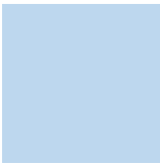


MODUL

PEMROGRAMAN BERBASIS PLATFORM



PERTEMUAN – 10

MODUL API 1

PROGRAM STUDI INFORMATIKA

FAKULTAS TEKNOLOGI INDUSTRI

UNIVERSITAS ATMA JAYA YOGYAKARTA





TUJUAN

Setelah menyelesaikan modul ini, praktikan diharapkan mampu :

1. Membuat Memahami konsep API
2. Memahami penggunaan migration, Route, Controller, dan Model pada Laravel
3. Memahami cara pembuatan Rest API menggunakan Laravel

DASAR TEORI

A. API

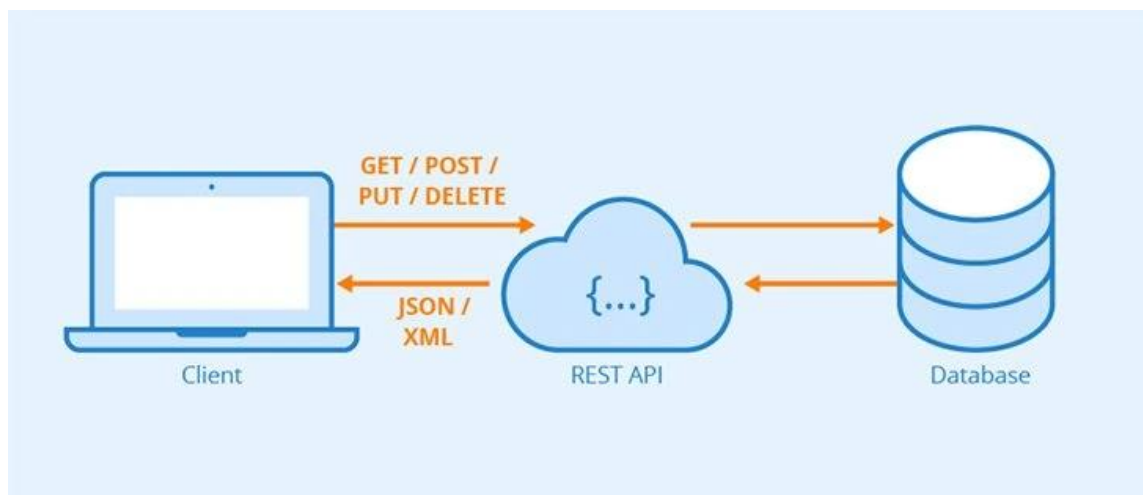
API adalah sekumpulan aturan dan protokol yang memungkinkan berbagai komponen perangkat lunak untuk saling berinteraksi. API menyediakan antarmuka yang diperlukan untuk mengakses dan memanipulasi data dari aplikasi yang berbeda. API menyediakan sekumpulan fungsi yang bisa digunakan oleh aplikasi lain. Dalam penggunaan API terdapat 3 peran: client, api, dan database. Client adalah consumer(pengguna) api. Client dapat berupa aplikasi, website, atau API lain.

Misalkan kita punya beberapa aplikasi yang saling terhubung ke database dan memiliki logika bisnis yang sama (proses login, CRUD, transaksi). Akan tidak efisien jika kita menulis ulang logika bisnis yang sama untuk tiap aplikasi. Maka dari itu API berperan untuk menampung semua logika bisnis dan juga menjembatani aplikasi dengan database.

Di modul ini kita akan membuat REST API, dimana API dan client berkomunikasi menggunakan HTTP request. REST adalah singkatan dari Representational State Transfer dan merupakan salah satu tipe arsitektur software. REST API memungkinkan aplikasi untuk berkomunikasi dengan aplikasi lain melalui protokol HTTP (Hypertext Transfer Protocol). Cara kerja REST API adalah sebagai berikut:

1. Client mengirim HTTP request ke server.

2. Server menerima request dan menjalankan aksi sesuai method dan url yang diterima.
3. Server membuat response dan mengirimnya ke client.
4. Client mendapat dan mengolah HTTP response.



Gambar 1 Cara kerja REST API

HTTP request terdiri dari beberapa bagian utama yaitu:

1. Method: menentukan action yang ingin dilakukan, terdapat beberapa jenis method:
 - a. GET : mengambil data.
 - b. POST : mengirim data baru.
 - c. PUT/PATCH : memperbarui data.
 - d. DELETE : menghapus data.
2. URL (Uniform Resource Locator): alamat yang ingin diakses, terdiri dari protocol (http/https), nama host, dan path. Contoh: 'https://nama-domain.com'
3. Headers: menyediakan informasi tambahan tentang permintaan, seperti jenis konten yang diterima oleh klien, jenis konten yang dikirimkan oleh klien, cookie, pengaturan cache, dan banyak lagi.
4. Body: digunakan pada method POST/PUT sebagai tempat untuk data yang akan ditambahkan atau diperbarui.
5. Parameters: digunakan untuk memberi data tambahan pada URL. Dapat berupa query parameters yang ditambahkan



setelah tanda tanya pada URL (?key=value), atau parameter path yang dimasukkan langsung ke dalam path (/users/{id}).

6. Authentication: menyimpan credential seperti token atau API Key, yang digunakan untuk mengidentifikasi dan memverifikasi pengguna.

B. Laravel

Laravel adalah sebuah framework berbasis PHP. Laravel masih menjadi framework yang populer karena fiturnya yang lengkap untuk database relasional. Kita akan membuat API menggunakan Laravel. Dalam guided ini, ada beberapa komponen yang akan kita pelajari:

1. Migration

Mekanisme untuk mengelola skema database dengan baris kode. Migration dapat digunakan untuk membuat, memodifikasi, dan menghapus table beserta kolom-kolom di dalamnya. Migration berlokasi dalam folder 'database/migration'.

referensi: <https://laravel.com/docs/10.x/migrations>

2. Model

Representasi tabel pada database dalam bentuk objek. Model memungkinkan kita untuk melakukan operasi CRUD (Create, Read, Update, Delete). Dengan menggunakan ORM (Object-Relational Mapping) Eloquent, kita dapat melakukan operasi tersebut tanpa harus menulis query SQL. Model berlokasi dalam folder 'app/Models'.

referensi: <https://laravel.com/docs/10.x/eloquent>

3. Controller

Controller menyimpan logika bisnis aplikasi. Controller kita gunakan untuk memproses input, berinteraksi dengan model dan mengirim response balikan. Controller berlokasi dalam folder 'app/http/Controllers'.

referensi: <https://laravel.com/docs/10.x/controllers>



4. Route

Mekanisme untuk menentukan endpoint URL yang dapat diakses oleh client. Dalam file ini kita dapat menentukan alamat URL, method yang diperbolehkan dan aksi yang dijalankan ketika ada request masuk. Route berlokasi dalam folder 'route/'. referensi: <https://laravel.com/docs/10.x/routing>

5. Factory:

Digunakan untuk membuat data dummy dengan bantuan library faker. Dalam factory kita bisa mendeskripsikan bagaimana kita ingin tiap data pada tabel terbuat. Contoh: untuk tabel user kita ingin kolom nama diisi dengan nama depan dan umur diisi angka kisaran 20 – 60 kelipatan 5.

<https://laravel.com/docs/10.x/eloquent-factories>

6. Seeder:

Layaknya migration untuk proses DML (Data Manipulation Language). Digunakan untuk inisiasi data ke database. Dalam seeder kita memanggil factory dan mengatur berapa banyak data yang ingin dibuat dalam sekali seeding. Seperti migration, kita menggunakan command untuk menjalankan seeder.

<https://laravel.com/docs/10.x/seeding>

Tambahan:

Berikut adalah fitur lain dari Laravel yang bisa kalian pelajari secara mandiri sebagai referensi kedepannya.

7. Relationship:

Fitur dari model untuk mempermudah proses CRUD dari tabel yang berelasi (One-to-One, One-to-Many, Many-to-Many).

<https://laravel.com/docs/10.x/eloquent-relationships>

C. Postman

Postman adalah tools yang bisa digunakan untuk mengirim dan menerima HTTP request/response. Postman akan kita gunakan untuk



122 melakukan testing API yang akan dibuat di modul ini. Bagi yang belum
123 mempunyai Postman dapat mengunduh melalui link berikut:

124 <https://www.postman.com/downloads/>.

125

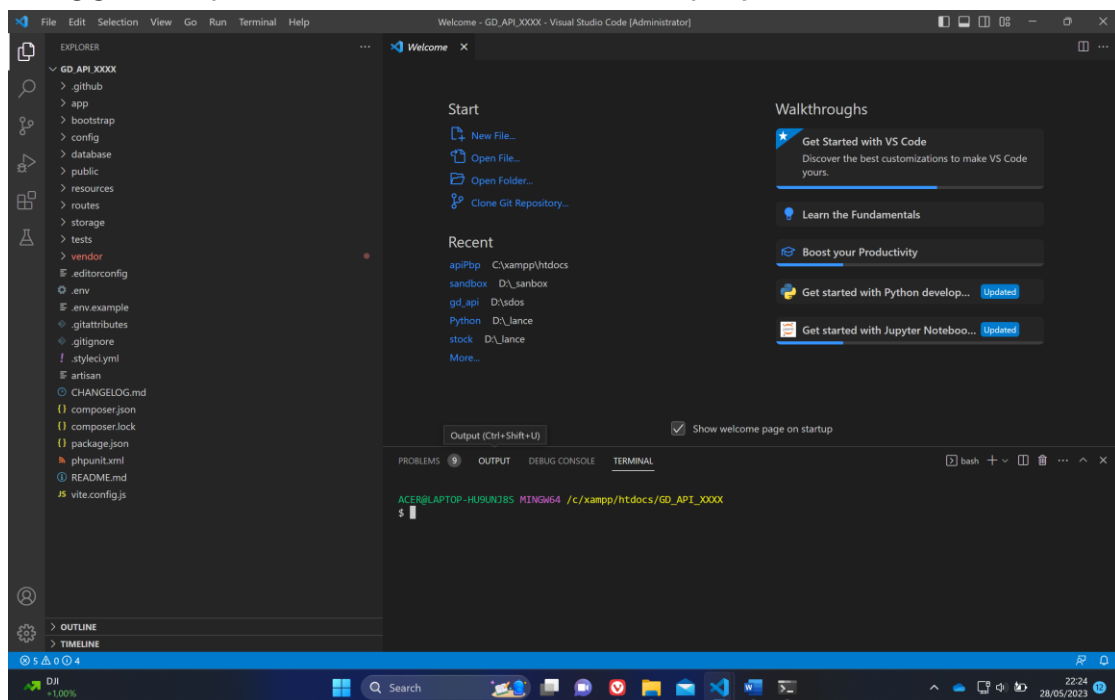
GUIDED 10 – API

Poin yang dipelajari dalam Guided 1 ini, yaitu :

1. Memahami cara membuat projek Laravel.
2. Memahami struktur folder dan file dalam Laravel.
3. Memahami fungsi migration, route, controller dan model.

Pada guided 10 ini, kita akan mencoba membuat REST API menggunakan Laravel 10. Silahkan ikuti langkah – langkah berikut ini :

1. Buka folder htdocs kalian dalam terminal lalu buat project baru dengan memasukkan command `composer create-project laravel/laravel GD_API_XXXX`.
2. Tunggu sampai selesai terinstall lalu buka project dalam VS Code.



Gambar 2 Tampilan project Laravel baru

3. Kita akan membuat Model, Migration, dan Controller sekaligus menggunakan command: `php artisan make:model Barang -mcfs --api`.

Note: disini kita menggunakan flag `-mcfs` untuk membuat Migration(m), Controller(c), Factory(f) dan Seeder(s) sekaligus.

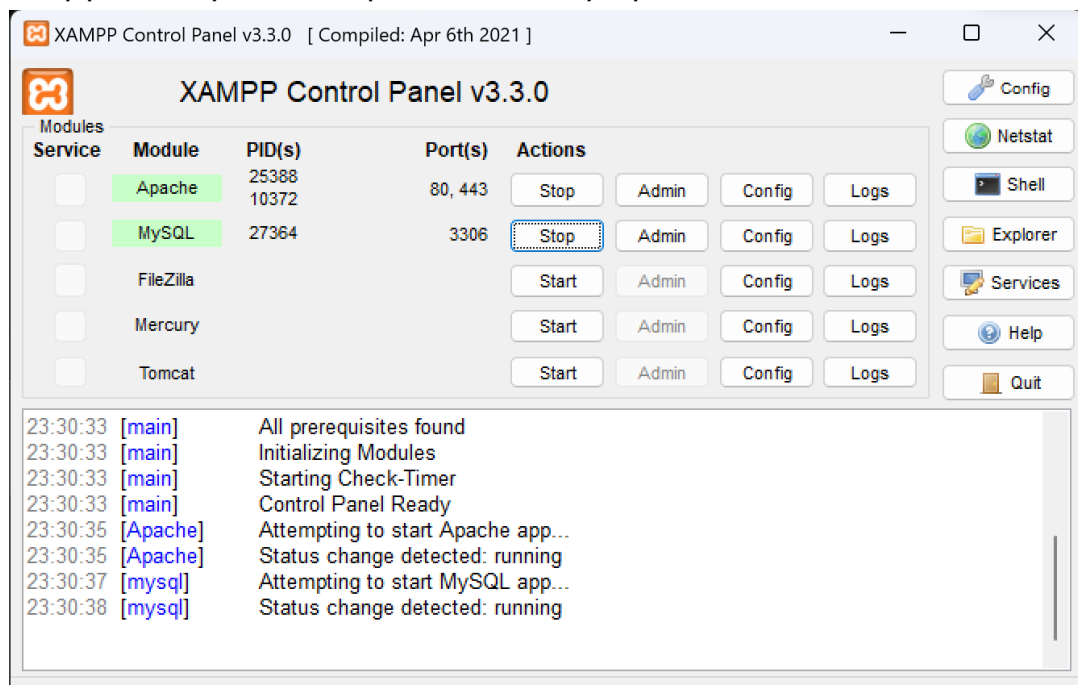
Sedangkan --api untuk membuat fungsi-fungsi kosong untuk proses CRUD di controller.

```
ACER@LAPTOP-HU9UNJ8S MINGW64 /d/xampp/htdocs/GD_API_XXXX
$ php artisan make:model Barang -mcfs --api

INFO Model [D:\xampp\htdocs\GD_API_XXXX\app\Models\Barang.php] created successfully.
INFO Factory [D:\xampp\htdocs\GD_API_XXXX\database\factories\BarangFactory.php] created successfully.
INFO Migration [D:\xampp\htdocs\GD_API_XXXX\database\Migrations\2023_11_09_105450_create_barangs_table.php] created successfully.
INFO Seeder [D:\xampp\htdocs\GD_API_XXXX\database\seeders\BarangSeeder.php] created successfully.
INFO Controller [D:\xampp\htdocs\GD_API_XXXX\app\Http\Controllers\BarangController.php] created successfully.
```

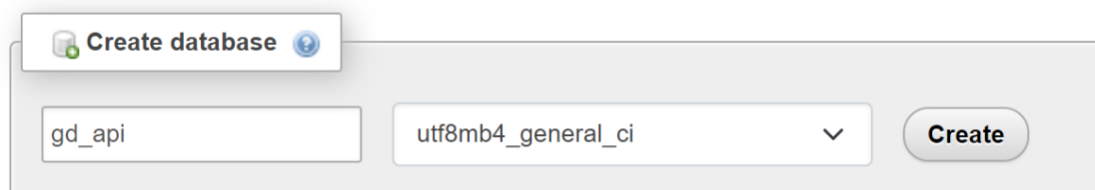
Gambar 3 Command untuk membuat model

- Kita akan mulai konfigurasi database terlebih dahulu, pertama buka xampp dan nyalakan apache dan mysql.



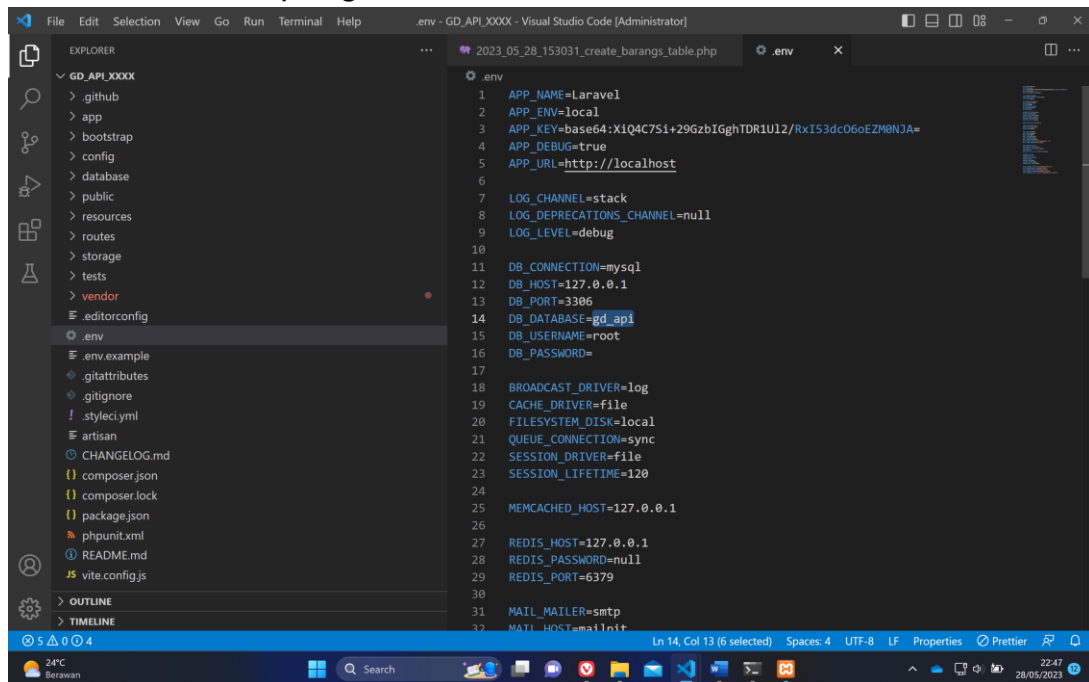
Gambar 4 Tampilan XAMPP

- Klik tombol admin pada bagian MySQL untuk masuk ke phpMyAdmin. Buat sebuah database baru dengan nama gd_api.



Gambar 5 Pembuatan database gd_api

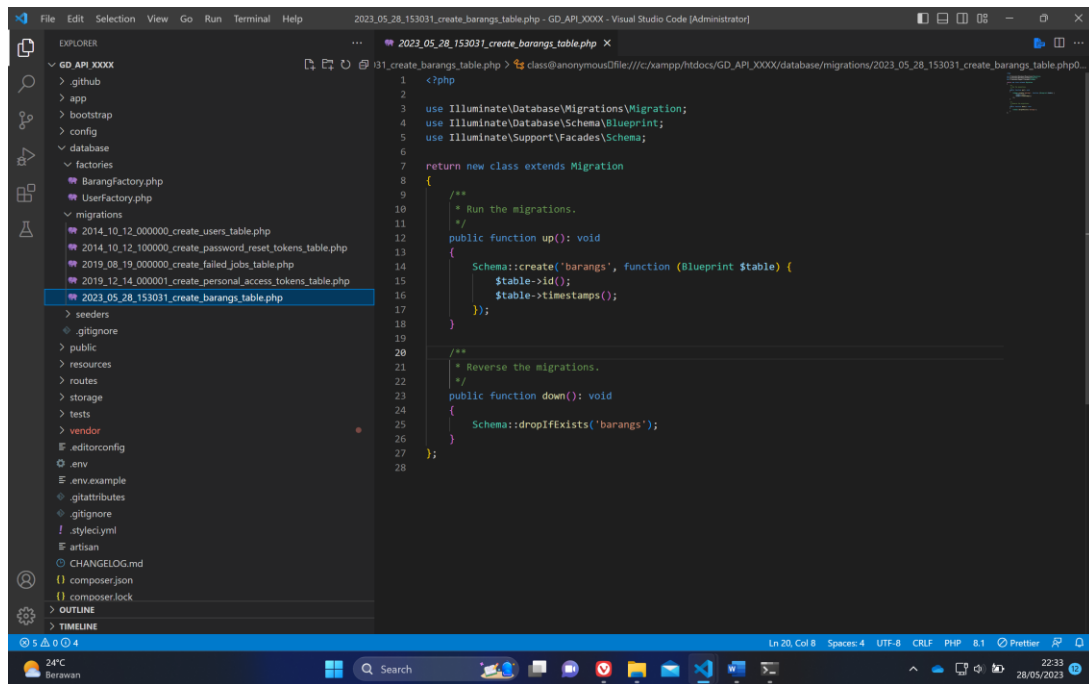
6. Kembali ke VS Code, buka file `.env` dan ubah `DB_DATABASE` dengan nama database yang telah dibuat tadi.



```
.env
1 APP_NAME=Laravel
2 APP_ENV=local
3 APP_KEY=base64:XlQ4C7S1+29GzbIGghTDR1U12/RxI53dc0GcEZM8NJA=
4 APP_DEBUG=true
5 APP_URL=http://localhost
6
7 LOG_CHANNEL=stack
8 LOG_DEPRECATIONS_CHANNEL=null
9 LOG_LEVEL=debug
10
11 DB_CONNECTION=mysql
12 DB_HOST=127.0.0.1
13 DB_PORT=3306
14 DB_DATABASE=gd_api
15 DB_USERNAME=root
16 DB_PASSWORD=
17
18 BROADCAST_DRIVER=log
19 CACHE_DRIVER=file
20 FILESYSTEM_DISK=local
21 QUEUE_CONNECTION=sync
22 SESSION_DRIVER=file
23 SESSION_LIFETIME=120
24
25 MEMCACHED_HOST=127.0.0.1
26
27 REDIS_HOST=127.0.0.1
28 REDIS_PASSWORD=null
29 REDIS_PORT=6379
30
31 MAIL_MAILER=smt
32 MAIL_HOST=mailhog
```

Gambar 6 Konfigurasi `.env`

7. Disini kita akan membuat tabel database menggunakan migrations. Buka folder `database/migrations` dan buka file paling bawah yang telah tergenerate melalui command pada Langkah 3.



Gambar 7 Konfigurasi migration

8. Definisikan kolom-kolom yang ingin kita buat pada tabel barangs. Ubah fungsi up menjadi seperti berikut:

```
public function up(): void
{
    Schema::create('barangs', function (Blueprint $table) {
        $table->id();
        $table->string("nama");
        $table->string("deskripsi");
        $table->integer("stok");
        $table->timestamps();
    });
}
```

Gambar 8 Definisi kolom pada tabel barangs

9. Lakukan migrasi dengan menjalankan command berikut: `php artisan migrate.`

```
ACER@LAPTOP-HU9UNJ8S MINGW64 /d/xampp/htdocs/GD_API_XXXX
$ php artisan migrate

INFO Preparing database.

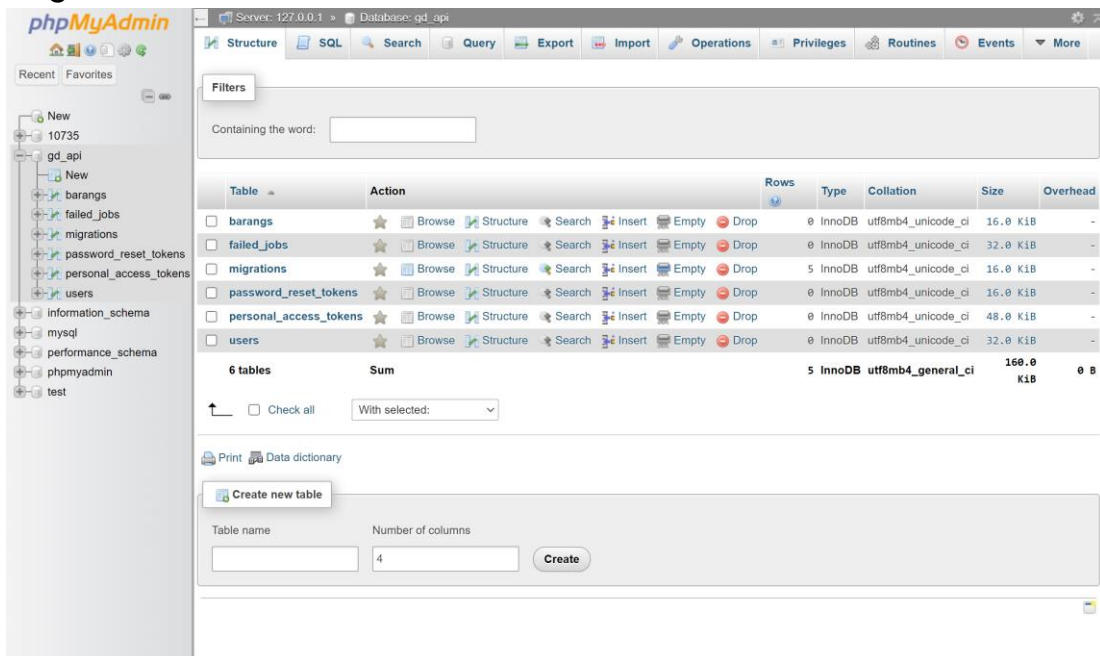
Creating migration table ..... 17ms DONE

INFO Running migrations.

2014_10_12_000000_create_users_table ..... 22ms DONE
2014_10_12_100000_create_password_reset_tokens_table ..... 24ms DONE
2019_08_19_000000_create_failed_jobs_table ..... 20ms DONE
2019_12_14_000001_create_personal_access_tokens_table ..... 31ms DONE
2023_11_09_105450_create_barangs_table ..... 9ms DONE
```

Gambar 9 Proses migrasi

10. Setelah migrasi, bisa kita lihat pada database akan muncul tabel barangs dengan kolom-kolom seperti yang sudah diatur pada migration.



Gambar 10 Hasil migration

11. Buka file app>Models>Barang.php.

Agar bisa mengirim data array ke model, kita perlu menyalakan mass assignment. Ada dua cara untuk ini, dengan membuat variabel fillable **atau** guarded:

- Fillable diisi dengan nama column yang **bisa** diterima model melalui array.
- Guarded diisi dengan nama column yang **tidak bisa** diterima model melalui array.

Referensi: <https://laravel.com/docs/10.x/eloquent#mass-assignment>



197

```
app > Models > Barang.php > ...
1  <?php
2
3  namespace App\Models;
4
5  use Illuminate\Database\Eloquent\Factories\HasFactory;
6  use Illuminate\Database\Eloquent\Model;
7
8  11 references | 0 implementations
9  class Barang extends Model
10 {
11     use HasFactory;
12
13     // pilih salah satu antara fillable atau guarded
14     0 references
15     protected $fillable = ['nama','deskripsi','stok'];
16     // protected $guarded = [];
```

198

199

Gambar 11 Isi model Barang.php

200

201 12. Buka file app>Http>Controllers>BarangController.php. Pertama kita
202 akan mengatur fungsi index untuk mendapatkan semua data dari
203 database. Ubah fungsi index menjadi berikut:



```
0 references | 0 overrides
public function index()
{
    try{
        $barang = Barang::all();
        return response()->json([
            "status" => true,
            "message" => 'Berhasil ambil data',
            "data" => $barang
        ], 200); //status code 200 = success
    }
    catch(\Exception $e){
        return response()->json([
            "status" => false,
            "message" => $e->getMessage(),
            "data" => []
        ], 400); //status code 400 = bad request
    }
}
```

Gambar 12 Fungsi index pada BarangController.php

13. Kemudian ubah fungsi show untuk mengambil data berdasarkan id.



```
0 references | 0 overrides
public function show($id)
{
    try{
        $barang = Barang::find($id);

        if(!$barang) throw new \Exception("Barang tidak ditemukan");

        return response()->json([
            "status" => true,
            "message" => 'Berhasil ambil data',
            "data" => $barang
        ], 200); //status code 200 = success
    }
    catch(\Exception $e){
        return response()->json([
            "status" => false,
            "message" => $e->getMessage(),
            "data" => []
        ], 400); //status code 400 = bad request
    }
}
```

Gambar 13 Fungsi show pada BarangController.php

14. Ubah fungsi store untuk insert data berdasarkan input yang diberikan.



```
0 references | 0 overrides
public function store(Request $request)
{
    try{
        //$request->all() untuk mengambil semua input dalam request body
        $barang = Barang::create($request->all());
        return response()->json([
            "status" => true,
            "message" => 'Berhasil insert data',
            "data" => $barang
        ], 200); //status code 200 = success
    }
    catch(\Exception $e){
        return response()->json([
            "status" => false,
            "message" => $e->getMessage(),
            "data" => []
        ], 400); //status code 400 = bad request
    }
}
```

Gambar 14 Fungsi store pada BarangController.php

15. Ubah fungsi update untuk update data berdasarkan input dan id yang diberikan.



```
0 references | 0 overrides
public function update(Request $request, $id)
{
    //
    try{
        $barang = Barang::find($id);

        if(!$barang) throw new \Exception("Barang tidak ditemukan");

        $barang->update($request->all());

        return response()->json([
            "status" => true,
            "message" => 'Berhasil update data',
            "data" => $barang
        ], 200); //status code 200 = success
    }
    catch(\Exception $e){
        return response()->json([
            "status" => false,
            "message" => $e->getMessage(),
            "data" => []
        ], 400); //status code 400 = bad request
    }
}
```

Gambar 15 Fungsi update pada BarangController.php

16. Terakhir ubah fungsi destroy untuk menghapus data berdasarkan id yang diberikan.


```
/**
 * Remove the specified resource from storage.
 */
0 references | 0 overrides
public function destroy($id)
{
    //
    try{
        $barang = Barang::find($id);

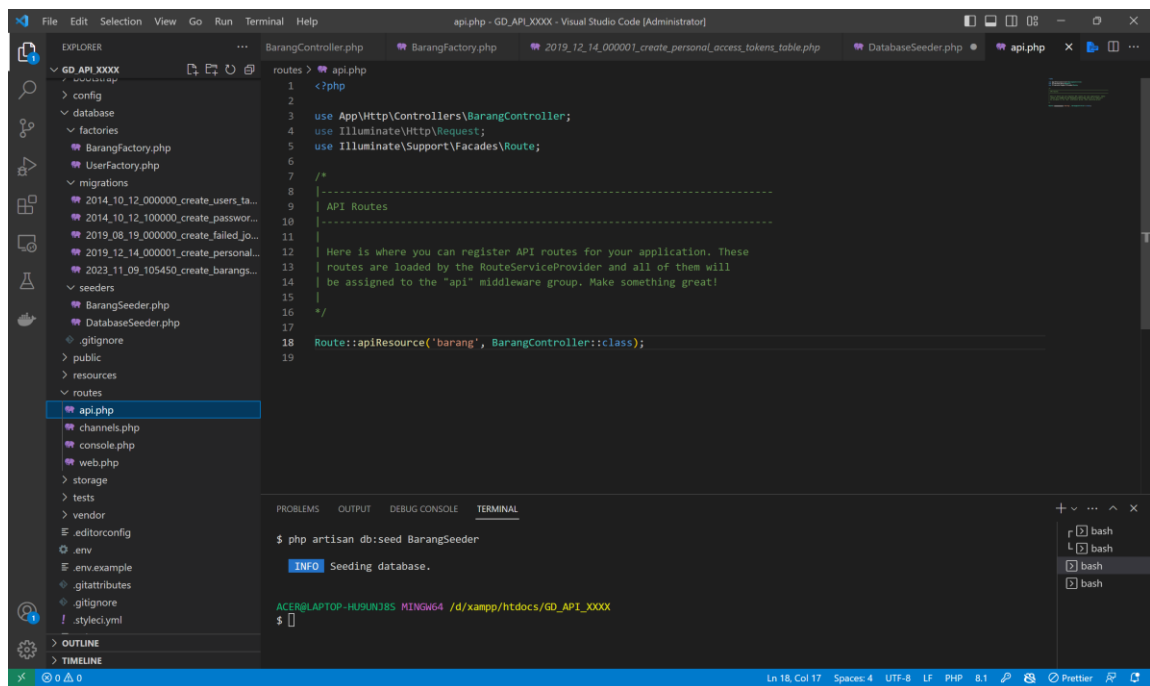
        if(!$barang) throw new \Exception("Barang tidak ditemukan");

        $barang->delete();

        return response()->json([
            "status" => true,
            "message" => 'Berhasil delete data',
            "data" => $barang
        ], 200); //status code 200 = success
    }
    catch(\Exception $e){
        return response()->json([
            "status" => false,
            "message" => $e->getMessage(),
            "data" => []
        ], 400); //status code 400 = bad request
    }
}
```

Gambar 16 Fungsi destroy pada BarangController.php

17. Sebelum bisa digunakan, kita harus menyambungkan fungsi-fungsi di controller dengan sebuah url masing-masing, untuk itu buka routes/api.php.



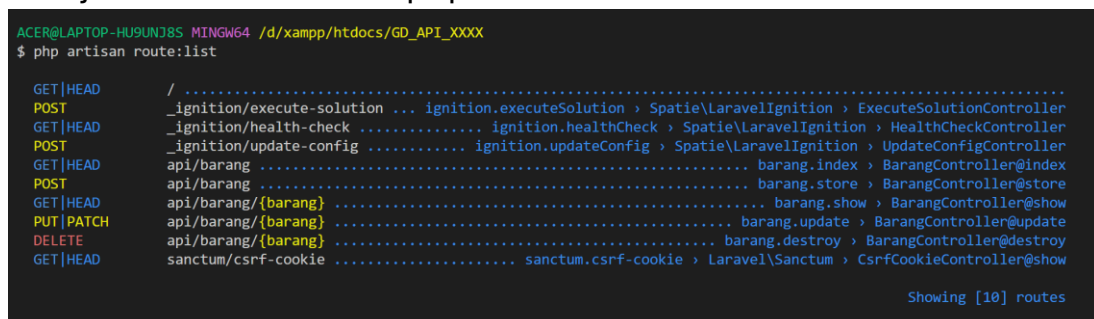
Gambar 17 Isi route api.php

18. Kita bisa menggunakan `Route::apiResource()` untuk langsung membuat route dari tiap fungsi tadi tanpa perlu membuatnya satu persatu. Tambahkan kode berikut pada file `api.php`:

```
Route::apiResource('barang', BarangController::class);
```

Gambar 18 Konfigurasi route untuk BarangController

19. Jika ingin mengecek route yang sudah terdaftar kita bisa menjalankan command `php artisan route:list`



Gambar 19 Daftar route dalam API

20. Buka `App/database/factories/BarangFactory.php`.

Sebelum mencoba API kita akan membuat data dummy menggunakan factory dan seeder. Disini kita menggunakan library faker untuk generate deskripsi dan stok.

```

0 references | 0 implementations
class BarangFactory extends Factory
{
    /**
     * Define the model's default state.
     *
     * @return array<string, mixed>
     */
    0 references | 0 overrides
    public function definition(): array
    {
        return [
            'nama' => '', //nama akan kita isi manual
            'deskripsi' => fake()->words(fake()->numberBetween(3,6),true),
            'stok' => fake()->numberBetween(5,10),
        ];
    }
}

```

Gambar 20 Isi BarangFactory.php

21. Buka App/database/seeder/ BarangSeeder.php. Disini kita memanggil factory dan menentukan berapa banyak data dummy yang ingin dibuat dalam sekali seeding.

```

<?php

namespace Database\Seeders;

use App\Models\Barang;
use Illuminate\Database\Console\Seeds\WithoutModelEvents;
use Illuminate\Database\Seeder;

0 references | 0 implementations
class BarangSeeder extends Seeder
{
    /**
     * Run the database seeds.
     */
    0 references | 0 overrides
    public function run(): void
    {
        //membuat 3 barang dengan nama yang ditentukan menggunakan sequence
        //bisa dicoba naikan count tanpa menambahkan nama baru untuk melihat bagaimana cara sequence bekerja
        Barang::factory()->count(3)->sequence(
            ['nama' => 'Apel'],
            ['nama' => 'Kelengkeng'],
            ['nama' => 'Durian'],
        )->create();
    }
}

```

Gambar 21 Isi BarangSeeder.php

22. Untuk menjalankan seeder masukkan command berikut pada terminal.

```
$ php artisan db:seed BarangSeeder
```

INFO Seeding database.

Gambar 22 Proses seeding

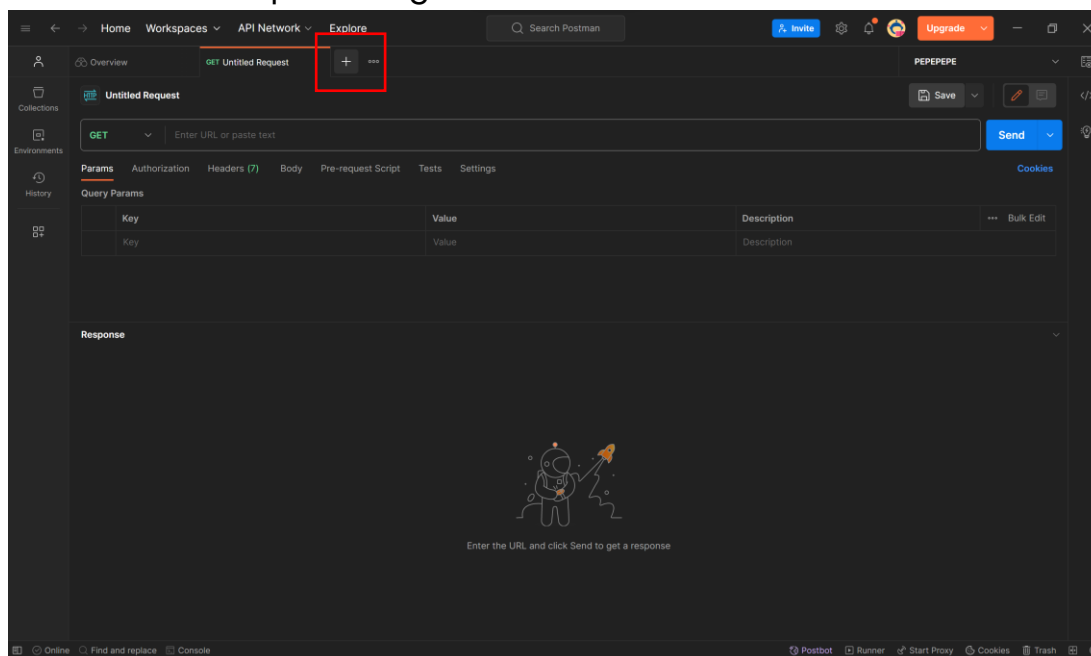
23. Terakhir kita akan mencoba API yang sudah dibuat. Pertama jalankan project dengan command berikut:

```
ACER@LAPTOP-HU9UNJ8S MINGW64 /c/xampp/htdocs/apiPbp (api)
$ php artisan serve
```

INFO Server running on [http://127.0.0.1:8000].

Gambar 23 Cara menjalankan API

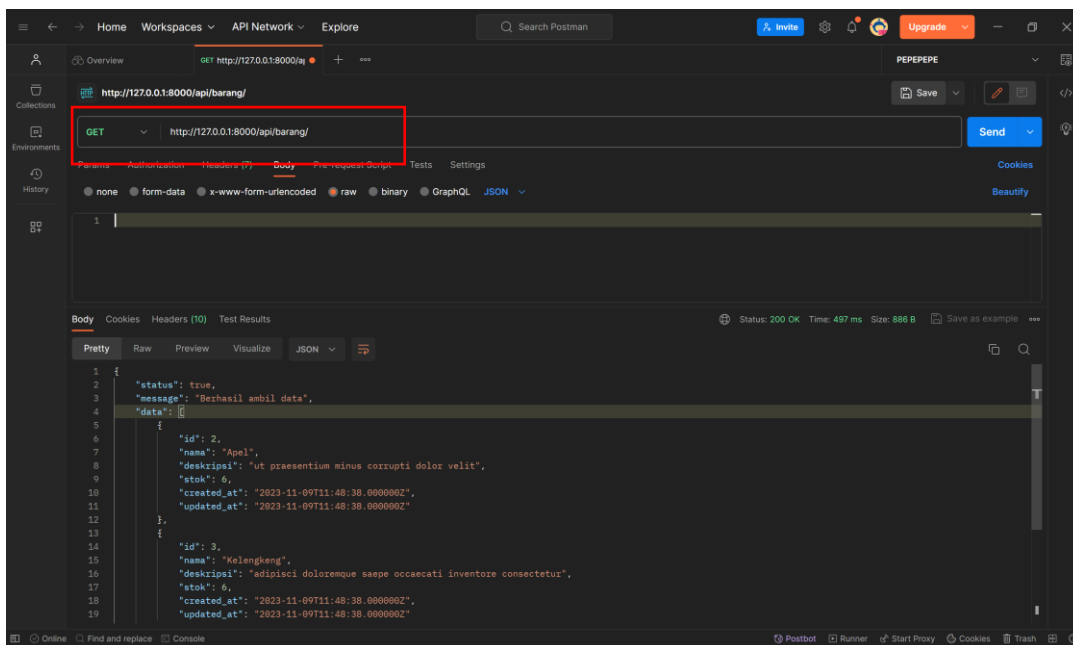
24. Buka postman dan buat sebuah request baru dengan menekan tombol tambah pada bagian atas.



Gambar 24 Tampilan Postman

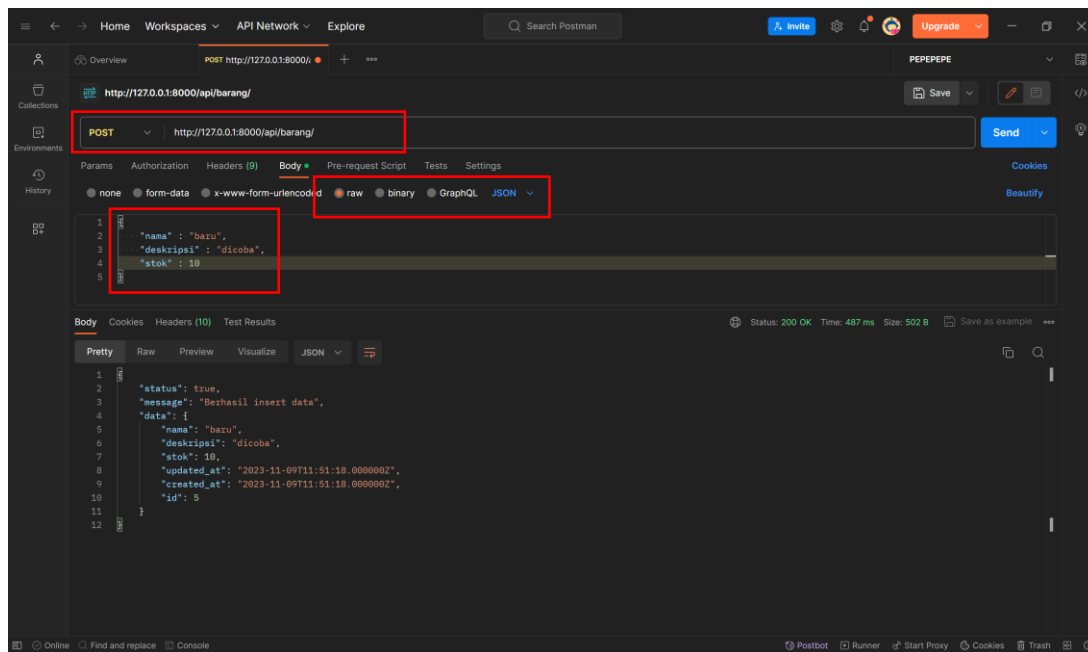
25. Masukkan url, method dan body sesuai skenario testing. Ikuti konfigurasi pada gambar kemudian tekan tombol send untuk mengirim request ke API.

Test Get:



Gambar 25 Testing Get

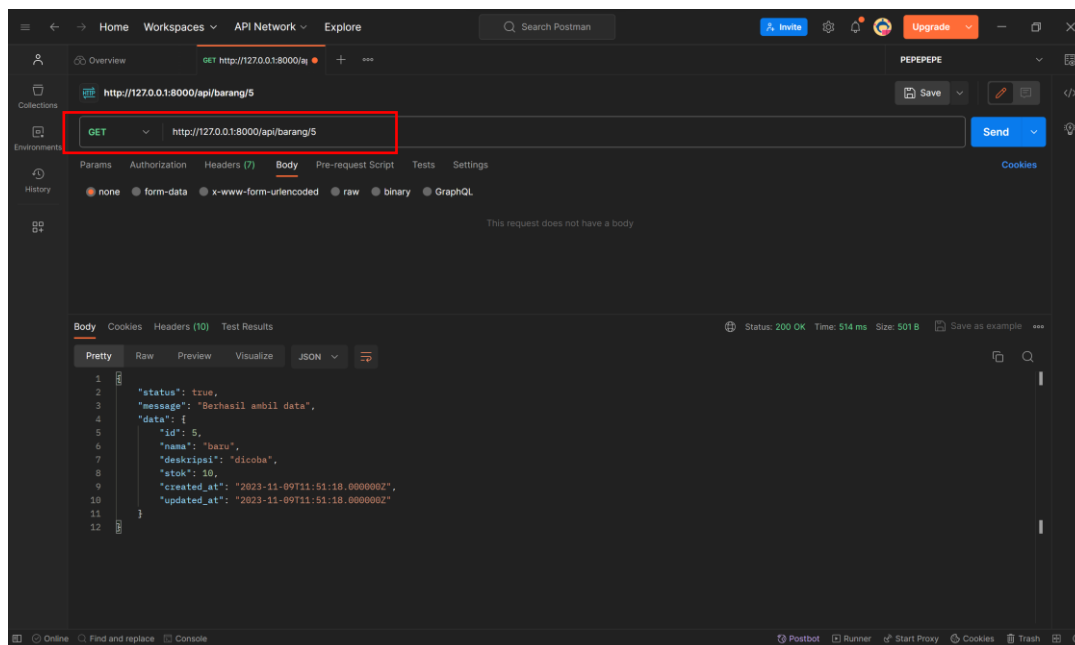
26. Test Create:



Gambar 26 Testing Create

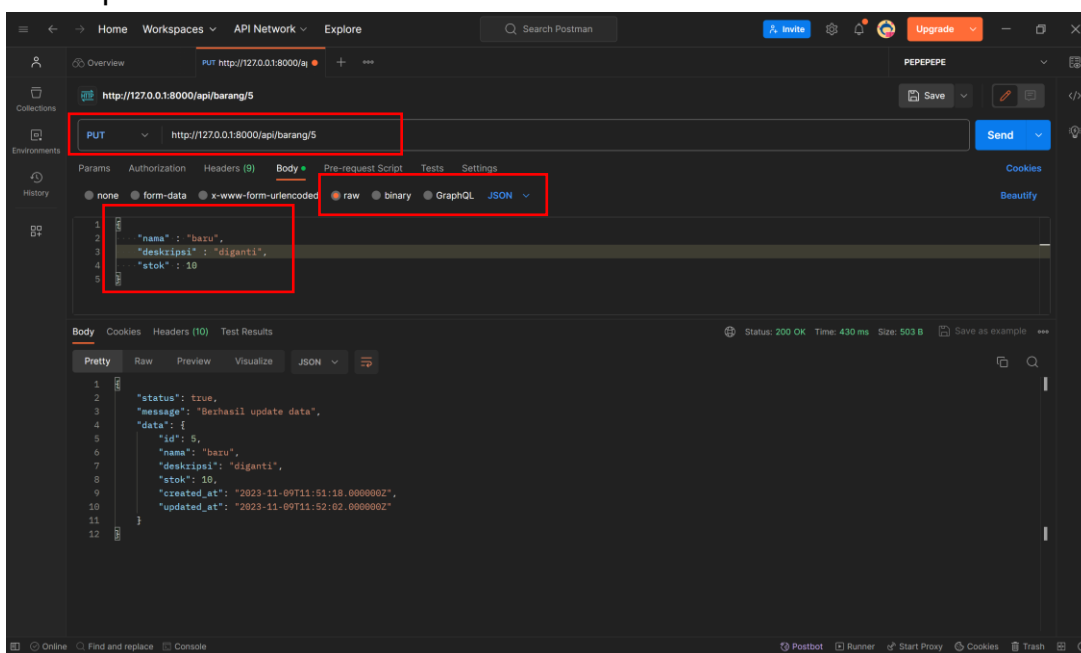


27. Test Search:



Gambar 27 Testing Search

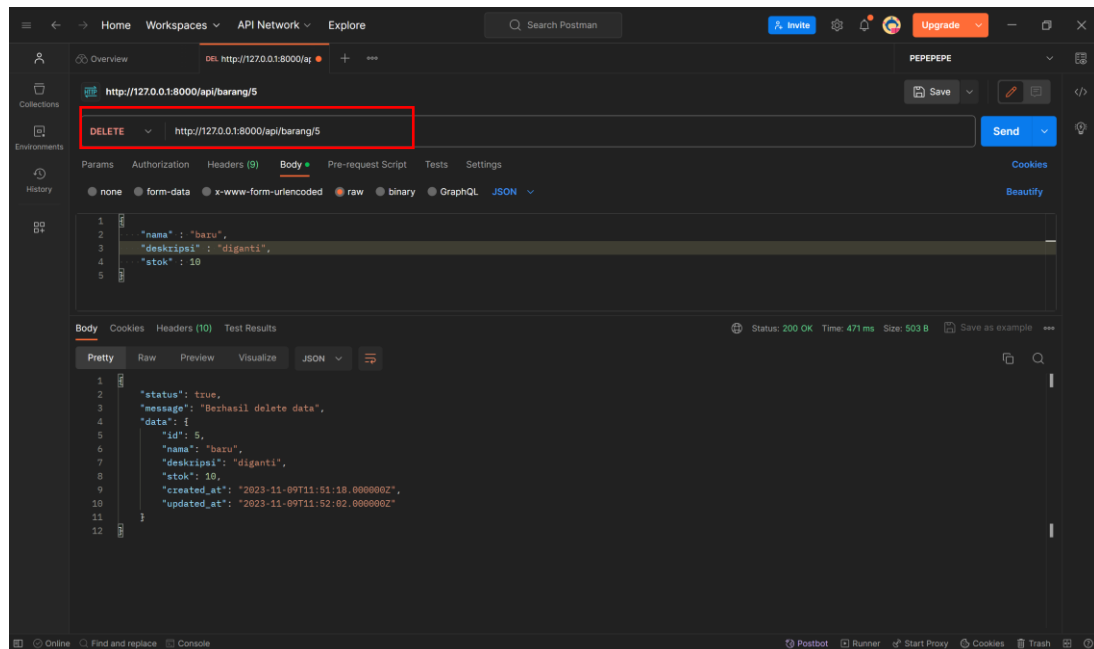
28. Test Update:



Gambar 28 Testing Update



29. Test Delete:



Gambar 29 Testing Delete

ATURAN Pengerjaan Guided :

- Guided dikerjakan selama waktu perkuliahan berlangsung.
- Penamaan proyek guided harus sesuai dengan yang sudah dicontohkan.
- Guided dikumpulkan melalui github dengan penamaan setiap file pada github adalah : **NAMAGUIDED_XXXX** (contoh : **Guided1_Intent_9999**)
 - o **NAMAGUIDED** → SESUAI DENGAN CONTOH DALAM MODUL INI
 - o **XXXX** → 4 DIGIT TERAKHIR NPM
- Setelah diupload melalui github, jangan lupa untuk mengumpulkan keseluruhan link file github melalui situs kuliah.
- Cara upload ke github, silahkan melihat pada modul **"UPLOAD GITHUB"**.