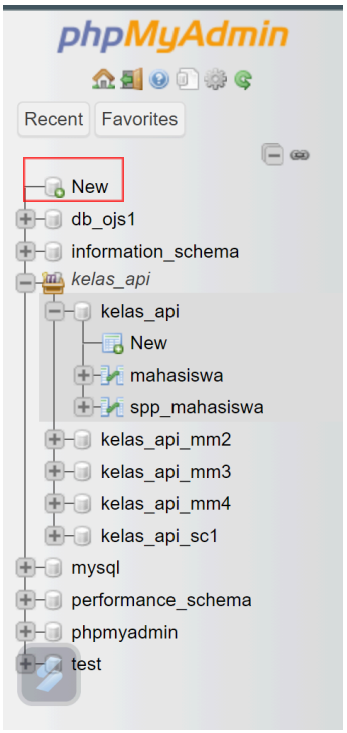
Konfigurasi Database

Sebelum membuat sebuah rest API langkah 1 adalah dengan membuat database dan melakukan export data yang ada di LMS dengan format "**.sql**".

Langkah 1. Create Database

Untuk membuat database dapat dengan menggunakan terminal dengan mengetikan "**CREATE DATABASE nama_database**" atau dengan klik "**New**" seperti gambar 1 kemudian ketikan "**nama Database**" kemudian klik "Create"



Gambar 1. Struktur Database

Langkah 2. Import Database

Setelah anda membuat database kemudian masuk ke database pada localhost dengan klik nama_database yang ada di panel kiri kemudian pilih tab "**Import**". Pada "**File to import:**" cari file **.sql** untuk di import, pada "**Format**" pilih sql, kemudian klik tombol "**impor**". Hasil Import kemudian akan

ada 2 tabel yaitu **mahasiswa** dan **spp_mahasiswa** seperti pada gambar 2.

	Table	Action	Rows	Type	Collation	Size	Overhead
☐	mahasiswa	Browse Structure Search Insert Empty Drop	7	InnoDB	utf8mb4_general_ci	16.0 KiB	-
☐	spp_mahasiswa	Browse Structure Search Insert Empty Drop	5	InnoDB	utf8mb4_general_ci	16.0 KiB	-
	2 tables	Sum	12	InnoDB	utf8mb4_general_ci	32.0 KiB	0 B

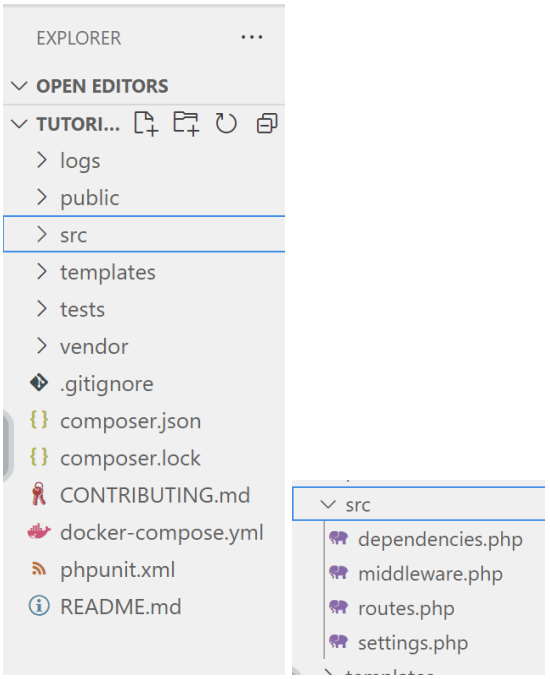
Gambar 2. Hasil Export

Konfigurasi Rest API

Rest API dibuat menggunakan bahasa pemrograman php dengan menggunakan framework Slim. Framework Slim sudah install menggunakan composer dan hasil install dapat di download pada laman https://github.com/zarkuz/restapislim.

Langkah 1 Konfigurasi db

Untuk konfigurasi database pertama adalah dengan masuk pada folder **src->settings.php** seperti pada gambar 3.



Gambar 3. Lokasi "settings.php"

Kemudian setelah masuk ubah host, user, pass, db, dan driver sesuai dengan konfigurasi pada server(contoh menggunakan XAMPP) yang anda gunakan. Posisi script Konfigurasi Database dapat dilihat seperti pada gambar 4.

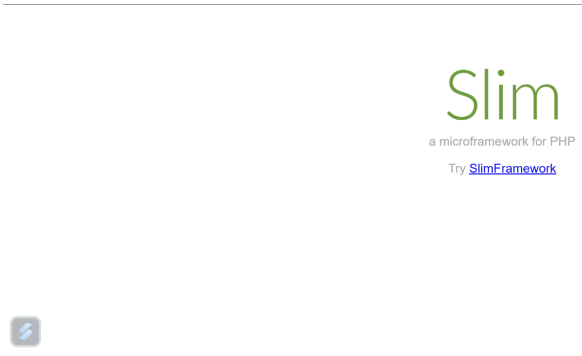
settings.php

```php
<?php
return [
    'settings' => [
        'displayErrorDetails' => true, // set to false in production
        'addContentLengthHeader' => false, // Allow the web server to send the content-length header

        // Renderer settings
        'renderer' => [
            'template_path' => __DIR__ . '/../templates/',
        ],

        // Monolog settings
        'logger' => [
            'name' => 'slim-app',
            'path' => isset($_ENV['docker']) ? 'php://stdout' : __DIR__ . '/../logs/app.log',
            'level' => \Monolog\Logger::DEBUG,
        ],

        'db' => [
            'host' => 'localhost',
            'user' => 'root',
            'pass' => '',
            'dbname' => 'kelas_api',
            'driver' => 'mysql'
        ]
    ],
];
```

Gambar 4. Konfigurasi Database

Langkah 2 Konfigurasi API

Setelah semua setting selesai maka bisa dicek apakah framework sudah dapat berjalan dengan membuka pada browser dengan link **"Link_disini/public"** hingga tampil seperti gambar 5.



Gambar 5. Tampilan Framework Slim

Setelah framework sudah berjalan kemudian dapat masuk ke konfigurasi Rest API dengan masuk pada **src->routes.php** kemudian edit pada bagian yang ada seperti pada gambar 6 yang diberi kotak merah

Gambar 6. Lokasi Script Setting Rest API

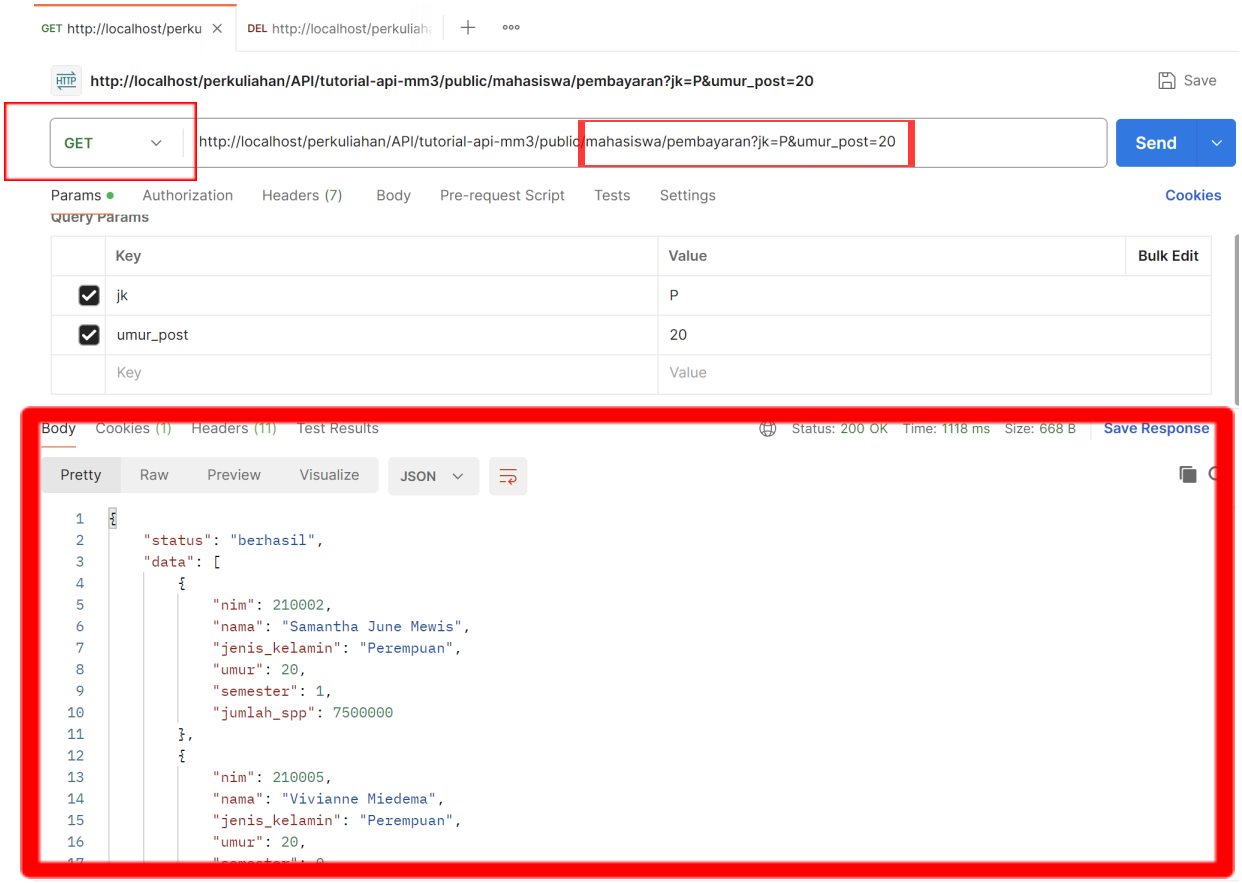
Skenario 1 Melihat data

Skenario 1 Melihat data dengan method GET dibuat untuk melihat data dengan data berasal dari table mahasiswa join dengan spp_mahasiswa. Kemudian Script yang ditambahkan adalah sebagai berikut:

```
$app->get('/mahasiswa/pembayaran', function(Request $request, Response $response){
        //disini dimasukan script sql
        $sql = "SELECT mahasiswa.nim, nama,
                CASE
                    WHEN jk='L' THEN 'Laki-Laki'
                    WHEN jk='P' THEN 'Perempuan'
                END as jenis_kelamin
                , umur, ifnull(semester,0) as semester, ifnull(jumlah,0) as jumlah_spp
                FROM mahasiswa LEFT JOIN spp_mahasiswa
                ON mahasiswa.nim = spp_mahasiswa.nim WHERE mahasiswa.nim<>0";
        $jk = $request->getQueryParam("jk");
        $umur = $request->getQueryParam("umur");

        if($jk<>null){
            $sql = $sql." AND jk='$jk'";
        };
```

```
        if($umur<>null){
            $sql = $sql." AND umur=$umur";
        };

        $stmt = $this->db->prepare($sql);
        $stmt->execute();
        $result = $stmt->fetchAll();
        return $response->withJson(["status"=>"success", "data"=>$result],200);
    });
```

Setelah menambahkan untuk mengecek dapat menggunakan postman dengan link sama seperti sebelumnya kemudian ditambahkan "**/mahasiswa/pembayaran**" sesuai dengan yang dideklarasikan pada script. Untuk menjalankan dilakukan dengan method GET dengan memasukan link diatas dengan parameter yang dapat digunakan adalah umur dan jk. Hasil method GET dapat dilihat pada gambar 7.



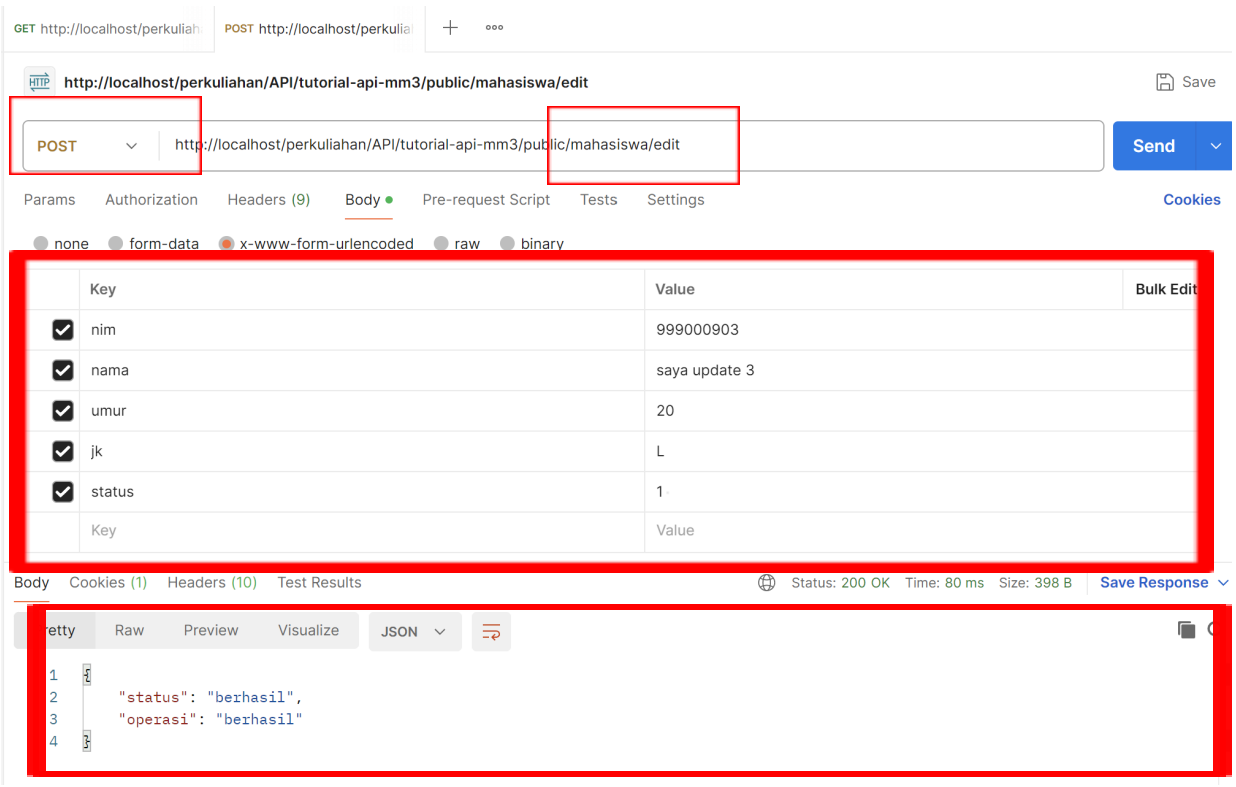
Gambar 7. Hasil method GET

Skenario 2 Menambahkan data dengan method POST

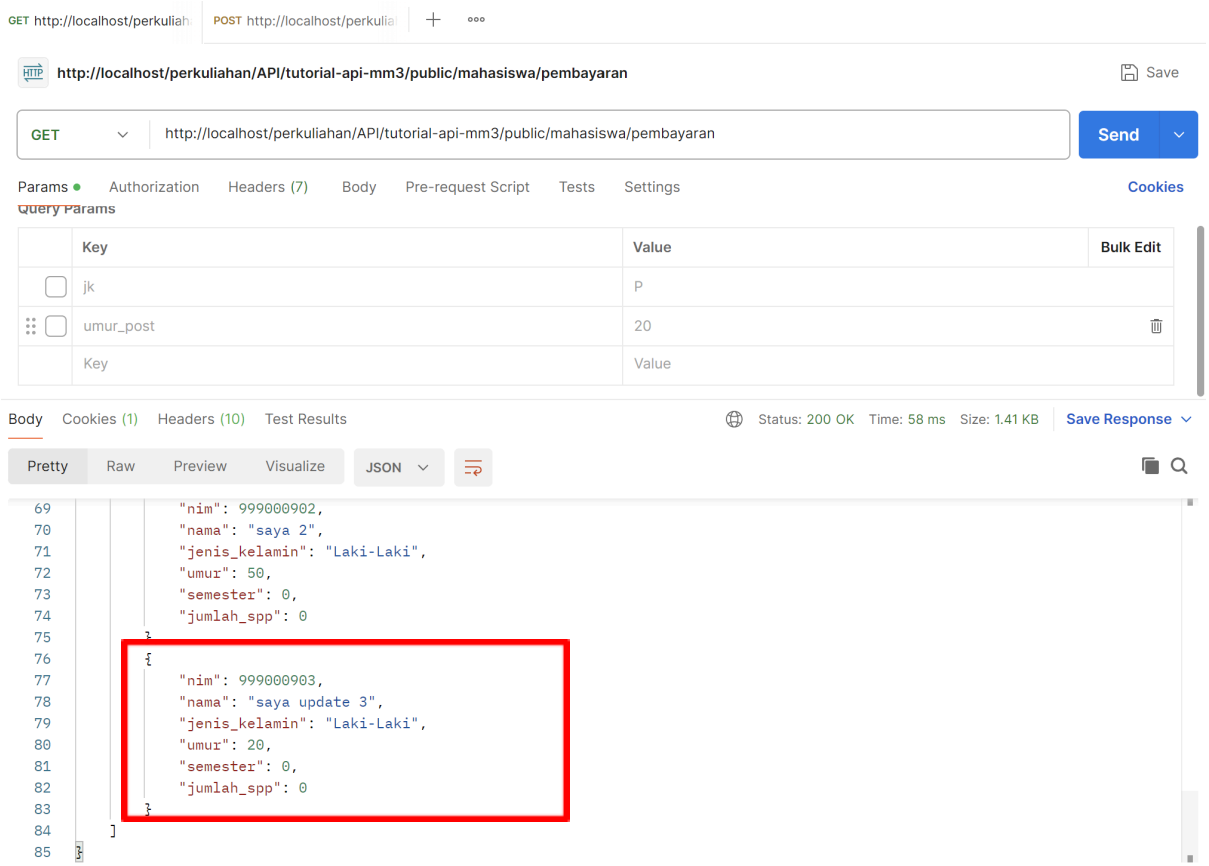
Skenario 2 Menambahkan data dengan method POST dibuat untuk menambahkan/insert data pada tabel mahasiswa. Kemudian Script yang ditambahkan adalah sebagai berikut:

```
$app->post('/mahasiswa/edit', function(Request $request, Response $response){
        $add_mahasiswa = $request->getParsedBody();
        $nim = $add_mahasiswa['nim'];
        $jk = $add_mahasiswa['jk'];
        $nama = $add_mahasiswa['nama'];
        $umur = $add_mahasiswa['umur'];

        //disini dimasukan script sql
        $sql = "INSERT INTO mahasiswa(nim, nama, jk, umur) VALUES
                ($nim, '$nama', '$jk', $umur)";

        $stmt = $this->db->prepare($sql);
        $stmt->execute();
        return $response->withJson(["status"=>"success", "data"=>"1"],200);
    });
```

Setelah menambahkan untuk mengecek dapat menggunakan postman dengan link sama seperti sebelumnya kemudian ditambahkan "**/mahasiswa/edit**" sesuai dengan yang dideklarasikan pada script. Untuk menjalankan dilakukan dengan method POST dengan memasukan link diatas dengan menambahkan data yang diinput pada Body -> x-www-form-urlencoded. Hasil method POST dapat dilihat pada gambar 8 kemudian dapat di kroscek kembali pada data untuk melihat dapat apakah sudah bertambah atau belum pada gambar 9.



Gambar 8. Hasil Method Post



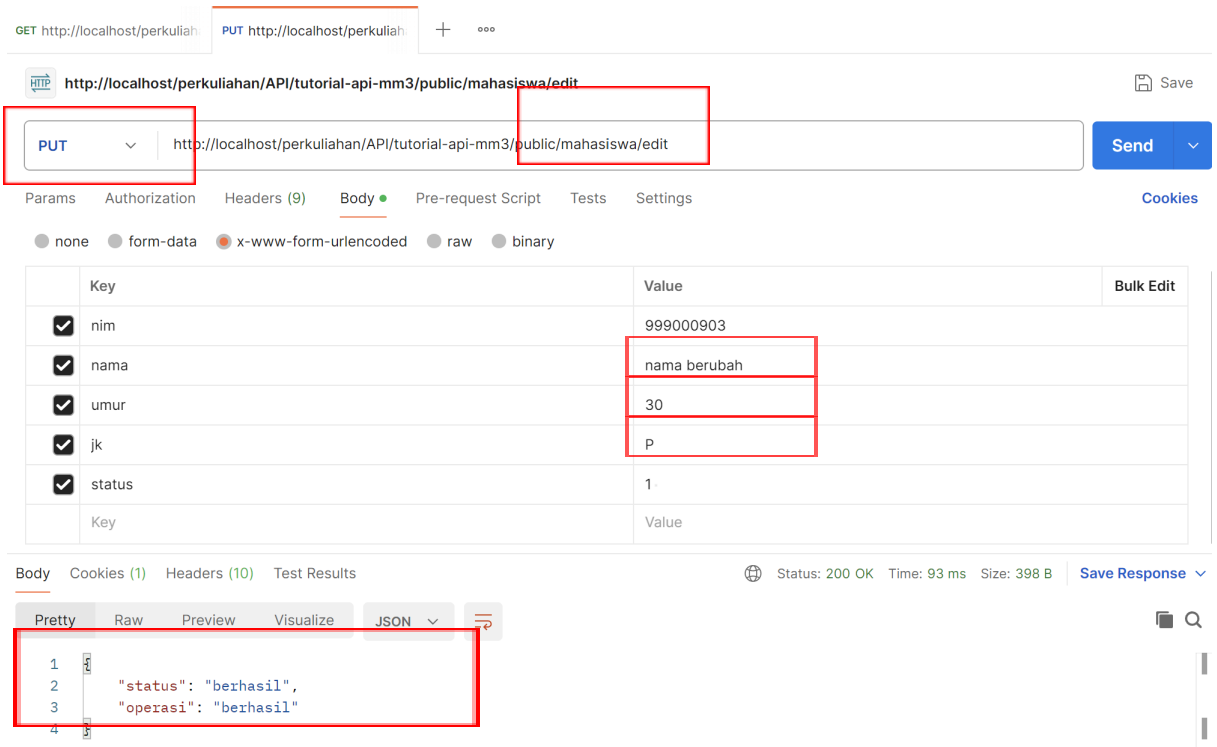
Gambar 9. Hasil Method GET (Melihat Hasil Penambahan Data)

Skenario 3 Mengubah data dengan method PUT

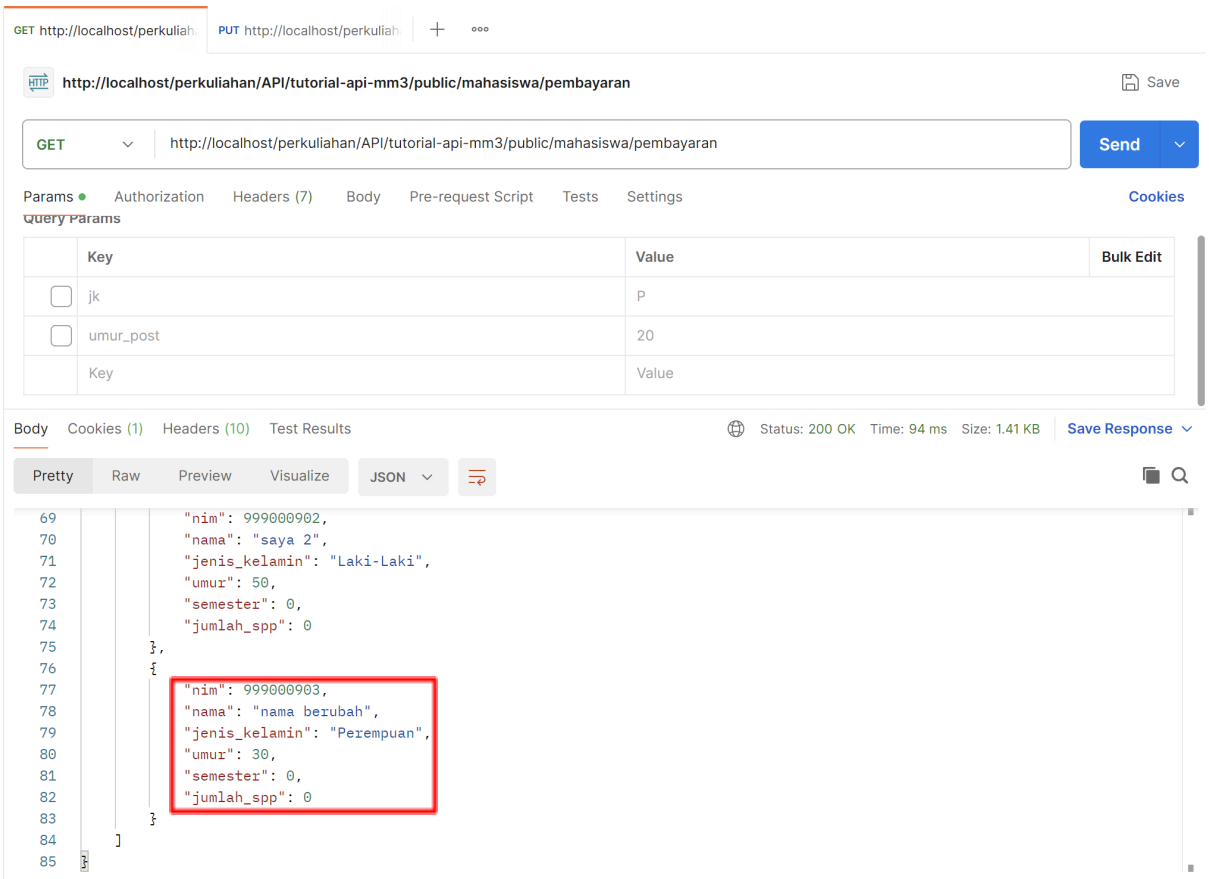
Skenario 3 Mengubah data dengan method PUT dibuat untuk mengubah/update data pada tabel mahasiswa. Kemudian Script yang ditambahkan adalah sebagai berikut:

```
$app->put('/mahasiswa/edit', function(Request $request, Response $response){
        $add_mahasiswa = $request->getParsedBody();
        $nim = $add_mahasiswa['nim'];
        $jk = $add_mahasiswa['jk'];
        $nama = $add_mahasiswa['nama'];
        $umur = $add_mahasiswa['umur'];

        $textupdate= "";
        if($nama<>null){
            $textupdate = $textupdate.", nama='$nama'";
        };
        if($jk<>null){
            $textupdate = $textupdate.", jk='$jk'";
        };
        if($umur<>null){
            $textupdate = $textupdate.", umur=$umur";
        };
        //disini dimasukan script sql
        $sql = "UPDATE mahasiswa SET nim=$nim".$textupdate." WHERE nim=$nim";

        $stmt = $this->db->prepare($sql);
        $stmt->execute();
        return $response->withJson(["status"=>"success", "data"=>"1"],200);
    });
```

Setelah menambahkan untuk mengecek dapat menggunakan postman dengan link sama seperti sebelumnya kemudian ditambahkan "**/mahasiswa/edit**" sesuai dengan yang dideklarasikan pada script. Untuk menjalankan dilakukan dengan method PUT dengan memasukan link diatas dengan menambahkan data yang diinput pada Body -> x-www-form-urlencoded dengan acuan data nim merupakan data nim yang akan di update. Hasil method PUT dapat dilihat pada gambar 10 kemudian dapat di kroscek kembali pada data untuk melihat dapat apakah sudah berubah atau belum pada gambar 11.



Gambar 10. Hasil Method PUT



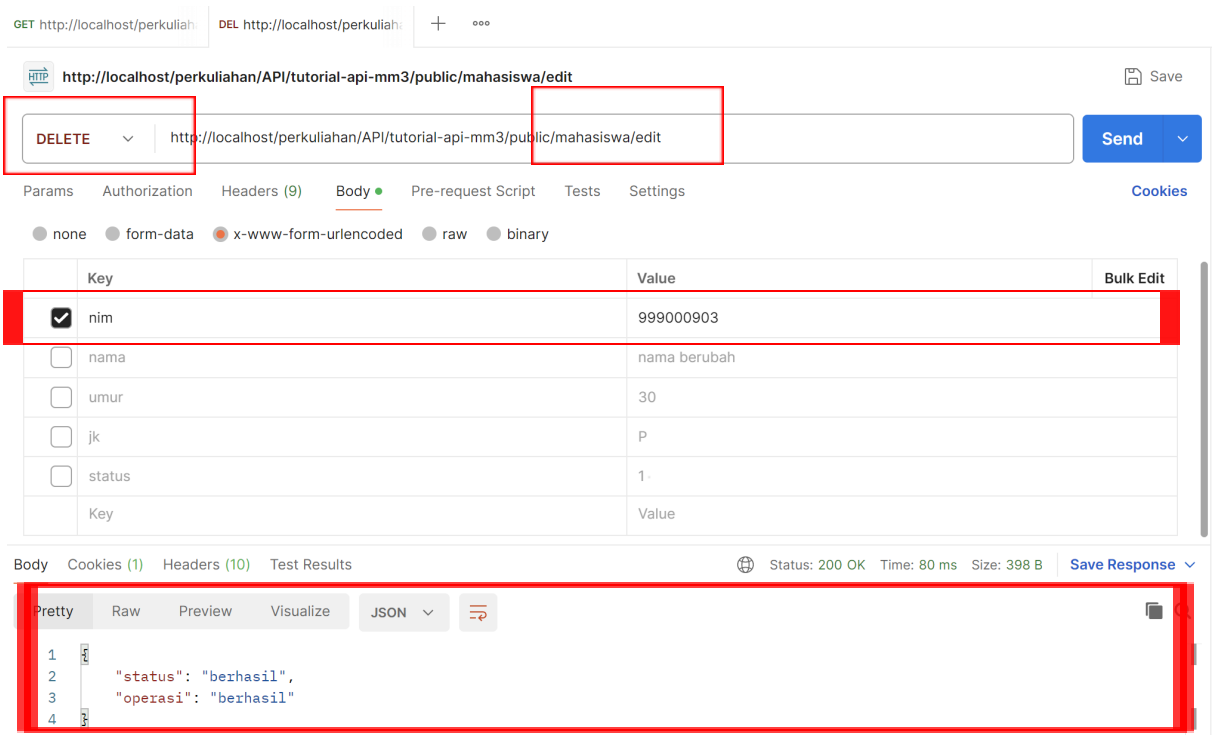
Gambar 11. Hasil Method GET (Melihat Hasil Pengubahan Data)

Skenario 4 Menghapus data dengan method DELETE

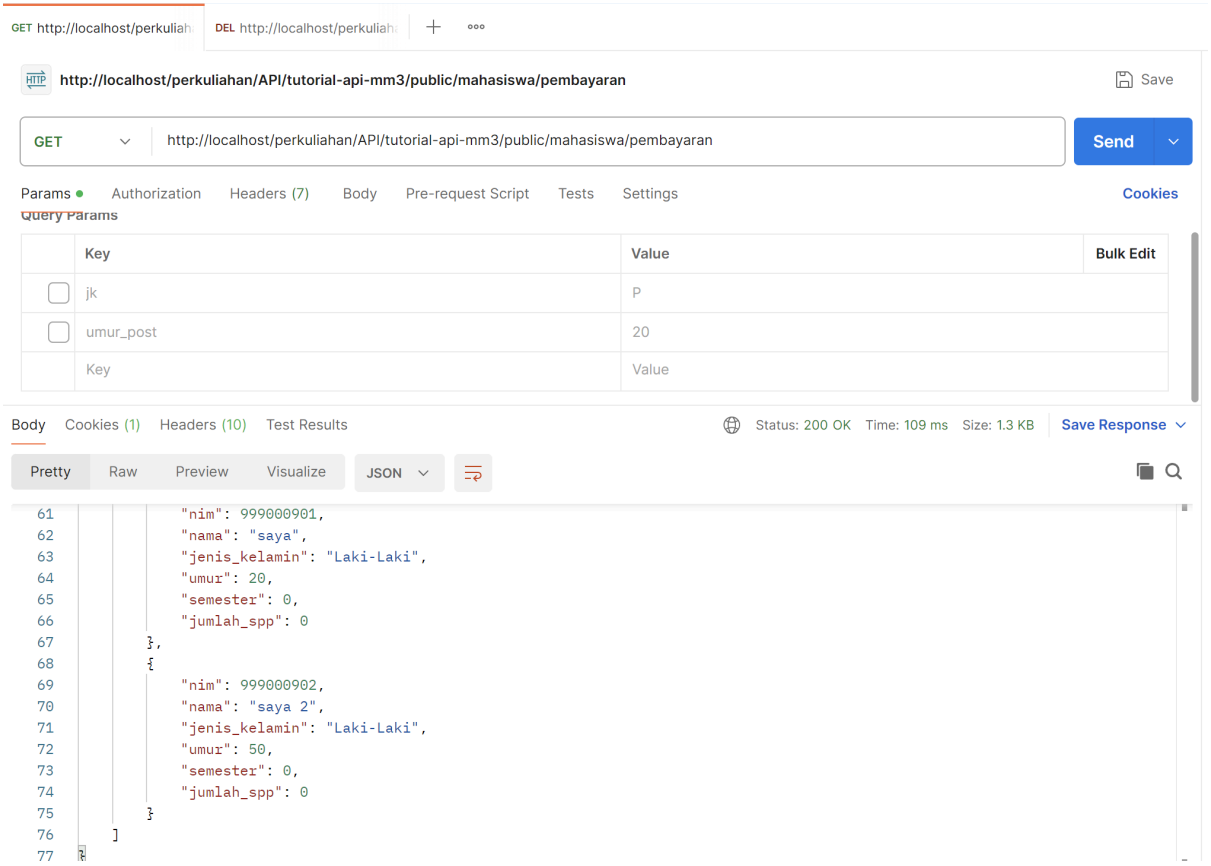
Skenario 4 Menghapus data dengan method DELETE dibuat untuk menghapus/delete data pada tabel mahasiswa. Kemudian Script yang ditambahkan adalah sebagai berikut:

```
$app->delete('/mahasiswa/edit', function(Request $request, Response $response){
        $add_mahasiswa = $request->getParsedBody();
        $nim = $add_mahasiswa['nim'];

        //disini dimasukan script sql
        $sql = "DELETE FROM mahasiswa WHERE nim=$nim";

        $stmt = $this->db->prepare($sql);
        $stmt->execute();
        return $response->withJson(["status"=>"success", "data"=>"1"],200);
    });
```

Setelah menambahkan untuk mengecek dapat menggunakan postman dengan link sama seperti sebelumnya kemudian ditambahkan "**/mahasiswa/edit**" sesuai dengan yang dideklarasikan pada script. Untuk pengujian dilakukan dengan method DELETE dengan memasukan link diatas dengan menambahkan data yang diinput pada Body -> x-www-form-urlencoded dengan acuan data nim untuk data yang akan dihapus. Hasil pengujian dapat dilihat pada gambar 12 kemudian dapat di kroscek kembali pada data untuk melihat dapat apakah sudah dihapus atau belum pada gambar 13.



Gambar 12. Hasil Method DELETE



Gambar 13. Hasil Method DELETE (Melihat Hasil Penghapusan Data)