

Bab 7

Application Programmable Interfaces (API)

Detail Materi

Indikator :
Memiliki Pemahaman mengenai API dan REST API pada beberapa bahasa pemrograman



Memahami konsep dasar API dan REST API



Membuat API Server menggunakan CodeIgniter



Membuat REST API Server menggunakan NodeJS



Membuat API Server menggunakan Python

Modul Sekolah DevOps Cilsy

Hak Cipta © 2020 **PT. Cilsy Fiolution Indonesia**

Hak Cipta dilindungi Undang-Undang. Dilarang memperbanyak atau memindahkan sebagian atau seluruh isi buku ini dalam bentuk apapun, baik secara elektronik maupun mekanis, termasuk mecropy, merekam atau dengan sistem penyimpanan lainnya, tanpa izin tertulis dari Penulis dan Penerbit.

Penulis : Widiani & Tresna Widiyaman

Editor: Muhammad Fakhri Abdillah, Tresna Widiyaman

Revisi Batch 4

Penerbit : **PT. Cilsy Fiolution Indonesia**

Web Site : <https://cilsyfiolution.com> , <https://devops.cilsy.id>

Sanksi Pelanggaran Pasal 113 Undang-undang Nomor 28 Tahun 2014 tentang Hak Cipta

1. Setiap orang yang dengan tanpa hak melakukan pelanggaran hak ekonomi sebagaimana dimaksud dalam pasal 9 ayat (1) huruf i untuk penggunaan secara komersial dipidana dengan pidana penjara paling lama 1 (satu) tahun dan atau pidana denda paling banyak Rp100.000.000,00 (seratus juta rupiah).
2. Setiap orang yang dengan tanpa hak dan atau tanpa izin pencipta atau pemegang hak cipta melakukan pelanggaran hak ekonomi pencipta sebagaimana dimaksud dalam pasal 9 ayat (1) huruf c, huruf d, huruf f, dan atau huruf h, untuk penggunaan secara komersial dipidana dengan pidana penjara paling lama 3 (tiga) tahun dan atau pidana denda paling banyak Rp500.000.000,00 (lima ratus juta rupiah)
3. Setiap orang yang dengan tanpa hak dan atau tanpa izin pencipta atau pemegang hak melakukan pelanggaran hak ekonomi pencipta sebagaimana dimaksud dalam pasal 9 ayat (1) huruf a, huruf b, huruf e, dan atau huruf g, untuk penggunaan secara komersial dipidana dengan pidana penjara paling lama 4 (empat) tahun dan atau pidana denda paling banyak Rp1.000.000.000,00 (satu miliar rupiah)
4. Setiap orang yang memenuhi unsur sebagaimana dimaksud pada ayat (3) yang dilakukan dalam bentuk pembajakan, dipidana dengan pidana penjara paling lama 10 (sepuluh) tahun dan atau pidana denda paling banyak Rp4.000.000.000,00 (empat miliar rupiah)

Daftar Isi

Cover.....	1
7. Application Programmable Interfaces (API).....	4
Learning Outcomes.....	4
Outline Materi.....	4
7.1. Pengenalan API.....	5
7.1.1. Apa itu API ?.....	5
7.1.2. REST API.....	6
7.1.3. Arsitektur API.....	7
7.1.4. Kelebihan dan Kekurangan API.....	9
7.2. Praktik API dengan NodeJS.....	10
7.2.1. Pengenalan NodeJS.....	10
7.2.2. Persiapan.....	11
7.2.2.1. Instalasi NodeJS Windows.....	11
7.2.2.2. Instalasi NodeJS Ubuntu.....	11
7.2.2.3. Instalasi Extension Tabbed Postman.....	12
7.2.3. Setup dan Praktek Pengerjaan.....	13
7.2.4. Memasukan fungsi CRUD.....	21
7.2.5. Pengujian.....	25
7.2.5.1. Pengujian Menggunakan POST.....	25
7.2.5.2. Pengujian Menggunakan GET.....	25
7.2.5.3. Pengujian Menggunakan DELETE.....	26
7.3. Summary.....	27
7.4. References.....	27

7.

Application Programmable Interfaces (API)

Learning Outcomes

Setelah selesai mempelajari bab ini, peserta mampu :

1. Memahami konsep dasar API dan REST API
2. Dapat membuat REST API Server menggunakan NodeJS

Outline Materi

1. Pengenalan API
2. REST API
3. API NodeJS

7.1. Pengenalan API

7.1.1. Apa itu API ?

API (*Application Programmable Interface*) adalah sebuah interface yang berupa kumpulan fungsi yang dapat di-'panggil' atau dijalankan oleh program lain. Kegunaan dari API adalah sebagai "jembatan" yang menghubungkan 2 program yang berbeda sehingga dapat memudahkan kegiatan yang akan dilakukan. Sebagai contoh, ketika kita ingin menggunakan fitur kamera di smartphone kita tidak perlu mengetikkan kode untuk mengakses interface kamera, kita hanya perlu menggunakan API untuk mengaksesnya saja.



Ilustrasi Penggunaan API

Penggunaan API saat ini sudah sangat luas penerapannya, penerapannya dapat dilakukan di berbagai bahasa pemrograman, library dan framework, sistem operasi, dan web API atau web service. API dapat dihubungkan untuk menghubungkan 2 program yang berbeda misalnya :

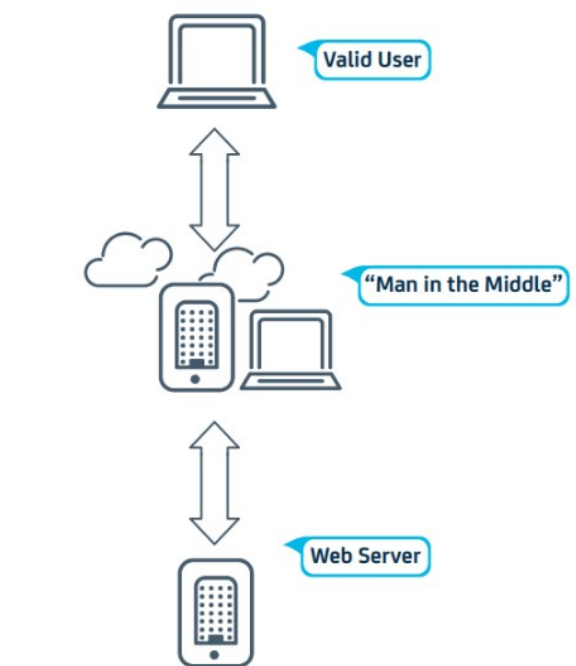
- Menghubungkan bahasa pemrograman php dengan mysql menggunakan perintah/script mysqli dan pdo. Perintah mysqli dan pdo tersebut merupakan API yang digunakan untuk membuat koneksi antara php dan database mysql.

- Pada framework/library Code Igniter, ketika kita akan mengambil data dari sebuah tabel kita tidak perlu menggunakan syntax mysql yang cukup panjang, kita hanya perlu memanggil dengan API "***\$this->db->get('nama_tabel');***". Ini merupakan API pada Code Igniter untuk mengambil seluruh data yang ada pada sebuah tabel.

7.1.2. REST API

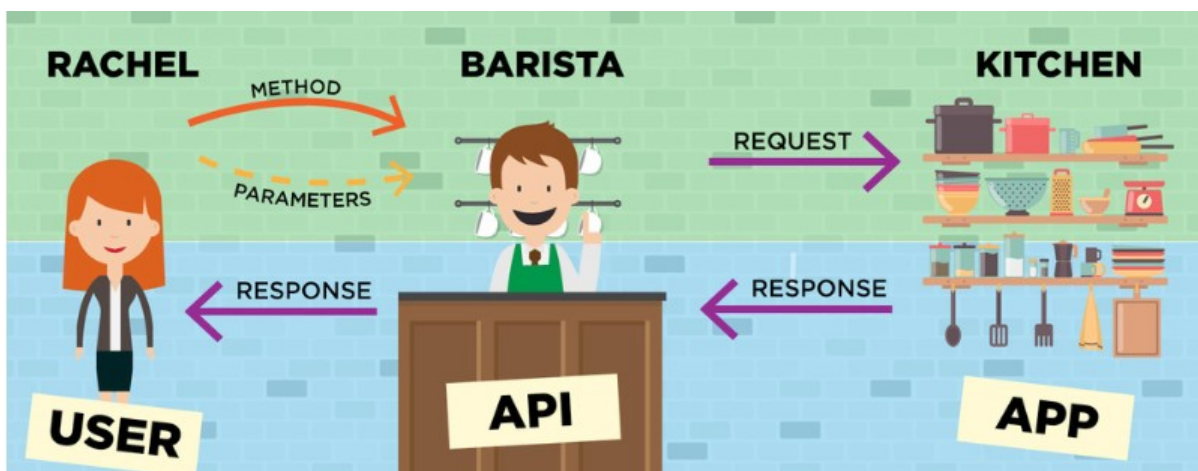
Rest (*REpresentational State Transfer*) API adalah sebuah gaya perancangan (*architectural style*). REST API memungkinkan terjadinya interaksi antar mesin, maksudnya ketika kita membuka sebuah web yang terjadi adalah interaksi antara manusia dengan mesin, sedangkan ketika menggunakan REST API maka yang terjadi adalah interaksi antara mesin dengan mesin. Salah satu contohnya misalnya aplikasi e-banking.

Aplikasi e-banking tidak mungkin memiliki akses langsung ke database milik bank, akan tetapi aplikasi tersebut dapat mengakses API yang dimiliki bank sehingga mereka dapat mendapatkan data-data dari berbagai bank tersebut.



Ilustrasi Alur Rest API

Kita dapat analogikan Rest API ini menjadi sebuah restoran. Disebuah restoran ada seorang pelanggan yang datang kemudian memesan makanan dari menu yang diberikan oleh pelayan, setelah selesai memilih pesanan maka pelayan akan menuliskan pesanan dari pelanggan kemudian memberikan pesanan tersebut pada koki di dapur. Setelah pesanan selesai maka pelayan akan mengantarkan makanan dari dapur menuju pelanggan.

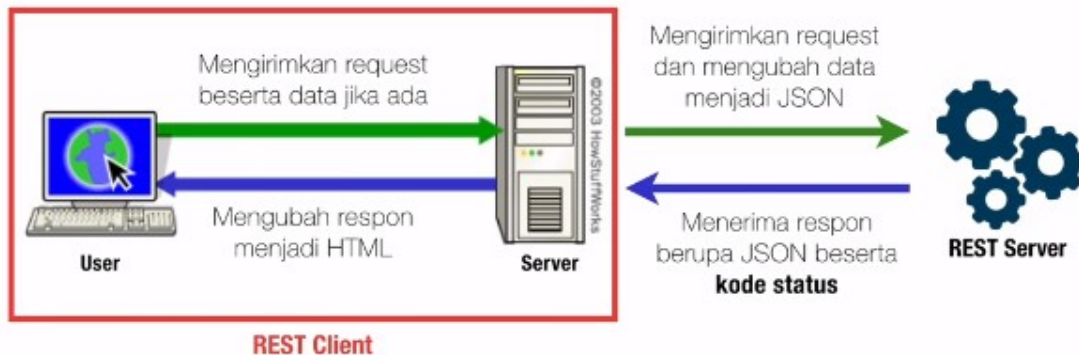


Analogi Restoran Rest API

Pelanggan adalah kita sebagai user yang memberikan request. **Pelayan** adalah API yang mencatat serta menyampaikan request dari user. **Menu** yang pelayan berikan adalah REST API karena fungsinya adalah sebagai aturan yang diberikan agar kita tidak memesan hal yang tidak ada dalam menu. Setelah makanan selesai dibuat dari dapur maka yang pelayan antarkan pada pelanggan adalah sebuah response.

7.1.3. Arsitektur API

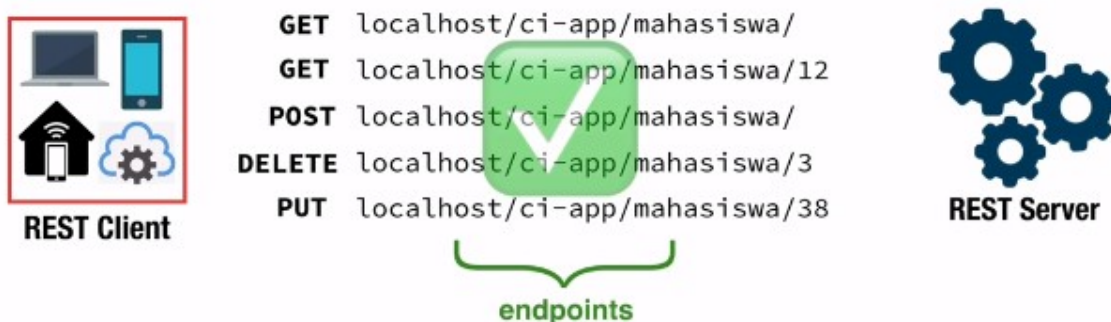
Hal pertama yang dibutuhkan pertama kali pada saat menggunakan REST API adalah REST Server, dimana bagian ini merupakan bagian yang akan menyediakan resource/data. Nantinya sebuah REST Client akan membuat HTTP Request ke server dan server akan merespon dengan mengirimkan sebuah HTTP Response sesuai request dari client.



Ilustrasi antara REST Client dan REST Server

HTTP Request memiliki beberapa komponen yang diataranya adalah sebagai berikut :

- **HTTP method** : GET, POST, PUT, DELETE dan lain lain yang sesuai dengan penggunaanya
- **URI** (*Uniform Resource Identifier*) untuk mengetahui lokasi data di server
- **HTTP Version** : contohnya HTTP v1.1
- **Request Header** : yang isinya berupa seperti Authorization, tipe client dan lain lain
- **Request Body** : data yang diberikan ke server misalnya URI params



Ilustrasi HTTP Methode dari REST Client

Sedangkan untuk HTTP Response memiliki komponen sebagai berikut :

- **Response code** : Berupa status server terhadap request, ini dibutuhkan karena yang terjadi adalah interaksi antar mesin. Mesin dapat mengetahui hasilnya dengan membaca status code nya misalnya 200, 401, 404 dan lainnya.
- HTTP Version : contohnya HTTP v1.1
- **Response Header** : berisi metadata seperti content type, cache tag dan lainnya
- **Response Body** : data resource yang diberikan oleh server baik berupa text, json atau xml

7.1.4. Kelebihan dan Kekurangan API

Meskipun API memiliki beragam fungsi yang memudahkan, ada beberapa hal yang menjadi kekurangan dan juga kelebihan dari service API ini yang diantaranya sebagai berikut.

Kelebihan API:

1. Bisa digunakan oleh banyak bahasa pemrograman dan banyak platform.
2. Lebih simple dibandingkan dengan SOAP.
3. Mudah dipelajari.
4. Menggunakan protokol HTTP.

Kekurangan API :

1. Waktu akses yang biasanya lebih lama dibandingkan dengan native library.
2. Lebih rentan dengan serangan keamanan karena harus melewati protokol HTTP.

7.2. Praktik API dengan NodeJS

7.2.1. Pengenalan NodeJS



Node.JS Logo

Node.js merupakan perangkat lunak yang didesain untuk mengembangkan aplikasi berbasis web dan ditulis dalam sintaks bahasa pemrograman JavaScript. Bila selama ini kita mengenal JavaScript sebagai bahasa pemrograman yang berjalan di sisi client / browser saja, maka Node.js ada untuk melengkapi peran JavaScript sehingga bisa juga berlaku sebagai bahasa pemrograman yang berjalan di sisi server, seperti halnya PHP, Ruby, Perl, dan sebagainya.

Node.js dapat berjalan di sistem operasi Windows, Mac OS X dan Linux tanpa perlu ada perubahan kode program. Node.js memiliki pustaka server HTTP sendiri sehingga memungkinkan untuk menjalankan server web tanpa menggunakan program server web seperti Apache atau Nginx.

Pada pembahasan ini kita tidak akan membahas bahasa pemrograman NodeJS ini lebih dalam, karena disini kita hanya akan mencoba sebagian penggunaan dasar API pada Node.JS.

Selebihnya untuk memperdalam bahasa NodeJS ini kalian dapat coba membaca beberapa referensi yang tersebar di internet.

7.2.2. Persiapan

Untuk mulai menggunakan NodeJS, hal pertama yang harus kita lakukan adalah menginstallkan NodeJS terlebih dahulu di komputer kita atau menggunakan Instance EC2 AWS yang sudah terinstall Webserver atau komputer lokal yang terinstall XAMPP/LAMPP.

7.2.2.1. Instalasi NodeJS Windows

Untuk pengguna sistem operasi windows, kalian dapat mengunduh terlebih dahulu NodeJS di website resminya, tahap installasinya tidak berbeda jauh dengan instalasi software seperti biasanya. Berikut merupakan link nya <https://nodejs.org/en/> .

7.2.2.2. Instalasi NodeJS Ubuntu

Sedangkan untuk instalasi di Ubuntu, pertama kita harus menambahkan dulu PPA di sistem sebelum menginstall nodejs nya. Setelah itu baru kita bisa menginstall langsung nodejs nya.

```
curl -sL https://deb.nodesource.com/setup_12.x | sudo -E bash -
sudo apt-get install -y nodejs
```

```
vagrant@ubuntu-xenial:~$ curl -sL https://deb.nodesource.com/setup_12.x | sudo -E bash -
## Installing the NodeSource Node.js 12.x repo...
```

Menambahkan Repositori Node.js

```
vagrant@ubuntu-xenial:~$ sudo apt-get install -y nodejs
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
  libpython-stdlib libpython2.7-minimal libpython2.7-stdlib python python-minimal python2.7 python2.7-minimal
```

Instalasi Node.js

Setelah proses instalasi NodeJS pada ubuntu maupun windows selesai, selanjutnya buka terminal/cmd lalu kita cek versi dari NodeJS yang kita gunakan dengan menggunakan perintah berikut.

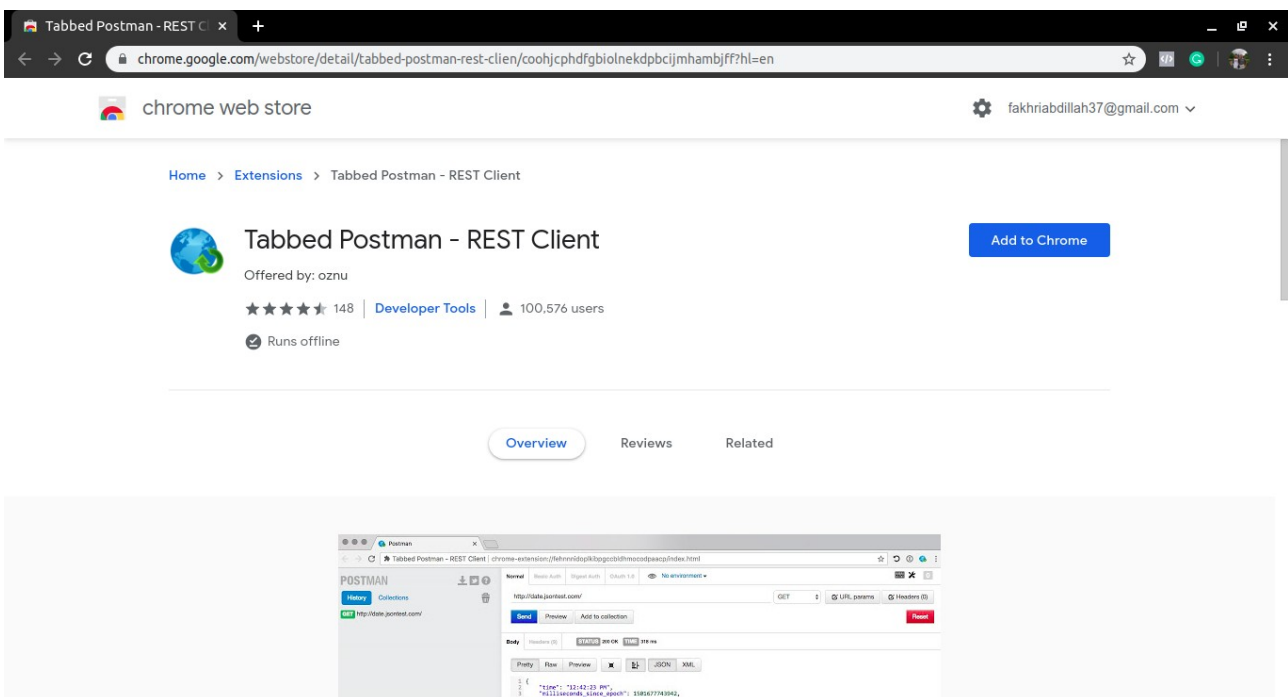
`npm -v`

```
vagrant@ubuntu-xenial:~$ npm -v
6.13.4
```

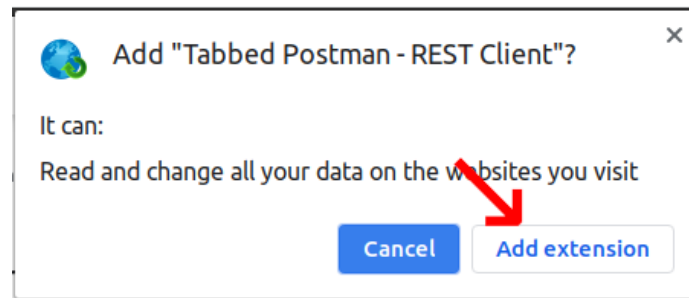
versi nodejs yang tersedia

7.2.2.3. Instalasi Extension Tabbed Postman

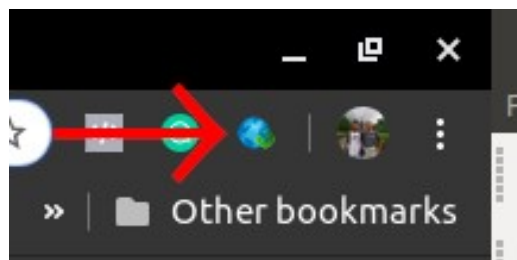
Buka link berikut (<https://bit.ly/2K6fosA>) untuk melakukan instalasi *extension pack* Postman pada *web browser* Google Chrome. Klik tombol biru bertuliskan **Add to Chrome**.



Pilih Add Extension



Untuk melakukan pengujian, Anda tinggal menekan *icon extension* tersebut yang berada di sebelah kanan atas browser anda.



Untuk alternatif lain, kita bisa menggunakan **Postman API Client** yang bisa kita unduh di link <https://www.postman.com/downloads/>.

7.2.3. Setup dan Praktek Pengerjaan

Hal yang perlu kita lakukan pertama adalah masuk ke direktori dimana kita akan mengerjakan project ini. Jika belum kalian buat terlebih dahulu direktory nya disini kami menggunakan direktori **nodejs-project**. Kemudian ketikan perintah seperti dibawah ini.

```
npm init
```

Fungsi dari perintah tersebut adalah untuk melakukan generate pada file **package.json**. Setelah itu, kita cukup menekan tombol enter saja sampai selesai jika ingin semua settingannya default. Meskipun begitu kita masih bisa mengubahnya konfigurasi tersebut.

```
vagrant@ubuntu-xenial:~$ mkdir nodejs-project
vagrant@ubuntu-xenial:~$ cd nodejs-project/
vagrant@ubuntu-xenial:~/nodejs-project$ npm init
This utility will walk you through creating a package.json file.
It only covers the most common items, and tries to guess sensible defaults.

See `npm help json` for definitive documentation on these fields
and exactly what they do.

Use `npm install <pkg>` afterwards to install a package and
save it as a dependency in the package.json file.

Press ^C at any time to quit.
package name: (nodejs-project)
version: (1.0.0)
description:
entry point: (index.js)
test command:
git repository:
keywords:
author:
license: (ISC)
About to write to /home/vagrant/nodejs-project/package.json:

{
  "name": "nodejs-project",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "author": "",
  "license": "ISC"
}

Is this OK? (yes)
vagrant@ubuntu-xenial:~/nodejs-project$
```

Konfigurasi nodejs npm init

Setelah itu, kita coba untuk membuat file bernama **server.js** yang akan menjadi handle request API kita. Lalu kita masukan script seperti berikut.

```
const express = require('express')
const app = express()
const port = 3000

app.get("/", (req, res) => {
  res.json(["Tony", "Lisa", "Michael", "Ginger", "Food"]);
});
```

```
app.listen(port, () => {
  console.log(`Example app listening at http://localhost:${port}`)
})
```

script dapat dilihat di

<https://gist.github.com/sdcilsy/9f33bf625f971e9e991b7dafcacc186c#file-server-init-js>

Selanjutnya, kita jalankan API server kita menggunakan perintah berikut.

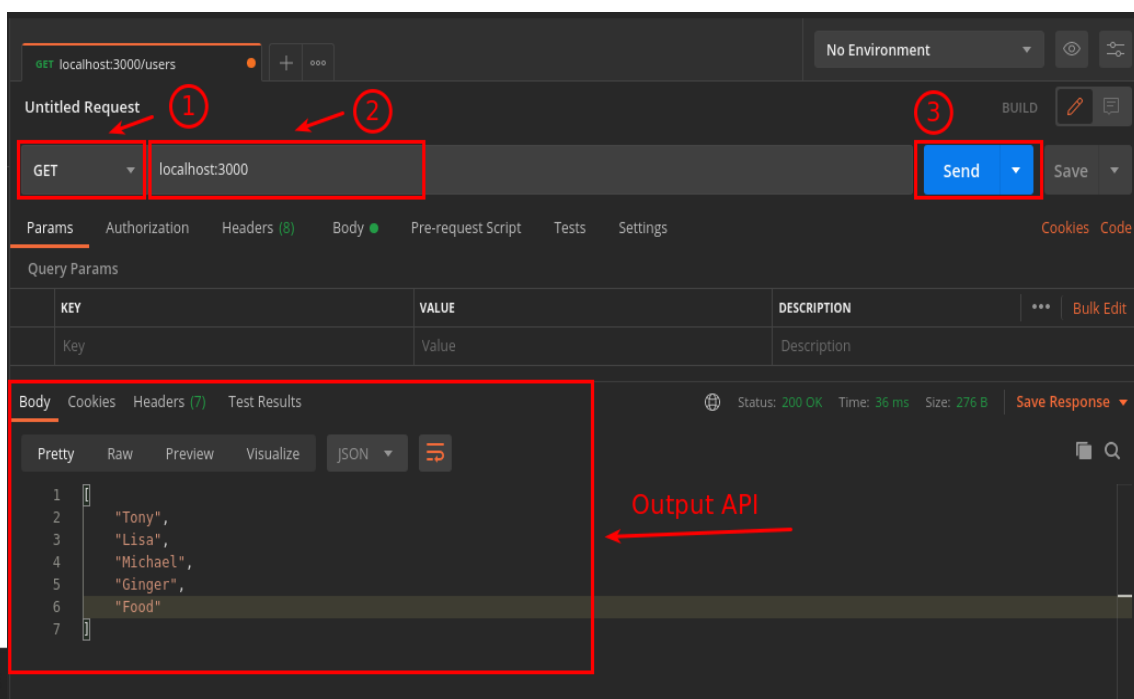
```
npm start
```

Jika server berjalan, maka output akan seperti berikut

```
taufik@hewlettpackard: $ npm start
> nodejs-project(hello-world-api)@1.0.0 start
> node server.js

Example app listening at http://localhost:3000
```

Lalu kita akan mencoba mengakses API server yang sudah kita buat menggunakan aplikasi **Postman** dengan method GET dan alamat **localhost:3000**.



Kita sudah berhasil membuat API server sederhana dengan method **GET**. Yaaaayyy.

Selanjutnya kita akan mengupgrade **Server API** kita dengan ke integrasi ke **MySQL** database, dan **CRUD** menggunakan method POST, GET, PUT dan DELETE.

Kita perlu menginstallkan beberapa package yang akan digunakan, caranya dengan menggunakan perintah seperti dibawah.

```
npm install --save mysql body-parser
```

```
vagrant@ubuntu-xenial:~/nodejs-project$ npm install --save express mysql body-parser
npm notice created a lockfile as package-lock.json. You should commit this file.
npm WARN nodejs-project@1.0.0 No description
npm WARN nodejs-project@1.0.0 No repository field.

+ body-parser@1.19.0
+ mysql@2.18.1
+ express@4.17.1
added 59 packages from 48 contributors and audited 171 packages in 4.227s
found 0 vulnerabilities

vagrant@ubuntu-xenial:~/nodejs-project$
```

Instalasi mysql dan body-parser

Tunggu sampai prosesnya selesai. Setelah itu masuk kedalam mysql dan buat sebuah database baru dengan nama **nodejs_api**.

```
mysql> create database nodejs_api;
Query OK, 1 row affected (0.00 sec)

mysql> use nodejs_api;
Database changed
```

Kemudian buat tabel baru dengan requirement sebagai berikut.

```
CREATE TABLE IF NOT EXISTS `person` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `first_name` varchar(50) NOT NULL DEFAULT '0',
  `last_name` varchar(50) NOT NULL DEFAULT '0',
  PRIMARY KEY (`id`)
);
```


Lakukan juga konfigurasi berikut.

```
ALTER USER 'root'@'localhost' IDENTIFIED WITH mysql_native_password BY
'<password>';
flush privileges;
```

Konfigurasi diatas berguna untuk mencegah adanya error NodeJS terhadap versi MySQL terbaru, karena adanya caching_sha2_password pada MySQL versi 8.0, sementara Node.js belum mengimplementasikannya.

Setelah database setup, selanjutnya kita edit file **server.js** di direktori **nodejs-project** yang sudah kita buat tadi. Isikan script berikut didalamnya.

```
var express = require('express'),
    app = express(),
    port = process.env.PORT || 3000,
    bodyParser = require('body-parser'),
    controller = require('./controller');

app.use(bodyParser.urlencoded({ extended: true }));
app.use(bodyParser.json());

var routes = require('./routes');
routes(app);

app.listen(port);
console.log('Learn Node JS With Kiddy, RESTful API server started on: ' + port);
```

```

server.js  x  conn.js  x  controller.js  x  res.js
1  var express = require('express'),
2      app = express(),
3      port = process.env.PORT || 3000,
4      bodyParser = require('body-parser'),
5      controller = require('./controller');
6
7  app.use(bodyParser.urlencoded({ extended: true }));
8  app.use(bodyParser.json());
9
10 var routes = require('./routes');
11 routes(app);
12
13 app.listen(port);
14 console.log('Learn Node JS With Kiddy, RESTful API server started on: ' + port);

```

Isi file server.js

script dapat dilihat di

<https://gist.github.com/sdcilasy/9f33bf625f971e9e991b7dafcacc186c#file-server-js>

Lanjut kita membuat file baru lagi dengan nama **conn.js** yang berfungsi sebagai tempat mengkoneksikan aplikasi ke database. Isikan script seperti dibawah ini.

```

var mysql = require('mysql');

var con = mysql.createConnection({
  host: "localhost",
  user: "root",
  password: "",
  database: "nodejs_api"
});

con.connect(function (err){mysql
  if(err) throw err;
});

module.exports = con;

```

script dapat dilihat di

<https://gist.github.com/sdcilisy/9f33bf625f971e9e991b7dafcacc186c#file-conn-js>

Buat juga file dengan nama **controller.js**, isikan dengan script seperti dibawah ini.

```
'use strict';

var response = require('./res');
var connection = require('./conn');

exports.users = function(req, res) {
    connection.query('SELECT * FROM person', function (error, rows, fields){
        if(error){
            console.log(error)
        } else{
            response.ok(rows, res)
        }
    });
};

exports.index = function(req, res) {
    response.ok("Hello from the Node JS RESTful side!", res)
};
```

script dapat dilihat di

<https://gist.github.com/sdcilisy/9f33bf625f971e9e991b7dafcacc186c#file-controller-init-js>

Kita juga perlu membuat file **res.js** yang berfungsi sebagai standarisasi respon API nya.

```
'use strict';

exports.ok = function(values, res) {
```

```
var data = {
  'status': 200,
  'values': values
};
res.json(data);
res.end();
};
```

script dapat dilihat di

<https://gist.github.com/sdcilsy/9f33bf625f971e9e991b7dafcacc186c#file-res-js>

Tahap selanjutnya kita buat file **routes.js** yang fungsinya sebagai tempat kita menyimpan routes atau endpoint yang ada pada rest api kita.

```
'use strict';
module.exports = function(app) {
  var todoList = require('./controller');

  app.route('/').get(todoList.index);

  app.route('/users').get(todoList.users);
};
```

script dapat dilihat di

<https://gist.github.com/sdcilsy/9f33bf625f971e9e991b7dafcacc186c#file-routes-init-js>

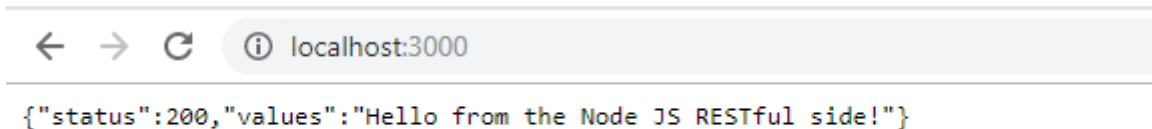
Setelah kita selesai membuat beberapa file yang kita butuhkan, selanjutnya kita coba untuk menjalankan nodejs yang sudah kita config. untuk menjalankannya kita perlu mengetikkan perintah seperti pada gambar dibawah ini.

Node server.js

```
vagrant@ubuntu-xenial:~/nodejs-project$ node server.js
Learn Node JS With Kiddy, RESTful API server started on: 3000
```

NodeJS sudah berjalan

Setelah Nodejs berhasil dijalankan seperti gambar diatas, selanjutnya kita coba akses nodejs tersebut di browser atau menggunakan postman dengan memasukan alamat komputer yang kita installkan NodeJS atau localhost bila menginstall di local <http://127.0.0.1:3000>



NodeJS Sudah berhasil di browser

7.2.4. Memasukan fungsi CRUD

Selanjutnya setelah kita selesai melakukan setup pada NodeJS, kita akan coba menerapkan CRUD agar data dapat kita modifikasi menggunakan method GET, DELETE, POST.

CRUD adalah singkatan dari **Create Read Update Delete** , yang sering digunakan pada aplikasi-aplikasi pengolahan data yang kebanyakan menggunakan fungsi CRUD didalamnya.

Untuk memulainya kita akan menambahkan script pada file **controller.js**, selanjutnya masukan script dibawah ini.

```
'use strict';

var response = require('./res');
var connection = require('./conn');

exports.users = function(req, res) {
  connection.query('SELECT * FROM person', function (error, rows, fields){
    if(error){
      console.log(error)
    } else{
```

```

        response.ok(rows, res)
    }
});

};

exports.index = function(req, res) {
    response.ok("Hello from the Node JS RESTful side!", res)
};

exports.findUsers = function(req, res) {

    var user_id = req.params.user_id;

    connection.query('SELECT * FROM person where id = ?',
    [ user_id ],
    function (error, rows, fields){
        if(error){
            console.log(error)
        } else{
            response.ok(rows, res)
        }
    });
};

exports.createUsers = function(req, res) {

    var first_name = req.body.first_name;
    var last_name = req.body.last_name;

    connection.query('INSERT INTO person (first_name, last_name) values (?,?)',
    [ first_name, last_name ],
    function (error, rows, fields){
        if(error){
            console.log(error)

```

```

    } else{
        response.ok("Berhasil menambahkan user!", res)
    }
});
};

exports.updateUsers = function(req, res) {

    var user_id = req.body.user_id;
    var first_name = req.body.first_name;
    var last_name = req.body.last_name;

    connection.query('UPDATE person SET first_name = ?, last_name = ? WHERE id
= ?',
    [ first_name, last_name, user_id ],
    function (error, rows, fields){
        if(error){
            console.log(error)
        } else{
            response.ok("Berhasil merubah user!", res)
        }
    });
};

exports.deleteUsers = function(req, res) {

    var user_id = req.body.user_id;

    connection.query('DELETE FROM person WHERE id = ?',
    [ user_id ],
    function (error, rows, fields){
        if(error){
            console.log(error)
        } else{

```

```

        response.ok("Berhasil menghapus user!", res)
    }
});
};

```

script dapat dilihat di

<https://gist.github.com/sdcilisy/9f33bf625f971e9e991b7dafcacc186c#file-controller-js>

Jika selesai, kita tambahkan juga script berikut pada file **routes.js**.

```

'use strict';
module.exports = function(app) {
    var todoList = require('./controller');

    app.route('/').get(todoList.index);

    app.route('/users').get(todoList.users);

    app.route('/users/:user_id').get(todoList.findUsers);

    app.route('/users').post(todoList.createUsers);

    app.route('/users').put(todoList.updateUsers);

    app.route('/users').delete(todoList.deleteUsers);
};

```

script dapat dilihat di

<https://gist.github.com/sdcilisy/9f33bf625f971e9e991b7dafcacc186c#file-routes-js>

Setelah kita menambahkan beberapa script baru, selanjutnya kita jalankan kembali NodeJS menggunakan perintah berikut.

```
node server.js
```



```
vagrant@ubuntu-xenial:~/nodejs-project$ node server.js
Learn Node JS With Kiddy, RESTful API server started on: 3000
```

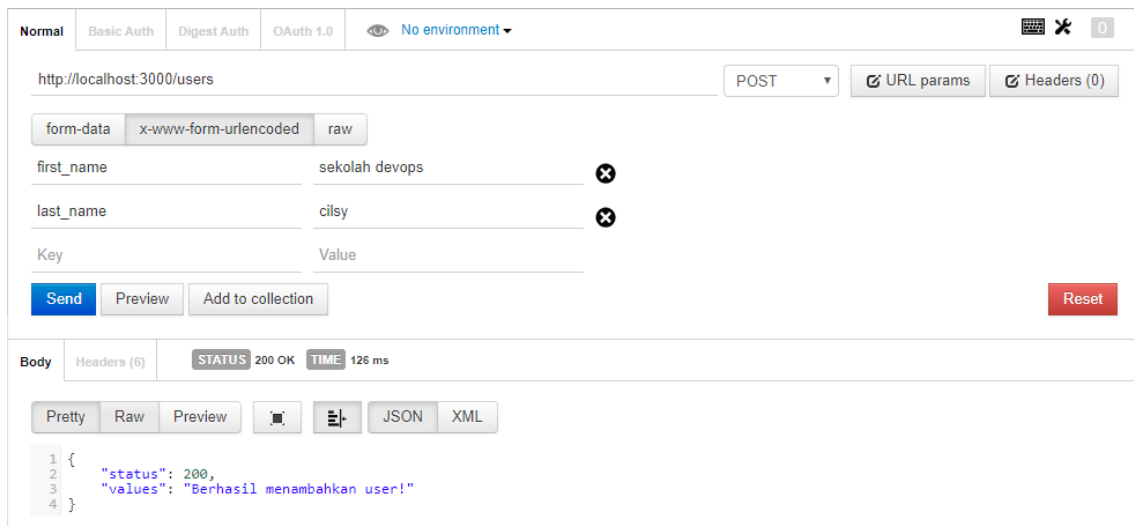
Hasil NodeJS berjalan

7.2.5. Pengujian

Pada tahap ini kita akan melakukan pengujian pada REST API NodeJS yang sudah kita konfigurasi tadi, kita akan test sama seperti pada pembahasan codeigniter menggunakan aplikasi postman.

7.2.5.1. Pengujian Menggunakan POST

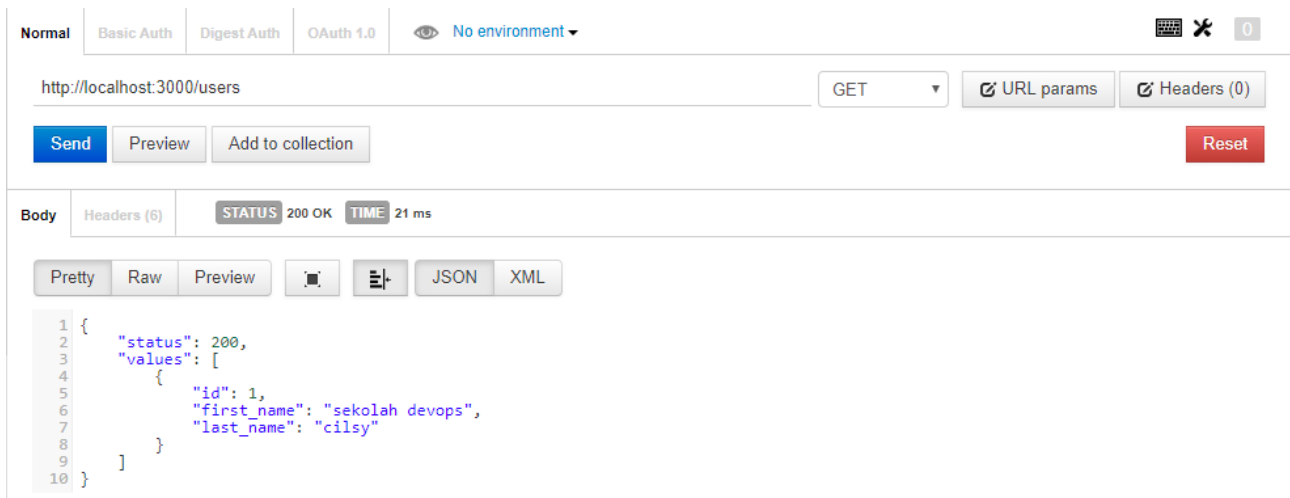
Pertama kita akan mencoba menggunakan metode POST pada Extension **Tabbed Postman** yang sudah di install sebelumnya dengan mengisi data baru pada aplikasi. Coba lakukan seperti pada gambar dibawah ini.



Hasil pengujian API NodeJS dengan POST

7.2.5.2. Pengujian Menggunakan GET

Selanjutnya kita akan coba menggunakan metode GET, kita dapat melihat isi dari aplikasi API yang kita buat tadi seperti dibawah ini.



Normal Basic Auth Digest Auth OAuth 1.0 No environment

http://localhost:3000/users GET URL params Headers (0)

Send Preview Add to collection Reset

Body Headers (6) STATUS 200 OK TIME 21 ms

Pretty Raw Preview JSON XML

```

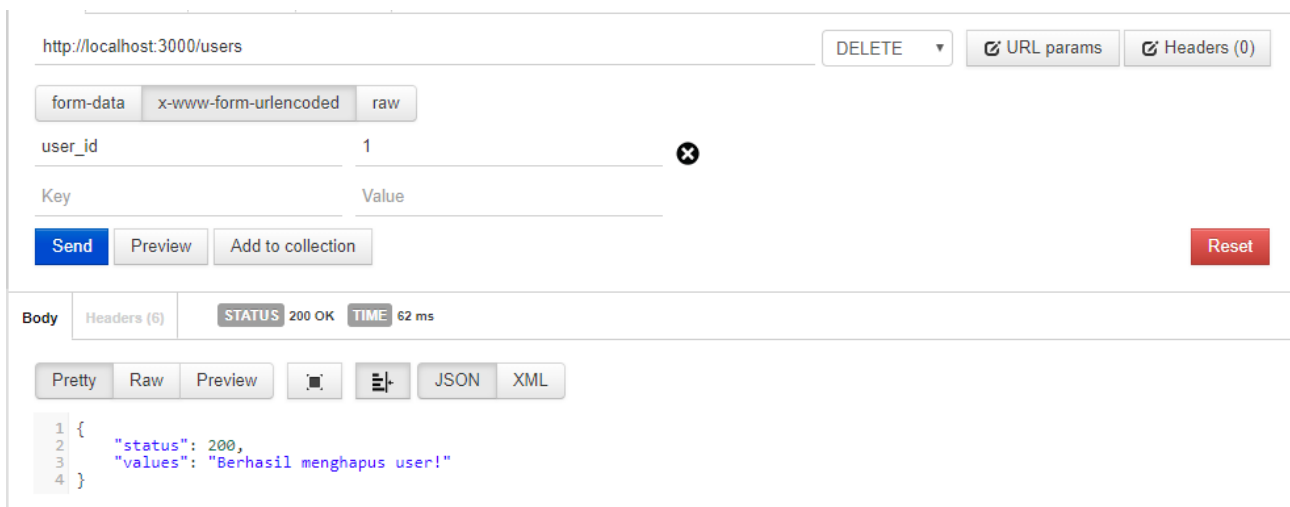
1 {
2   "status": 200,
3   "values": [
4     {
5       "id": 1,
6       "first_name": "sekolah devops",
7       "last_name": "cilsy"
8     }
9   ]
10 }

```

Hasil pengujian API NodeJS dengan GET

7.2.5.3. Pengujian Menggunakan DELETE

Pada bagian ini kita akan coba menghapus user berdasarkan id nya seperti dibawah ini.



http://localhost:3000/users DELETE URL params Headers (0)

form-data x-www-form-urlencoded raw

user_id 1

Key Value

Send Preview Add to collection Reset

Body Headers (6) STATUS 200 OK TIME 62 ms

Pretty Raw Preview JSON XML

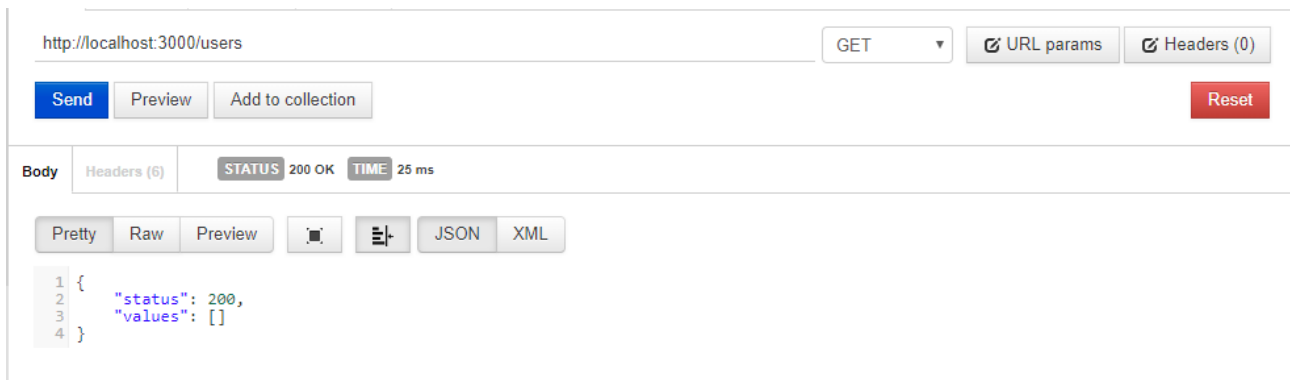
```

1 {
2   "status": 200,
3   "values": "Berhasil menghapus user!"
4 }

```

Pengujian API NodeJS dengan Delete

Jika berhasil, maka ketika kita menggunakan metode GET data akan terhapus.



Hasil Pengujian API NodeJS dengan Delete

Berikut merupakan serangkaian dari percobaan API menggunakan NodeJS.

7.3. Summary

Berikut merupakan rangkuman dari point-point yang sudah dipelajari pada BAB ini :

1. API merupakan sebuah interface yang berfungsi sebagai jembatan yang menghubungkan 2 program yang berbeda sehingga dapat memudahkan kegiatan yang akan dilakukan.
2. Rest API merupakan sebuah gaya perancangan yang memungkinkan terjadinya interaksi antar mesin contohnya misalnya aplikasi e-banking.
3. Pada praktik REST API kita menggunakan aplikasi codeigniter, nodejs, dan python sederhana dengan memanfaatkan extention postman pada chrome untuk melakukan pengujiannya.
4. Pengujian REST API menggunakan beberapa methode diantaranya GET, POST, DELETE, dan PUT

7.4. References

- <https://kiddyxyz.medium.com/tutorial-restful-api-node-js-express-mysql-part-1-527868bfb9d1>

- <https://kiddyxyz.medium.com/tutorial-restful-api-node-js-express-mysql-part-2-eff96ebb4b7f>
- <https://expressjs.com/en/starter/hello-world.html>
- <https://medium.com/@onejohi/building-a-simple-rest-api-with-nodejs-and-express-da6273ed7ca9>