**LAPORAN**

**REST FULL API MENGGUNAKAN PHP DAN MYSQL**

**KEAMANAN PERANGKAT LUNAK**



Dosen Pengampu :

Asmunin, S.Kom., M.Kom.

Disusun Oleh :

Mokhammad Brian Yusuf

(20051397036)

**PROGRAM STUDI MANAJEMEN INFORMATIKA**

**PROGRAM VOKASI**

**UNIVERSITAS NEGERI SURABAYA**

**2022**

RESTful API / REST API merupakan implementasi dari API (Application Programming Interface). REST (Representional State Transfer) adalah suatu arsitektur metode komunikasi yang menggunakan protokol HTTP untuk pertukaran data dan metode ini sering diterapkan dalam pengembangan aplikasi. Dimana tujuannya adalah untuk menjadikan sistem yang memiliki performa yang baik, cepat dan mudah untuk di kembangkan (scale) terutama dalam pertukaran dan komunikasi data.

Rest API atau sering juga disebut dengan istilah API saja sering digunakan sebagai penghubung antara ***front*–*end*** dan ***back*–*end*** aplikasi. *Front*–*end* sendiri aplikasi *interface* yang digunakan *user*atau pengguna aplikasi. Contoh dari *front-end*aplikasi yaitu aplikasi android dan iOS. *Back*–*end* aplikasi yang digunakan *administrator* untuk manajemen data. Sehingga dengan Rest API data dari front-end ke back-end ataupun sebaliknya dapat diolah seperti menampilkan, menambahkan, mengubah dan menghapus data.

**RESTful API memiliki 4 komponen penting di dalamnya diantaranya adalah :**

1. **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
2. **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 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.

1. **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.

Dan berikut ini 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.

1. **Format Response**  
   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”

**CARA MEMBUAT REST API**

1. } **Endpoint**  
   Dalam membuat API, **endpoint** adalah istilah yang digunakan untuk menyebutkan URL pada API yang sedang dibangun. Misalnya kita punya URL [http://belajaraplikasi.com/auth/token](https://belajaraplikasi.com/auth/token), kita bisa menyebutnya endpoint **/auth/token**. Format endpoint dapat disesuaikan dengan kebutuhan.

|  |  |  |  |
| --- | --- | --- | --- |
| **Route** | **Request Method** | **Body Request** | **Description** |
| /mahasiswa | GET | – | Get all mahasiswa data |
| /mahasiswa/{id} | GET | – | Get a single mahasiswa data |
| /mahasiswa | POST | id nim jk alamat jurusan | Insert new mahasiswa record into database |
| /mahasiswa/{id} | POST | nim jk alamat jurusan | Update mahasiswa record into database. |
| /mahasiswa/{id} | DELETE | – | Delete particular mahasiswa record from database |

1. **Database**

Buatlah database dengan nama db\_kampus, lalu eksekusi perintah sql dibawah ini untuk membuat table.

USE `db\_kampus`;

*/\*Table structure for table `tbl\_mahasiswa` \*/*

**DROP** **TABLE** IF EXISTS `tbl\_mahasiswa`;

**CREATE** **TABLE** `tbl\_mahasiswa` (

  `id` **int**(11) NOT NULL AUTO\_INCREMENT,

  `nim` **char**(8) **DEFAULT** NULL,

  `nama` **varchar**(200) **DEFAULT** NULL,

  `jk` **char**(1) **DEFAULT** NULL,

  `alamat` text,

  `jurusan` **varchar**(200) **DEFAULT** NULL,

**PRIMARY** **KEY** (`id`)

) ENGINE=InnoDB AUTO\_INCREMENT=3 **DEFAULT** CHARSET=latin1;

*/\*Data for the table `tbl\_mahasiswa` \*/*

**insert**  **into** `tbl\_mahasiswa`(`id`,`nim`,`nama`,`jk`,`alamat`,`jurusan`) **values**

(1,'15119999','muhammad al fatih','L','Jl. P. Puger 2 No 19','Informatika'),

(2,'16119212','Fatimah','P','Jl. Ringin Raya','Sistem Informasi');

1. **Koneksi**  
   Terlebih dahulu membuat folder untuk tempat projek dengan nama **api**lalu membuat file **koneksi.php**didalam folder tersebut, file koneksi.php bertujuan untuk membuat koneksi ke database. Berikut sintaksnya:

<?php

*// Membuat variabel, ubah sesuai dengan nama host dan database pada hosting*

$host = "localhost";

$user = "root";

$pass = "";

$db   = "db\_kampus";

*//Menggunakan objek mysqli untuk membuat koneksi dan menyimpan nya dalam variabel $mysqli*

$mysqli = **new** mysqli($host, $user, $pass, $db);

?>

1. **Mempercantik URL dengan .htaccess**  
   Buatlah file .htaccess dan tuliskan beberapa rule seperti di bawah:

RewriteEngine On # Turn on the rewriting engine

RewriteRule ^mahasiswa/?$ mahasiswa.php [NC,L]

RewriteRule ^mahasiswa/([0-9]+)/?$ mahasiswa.php?id=$1 [NC,L]

1. **Method**  
   Selanjutnya kita akan membuat beberapa method untuk mengakses database dan membuat kembalian data dalam format **JSON**, Buatlah file **method.php**dan Berikan sintaks seperti di bawah ini:

<?php

**require\_once** "koneksi.php";

**class** Mahasiswa

{

**public**  **function** get\_mhss()

   {

**global** $mysqli;

      $query="SELECT \* FROM tbl\_mahasiswa";

      $data=**array**();

      $result=$mysqli->query($query);

**while**($row=mysqli\_fetch\_object($result))

      {

         $data[]=$row;

      }

      $response=**array**(

                     'status' => 1,

                     'message' =>'Get List Mahasiswa Successfully.',

                     'data' => $data

                  );

      header('Content-Type: application/json');

      echo json\_encode($response);

   }

**public** **function** get\_mhs($id=0)

   {

**global** $mysqli;

      $query="SELECT \* FROM tbl\_mahasiswa";

**if**($id != 0)

      {

         $query.=" WHERE id=".$id." LIMIT 1";

      }

      $data=**array**();

      $result=$mysqli->query($query);

**while**($row=mysqli\_fetch\_object($result))

      {

         $data[]=$row;

      }

      $response=**array**(

                     'status' => 1,

                     'message' =>'Get Mahasiswa Successfully.',

                     'data' => $data

                  );

      header('Content-Type: application/json');

      echo json\_encode($response);

   }

**public** **function** insert\_mhs()

      {

**global** $mysqli;

         $arrcheckpost = **array**('nim' => '', 'nama' => '', 'jk' => '', 'alamat' => '', 'jurusan'   => '');

         $hitung = count(array\_intersect\_key($\_POST, $arrcheckpost));

**if**($hitung == count($arrcheckpost)){

               $result = mysqli\_query($mysqli, "INSERT INTO tbl\_mahasiswa SET

               nim = '$\_POST[nim]',

               nama = '$\_POST[nama]',

               jk = '$\_POST[jk]',

               alamat = '$\_POST[alamat]',

               jurusan = '$\_POST[jurusan]'");

**if**($result)

               {

                  $response=**array**(

                     'status' => 1,

                     'message' =>'Mahasiswa Added Successfully.'

                  );

               }

**else**

               {

                  $response=**array**(

                     'status' => 0,

                     'message' =>'Mahasiswa Addition Failed.'

                  );

               }

         }**else**{

            $response=**array**(

                     'status' => 0,

                     'message' =>'Parameter Do Not Match'

                  );

         }

         header('Content-Type: application/json');

         echo json\_encode($response);

      }

**function** update\_mhs($id)

      {

**global** $mysqli;

         $arrcheckpost = **array**('nim' => '', 'nama' => '', 'jk' => '', 'alamat' => '', 'jurusan'   => '');

         $hitung = count(array\_intersect\_key($\_POST, $arrcheckpost));

**if**($hitung == count($arrcheckpost)){

              $result = mysqli\_query($mysqli, "UPDATE tbl\_mahasiswa SET

              nim = '$\_POST[nim]',

              nama = '$\_POST[nama]',

              jk = '$\_POST[jk]',

              alamat = '$\_POST[alamat]',

              jurusan = '$\_POST[jurusan]'

              WHERE id='$id'");

**if**($result)

            {

               $response=**array**(

                  'status' => 1,

                  'message' =>'Mahasiswa Updated Successfully.'

               );

            }

**else**

            {

               $response=**array**(

                  'status' => 0,

                  'message' =>'Mahasiswa Updation Failed.'

               );

            }

         }**else**{

            $response=**array**(

                     'status' => 0,

                     'message' =>'Parameter Do Not Match'

                  );

         }

         header('Content-Type: application/json');

         echo json\_encode($response);

      }

**function** delete\_mhs($id)

   {

**global** $mysqli;

      $query="DELETE FROM tbl\_mahasiswa WHERE id=".$id;

**if**(mysqli\_query($mysqli, $query))

      {

         $response=**array**(

            'status' => 1,

            'message' =>'Mahasiswa Deleted Successfully.'

         );

      }

**else**

      {

         $response=**array**(

            'status' => 0,

            'message' =>'Mahasiswa Deletion Failed.'

         );

      }

      header('Content-Type: application/json');

      echo json\_encode($response);

   }

}

 ?>

1. **Request Method**  
   Selanjutnya kita akan membuat file **mahasiswa.php**yang akan diakses oleh endpoint, isi dari file adalah untuk pengecekan request method yang dilakukan dan menjalankan fungsi pada **method.php**sesuai dengan request methodnya. Berikut sintaks nya:

<?php

**require\_once** "method.php";

$mhs = **new** Mahasiswa();

$request\_method=$\_SERVER["REQUEST\_METHOD"];

**switch** ($request\_method) {

**case** 'GET':

**if**(!empty($\_GET["id"]))

         {

            $id=intval($\_GET["id"]);

            $mhs->get\_mhs($id);

         }

**else**

         {

            $mhs->get\_mhss();

         }

**break**;

**case** 'POST':

**if**(!empty($\_GET["id"]))

         {

            $id=intval($\_GET["id"]);

            $mhs->update\_mhs($id);

         }

**else**

         {

            $mhs->insert\_mhs();

         }

**break**;

**case** 'DELETE':

          $id=intval($\_GET["id"]);

            $mhs->delete\_mhs($id);

**break**;

**default**:

*// Invalid Request Method*

         header("HTTP/1.0 405 Method Not Allowed");

**break**;

**break**;

}

?>

Selanjutnya kita testing **API** menggunakan **POSTMAN**seperti berikut:









