MODUL 11
PEMROGRAMAN WEB
RESTful API

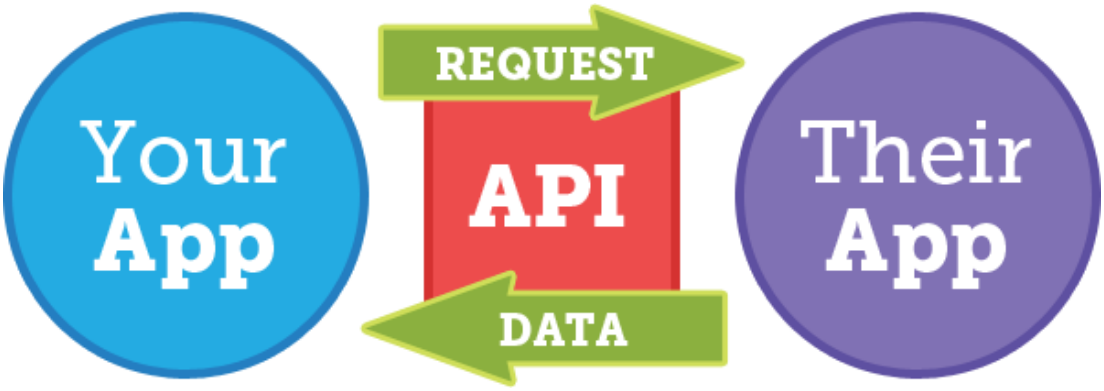
A. TUJUAN PERKULIAHAN

a. Mahasiswa memahami API dalam pemrograman komputer
b. Mahasiswa mengenali struktur penulisan REST
c. Mahasiswa mampu mengimplementasikan REST dengan PHP

B. ALOKASI WAKTU 3 x 50 menit

C. DASAR TEORI

1. API



API kependekan dari *Application Programming Interface* merupakan sebuah perangkat lunak yang dapat menerima panggilan atau permintaan dari perangkat lunak lainnya seperti aplikasi dan *website* yang memberikan pelayanan. Dengan kata lain API merupakan sebuah perangkat lunak yang mengijinkan atau menghubungkan dua aplikasi. Tidak hanya itu API juga digunakan untuk membuat *software* dan aplikasi oleh para *developer*.

API dapat Anda temui dalam kehidupan sehari-hari seperti saat Anda memesan hotel, mengirimkan pesan, memesan makanan secara *online* maupun ketika mengunduh sebuah *software*. Bagaimana hal tersebut terjadi? Prosesnya seperti ini, ketika Anda menggunakan aplikasi maupun melakukan aktifitas tersebut, aplikasi terhubung ke internet dan mengirimkan data ke *server*. Setelah itu, data tersebut diambil dan diterjemahkan oleh *server* yang kemudian melakukan aksi yang diperlukan sebagai hasilnya kemudian dikirimkan kembali ke *smartphone* Anda. Setelah menerima data tersebut, aplikasi

menerjemahkannya lalu memberikan informasi yang anda inginkan dengan cara yang mudah dibaca. Dengan adanya API semua dapat berjalan dengan lancar dan memudahkan komunikasi antar perangkat.

API membuat pemrograman menjadi lebih mudah dan mungkin. Seperti yang sudah disebutkan di atas, kebutuhan kita sebagai pelanggan dan khususnya bagi *developer* sangat dimudahkan dengan adanya API. Dengan melihat hal tersebut, peran dari API sendiri sangat berat terlebih untuk membuat tampilan sebuah aplikasi menjadi interaktif, mudah untuk digunakan, dan bersahabat untuk pengguna. Tidak hanya itu, API juga digunakan untuk berkomunikasi antara layanan-layanan. API memiliki peran yang sangat penting dalam teknologi.

2. Keuntungan API

a. Meningkatkan produktivitas

Karena permintaan untuk software modern terus meningkat, perusahaan mencari jalan yang cepat untuk membuat prototype dan membuat poduk baru. Jika Developers menghabiskan waktu untuk membangun aplikasi dari awal atau dari bawah keatas, ketika solusi yang diberikan serupa dengan dengan API yang sudah terekspos, maka hal tersebut akan berdampak negatif untuk produtivitas suatu perusahaan.

API menawarkan mekanisme untuk melakukan pengembangan yang lebih cepat. Dengan API, Developer dapat dengan cepat melakukan implementasi fungsi yang sudah ada dibandingkan membuat fungsi dari scratch atau dari awal.

b. Menghemat biaya (Aplikasi skala besar)

Biaya untuk membangun suatu aplikasi berbeda-beda tergantung pada beberapa faktor, termasuk pada kompleksitas proyek, jenis teknologi yang digunakan, dan keahlian developer.

Salah satu keuntungan terbesar API untuk bisnis adalah kemampuan untuk penghematan biaya. Karena API secara signifikan mengurangi upaya pengembangan, penggunaannya untuk membuat aplikasi merupakan cara yang baik untuk menghemat biaya.

Dengan API, developer dapat mengambil sebagian fungsi yang dibutuhkan untuk membuat aplikasi dari tempat lain tanpa harus membuat aplikasi dari awal. Dibandingkan harus menghabiskan sumber daya dan waktu yang berharga, developer bisa mengefektifkan biaya API dari penyedia pihak ketiga atau menggunakan API internal sendiri.

c. **Meningkatkan konektivitas dan kolaborasi**
   Private atau Internal API dapat meningkatkan kolaborasi dan komunikasi internal dalam suatu perusahaan. Fungsi utama dari API adalah Konekstivatas, API memungkinkan sistem, aplikasi dan platform yang berbeda dapat terhubung dan sharing data dengan yang lainnya dan melakukan berbagai jenis fungsi.

d. **Meningkatkan pengalaman customer**
   Dengan memanfaatkan kemampuan API, perusaahan dapat membuat produk baru dan merupakan cara efektif untuk berinteraksi dengan customer, khususnya di era digital seperti sekarang ini.

   Pada umumnya, customer saat ini tertarik pada pengalaman yang dipersonalisasi, daripada harus membuat solusi bisnis satu ukuran untuk semua. Dengan API, Developer dapat membuat solusi secara spesifik yang memenuhi harapan customer.

e. **Kemampuan untuk membuat produk baru**
   Menyelesaikan fungsionalitas suatu organisasi dan mengeksposnya menjadi sebuah API, menjadi tawaran yang menggiurkan untuk develeper external untuk membuat aplikasi diatasnya. Selain membangun hubungan partnership yang kuat, hal ini memungkinkan untuk menciptakan lebih banyak fitur khusus yang tidak dapat dibayangkan sebelumnya.

3. **Kelemahan API**

a. **Biaya besar (Aplikasi skala kecil)**
   API membutuhkan suatu biaya yang terbilang relatif mahal jika diimplementasikan pada aplikasi skala kecil, dalam hal waktu pengembangan, pemeliharaan berkelanjutan, menyediakan suatu dokumentasi API di suatu website.

b. **Potensi serangan yang besar**
   API memiliki keamanan yang standar sehingga potensi serangan ke web API sangat besar. Setiap URL API yang terekspos ke publik harus tidak memiliki celah sehingga perlu adanya autentikasi seperti penggunaan token atau password.

c. **Data kurang spesifik**
   API belum menentukan suatu hasil yang sesuai dengan kebutuhan atau ekspetasi kita. Misalnya sebuah API memberikan data tentang Kasus Covid di Seluruh Indonesia. Namun kita hanya memerlukan data untuk Penduduk DKI Jakarta, sehingga kita tidak butuh data daerah lain. Ini yang dimaksud belum sesuai kebutuhan atau ekspektasi. Dengan demikian, perlu menambahkan coding filtering khusus daerah tertentu

4. **Output API**

   Output dari RESTful ada XML dan JSON.

   JSON (JavaScript Object Notation), beberapa ciri-ciri JSON.

   1. Mudah di urai, bahkan oleh javascript pada sisi client, hanya object biasa
   2. Data transport lebih kecil
   3. Proses urai lebih cepat

   XML (Extensible Markup Language), beberapa ciri-ciri XML

   1. Butuh XML Document untuk mengurai (Xpath)
   2. Data transport lebih besar
   3. Proses urai lebih lambat

5. **REST dan RESTful**

   REST (Representational State Transfer) itu arsitektur sebuah software, sedangkan RESTful API itu merupakan salah satu model implementasi dari web service. RESTful API merupakan implementasi dari API. RESTful adalah protokol/aturan untuk melakukan REST. RESTful biasanya digunakan untuk merujuk ke layanan web yang mengimplementasikan arsitektur REST.

   Ada beberapa framework REST yang bagus bagi PHP, misalnya: Phalcon, Lumen, Slim, Laravel, Silex. Berikut adalah tabel metode dalam REST

Summary of HTTP Methods for RESTful APIs

The below table summarises the use of HTTP methods discussed above.

HTTP METHOD	CRUD	ENTIRE COLLECTION (E.G. /USERS)	SPECIFIC ITEM (E.G. /USERS/123)
POST	Create	201 (Created), 'Location' header with link to /users/{id} containing new ID.	Avoid using POST on single resource
GET	Read	200 (OK), list of users. Use pagination, sorting and filtering to navigate big lists.	200 (OK), single user. 404 (Not Found), if ID not found or invalid.
PUT	Update/Replace	405 (Method not allowed), unless you want to update every resource in the entire collection of resource.	200 (OK) or 204 (No Content). Use 404 (Not Found), if ID not found or invalid.
PATCH	Partial Update/Modify	405 (Method not allowed), unless you want to modify the collection itself.	200 (OK) or 204 (No Content). Use 404 (Not Found), if ID not found or invalid.
DELETE	Delete	405 (Method not allowed), unless you want to delete the whole collection — use with caution.	200 (OK). 404 (Not Found), if ID not found or invalid.

6. Key element pada RESTful API

a. Resources

Elemen kunci pertama adalah resource dari aplikasi itu sendiri sendiri. Mari kita asumsikan bahwa aplikasi web suatu perusahaan di server memiliki database karyawan. Contoh URL aplikasi web adalah **http://demo.com**. Sekarang untuk mengakses resource database karyawan melalui layanan REST, seseorang dapat memasukkan perintah **http://demo.com/employee/1**. Perintah ini memberi tahu server web untuk memberikan rincian karyawan yang nomor karyawannya 1.

b. Request Method

Elemen ini menjelaskan apa yang ingin Anda lakukan dengan resourcenya. Browser mengeluarkan perintah GET untuk menginstruksikan titik akhir yang diinginkan untuk mendapatkan data. Namun, ada banyak metode lain yang tersedia termasuk hal-hal seperti POST, PUT, dan DELETE. Jadi dalam kasus contoh **http://demo.com/employee/**1 , browser web sebenarnya mengeluarkan GET karena ingin mendapatkan detail catatan karyawan.

c. Request Headers

Request headers adalah instruksi tambahan yang dikirim dengan request. Elemen ini menentukan jenis respons yang diperlukan atau detail otorisasi

d. Request Body

Data biasanya dikirim dalam permintaan ketika permintaan dengan method POST dibuat ke layanan web REST. Dalam panggilan POST, klien sebenarnya memberi tahu layanan web REST bahwa ia ingin menambahkan resource ke server. Oleh karena itu, request perlu diisi dengan detail data yang diperlukan untuk ditambahkan ke server.

e. Response Body

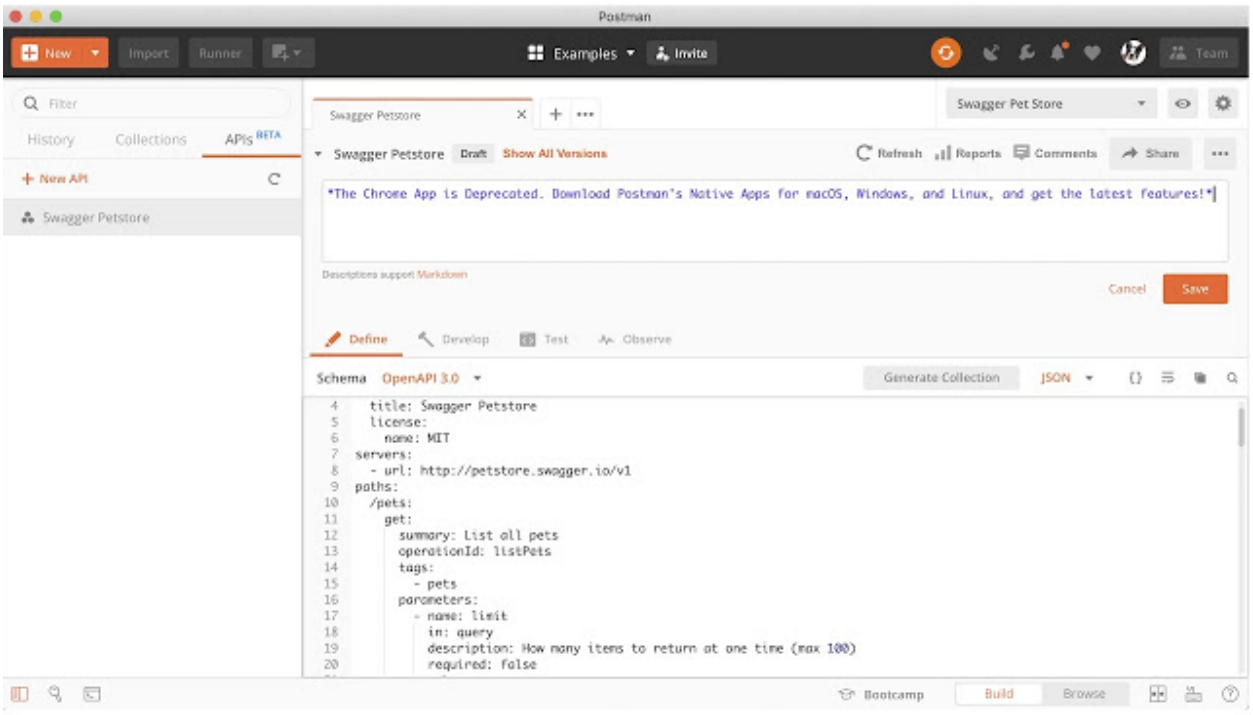
Response body adalah tubuh utama dari respon yang diharapkan. Dalam contoh RESTful API, jika kita meminta server web melalui permintaan **http://demo.com/employee/1** , server web akan mengembalikan dokumen berupa JSON atau XML dengan semua detail karyawan di response body.

f. Response Status codes

Kode-kode ini adalah kode umum yang dikembalikan bersama dengan respons dari server web. Contohnya adalah kode 200 yang biasanya dikembalikan jika tidak ada kesalahan saat mengembalikan respons ke klien. Atau kode 404 apabila alamat tidak ditemukan.

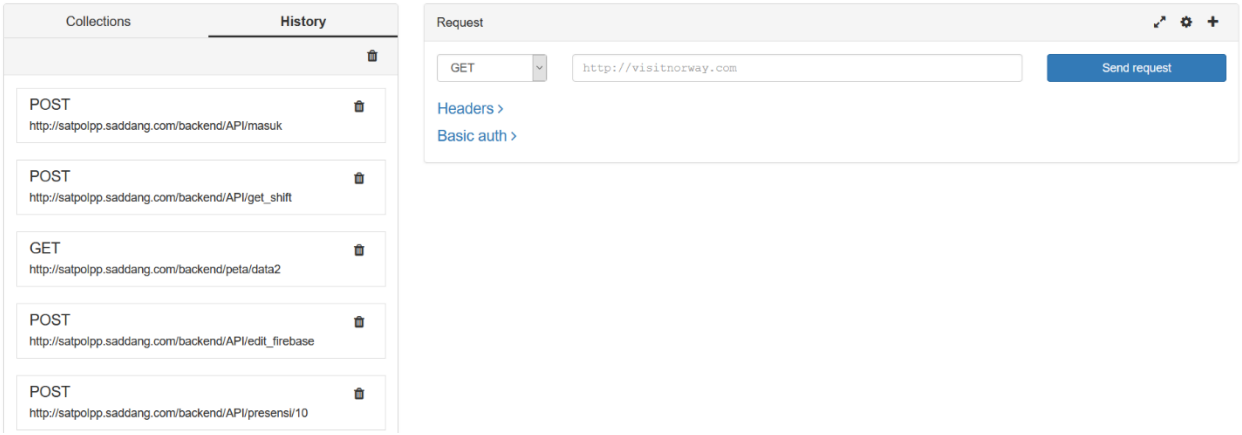
7. Tools dalam testing RESTful API

1. Postman

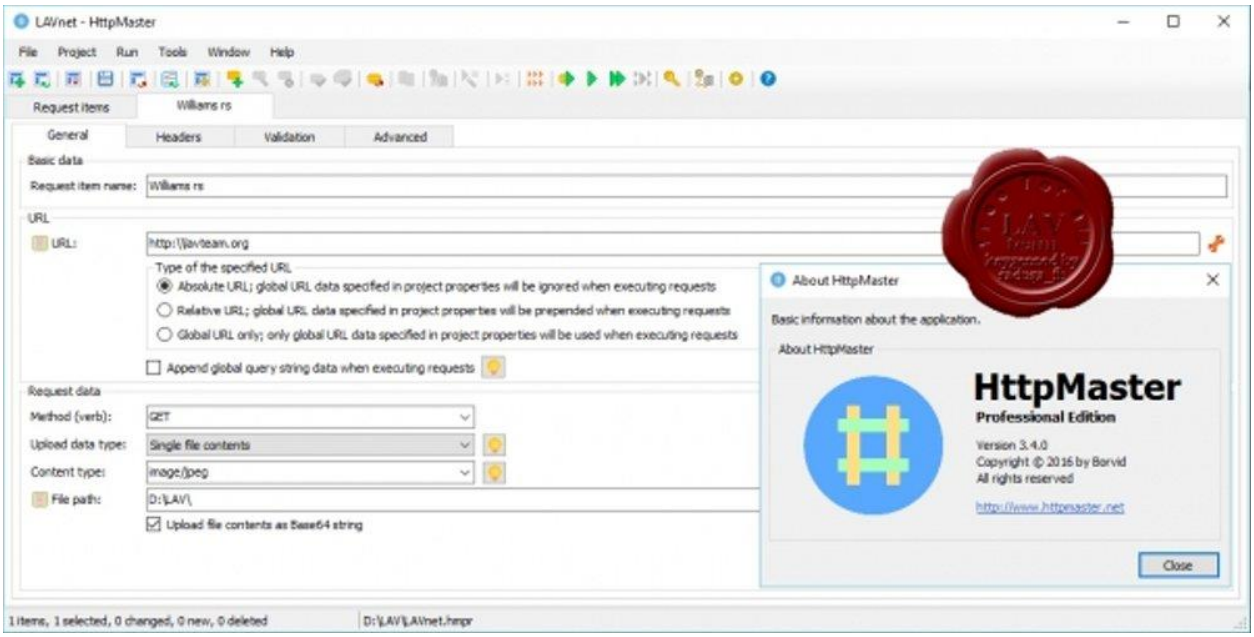


2. RESTED

</> RESTED



3. HTTPMaster



D. PRAKTIKUM : MEMBUAT RESTful API SEDERHANA

a. Brief Overview

Kita akan membuat REST API untuk modul pegawai yang akan menangani request HTTP Get, Post, Put dan Delete untuk mengambil, menambahkan, mengupdate, dan menghapus record-record dari database MySQL.

Buat folder 'pemweb/pertemuan11'di dalam root diectory dari web server, misalnya /var/www/html/ atau c:\xampp\htdocs\.

File-file yang akan digunakan selama percobaan turorial REST API ini:

1 index.php : Ini adalah file entri, sekaligus mencegah navigasi terhadap file-file di dalam foler.

2 koneksi.php : File ini digunakan untuk membangun koneksi ke database server MySQL.

3 .htaccess : File ini digunakan untuk mengatur redirect dan routing

Route	Method	JSON field	Deskripsi
/karyawan	GET	-	Meminta semua data pegawai
/karyawan/{id}	GET	-	Meminta data satu pegawai

/karyawan	POST	{“nama”:..., “email”:...., “jabatan”:...., “gaji”:... }	Menambahkan record pegawai baru ke dalam database
/karyawan/{id}	PUT	{“nama”:..., “email”:...., “jabatan”:...., “gaji”:... }	Mengupdate record pegawai tertentu
/karyawan/{id}	DELETE	-	Menghapus data pegawai tertentu

Tabel tersebut menunjukkan routing alamat yang dapat diakses serta hasil yang diharapkan dari akses dari masing-masing route

b. Membuat Skema Tabel Database

Buatlah suatu database MySQL bernama **“kantor”**, atau jika anda sudah mempunyai database dengan nama lain, maka cukup buat field berikut di dalam tabel bernama **tb_karyawan**.

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra
1	id	int(11)			No	None		AUTO_INCREMENT
2	nama	varchar(255)	utf8mb4_general_ci		No	None		
3	email	varchar(255)	utf8mb4_general_ci		No	None		
4	divisi	varchar(100)	utf8mb4_general_ci		No	None		
5	gaji	double			No	None		

c. Membuat File Koneksi

Buat file bernama **koneksi.php** dan menambahkan string koneksi MySQL untuk menghubungi MySQL dengan menggunakan metode mysqli_connect().

```php
<?php
class koneksiDB{
    function getKoneksi(){
        $host = "localhost";
        $username = "root";
        $pass = "";
        $db = "kantor";
        $konek = mysqli_connect($host, $username, $pass, $db) or die("Koneksi gagal ".
            mysqli_connect_error());

        if(mysqli_connect_errno()){
                exit();
        }
        return $konek;
    }
}
```

d. Membuat File .htaccess

buat file **.htaccess** (diawali dengan titik) di dalam folder dan memasukkan beberapa baris aturan untuk mengatur akses ke REST API dengan URL yang lebih singkat, bagus dan aman. Inilah baris-baris yang harus dimasukkan dalam file .htaccess tersebut.

```
RewriteEngine On
RewriteRule ^karyawan/?$ index.php [NC,L]
RewriteRule ^karyawan/([0-9]+)/?$ index.php?id=$1 [NC,L]
```

e. REST GET untuk mengambil semua data pegawai

Kita akan membuat suatu request REST HTTP jenis **GET** untuk mengakses semua record pegawai dari database MySQL. Kita akan menggunakan query MySQL untuk mengambil data dari tabel **tb_karyawan** dan mengirimkan array data JSON ke client sebagai obyek respon.Sekarang kita buat file bernama **index.php** dan tambahkan file koneksi MySQL untuk mengakses data di dalam tabel di server database MySQL.

```php
include_once("koneksi.php");
$db = new koneksiDB();
$koneksi = $db->getKoneksi();
$request = $_SERVER['REQUEST_METHOD'];
$uri_path = parse_url($_SERVER['REQUEST_URI'], PHP_URL_PATH);
$uri_segment = explode('/', $uri_path);
```

Kita menggunakan metode **$_SERVER** untuk mendapatkan informasi mengenai aksi, seperti request REST untuk menambahkan, mengedit atau menghapus.

Perintah **switch..case** digunakan untuk menyeleksi dan menentukan metode akses yang tepat untuk menangan resquest REST yang masuk

```
switch($request){
    case 'GET' :
        if(!empty($uri_segment[4])){
            $id = intval($uri_segment[4]);
            get_karyawan($id);
        } else {
            get_karyawan();
        }
        break;

    default:
    header("HTTP/1.0 405 Method Tidak Terdaftar");
    break;
}
```

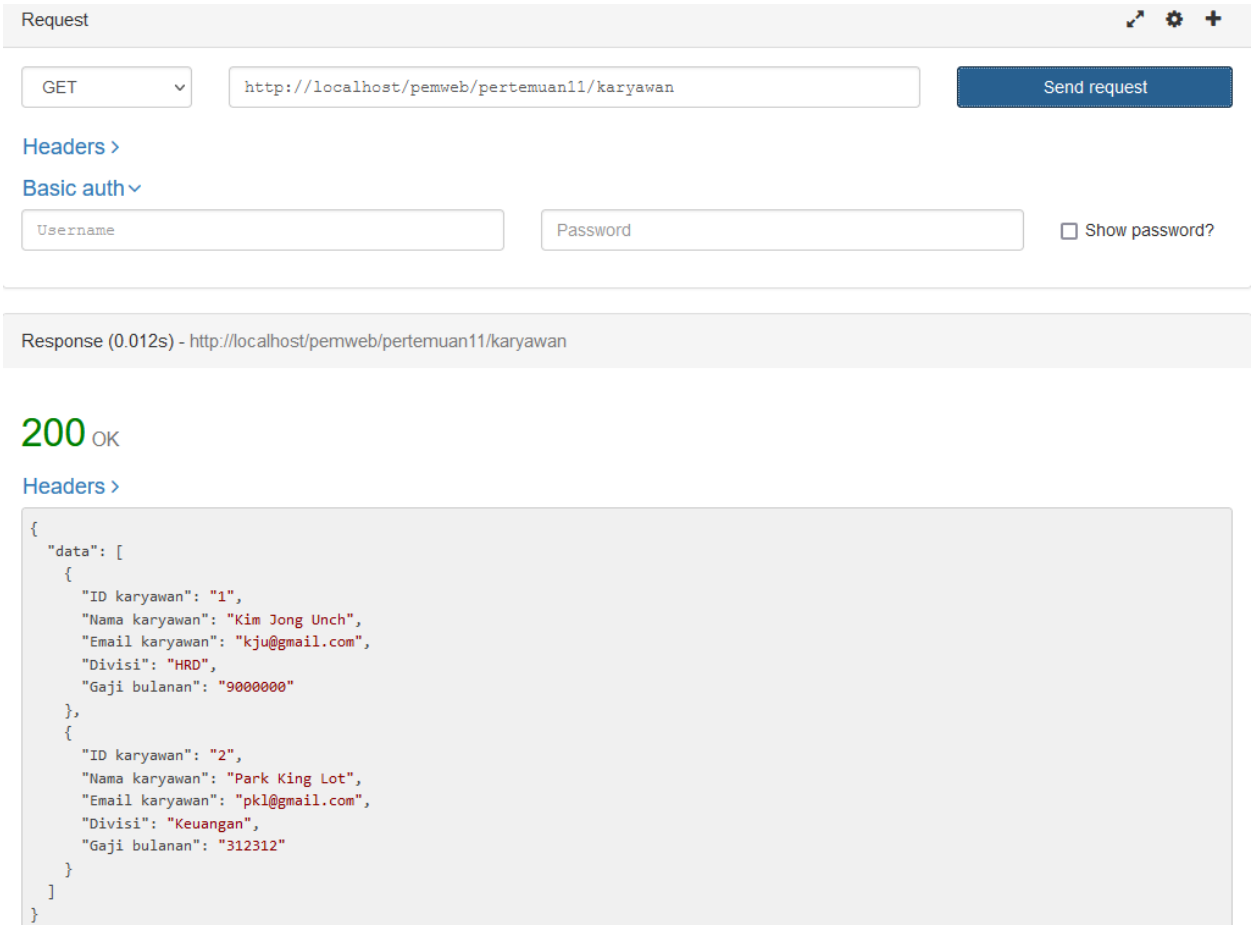
Kita memanfaatkan request GET untuk mengambil data semua pegawai dari database, sedangkan untuk pegawai tunggal (tertentu) perlu melewatkan karyawan id. Kita mendefinisikan metode **get_karyawan**() yang berisi kode di bawah ini:

```
function get_karyawan(){
    global $koneksi;
    $query= "SELECT * FROM tb_karyawan";
    $respon = array();
    $result = mysqli_query($koneksi, $query);
    $i = 0;
    while($row=mysqli_fetch_array($result)){
        $respon[] = $row;
    }
    header('Content-Type: application/json');
    echo json_encode($respon);
}
```

Metode **mysqli_query()** mengambil data dari tabel **tb_karyawan** di MySQL dan menyimpannya sebagai obyek ke dalam variabel 'result'. Metode **json_encode()** digunakan untuk mengkonversi array data ke dalam string json.

Silakan tambahkan satu atau dua record dahulu ke dalam database melalui PHPMyAdmin. Selanjutnya akses URL REST API dengan menggunakan tools testing API dengan alamat http://localhost/pemweb/pertemuan11/karyawan menggunakan web browser dan kita akan mendapatkan informasi tentang semua record pegawai yang ada di dalam tabel **tb_karyawan**.

Berikut contoh tampilan GET dari RESTED



f. REST GET untuk mengambil data pegawai tertentu

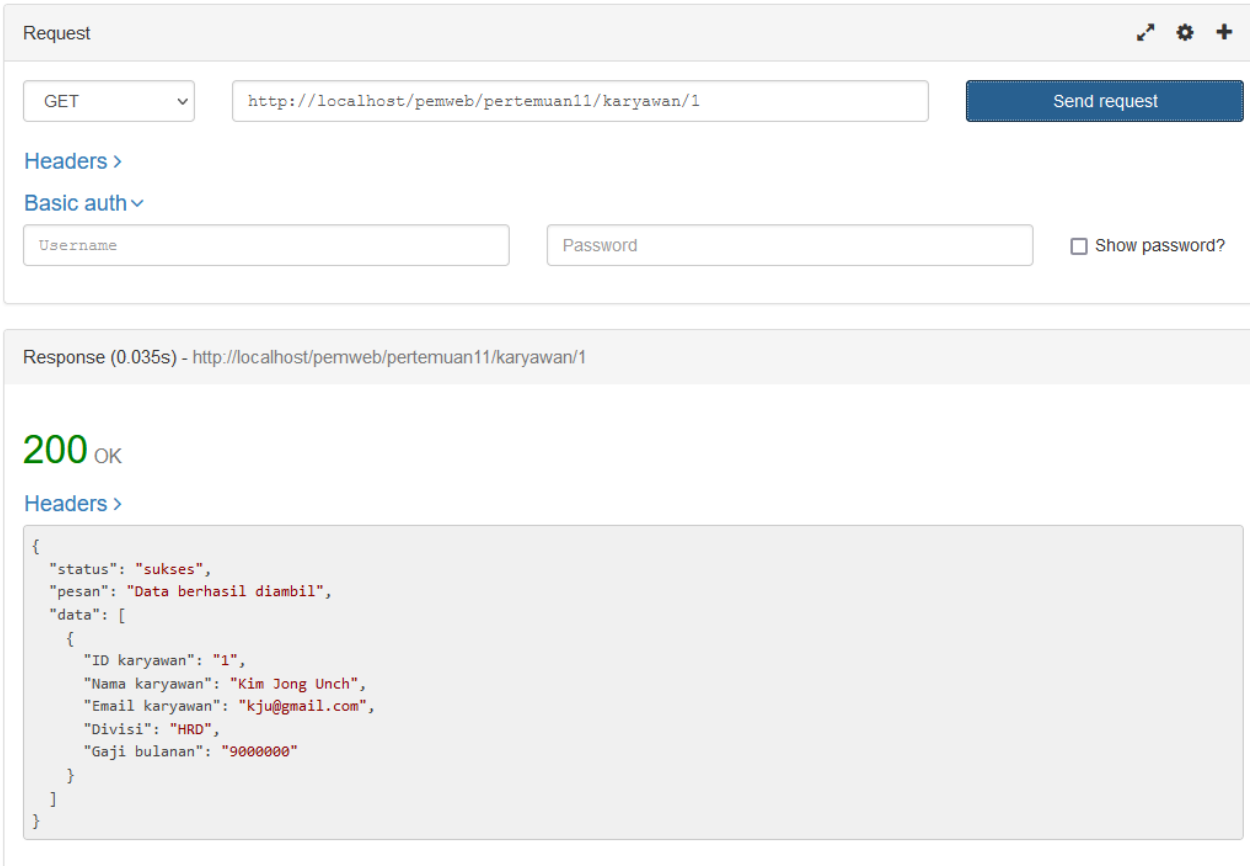
Kita akan menggunakan suatu request REST HTTP jenis GET untuk mengakses record tunggal(tertentu) dari pegawai dari database. Hampir sama dengan sebelumnya, tetapi diperlukan id dari pegawai yang akan diminta datanya. Fungsi **get_karyawan** di bawah ini merupakan modifikasi dari fungsi sebelumnya yang bernama sama.

Adapun kali ini kita tambahkan index **kode, status**, dan **data** sebagai penanda bahwa data yang kita kirim berhasil mengembalikan nilai atau tidak.

```php
function get_karyawan($id=""){
    global $koneksi;
    $query = "SELECT * FROM tb_karyawan";

    if(!empty($id)){
        $query .=" WHERE id=$id LIMIT 1";
    }

    $respon = array();
    $result = mysqli_query($koneksi, $query);
    $i=0;

    if($result){
        $respon['status']="sukses";
        $respon['pesan']="Data berhasil diambil";
        while ($row=mysqli_fetch_array($result)) {
            $respon['data'][$i]['ID karyawan'] = $row['id'];
            $respon['data'][$i]['Nama karyawan'] = $row['nama'];
            $respon['data'][$i]['Email karyawan'] = $row['email'];
            $respon['data'][$i]['Divisi'] = $row['divisi'];
            $respon['data'][$i]['Gaji bulanan'] = $row['gaji'];
            $i++;
        }
    } else {
        $respon['status']="gagal";
        $respon['pesan']="Data tidak berhasil diambil";
    }
    header('Content-type:application/json');
    echo json_encode($respon);
}
```

Silakan akses URL REST API http://localhost/pemweb/pertemuan11/karyawan/1 dari web browser atau tools API dan harusnya diperoleh sebaris record dari pegawai yang terdapat di dalam tabel tb_karyawan, tepatnya yang mempunyai Id = 1.

Berikut contoh tampilan method GET dari RESTED



g. REST POST untuk menambah data pegawai

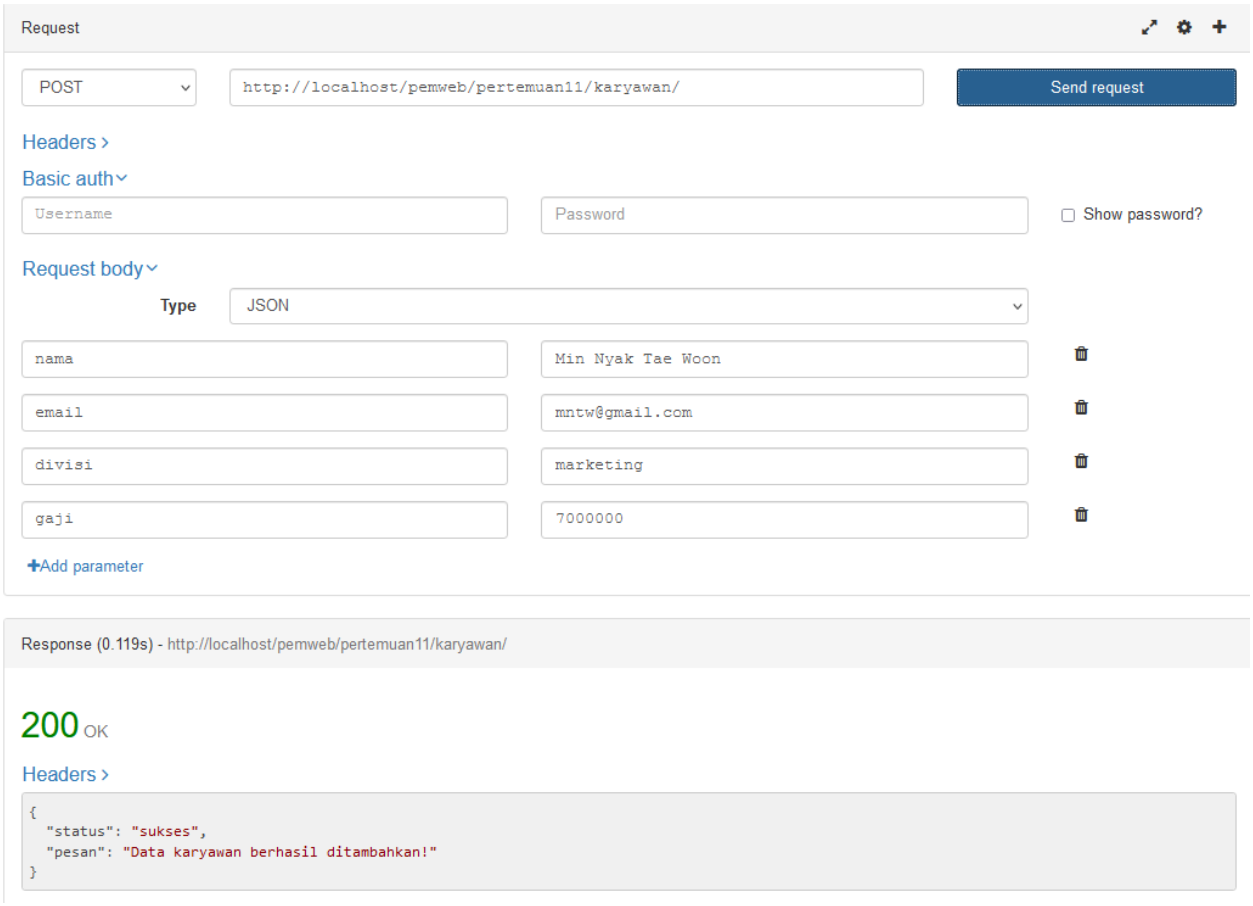
Kita akan membuatkan suatu REST API baru untuk menambahkan (insert) record pegawai baru ke dalam MySQL menggunakan PHP. Kita menggunakan request REST jenis **POST** karena kita akan menge"pos"kan beberapa data JSON ke server. Dalam struktur **switch** sebelumnya, silakan tambahkan case untuk "**POST**", seperti di bawah ini:

```
case 'POST' :
    insert_karyawan();
    break;
```

Fungsi yang akan menangani saat ada permintaan penambahan data baru adalah metode **insert_karyawan()** di dalam file **index.php**.

```
function insert_karyawan(){
    global $koneksi;
    $data = json_decode(file_get_contents('php://input'),true);
    $nama = $data['nama'];
    $email = $data['email'];
    $divisi = $data['divisi'];
    $gaji = $data['gaji'];

    $query = "INSERT INTO tb_karyawan SET nama='".$nama."',  email='".$email."',  divisi='".$divisi."
        ',  gaji='".$gaji."'";

    if(mysqli_query($koneksi, $query)){
        $respon = [
            'status'=>'sukses',
            'pesan'=>'Data karyawan berhasil ditambahkan!'
        ];
    } else {
        $respon = [
            'status'=>'gagal',
            'pesan'=>'Data karyawan gagal ditambahkan!'
        ];
    }
    header('Content-type:application/json');
    echo json_encode($respon);
}
```

Berikut contoh tampilan POST dari RESTED



Panggil method GET untuk mendapatkan seluruh data, amati penambahan data yang terjadi!

h. REST PUT untuk mengubah data pegawai

Selanjutnya adalah membuatkan kode program untuk menangani request REST API jenis **HTTP PUT**. Request ini ditujukan untuk mengupdate data yang telah ada di dalamdatabase MySQL. Update didasarkan pada Id dari pegawai, karena itu Id pegawai harus dijadikan sebagai parameter dari fungsi untuk menangani pembaruan record pegawai tersebut. Sebelumnya, kita harus menambahkan case baru "**PUT**" ke dalam struktur **switch...case**, seperti di bawah ini:

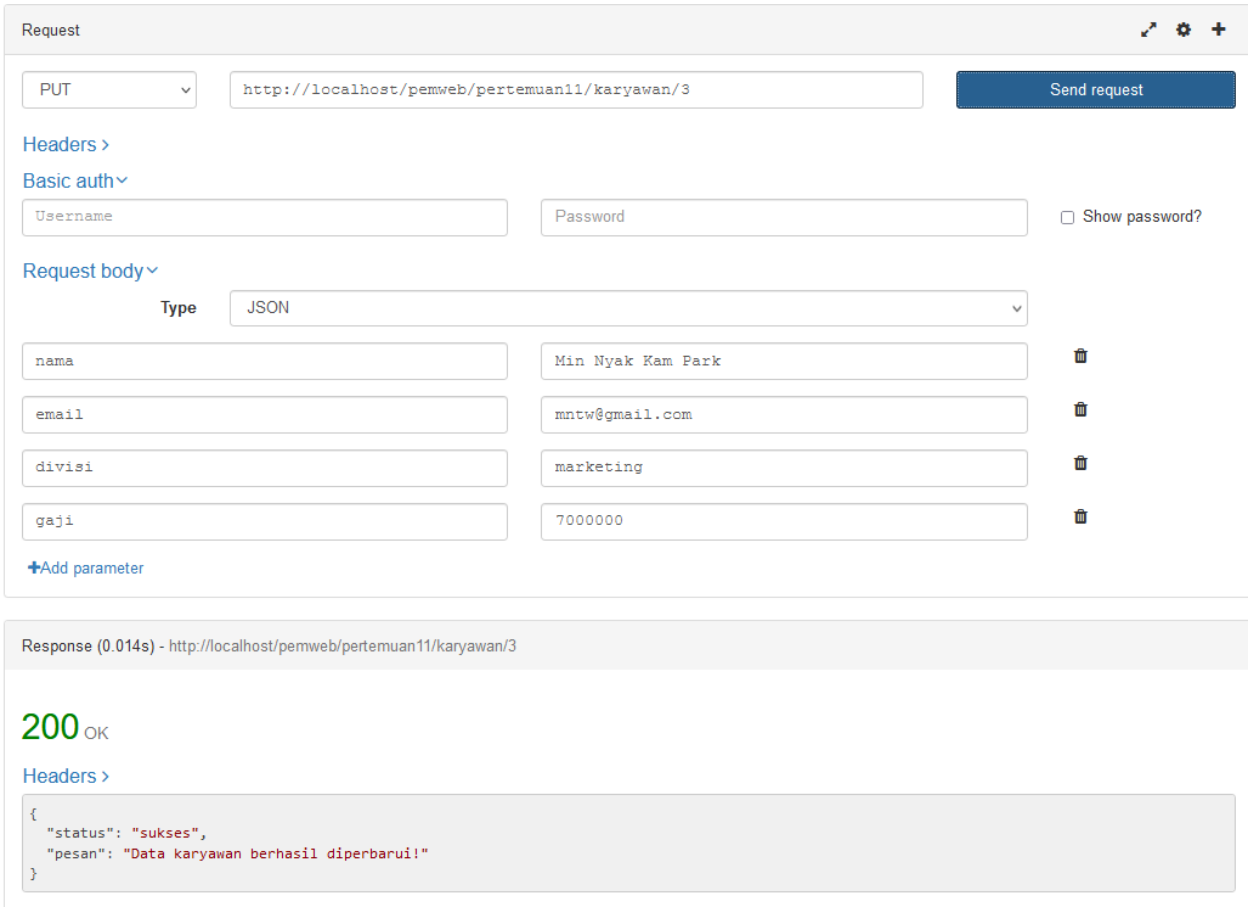
```
case 'PUT' :
    $id = intval($uri_segment[4]);
    update_karyawan($id);
    break;
```

Selanjutnya kita buat method **update_karyawan()** di dalam karyawans.php. Kita akan gunakan **id** yang akan digunakan sebagai parameter untuk update data dalam format json.

```
function update_karyawan($id){
    global $koneksi;
    $data = json_decode(file_get_contents('php://input'),true);
    $nama = $data['nama'];
    $email = $data['email'];
    $divisi = $data['divisi'];
    $gaji = $data['gaji'];

    $query = "UPDATE tb_karyawan SET nama='".$nama."',  email='".$email."',  divisi='".$divisi."',
        gaji='".$gaji."' WHERE id=".$id;

    if(mysqli_query($koneksi, $query)){
        $respon = [
            'status'=>'sukses',
            'pesan'=>'Data karyawan berhasil diperbarui!'
        ];
    } else {
        $respon = [
            'status'=>'gagal',
            'pesan'=>'Data karyawan gagal diperbarui!'
        ];
    }
    header('Content-type:application/json');
    echo json_encode($respon);
}
```

Perhatikan, jika update berhasil, pesan keberhasilan dikirimkan ke client REST API dalam format JSON.

Berikut contoh tampilan PUT dari RESTED



i. **REST DELETE untuk menghapus data pegawai**

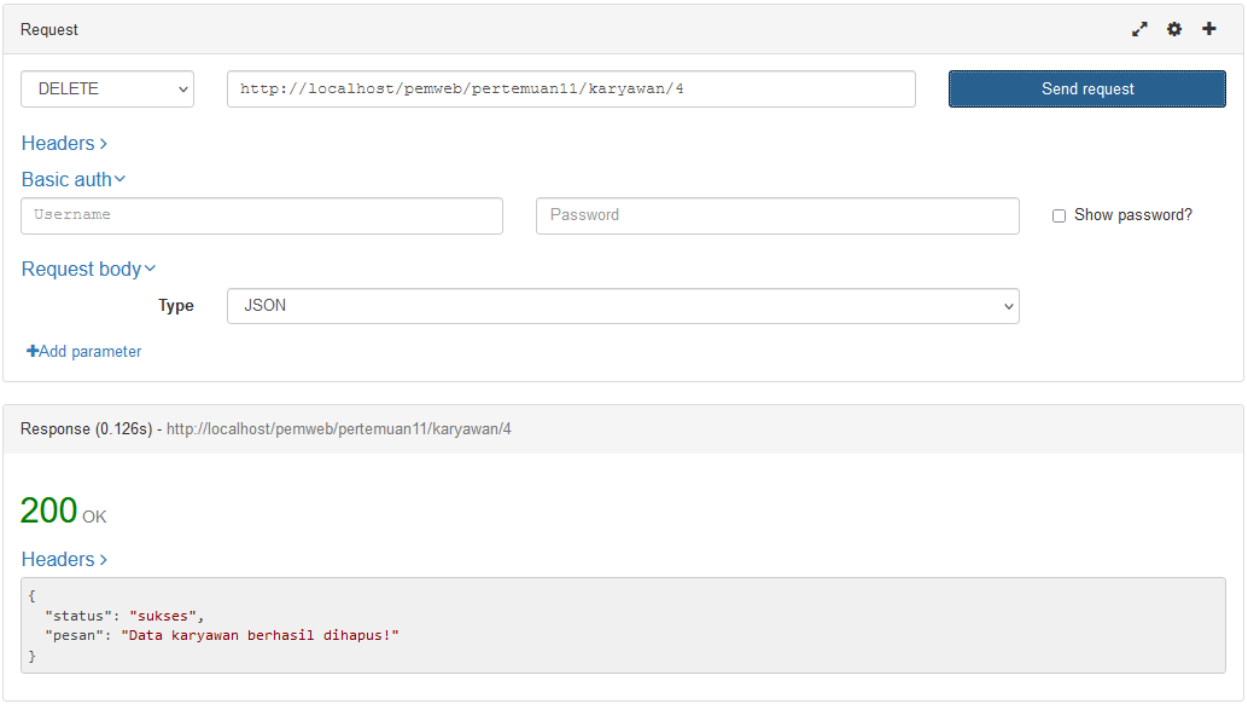
Terakhir, kita harus membuat fungsi untuk menangani request REST API jenis **DELETE**. Request ini meminta server API untuk menghapus record tertentu. Seperti pada CREATE dan UPDATE, DELETE juga memelukan Id Pegawai sebagai parameternya. Sebelumnya, kita perlu menambahkan case "**DELETE**" ke dalam struktur **switch ...case**, seperti berikut ini:

```
case 'DELETE' :
    $id = intval($uri_segment[4]);
    delete_karyawan($id);
    break;
```

Metode **delete_karyawan()** di dalam file **index.php** bertanggungjawab menangani penghapusan data pegawai tertentu. Field id digunakan sebagai kunci penghapusan tersebut.

```
function delete_karyawan($id){
    global $koneksi;

    $query = "DELETE FROM tb_karyawan WHERE id=".$id;

    if(mysqli_query($koneksi, $query)){
        $respon = [
            'status'=>'sukses',
            'pesan'=>'Data karyawan berhasil dihapus!'
        ];
    } else {
        $respon = [
            'status'=>'gagal',
            'pesan'=>'Data karyawan gagal dihapus!'
        ];
    }
    header('Content-type:application/json');
    echo json_encode($respon);
}
```

Berikut contoh tampilan DELETE dari RESTED



E. KETENTUAN TUGAS PRAKTIKUM

1. Buatlah Restful API seperti di contoh langkah di atas, namun ubahlah data yang digunakan menjadi "**universitas**", jadi gunakan tabel baru dengan nama "**mahasiswa**"
2. Field yang digunakan adalah "**NIM(varchar-15-Primary Key), nama (varchar-255), angkatan(int-4), semester(int-2), IPK(double), email(varchar-255), telepon(varchar-15)**"
3. File yang dikirim adalah .**htaccess, index.php, koneksi.php (jika terpisah dari index.php),** serta screenshot hasil uji **POST, GET, PUT** dan **DELETE**.
4. Tools pengujian API bebas, bisa gunakan Postman, RESTED, atau software lainnya.

F. REFERENSI

- https://medium.com/@ahmad.fight/perbedaan-rest-dengan-restfull-api-c08025d6d59e
- https://komputasi.files.wordpress.com/2019/03/membuat-restful-api-dengan-php-dan-mysqli.pdf
- https://www.restapitutorial.com/
- https://jsonplaceholder.typicode.com/
- https://dummy.restapiexample.com/
- https://manpro.id/blog/mengenal-manfaat-penggunaan-api/
- https://www.guru99.com/restful-web-services.html