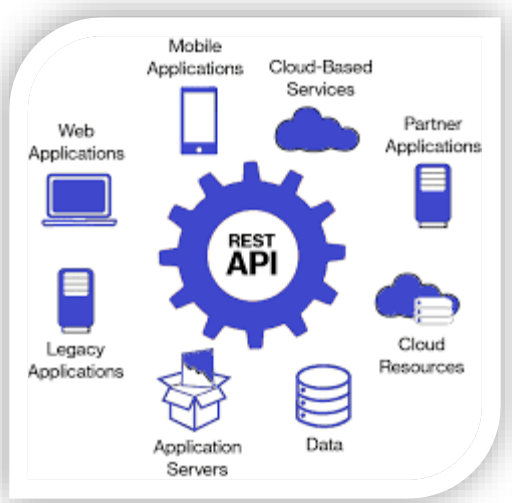
- HTTP method seperti GET, POST, PUT, DELETE dll sesuai dengan tugasnya masing-masing
- URI untuk mengetahui lokasi resource/data pada server
- HTTP Version, seperti HTTP v1.1
- Request Header, berisi metadata seperti Authorization, tipe client/browser, format yang didukung oleh client, format dari body pesan, seting cache dll.
- Request Body, konten dari data yang diberikan client ke server seperi URI params

**RESTfull API**

API dapat dikatakan **“RESTful”** jika memiliki fitur berikut :

- Client – server : client menangani front end dan server menangani back end dan keduanya dapat diganti secara independen satu sama lain.
- Stateless : Tidak ada data klien yang disimpan di server ketika ada permintaan dan status sesi disimpan di klien.
- Cacheable : Klien dapat men-cache respon (seperti browser yang men-cache elemen statis halaman web) untuk meningkatkan kinerja.

Keuntungan terbesar dari Restful API adalah Anda tidak perlu memasang apa pun di sisi klien. SDK atau framework tidak diperlukan. Yang harus Anda lakukan adalah membuat permintaan HTTP sederhana ke layanan end point dari API target, biarkan server melakukan layanannya untuk Anda dan dapatkan hasilnya kembali. Sangat mudah dilakukan.




Contoh paling sederhana adalah Google API untuk login. Anda tidak perlu menghabiskan banyak waktu hanya untuk membuat sistem login untuk member di website. Anda hanya perlu memanggil API dari google, pengguna hanya perlu login ke Google kemudian Anda akan mendapatkan data semisal alamat email / nama dari pengguna tersebut. Tentunya jika pengguna telah memberikan izin mengunakan data mereka

Jika kita jabarkan kembali, maka RESTful API memiliki 4 komponen penting di dalamnya diantaranya adalah:

- ✓ **URL Design**
- ✓ **HTTP Verbs**
- ✓ **HTTP Response Code**
- ✓ **Format Response**

**URL Design**

RESTful API diakses menggunakan protokol HTTP. Penamaan dan struktur URL yang konsisten akan menghasilkan API yang baik dan mudah untuk dimengerti developer. URL API biasa disebut endpoint dalam pemanggilannya. Contoh penamaan URL / endpoint yang baik adalah seperti berikut :

```
/users
/users/1234
/users/1234/photos
/users/1234/photos/abc
```

**HTTP Verbs**

Setiap request yang dilakukan terdapat metode yang dipakai agar server mengerti apa yang sedang di request client, diantaranya yang umum dipakai adalah :

**GET**

GET adalah metode HTTP Request yang paling simpel, metode ini digunakan untuk membaca atau mendapatkan data dari sumber.

**Contoh :**

| |
|---|
| *GET /users : Mengembalikan daftar user*<br>*GET /users/1234 : Mengembalikan data user dengan ID 1234* |

**POST**

POST adalah metode HTTP Request yang digunakan untuk membuat data baru dengan menyisipkan data dalam body saat request dilakukan.

**Contoh :**

| |
|---|
| *POST /users : Membuat data user baru* |

**PUT**

PUT adalah metode HTTP Request yang biasanya digunakan untuk melakukan update data resource.

**Contoh:**

| |
|---|
| *PUT /users/1234 : Mengupdate data user dengan ID 1234* |

**DELETE**

DELETE adalah metode HTTP Request yang digunakan untuk menghapus suatu data pada resource.

**Contoh:**

| |
|---|
| *DELETE /users/1234 : Menghapus data user dengan ID 1234* |

Selain HTTP Verbs diatas, masih ada metode HEAD dan PATCH dalam HTTP Request, tetapi jarang sekali digunakan.

**HTTP Response Code**

HTTP response code adalah kode standarisasi dalam menginformasikan hasil request kepada client. Secara umum terdapat 3 kelompok yang biasa kita jumpai pada RESTful API yaitu :

- **2XX** : adalah response code yang menampilkan bahwa request berhasil.
- **4XX** : adalah response code yang menampilkan bahwa request mengalami kesalahan pada sisi client.

- **5XX** : adalah response code yang menampilkan bahwa request mengalami kesalahan pada sisi server.

Berikut adalah response code yang biasa digunakan pada REST:

- **200 OK**
  Response code ini menandakan bahwa request yang dilakukan berhasil.

- **201 Created**
  Response code ini menandakan bahwa request yang dilakukan berhasil dan data telah dibuat.
  Kode ini digunakan untuk mengkonfirmasi berhasilnya request PUT atau POST.

- **400 Bad Request**
  Response code ini menandakan bahwa request yang dibuat salah atau data yang dikirim tidak ada.

- **401 Unauthorized**
  Response code ini menandakan bahwa request yang dibuat membutuhkan authentication sebelum mengakses resource.

- **404 Not Found**
  Response Code ini menandakan bahwa resource yang di dipanggil tidak ditemukan.

- **405 Method Not Allowed**
  Response code ini menandakan bahwa request endpoint ada tetapi metode HTTP yang digunakan tidak diizinkan.

- **409 Conflict**
  Response code ini menandakan bahwa request yang dibuat terdapat duplikasi, biasanya informasi yang dikirim sudah ada sebelumnya.

- **500 Internal Server Error**
  Response code ini menandakan bahwa request yang dilakukan terdapat kesalahan pada sisi server atau resource.

**Format Response**

- **Status/response code**, mengindikasikan status server terhadap resource yang direquest. misal : 404, artinya resource tidak ditemukan dan 200 response OK.

- **HTTP Version**, menunjukkan versi dari HTTP yang digunakan, contoh HTTP v1.1.

- **Response Header,** berisi metadata untuk HTTP Response.
  Contoh, type server, panjang content, tipe content, waktu response, dan lainnya.

- **Response Body**, konten dari data yang diberikan

Setiap request yang dilakukan client akan menerima data response dari server, response tersebut biasanya berupa data XML ataupun JSON. Setelah mendapatkan data response tersebut barulah client bisa menggunakannya dengan cara memparsing data tersebut dan diolah sesuai kebutuhan.

**Contoh:**

**XML**

```
HTTP/1.1 200 OK
Date: Sat, 06 Oct 2001 23:20:04 GMT
Server: Apache.1.3.12 (Unix)
Connection: close
Content-Type: text/xml
Content-Length: 124

<?xml version="1.0"?>
<methodResponse>
<params>
<param>
<value><double>18.24668429131</double></value>
</param>
</params>
</methodResponse>
```

**JSON**

```
GET /users/1234

HTTP/1.1 200 OK
Content-Type: application/vnd.api+json
{
"id": "1234",
"first_name": "jhon",
"last_name": "doe",
"created": "2015-05-22T14:56:29.000Z",
"updated": "2015-05-22T14:56:29.000Z"
}
```

**EOF**