

# **LAPORAN UAS WEB SERVICE**

## **MEMBUAT REST A.P.I (C.R.U.D)**

**Untuk Memenuhi Salah Satu Tugas**

**Mata Kuliah Web Service**

**Dosen Pengampu : Pahrul Irfan, S.Kom, M.Kom**



Disusun Oleh :

**Muhammad Hadi Wirawan ( 1710520196 )**

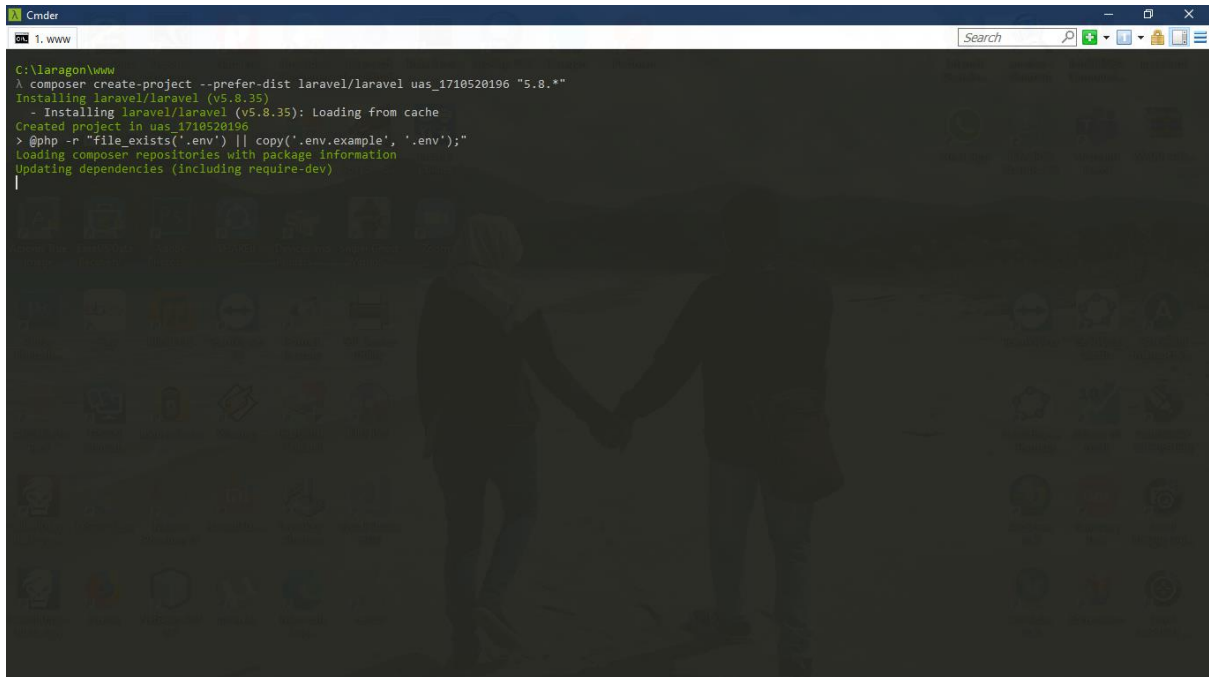
**RPL**

**Universitas Bumigora Mataram**

**2019/2020**

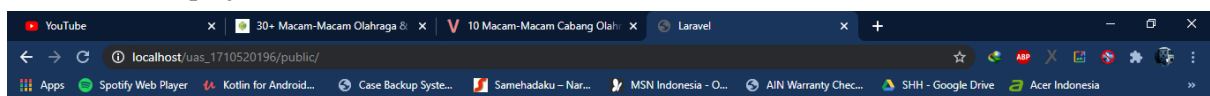
1710520196

1. Membuat project Laravel dengan nama uas\_1710520196 menggunakan fitur terminal di aplikasi Laragon dengan versi Laravel 5.8



```
C:\laragon\www
λ composer create-project --prefer-dist laravel/laravel uas_1710520196 "5.8.*"
Installing laravel/laravel (v5.8.35)
- Installing laravel/laravel (v5.8.35): Loading from cache
Created project in uas_1710520196
> @php -r "file_exists('.env') || copy('.env.example', '.env');"
Loading composer repositories with package information
Updating dependencies (including require-dev)
```

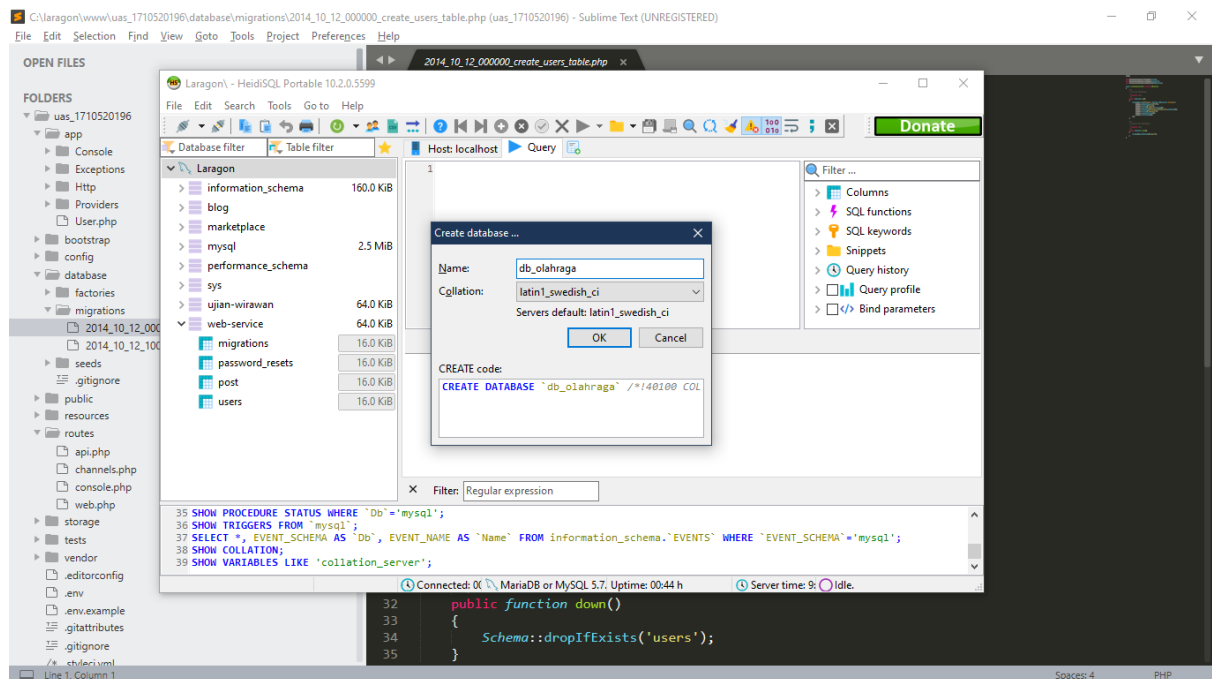
2. Ketika project sudah berhasil dibuat



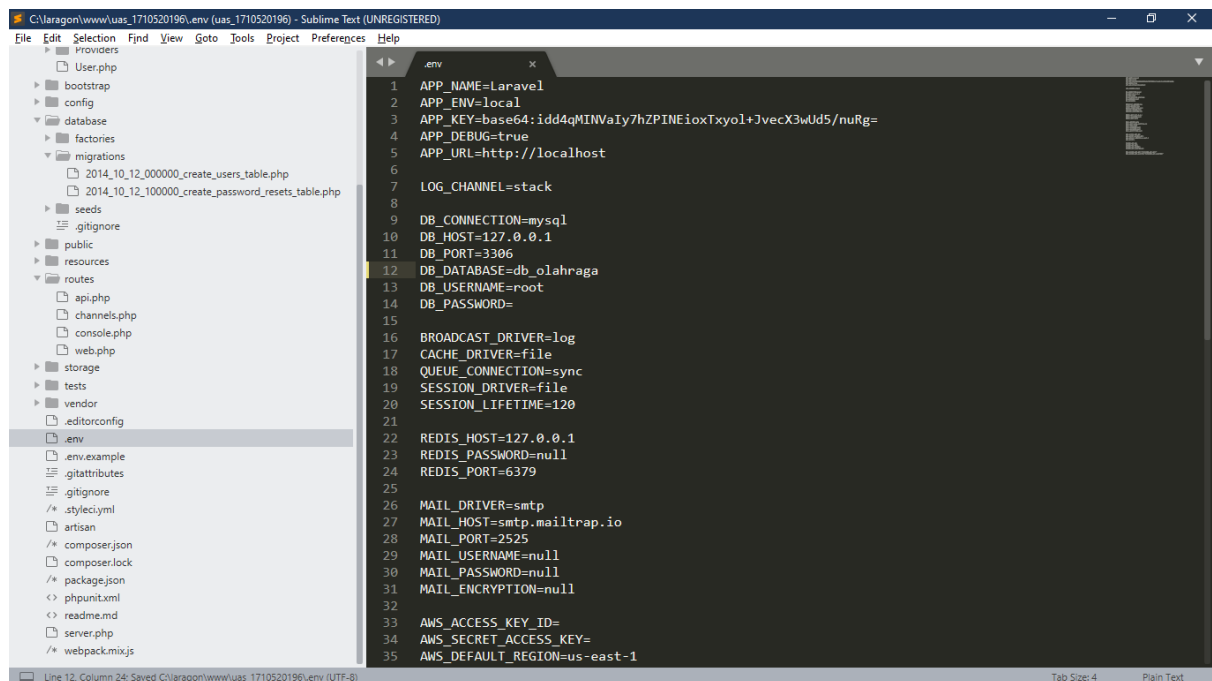
# Laravel

[DOCS](#) [LARACASTS](#) [NEWS](#) [BLOG](#) [NOVA](#) [FORGE](#) [GITHUB](#)

### 3. Membuat database dengan nama db\_olahraga pada fitur Database di aplikasi Laragon



### 4. Kemudian ubah konfigurasi pada file .env menggunakan text editor dan isikan nama database sesuai dengan yang sudah dibuat sebelumnya diatas



## 5. Membuat tabel pada database db\_olahraga dengan nama “olahraga”

```

C:\laragon\www\uas_1710520196
λ php artisan make:migration create_olahraga_table
Created Migration: 2020_07_30_134040_create_olahraga_table

C:\laragon\www\uas_1710520196
λ
  
```

## 6. Membuat kolom pada tabel “olahraga” dengan merubah isi script pada database/migrations/”sesuai nama tabel yang sudah dibuat sebelumnya”

```

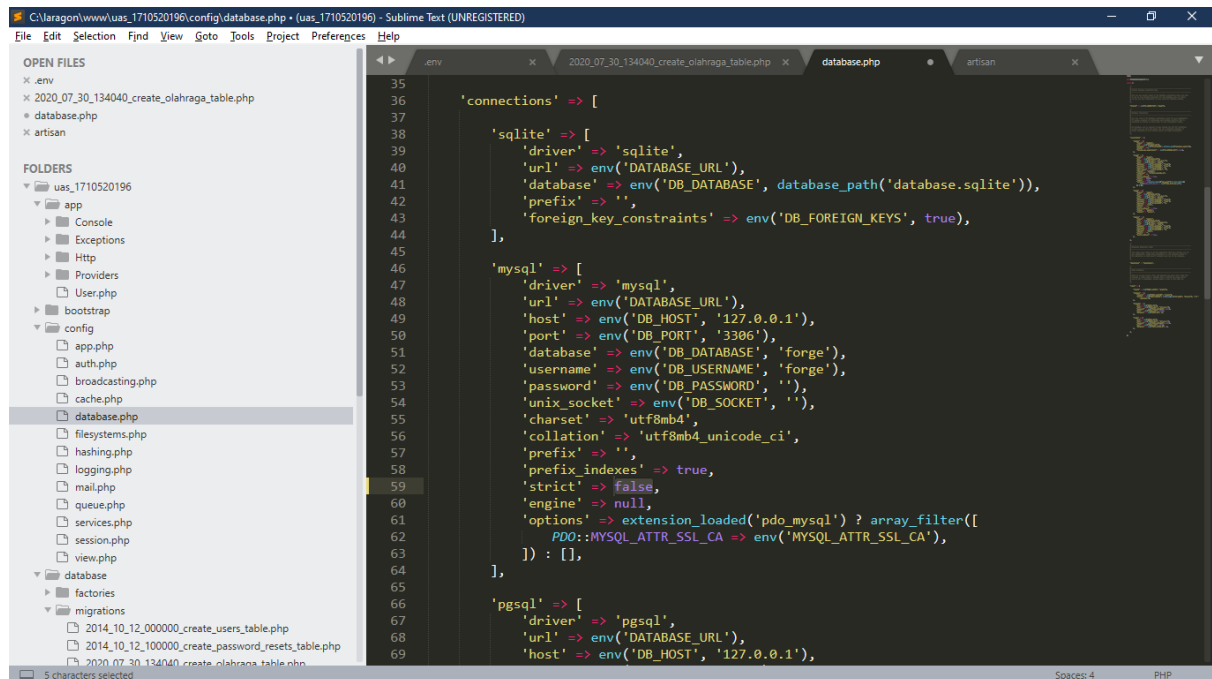
<?php
use Illuminate\Support\Facades\Schema;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Database\Migrations\Migration;

class CreateOlahragaTable extends Migration
{
    /**
     * Run the migrations.
     *
     * @return void
     */
    public function up()
    {
        Schema::create('olahraga', function (Blueprint $table) {
            $table->increments('id');
            $table->string('nama_olahraga');
            $table->string('jenis_olahraga');
            $table->integer('jumlah_pemain');
            $table->string('lokasi_main');
            $table->string('alat_yang_digunakan');
            $table->integer('waktu_main');
            $table->timestamps();
        });
    }

    /**
     * Reverse the migrations.
     *
     * @return void
     */
    public function down()
    {
        Schema::dropIfExists('olahraga');
    }
}
  
```

1710520196

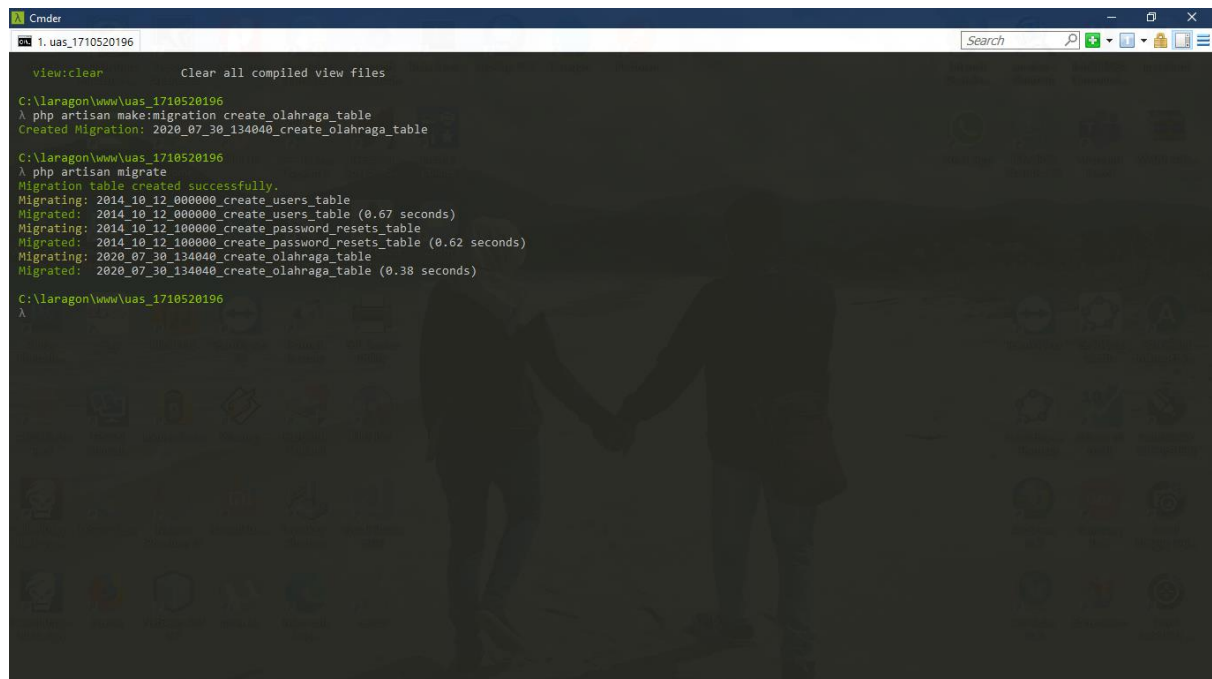
7. Sebelum menjalankan perintah *migration* kita harus melakukan sedikit konfigurasi pada file database.php yang ada pada folder proyek laravel yang baru dibuat atau pada folder uas-1710520196/config/database.php pada baris ke **53** ubahlah **value strict** dari **true** menjadi **false** seperti pada script dibawah



```
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69

'connections' => [
    'sqlite' => [
        'driver' => 'sqlite',
        'url' => env('DATABASE_URL'),
        'database' => env('DB_DATABASE', database_path('database.sqlite')),
        'prefix' => '',
        'foreign_key_constraints' => env('DB_FOREIGN_KEYS', true),
    ],
    'mysql' => [
        'driver' => 'mysql',
        'url' => env('DATABASE_URL'),
        'host' => env('DB_HOST', '127.0.0.1'),
        'port' => env('DB_PORT', '3306'),
        'database' => env('DB_DATABASE', 'forge'),
        'username' => env('DB_USERNAME', 'forge'),
        'password' => env('DB_PASSWORD', ''),
        'unix_socket' => env('DB_SOCKET', ''),
        'charset' => 'utf8mb4',
        'collation' => 'utf8mb4_unicode_ci',
        'prefix' => '',
        'prefix_indexes' => true,
        'strict' => false,
        'engine' => null,
        'options' => extension_loaded('pdo_mysql') ? array_filter([
            PDO::MYSQL_ATTR_SSL_CA => env('MYSQL_ATTR_SSL_CA'),
        ]) : [],
    ],
    'pgsql' => [
        'driver' => 'pgsql',
        'url' => env('DATABASE_URL'),
        'host' => env('DB_HOST', '127.0.0.1'),
```

8. Lakukan *migrate* dengan perintah *php artisan migrate* untuk menghasilkan tabel pada database db\_olahraga



```
view:clear Clear all compiled view files

C:\laragon\www\uas_1710520196
λ php artisan make:migration create_olahraga_table
Created Migration: 2020_07_30_134040_create_olahraga_table

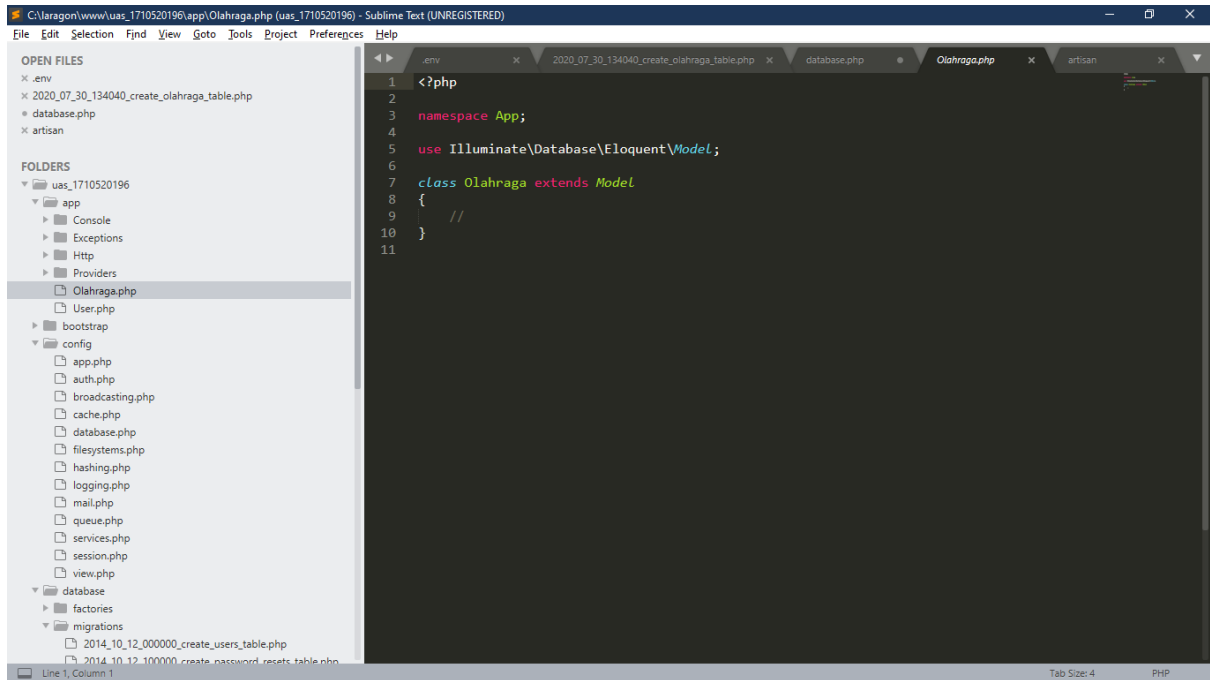
C:\laragon\www\uas_1710520196
λ php artisan migrate
Migration table created successfully.
Migrating: 2014_10_12_000000_create_users_table
Migrated: 2014_10_12_000000_create_users_table (0.67 seconds)
Migrating: 2014_10_12_100000_create_password_resets_table
Migrated: 2014_10_12_100000_create_password_resets_table (0.62 seconds)
Migrating: 2020_07_30_134040_create_olahraga_table
Migrated: 2020_07_30_134040_create_olahraga_table (0.38 seconds)

C:\laragon\www\uas_1710520196
λ
```

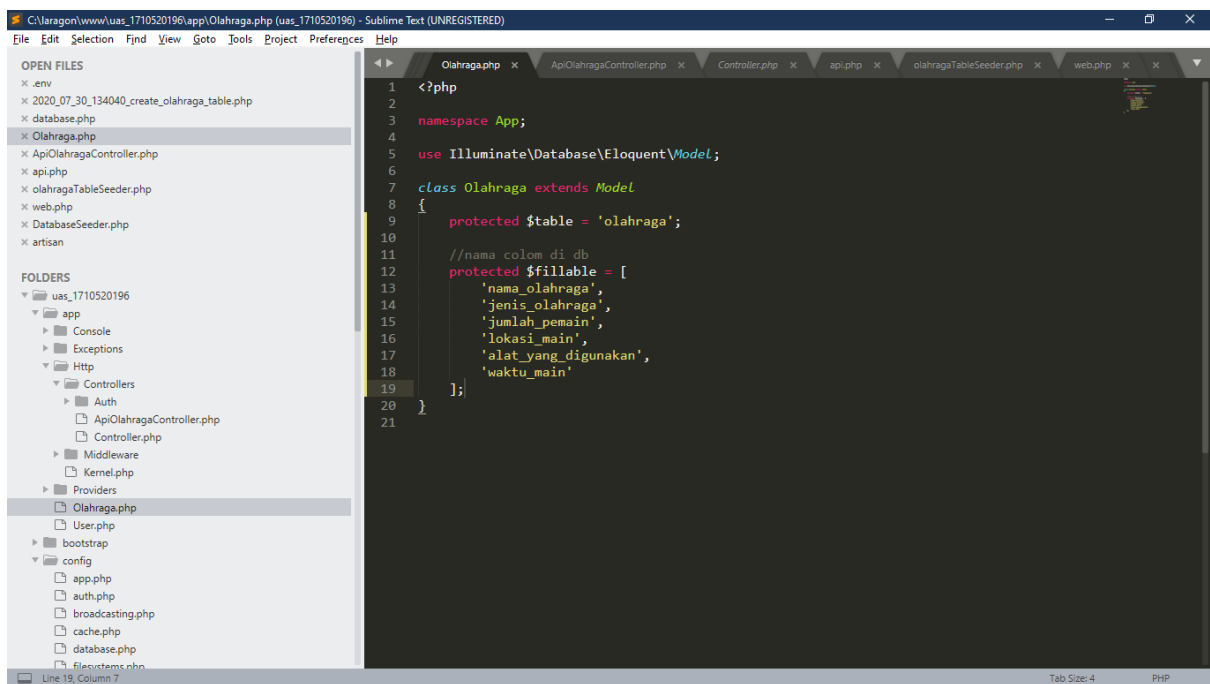
The screenshot displays a Windows desktop environment. In the foreground, a terminal window titled 'cmd' shows the execution of a Laravel migration command: `php artisan migrate`. The command output indicates that the migration was successful, creating the 'olahraga' table. Below the terminal, a HeidiSQL database client window is open, showing the 'olahraga' table structure. The table has five columns: 'id' (INT, AUTO\_INCREMENT), 'nama\_olahraga' (VARCHAR, 255), 'jenis\_olahraga' (VARCHAR, 255), 'jumlah\_pemain' (INT, 11), and 'lokasi\_main' (VARCHAR, 255). The table is currently empty. The background shows a Windows 10 desktop with various application icons and a taskbar.

The screenshot shows a Windows desktop with a dark background featuring a silhouette of two people holding hands. A Windows Explorer window is open, displaying the contents of the directory `C:\laragon\www\uas_1710520196`. The window title is `1. uas_1710520196`. The address bar shows the same path. The file list includes folders like `laragon` and `uas_1710520196`, and files like `uas_1710520196` and `uas_1710520196`.

11. Dan akan menghasilkan satu file baru pada folder penjualan/app dengan nama Olahraga.php. Hasil dari perintah artisan di atas dapat dilihat pada gambar dibawah

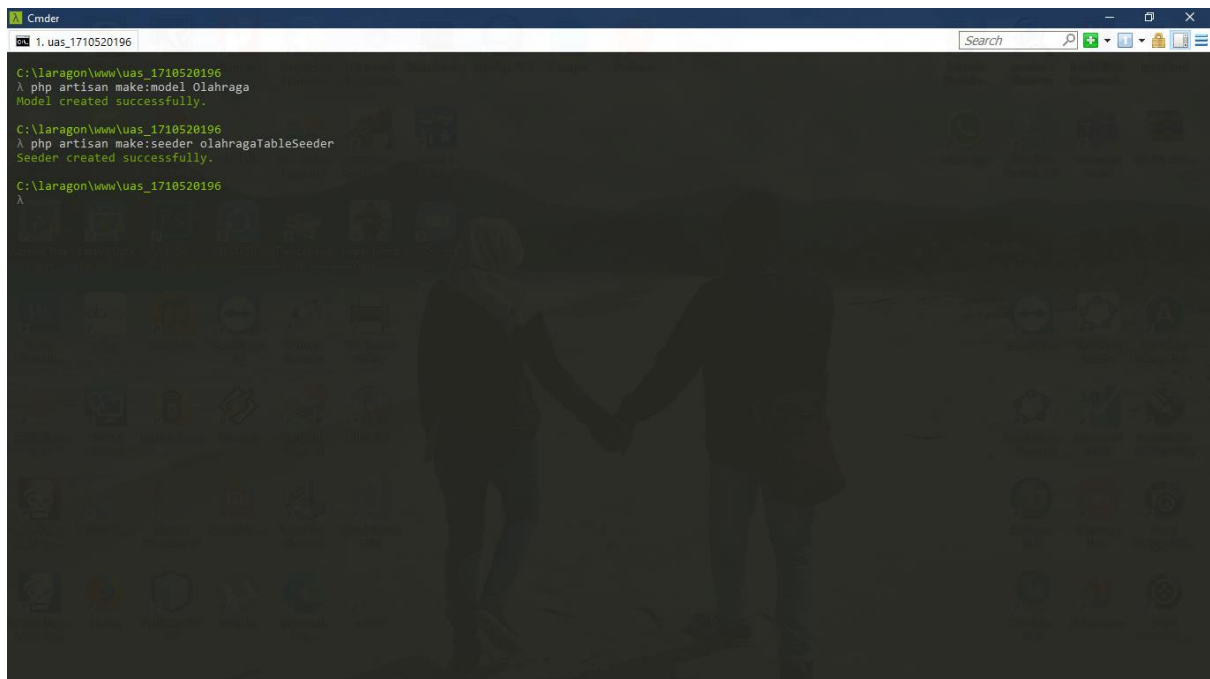


12. Di dalam class model diatas kita mendefinsikan satu variabel bernama table, value dari variabel ini akan dianggap sebagai nama tabel yang diwakili oleh model ini



1710520196

### 13. Membuat *seeder* untuk mencoba inputkan data pada database. Sintaksnya sebagai berikut

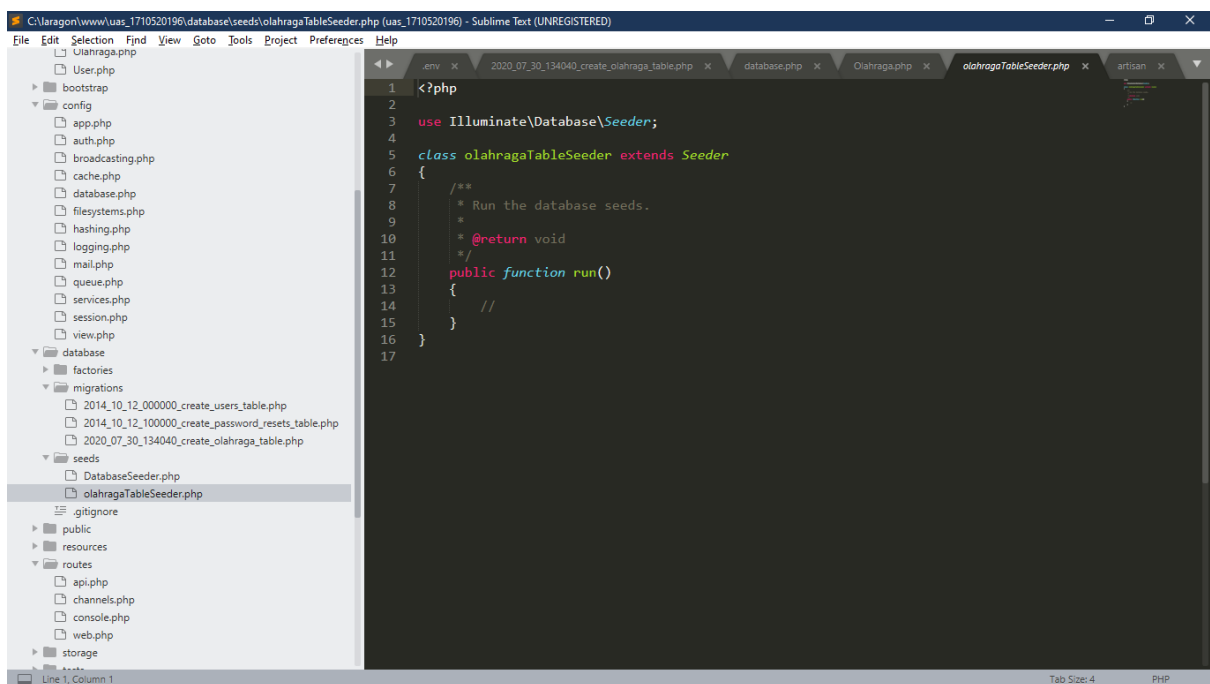


```
C:\laragon\www\uas_1710520196
λ php artisan make:model Olahraga
Model created successfully.

C:\laragon\www\uas_1710520196
λ php artisan make:seeder olahragaTableSeeder
Seeder created successfully.

C:\laragon\www\uas_1710520196
λ
```

### 14. Perintah *seeder* diatas akan menghasilkan file baru pada folder database/seeds dengan nama olahragaTableSeeder.php



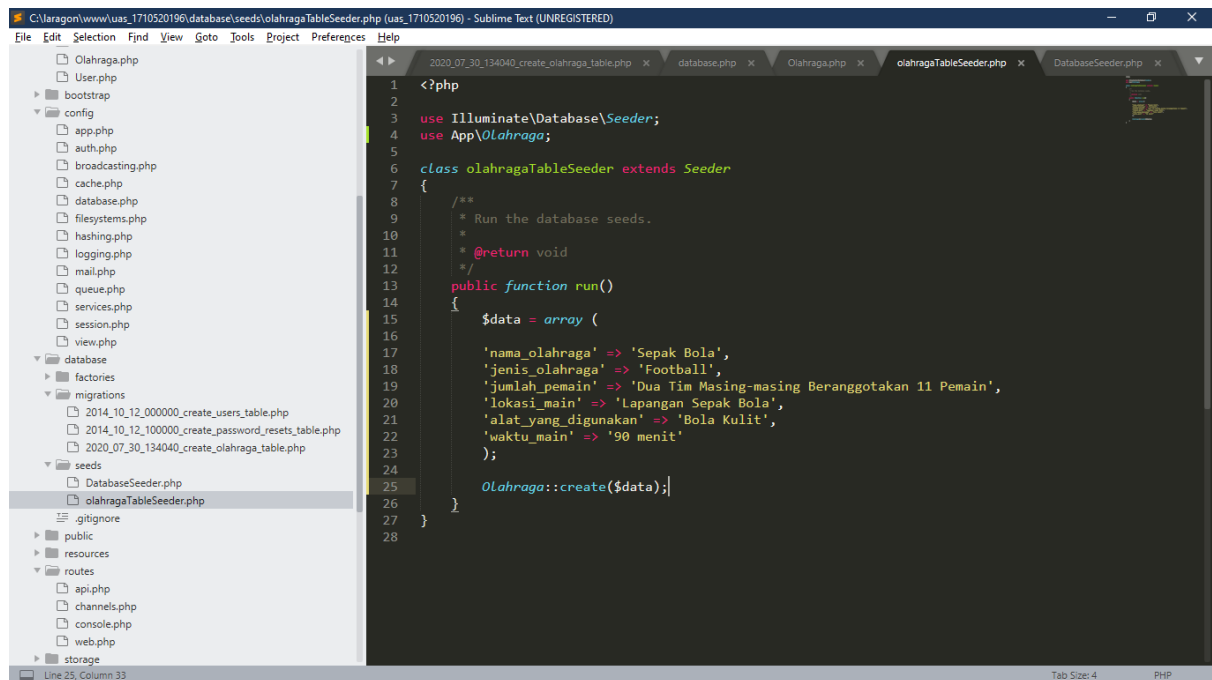
```
C:\laragon\www\uas_1710520196\database\seeds\olahragaTableSeeder.php (uas_1710520196) - Sublime Text (UNREGISTERED)

File Edit Selection Find View Goto Tools Project Preferences Help

1 <?php
2
3 use Illuminate\Database\Seeder;
4
5 class olahragaTableSeeder extends Seeder
6 {
7     /**
8      * Run the database seeds.
9      *
10     * @return void
11     */
12     public function run()
13     {
14         //
15     }
16 }
17
```



15. Selanjutnya tambahkan *script* pada *method run* pada olahragaTableSeeder.php , jangan lupa untuk menambahkan *namespace* dari *Controller Olahraga*

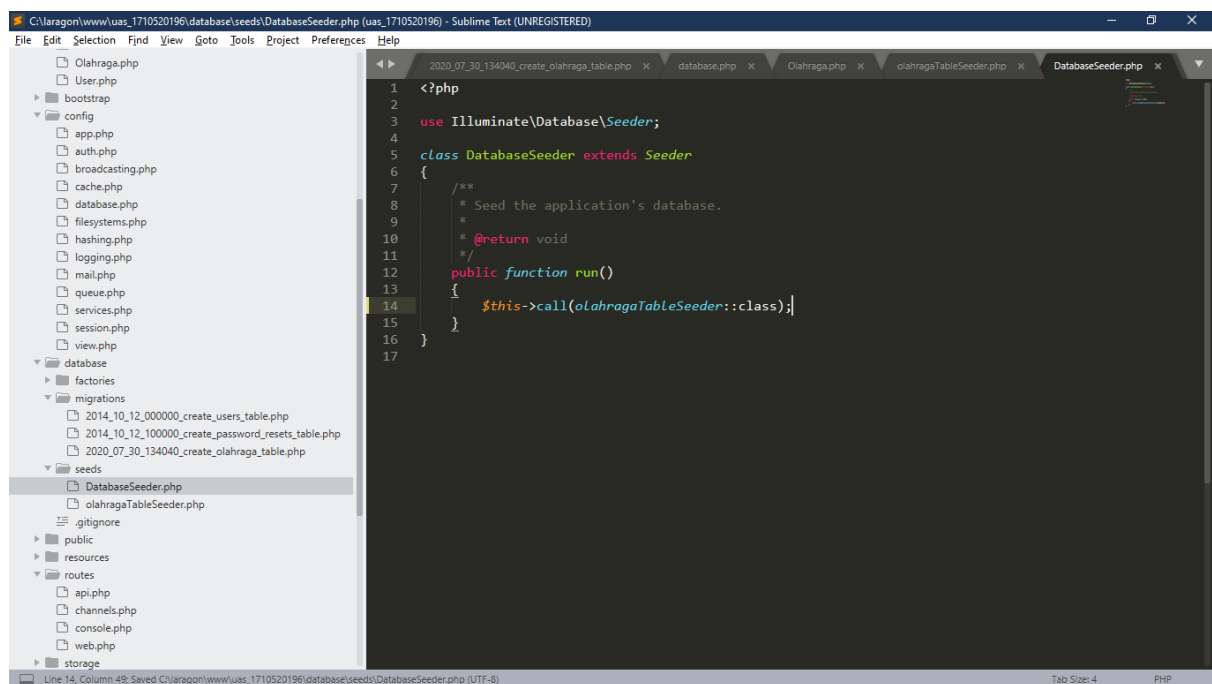


```

1 <?php
2
3 use Illuminate\Database\Seeder;
4 use App\Olahraga;
5
6 class olahragaTableSeeder extends Seeder
7 {
8     /**
9      * Run the database seeds.
10     *
11     * @return void
12     */
13     public function run()
14     {
15         $data = array (
16
17             'nama_olahraga' => 'Sepak Bola',
18             'jenis_olahraga' => 'Football',
19             'jumlah_pemain' => 'Dua Tim Masing-masing Beranggotakan 11 Pemain',
20             'lokasi_main' => 'Lapangan Sepak Bola',
21             'alat_yang_digunakan' => 'Bola Kulit',
22             'waktu_main' => '90 menit'
23         );
24
25         Olahraga::create($data);
26     }
27 }
28

```

16. Kemudian panggil *seeder* yang baru dibuat pada file DatabaseSeeder seperti berikut



```

1 <?php
2
3 use Illuminate\Database\Seeder;
4
5 class DatabaseSeeder extends Seeder
6 {
7     /**
8      * Seed the application's database.
9     *
10     * @return void
11     */
12     public function run()
13     {
14         $this->call(olahragaTableSeeder::class);
15     }
16 }
17

```

17. Eksekusi *seeder* yang sudah dibuat dengan perintah sebagai berikut

```

C:\laragon\www\uas_1710520196
λ php artisan db:seed
Seeding: olahragaTableSeeder
Database seeding completed successfully.

C:\laragon\www\uas_1710520196
λ

```

18. Maka pada tabel olahraga yang sudah di *seeder* akan terdapat tampilan sebagai berikut

The screenshot shows the HeidiSQL interface with the 'db\_olahraga' database selected. The 'olahraga' table is highlighted in the left sidebar. The main window displays the table's data:

| id | nama_olahraga | jenis_olahraga | jumlah_pemain                                 | lokasi_main         |
|----|---------------|----------------|---|---------------------|
| 1  | Sepak Bola    | Football       | Dua Tim Masing-masing Beranggotakan 11 Pemain | Lapangan Sepak Bola |

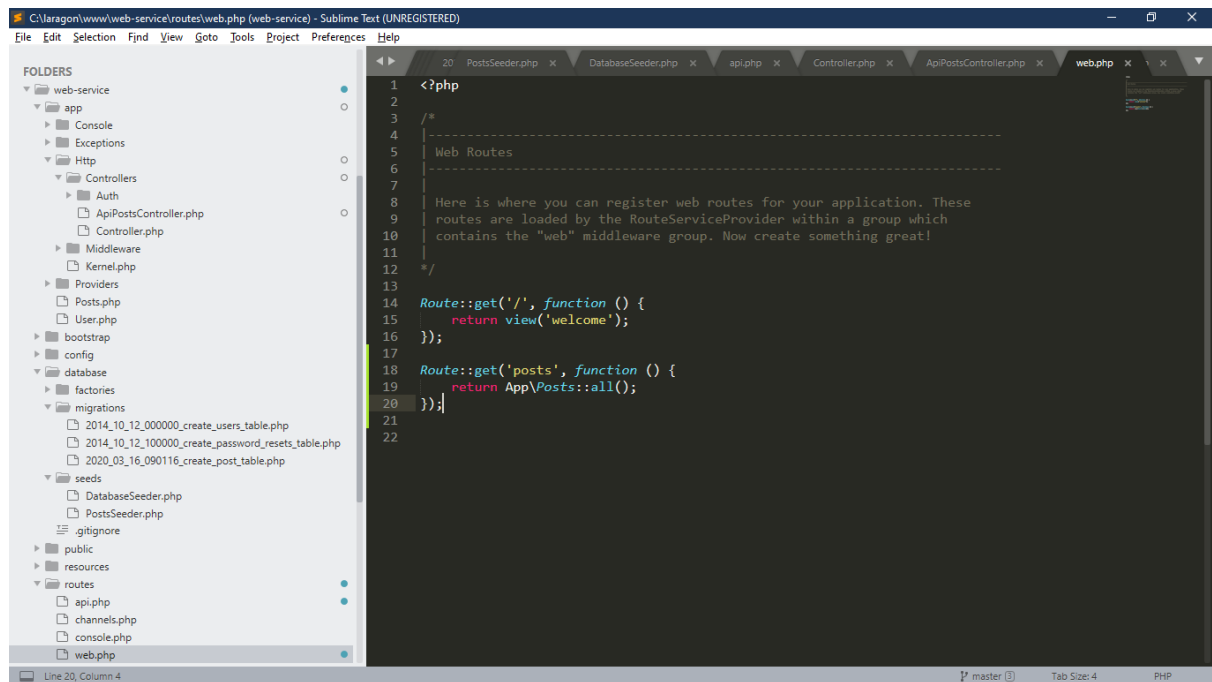
Below the table, the SQL commands used to create and seed the table are visible:

```

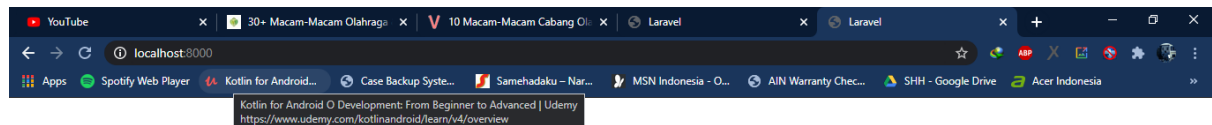
65 SHOW CREATE TABLE `db_olahraga`.`olahraga`;
66 UPDATE `db_olahraga`.`olahraga` SET `id`='1' WHERE `id`=4;
67 SELECT `id`, `nama_olahraga`, `jenis_olahraga`, `jumlah_pemain`, `lokasi_main`, `alat_yang_digunakan`, `waktu_main`, `created`
68 SELECT * FROM `db_olahraga`.`olahraga` LIMIT 1000;
69 SHOW CREATE TABLE `db_olahraga`.`olahraga`;

```

## 19. Membuat *route* pada file routes/web.php seperti berikut



## 20. Mencoba *route* tersebut dengan cara menjalankan *php artisan serve* pada terminal kemudian akses menggunakan browser seperti berikut



# Laravel

DOCS LARACASTS NEWS BLOG NOVA FORGE GITHUB

## 21. Membuat *Controller* yang lengkap sebagai metode untuk *route*

```

C:\laragon\www\uas_1710520196
λ php artisan make:controller ApiOlahragaController --resource
Controller created successfully.
C:\laragon\www\uas_1710520196
λ

```

22. Perintah diatas akan menghasilkan file baru bernama ApiOlahragaController.php yang terletak di folder app\Http\Controller dengan fungsi yang lengkap, mulai dari index, create, store, edit, update dan destroy.

```

C:\laragon\www\uas_1710520196\app\Http\Controllers\ApiOlahragaController.php (uas_1710520196) - Sublime Text (UNREGISTERED)
File Edit Selection Find View Goto Tools Project Preferences Help

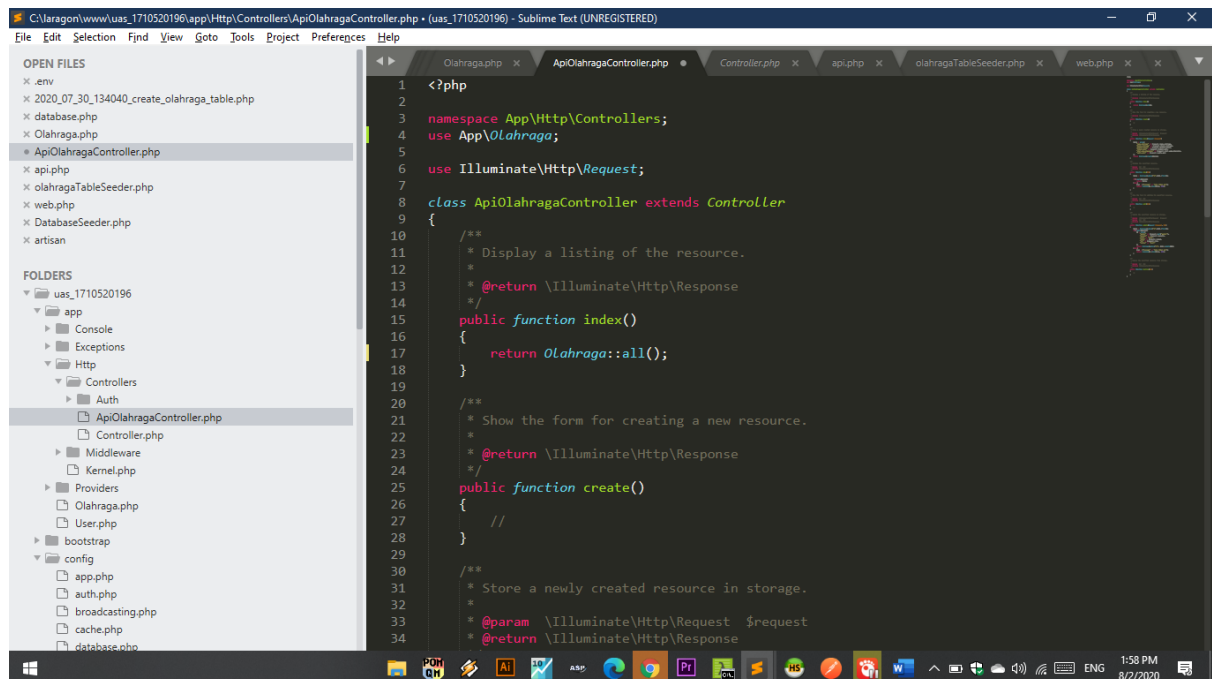
OPEN FILES
× .env
× 2020_07_30_134040_create_olahraga_table.php
× database.php
× Olahraga.php
× api.php
× olahragaTableSeeder.php
× web.php
× DatabaseSeeder.php
× artisan

FOLDERS
▼ uas_1710520196
  ▼ app
    ▶ Console
    ▶ Exceptions
    ▼ Http
      ▼ Controllers
        ▶ Auth
        ▶ ApiOlahragaController.php
        ▶ Controller.php
        ▶ Middleware
        ▶ Kernel.php
        ▶ Providers
        ▶ Olahraga.php
        ▶ User.php
      ▶ bootstrap
    ▼ config
      app.php
      auth.php
      broadcasting.php
      cache.php
      database.php
      filesystems.php
      hashing.php

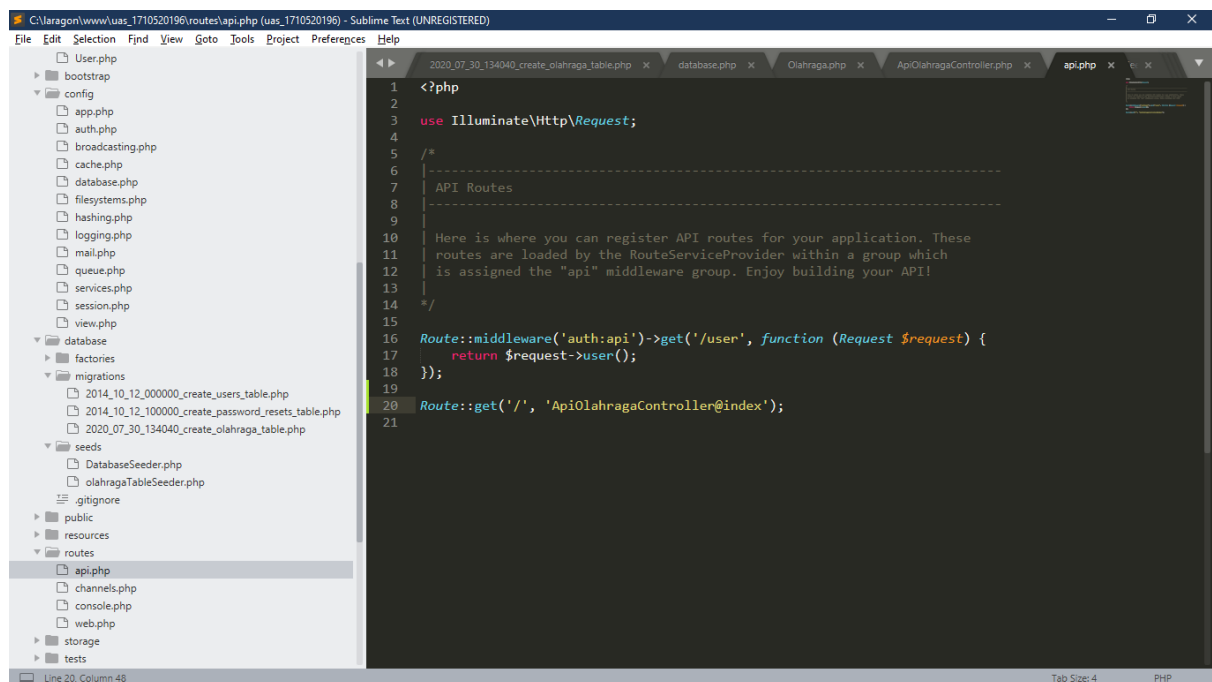
1  <?php
2  namespace App\Http\Controllers;
3
4  use Illuminate\Http\Request;
5
6  class ApiOlahragaController extends Controller
7  {
8
9      /**
10       * Display a listing of the resource.
11       *
12       * @return \Illuminate\Http\Response
13       */
14     public function index()
15     {
16         //
17     }
18
19     /**
20      * Show the form for creating a new resource.
21      *
22      * @return \Illuminate\Http\Response
23      */
24     public function create()
25     {
26         //
27     }
28
29     /**
30      * Store a newly created resource in storage.
31      *
32      * @param \Illuminate\Http\Request $request
33      * @return \Illuminate\Http\Response
34      */
35     public function store(Request $request)

```

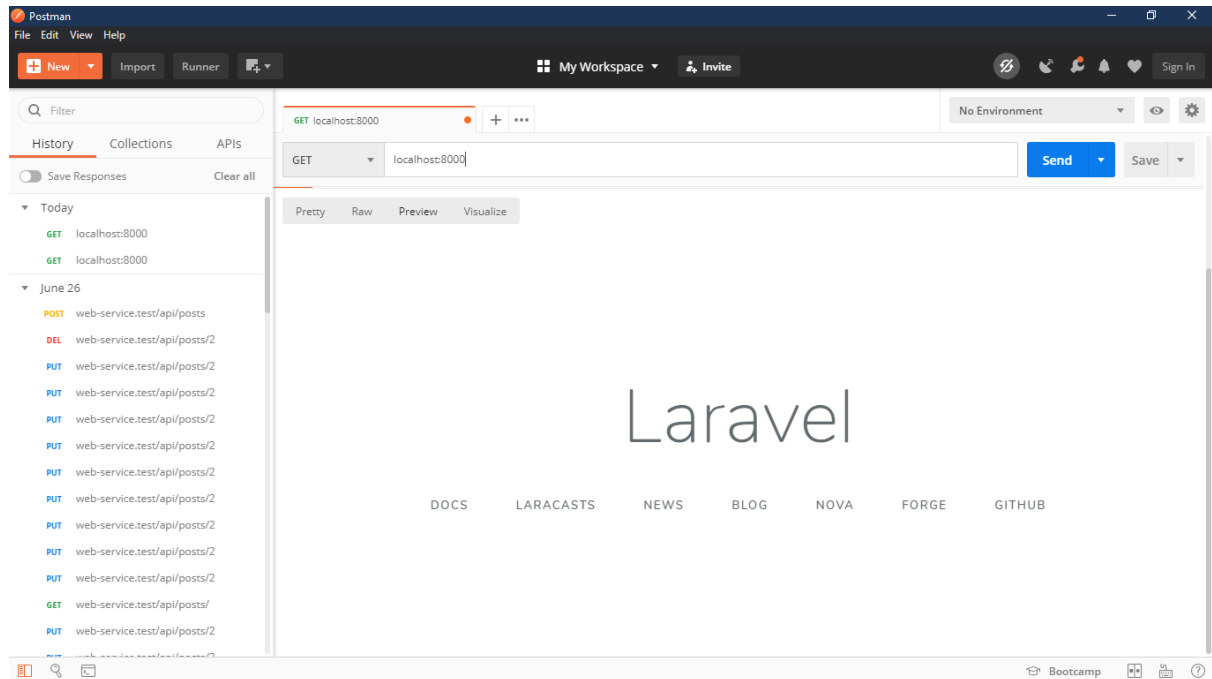
### 23. Kemudian konfigurasi seperti dibawah ini, tambahkan sintaks berikut pada fungsi index



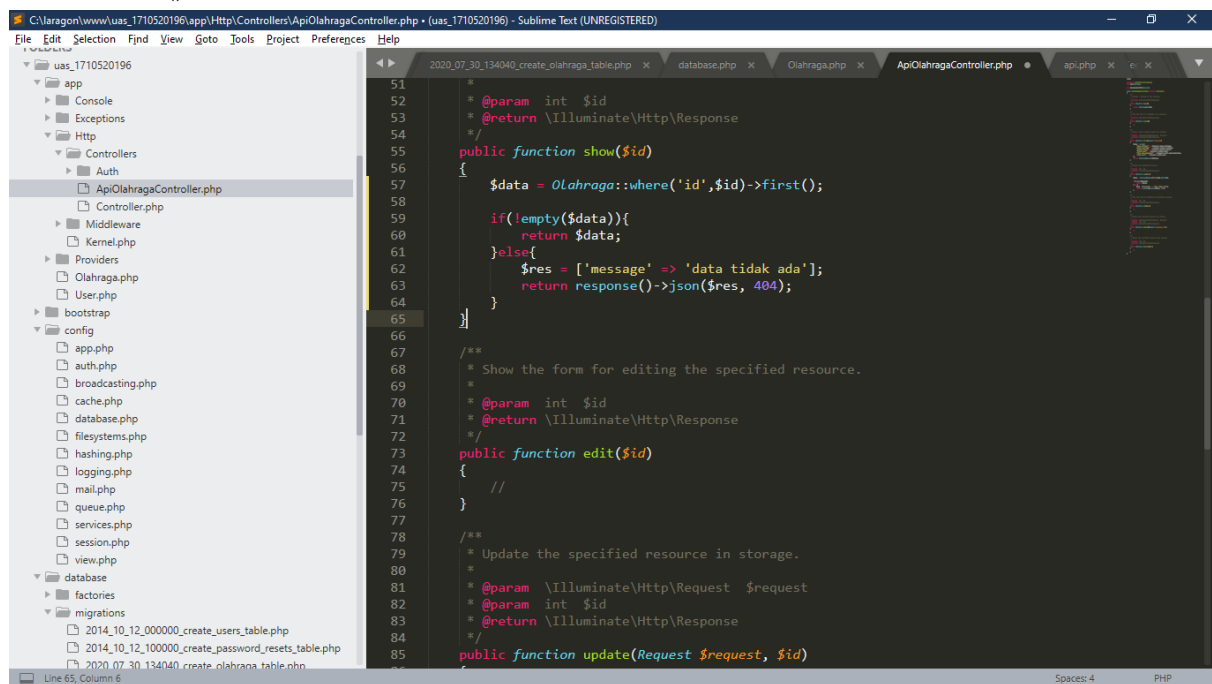
### 24. Dan pada file routes yaitu api.php seperti berikut



25. Kemudian lakukan tes menggunakan aplikasi Postman, setelah melakukan perintah *php artisan serve*



26. Kemudian pada untuk menampilkan data dari database tambahkan sintaks berikut pada fungsi *show()*



## 27. Kemudian route nya seperti berikut

```

1 <?php
2 use Illuminate\Http\Request;
3
4
5 /*
6 -----
7 API Routes
8 -----
9
10 Here is where you can register API routes for your application. These
11 routes are loaded by the RouteServiceProvider within a group which
12 is assigned the 'api' middleware group. Enjoy building your API!
13 */
14
15 Route::middleware('auth:api')->get('/user', function (Request $request) {
16     return $request->user();
17 });
18
19
20 Route::get('/', 'ApiOlahragaController@index');
21 Route::get('olahraga/{id}', 'ApiOlahragaController@show');
22

```

## 28. Jika di tes menggunakan Postman seperti berikut

GET localhost:8000/api/olahraga/2

Params Authorization Headers (7) Body Pre-request Script Tests Settings Cookies Code

Query Params

| KEY | VALUE | DESCRIPTION |
|-----|-------|-------------|
| Key | Value | Description |

Body Cookies (2) Headers (9) Test Results Status: 200 OK Time: 1051 ms Size: 615 B Save Response

Pretty Raw Preview Visualize JSON

```

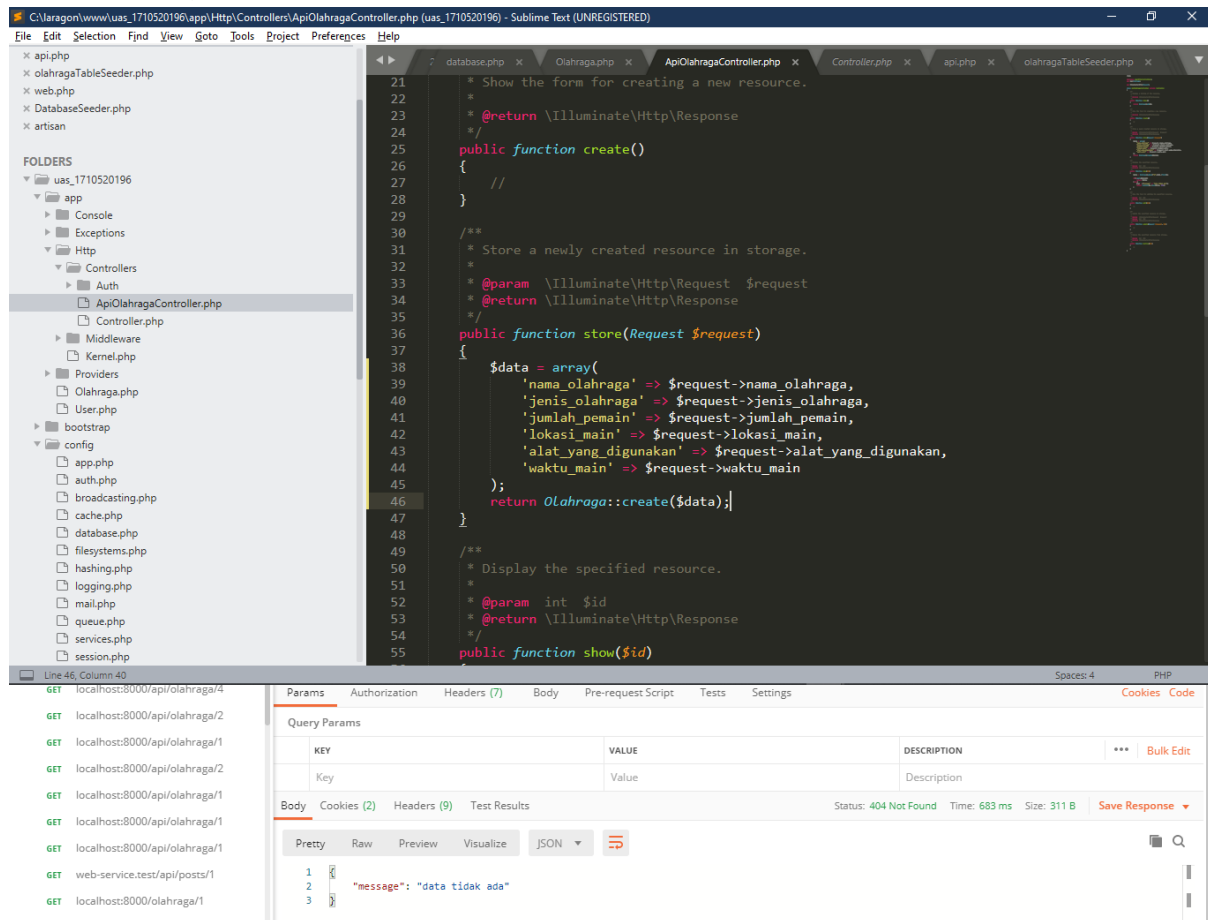
1 {
2   "id": 2,
3   "nama_olahraga": "Bulu Tangkis",
4   "jenis_olahraga": "Raket",
5   "jumlah_pemain": "dua orang pada sektor tunggal atau dua pasangan pada sektor ganda",
6   "lokasi_main": "Lapangan Bulu Tangkis",
7   "alat_yang_digunakan": "Raket, Bola Kok",
8   "waktu_main": "45 menit (Tergantung Wasit)",
9   "created_at": "2020-07-30 16:32:41",
10  "updated_at": "2020-07-30 16:32:41"
11 }

```

1710520196

Angka 2 menunjukkan id dari Olahraga yang hendak ditampilkan, jika data yang dicari tidak ada, maka akan menampilkan pesan error seperti berikut

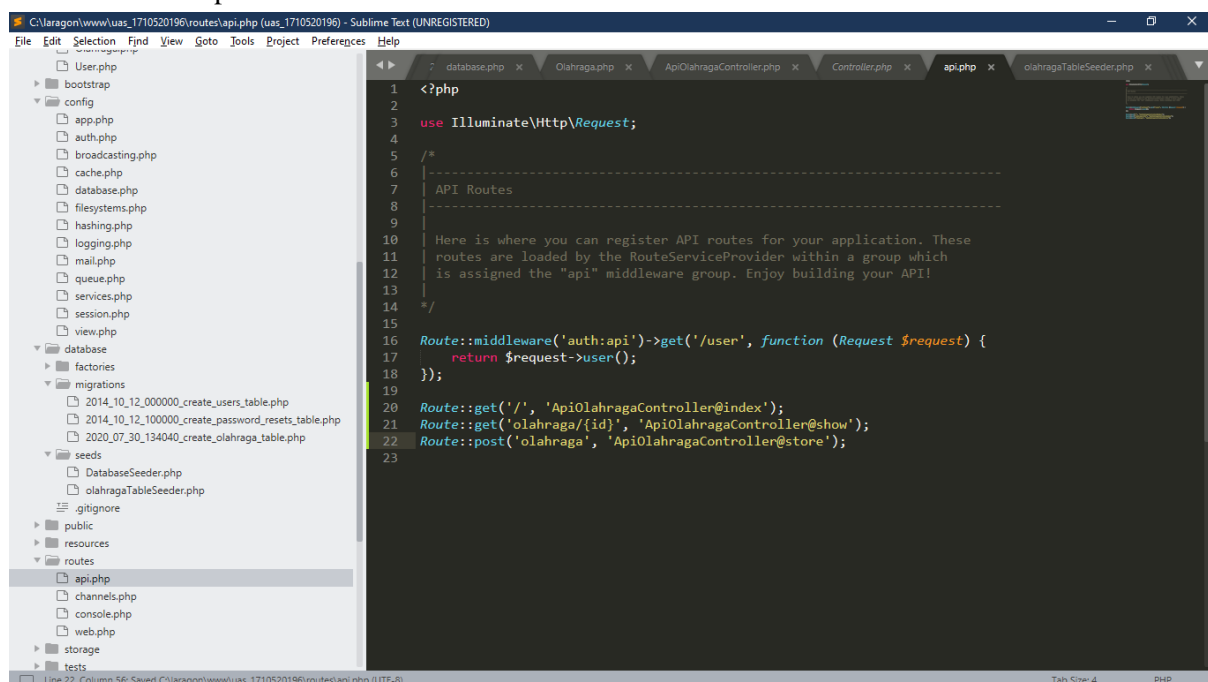
29. Menyimpan data ke database digunakan **controller** pada fungsi **store()** dengan script seperti berikut,



The screenshot displays a web application interface with a REST client and a PHP controller. The REST client shows a GET request to localhost:8000/api/olahraga/1 with a 404 Not Found status and a response body of {\"message\": \"data tidak ada\"}. The PHP controller shows the store function in ApiOlahragaController.php.

```
21 * Show the form for creating a new resource.
22 *
23 * @return \\Illuminate\\Http\\Response
24 */
25 public function create()
26 {
27     //
28 }
29
30 /**
31 * Store a newly created resource in storage.
32 *
33 * @param \\Illuminate\\HttpRequest $request
34 * @return \\Illuminate\\Http\\Response
35 */
36 public function store(Request $request)
37 {
38     $data = array(
39         'nama_olahraga' => $request->nama_olahraga,
40         'jenis_olahraga' => $request->jenis_olahraga,
41         'jumlah_pemain' => $request->jumlah_pemain,
42         'lokasi_main' => $request->lokasi_main,
43         'alat_yang_digunakan' => $request->alat_yang_digunakan,
44         'waktu_main' => $request->waktu_main
45     );
46     return Olahraga::create($data);
47 }
48
49 /**
50 * Display the specified resource.
51 *
52 * @param int $id
53 * @return \\Illuminate\\Http\\Response
54 */
55 public function show($id)
```

30. Method **POST** digunakan untuk input data (mengakses fungsi **store** pada **controller**) ke database pada **route** berikut

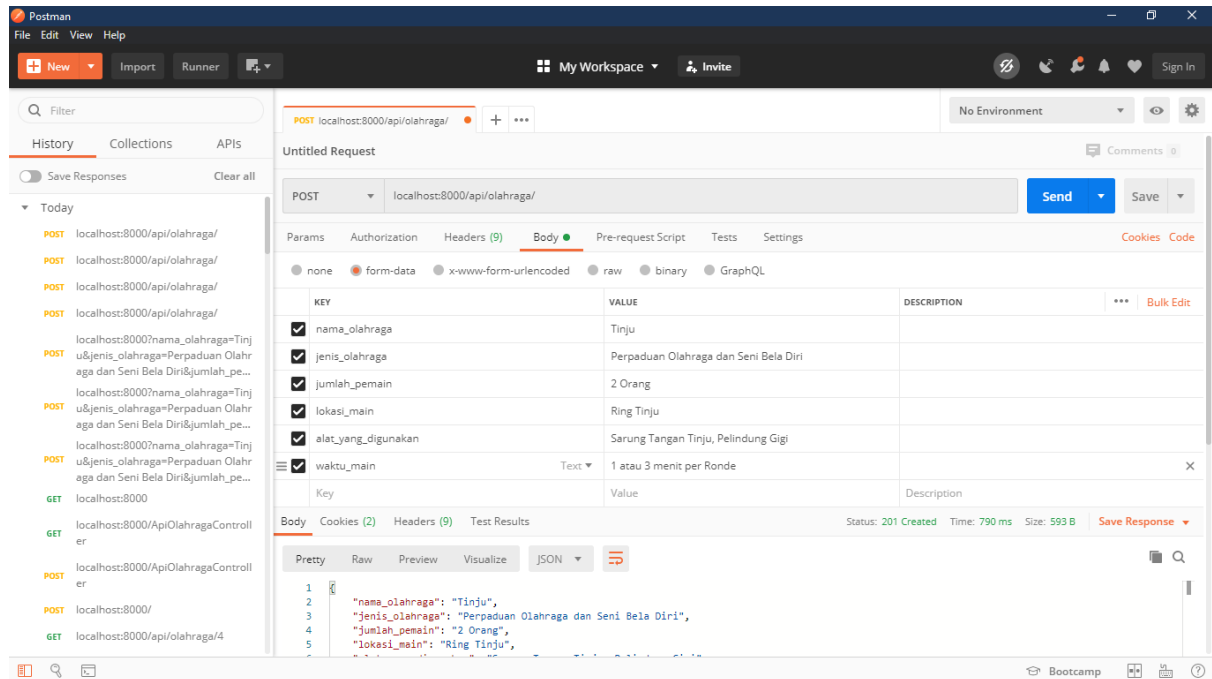


The screenshot displays a web application interface with a REST client and a PHP route file. The REST client shows a POST request to localhost:8000/api/olahraga/1 with a 404 Not Found status and a response body of {\"message\": \"data tidak ada\"}. The PHP route file shows the store function in ApiOlahragaController.php.

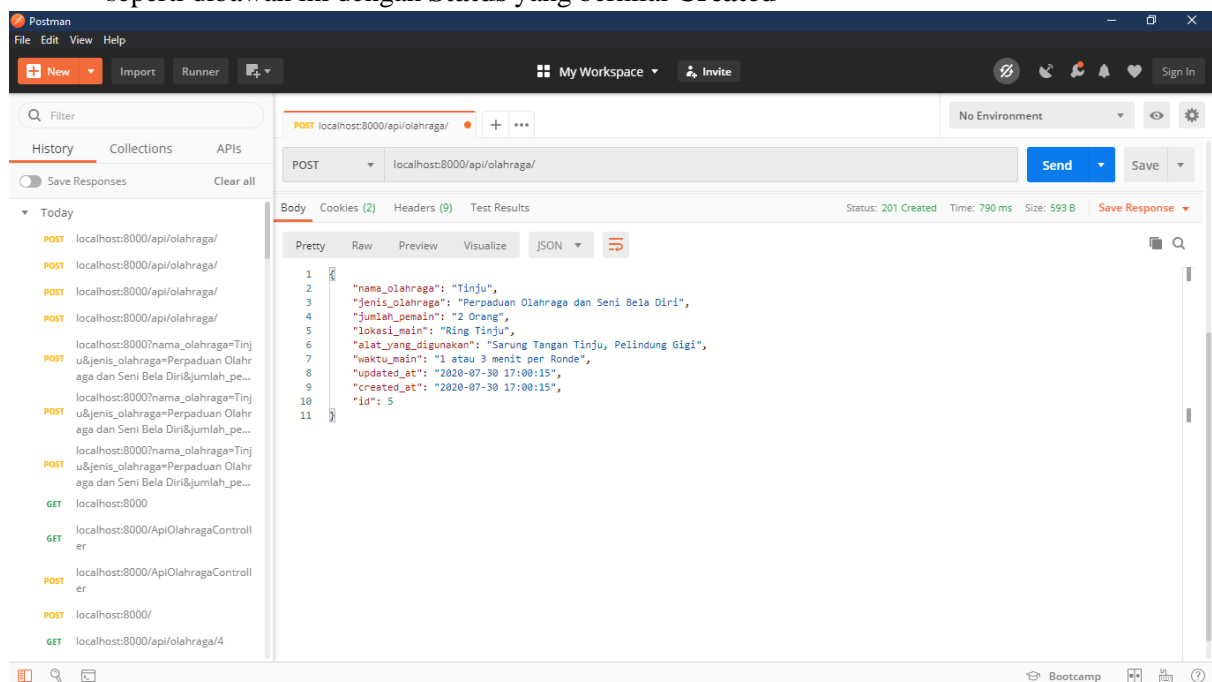
```
1 <?php
2
3 use Illuminate\\Http\\Request;
4
5 /*
6 |-----
7 | API Routes
8 |-----
9 |
10 | Here is where you can register API routes for your application. These
11 | routes are loaded by the RouteServiceProvider within a group which
12 | is assigned the "api" middleware group. Enjoy building your API!
13 |
14 | */
15
16 Route::middleware('auth:api')->get('/user', function (Request $request) {
17     return $request->user();
18 });
19
20 Route::get('/', 'ApiOlahragaController@index');
21 Route::get('olahraga/{id}', 'ApiOlahragaController@show');
22 Route::post('olahraga', 'ApiOlahragaController@store');
23
```



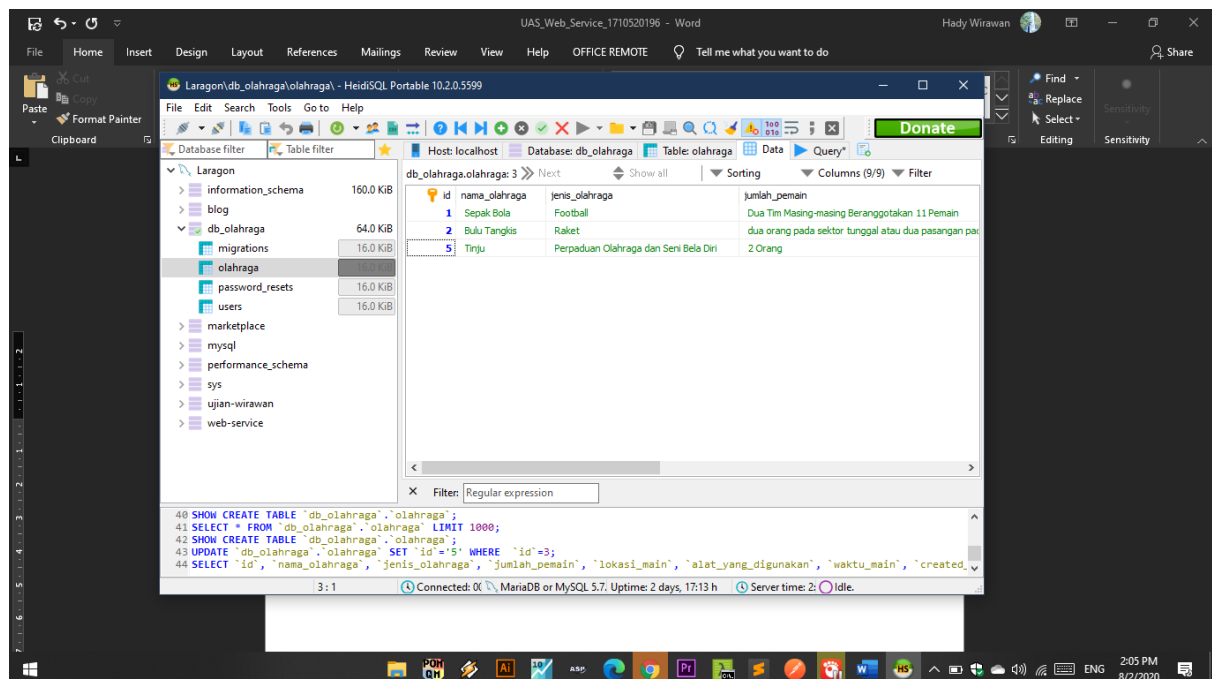
31. Untuk mencobanya, gunakan pula method **POST** pada aplikasi Postman, dan menggunakan **form-data** di tab menu **Body**, pada bagian **key** diisi sesuai dengan nama variabel di database (db\_olahraga) kemudian isikan nilai **value** sesuai dengan yang harus diisikan



Jika berhasil maka data akan berhasil ditambahkan ke database db\_olahraga dan akan tampil seperti dibawah ini dengan **Status** yang bernilai **Created**

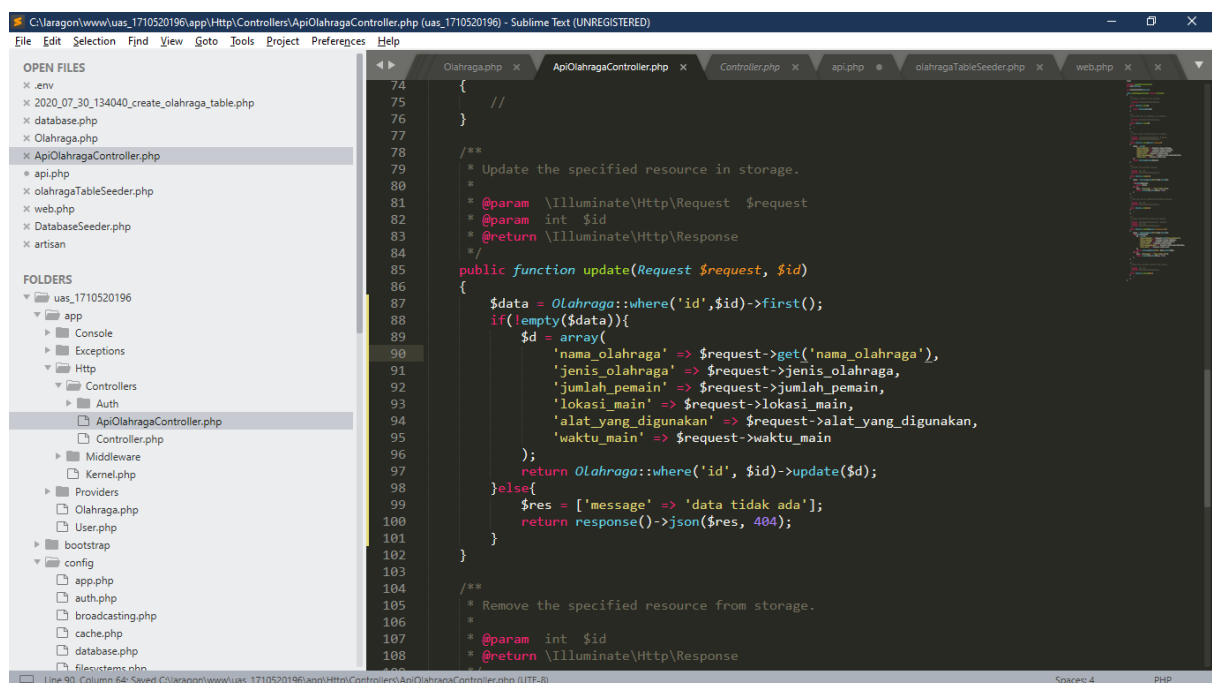


32. Jika di cek pada database maka data yang tadi diinputkan akan tampil seperti berikut



Catatan : id yang tercetak adalah 5 karena 2x input sebelumnya error karena kesalahan dalam percobaan kemudian id ke 3 dan 4 dihapus manual, dan id terus akan melanjutkan *increment* kecuali melakukan *refresh* pada *migrate* sehingga semua data yang ada di db\_olahraga akan terhapus (*php artisan migrate:refresh*)

33. Melakukan update data di database maka digunakanlah **Controller** pada fungsi *update()* dengan script sebagai berikut (pada kasus ini penulis akan mencoba edit data dengan id 5)



Dan route seperti berikut

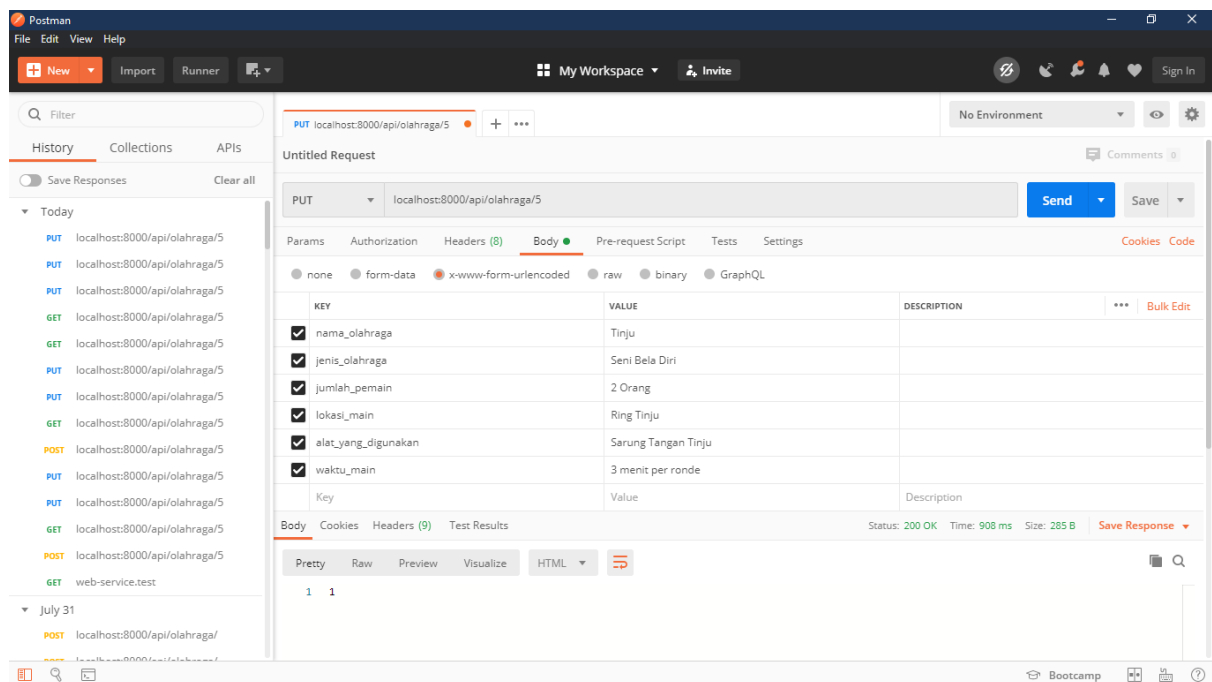
```

5  /
6  -----
7  | API Routes
8  | -----
9  |
10 | Here is where you can register API routes for your application. These
11 | routes are loaded by the RouteServiceProvider within a group which
12 | is assigned the "api" middleware group. Enjoy building your API!
13 |
14 | /
15 |
16 | Route::middleware('auth:api')->get('/user', function (Request $request) {
17 |     return $request->user();
18 | });
19 |
20 | Route::get('/', 'ApiOlahragaController@index');
21 | Route::get('olahraga/{id}', 'ApiOlahragaController@show');
22 | Route::post('olahraga', 'ApiOlahragaController@store');
23 | Route::put('olahraga/{id}', 'ApiOlahragaController@update');
24 |

```

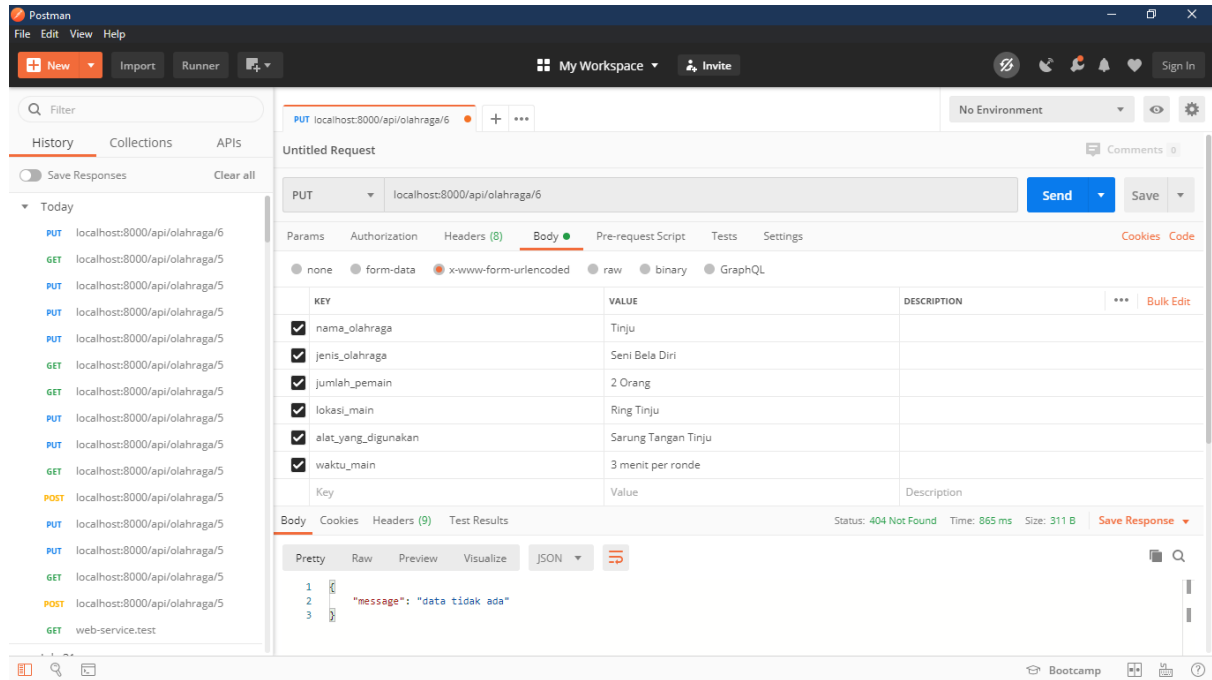
Untuk melakukan update data pada database digunakanlah **method dan route PUT**

34. Hampir sama seperti saat melakukan input data, namun untuk melakukan proses update dilakukan pada tab menu **Body** kemudian di **x-www-form-urlencoded** isikan **key dan value** yang ingin dirubah. Ketika data berhasil diupdate maka akan tampil angka 1

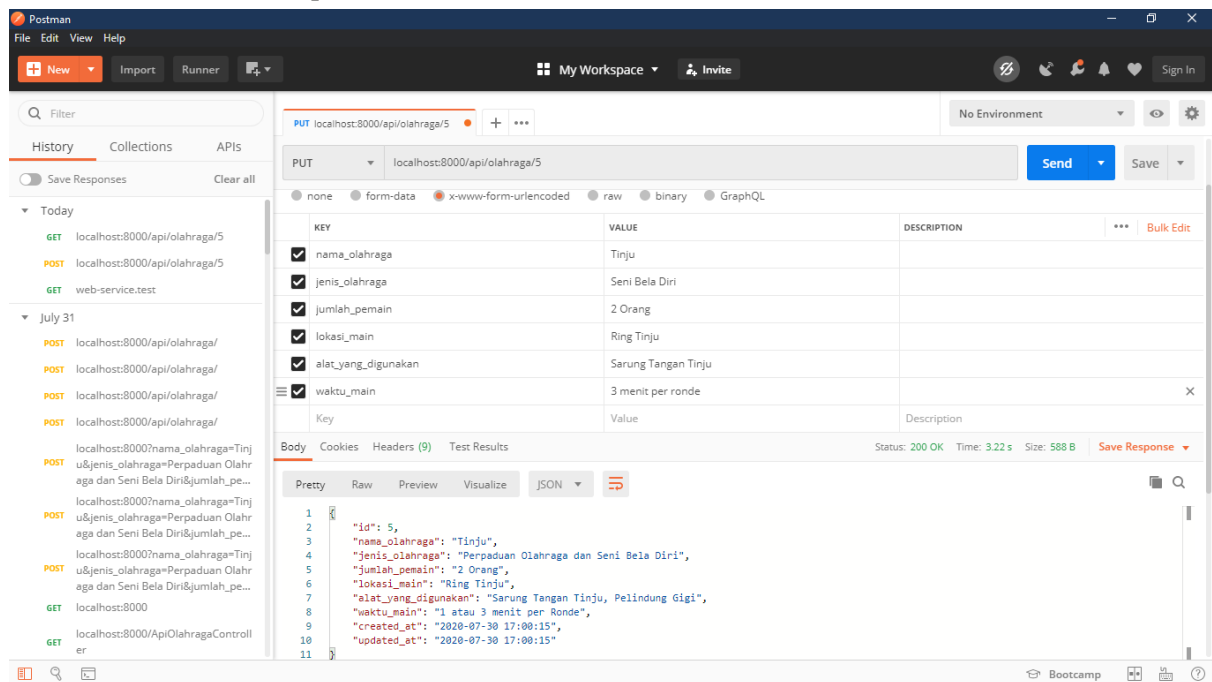


1710520196

Ketika data yang diupdate tidak ada, maka akan ditampilkan pesan berupa json text sesuai dengan script diatas pada fungsi **update()** seperti berikut

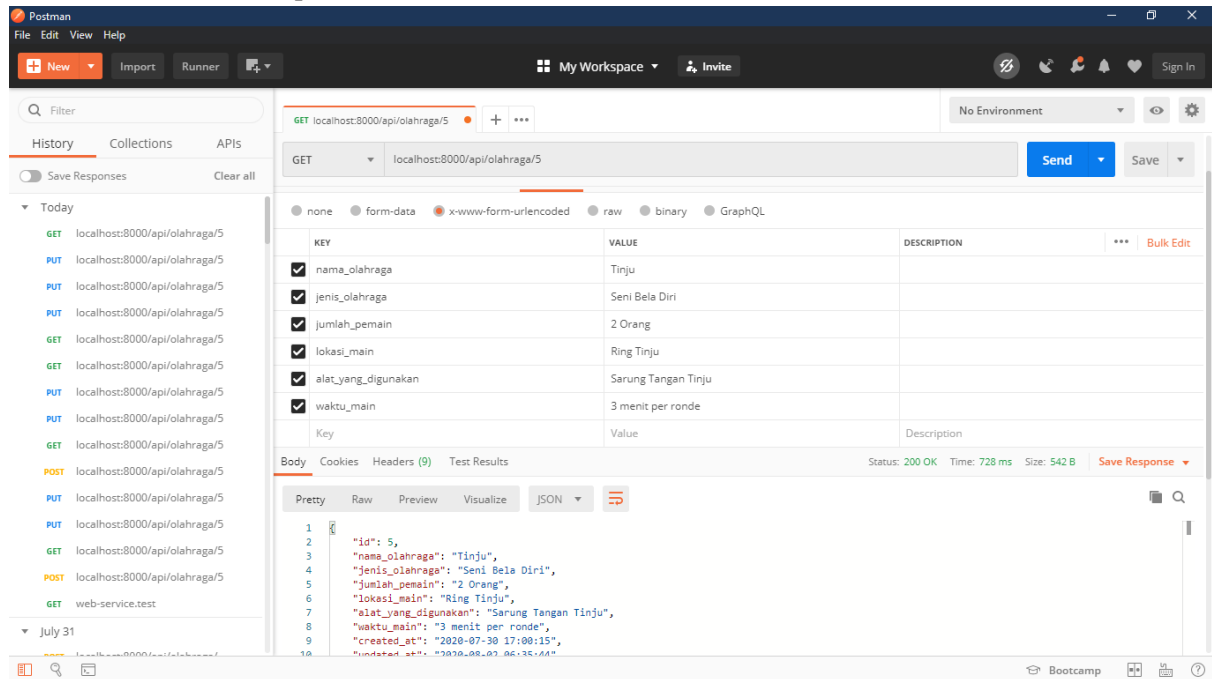


Data sebelum diupdate



1710520196

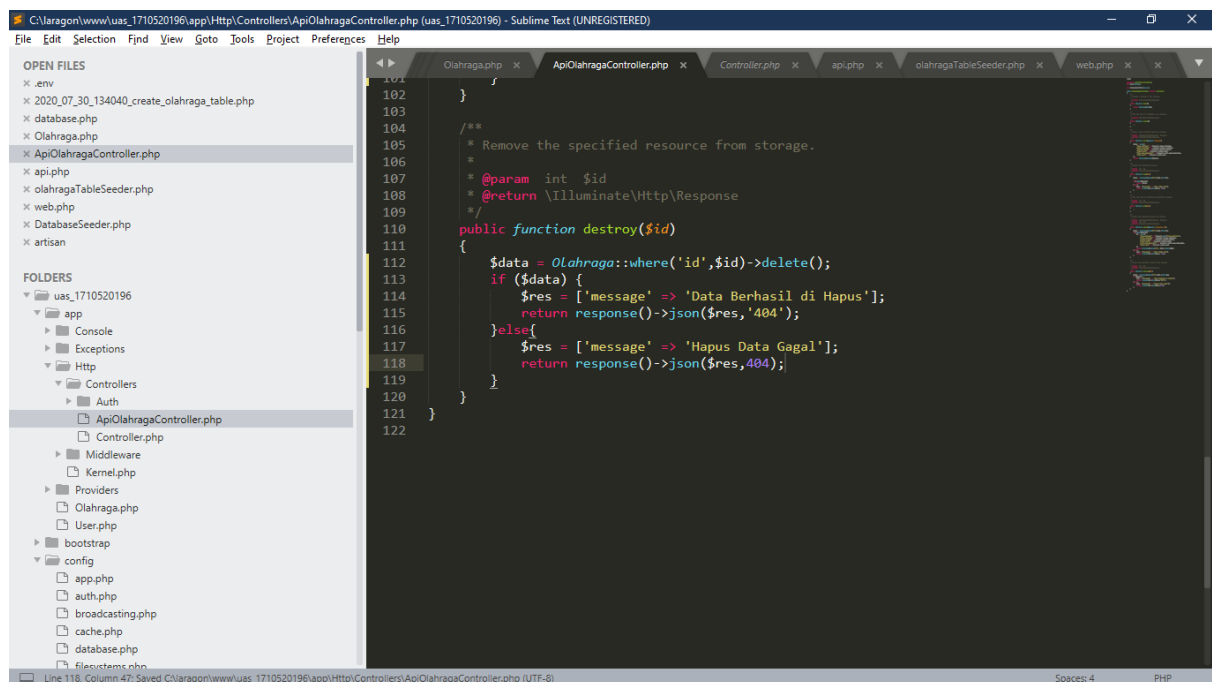
## Data setelah diupdate



Postman interface showing a GET request to `localhost:8000/api/olahraga/5`. The response status is 200 OK. The response body is a JSON object:

```
{
  "id": 5,
  "nama_olahraga": "Tinju",
  "jenis_olahraga": "Seni Bela Diri",
  "jumlah_pemain": "2 Orang",
  "lokasi_main": "Ring Tinju",
  "alat_yang_digunakan": "Sarung Tangan Tinju",
  "waktu_main": "3 menit per ronde",
  "created_at": "2020-07-30 17:00:15",
  "updated_at": "2020-08-03 06:35:44"
}
```

## 35. Menghapus data di database maka digunakan fungsi pada **Controller** yakni *destroy()* Dengan *script* seperti berikut

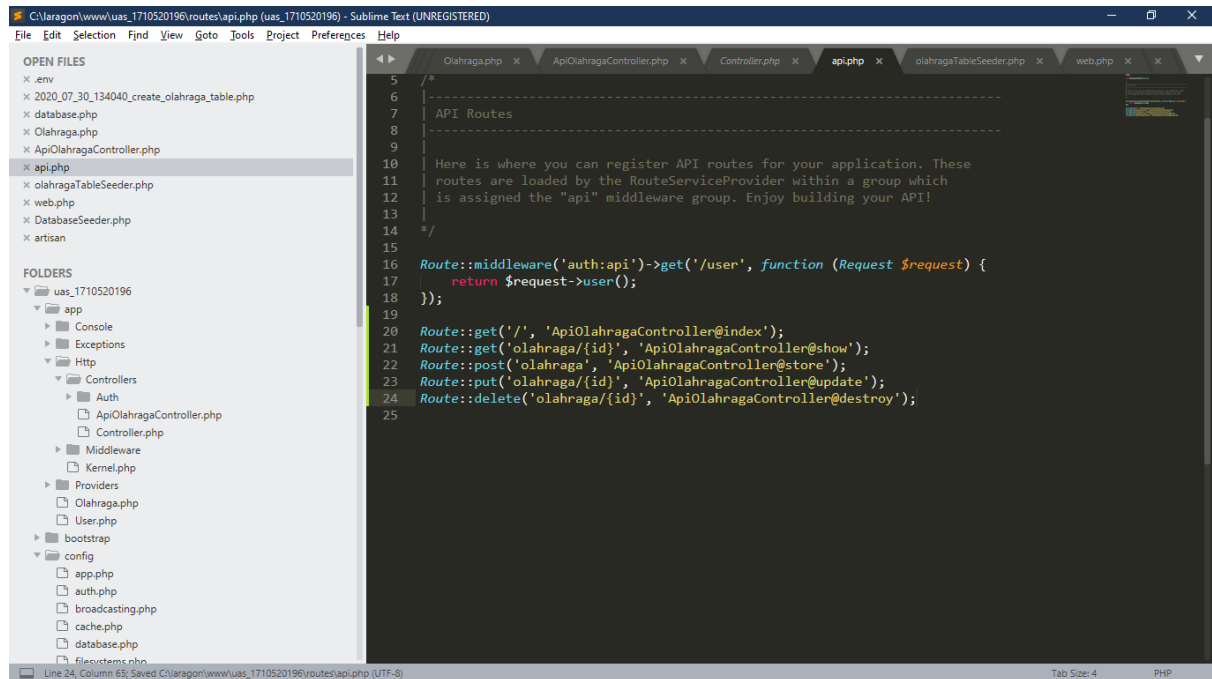


Sublime Text editor showing the `destroy()` function in `ApiOlahragaController.php`:

```
101 }
102
103
104 /**
105  * Remove the specified resource from storage.
106  *
107  * @param int $id
108  * @return \Illuminate\Http\Response
109  */
110 public function destroy($id)
111 {
112     $data = Olahraga::where('id',$id)->delete();
113     if ($data) {
114         $res = ['message' => 'Data Berhasil di Hapus'];
115         return response()->json($res, 404);
116     }else{
117         $res = ['message' => 'Hapus Data Gagal'];
118         return response()->json($res, 404);
119     }
120 }
121
122 }
```

1710520196

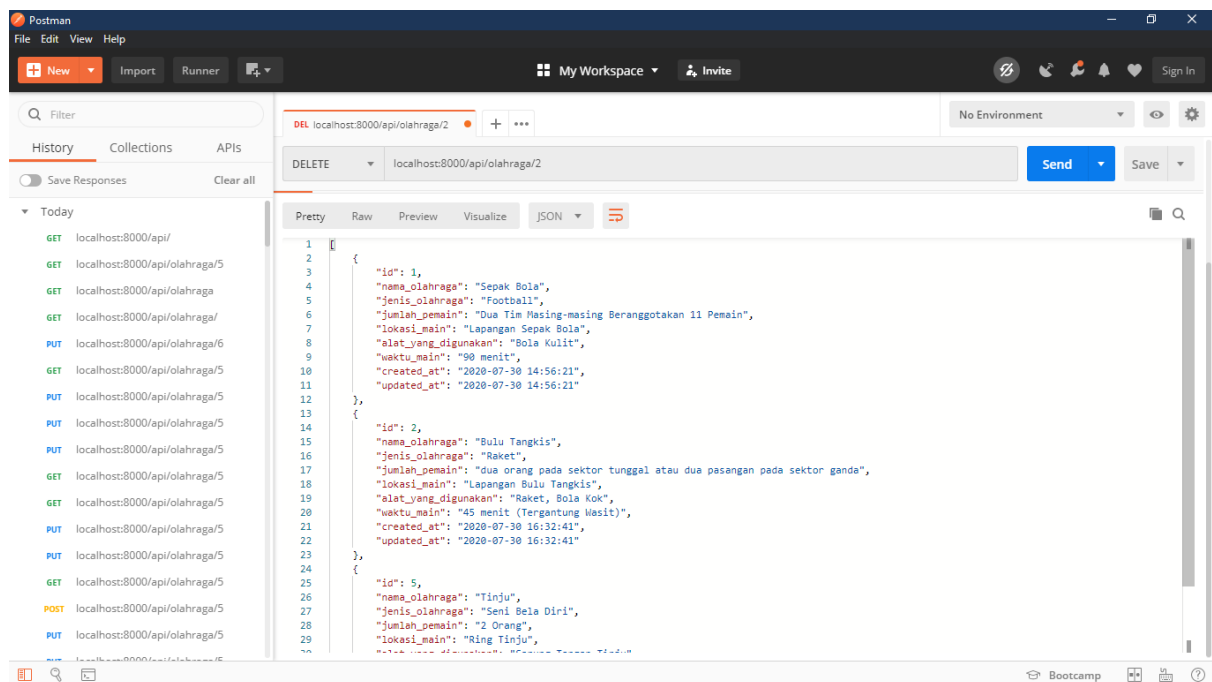
Dan route seperti berikut



```
5  /*
6  -----
7  API Routes
8  -----
9
10 Here is where you can register API routes for your application. These
11 routes are loaded by the RouteServiceProvider within a group which
12 is assigned the "api" middleware group. Enjoy building your API!
13
14 */
15
16 Route::middleware('auth:api')->get('/user', function (Request $request) {
17     return $request->user();
18 });
19
20 Route::get('/', 'ApiOlahragaController@index');
21 Route::get('olahraga/{id}', 'ApiOlahragaController@show');
22 Route::post('olahraga', 'ApiOlahragaController@store');
23 Route::put('olahraga/{id}', 'ApiOlahragaController@update');
24 Route::delete('olahraga/{id}', 'ApiOlahragaController@destroy');
```

36. Untuk mengujinya pada **Postman** digunakan method **Delete**

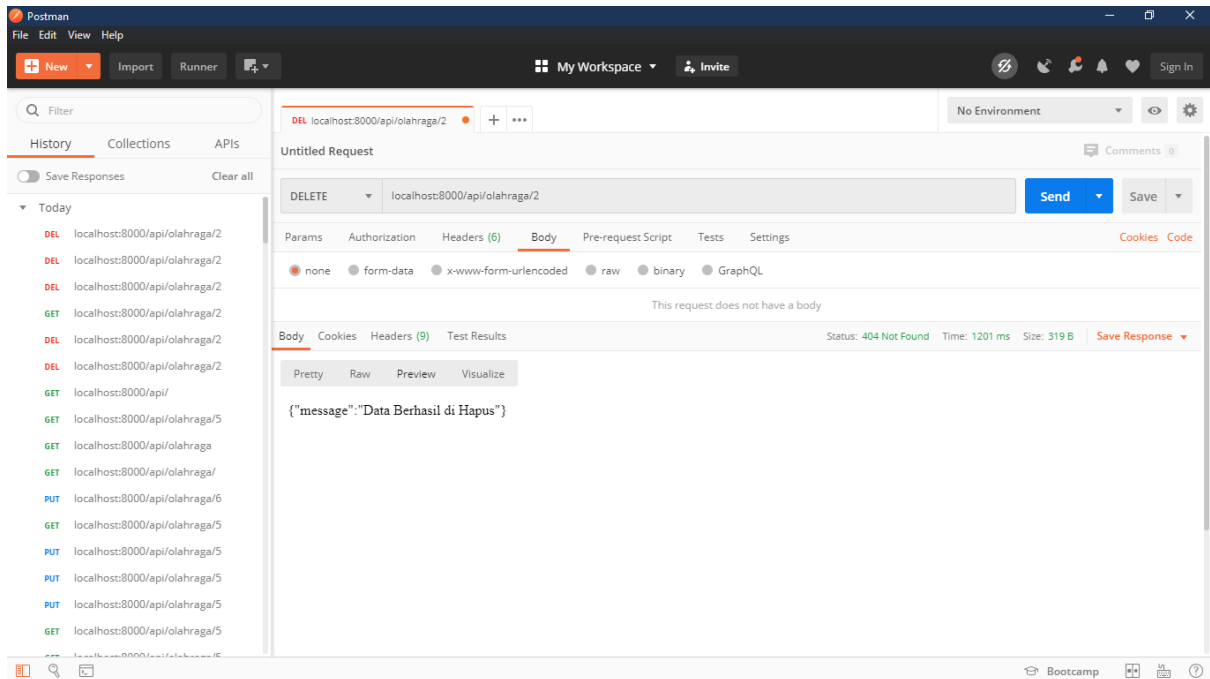
Pada kasus ini penulis akan mencoba melakukan delete terhadap data di database dengan id ke 2



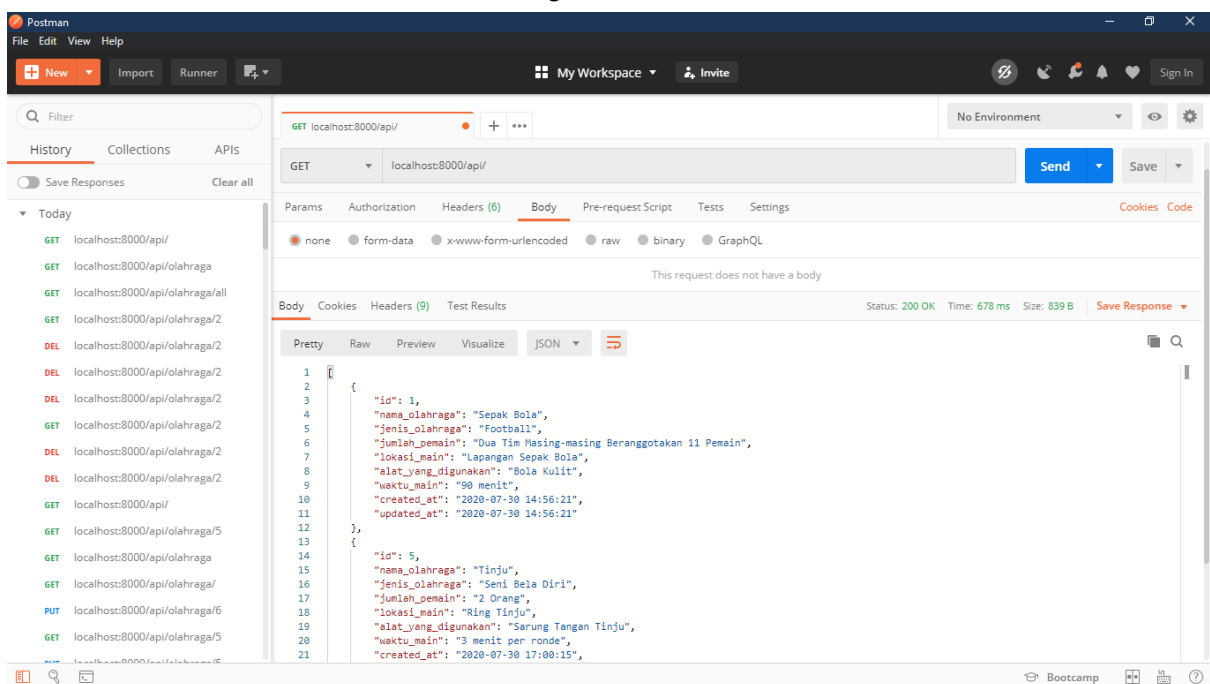
```
1 {
2   {
3     "id": 1,
4     "nama_olahraga": "Sepak Bola",
5     "jenis_olahraga": "Football",
6     "jumlah_pemain": "Dua Tim Masing-masing Beranggotakan 11 Pemain",
7     "lokasi_main": "Lapangan Sepak Bola",
8     "alat_yang_digunakan": "Bola Kulit",
9     "waktu_main": "90 menit",
10    "created_at": "2020-07-30 14:56:21",
11    "updated_at": "2020-07-30 14:56:21"
12  },
13  {
14    "id": 2,
15    "nama_olahraga": "Bulu Tangkis",
16    "jenis_olahraga": "Raket",
17    "jumlah_pemain": "dua orang pada sektor tunggal atau dua pasangan pada sektor ganda",
18    "lokasi_main": "Lapangan Bulu Tangkis",
19    "alat_yang_digunakan": "Raket, Bola Kok",
20    "waktu_main": "45 menit (Tergantung Wasit)",
21    "created_at": "2020-07-30 16:32:41",
22    "updated_at": "2020-07-30 16:32:41"
23  },
24  {
25    "id": 5,
26    "nama_olahraga": "Tinjau",
27    "jenis_olahraga": "Seni Bela Diri",
28    "jumlah_pemain": "2 Orang",
29    "lokasi_main": "Ring Tinju",
30    "created_at": "2020-07-30 16:32:41",
31    "updated_at": "2020-07-30 16:32:41"
32  }
33 }
```

1710520196

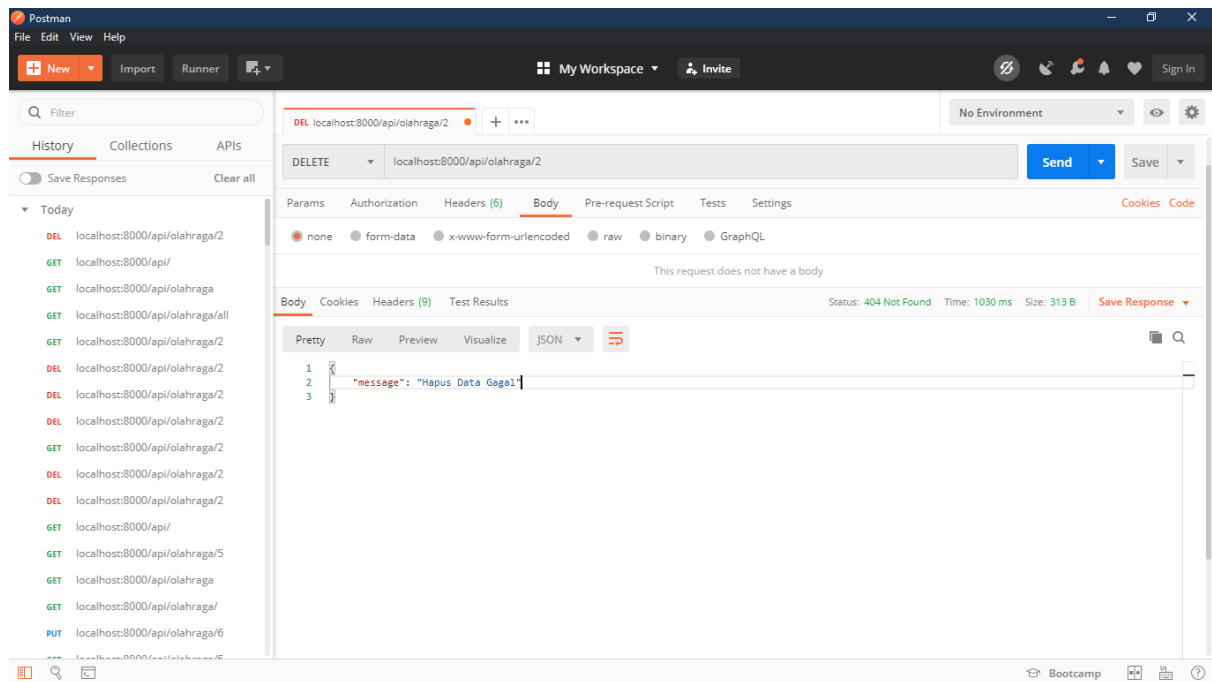
## Hasilnya



Saat di cek maka data di database dengan id ke 2 tidak ada nanda



Ketika hendak menghapus data dengan id yang tidak ada pula akan ditampilkan json text seperti berikut, sesuai dengan text yang diinputkan pada script pada fungsi ***destroy()***



Kesimpulannya, sebelum membuat Rest API menggunakan Laravel, kita harus memahami konsep dari MVC (Model, View, Controller) namun pada pertemuan dan penjelasan diatas hanya tentang bagaimana untuk membuat dan memanfaatkan Controller, mulai dari membuat database hingga memanipulasi isinya menggunakan konsep Controller. Tentunya kita harus memahami tiap-tiap metode yang digunakan pada Controller sehingga penggunaannya tidak salah. Dalam penjelasan ini penulis menggunakan Laravel versi 5.8.

Alamat github : [https://github.com/Hadywirawan18/Uas\\_RestApiLaravel\\_1710520196](https://github.com/Hadywirawan18/Uas_RestApiLaravel_1710520196)

Alamat Heroku : <https://uas-webservice-1710520196.herokuapp.com/>