

**Laporan Praktikum Pemrograman Web Lanjut**

**Jobsheet 10: Restful API**



Oleh :

Danica Nasywa Putrinier (2341760122)

Kelas SIB 2B / 05

**PROGRAM STUDI D-IV SISTEM INFORMASI BISNIS JURUSAN**

**TEKNOLOGI INFORMASI**

**POLITEKNIK NEGERI MALANG**

Jl. Soekarno Hatta No.9, Jatimulyo, Kec. Lowokwaru, Kota Malang, Jawa Timur

65141

## Praktikum 1 - Membuat RESTful API register

1. Sebelum memulai membuat REST API, terlebih dahulu download aplikasi Postman di <https://www.postman.com/downloads>

Aplikasi ini akan digunakan untuk mengerjakan semua tahap praktikum pada Jobsheet ini.

2. Lakukan instalasi JWT dengan mengetikkan perintah berikut:

```
PS E:\Aplik\laragon\www\Pemrograman_Web_Lanjut_2025\Minggu 9> composer require tymon/jwt-auth:2.1.1
./composer.json has been updated
Running composer update tymon/jwt-auth
Loading composer repositories with package information
Updating dependencies
Lock file operations: 4 installs, 0 updates, 0 removals
- Locking lcobucci/clock (2.3.0)
- Locking lcobucci/jwt (4.0.4)
- Locking stella-maris/clock (0.1.7)
- Locking tymon/jwt-auth (2.1.1)
Writing lock file
Installing dependencies from lock file (including require-dev)
Package operations: 4 installs, 1 update, 0 removals
- Downloading stella-maris/clock (0.1.7)
- Downloading lcobucci/clock (2.3.0)
- Downloading lcobucci/jwt (4.0.4)
- Downloading tymon/jwt-auth (2.1.1)
```

`composer require tymon/jwt-auth:2.1.1`

Pastikan Anda terkoneksi dengan internet.

3. Setelah berhasil menginstall JWT, lanjutkan dengan publish konfigurasi file dengan perintah berikut:

```
NO security vulnerability advisories found.
PS E:\Aplik\laragon\www\Pemrograman_Web_Lanjut_2025\Minggu 9> php artisan vendor:publish --provider=
Tymon\JWTAuth\Providers\LaravelServiceProvider

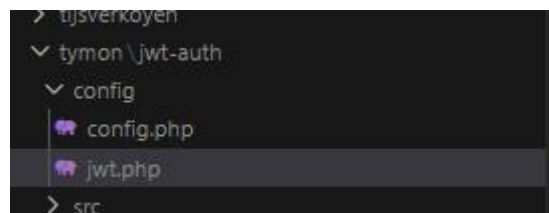
INFO Publishing assets.

Copying file [E:\Aplik\laragon\www\Pemrograman_Web_Lanjut_2025\Minggu 9\vendor\tymon\jwt-auth\conf
g\config.php] to [E:\Aplik\laragon\www\Pemrograman_Web_Lanjut_2025\Minggu 9\config\jwt.php] DONE
```

`php artisan vendor:publish --`

`provider="Tymon\JWTAuth\Providers\LaravelServiceProvider"`

4. Jika perintah di atas berhasil, maka kita akan mendapatkan 1 file baru yaitu config/jwt.php. Pada file ini dapat dilakukan konfigurasi jika memang diperlukan.



5. Setelah itu jalankan perintah berikut untuk membuat secret key JWT.

`php artisan jwt:secret`

Jika berhasil, maka pada file .env akan ditambahkan sebuah baris berisi nilai key JWT\_SECRET.

```
PS E:\Aplik\laragon\www\Pemrograman_Web_Lanjut_2025\Minggu 9> php artisan jwt:secret
jwt-auth secret [LpAgFCfOvRwLGtyJTADZV8a5ihrt2GE5YnaBJr5ZPrqiddAcpszqGXhtybOwmh1RR] set successfully.
PS E:\Aplik\laragon\www\Pemrograman_Web_Lanjut_2025\Minggu 9>
```

6. Selanjutnya lakukan konfigurasi guard API. Buka config/auth.php. Ubah bagian 'guards' menjadi seperti berikut.

```
'guards' => [
    'web' => [
        'driver' => 'session',
        'provider' => 'users',
    ],
    'api' => [
        'driver' => 'jwt',
        'provider' => 'users',
    ],
],
```

7. Kita akan menambahkan kode di model UserModel, ubah kode seperti berikut:

```
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Model;
use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Relations\BelongsTo;
use Illuminate\Foundation\Auth\User as Authenticatable; // implementasi class
use Tymon\JWTAuth\Contracts\JWTSubject;

class UserModel extends Authenticatable implements JWTSubject
{
    use HasFactory;

    public function getJWTIdentifier(){
        return $this->getKey();
    }

    public function getJWTCustomClaims(){
        return [];
    }

    protected $table = 'm_user'; //mendefinisikan nama tabel yang digunakan model
    protected $primaryKey = 'user_id'; //mendefinisikan primary key dari tabel
```

8. Berikutnya kita akan membuat controller untuk register dengan menjalankan perintah berikut.

```
PS E:\Aplik\laragon\www\Pemrograman_Web_Lanjut_2025\Minggu 9> php artisan make:controller Api/RegisterController

INFO Controller [E:\Aplik\laragon\www\Pemrograman_Web_Lanjut_2025\Minggu 9\app\Http\Controllers\Api\RegisterController.php] created successfully.
```

php artisan make:controller Api/RegisterController

Jika berhasil maka akan ada tambahan controller pada folder Api dengan nama

RegisterController.

9. Buka file tersebut, dan ubah kode menjadi seperti berikut

```
class RegisterController extends Controller
{
    public function __invoke(Request $request){
        //set validation
        $validator = Validator::make($request->all(), [
            'username' => 'required',
            'nama' => 'required',
            'password' => 'required|min:5|confirmed',
            'level_id' => 'required'
        ]);

        //if validations fails
        if ($validator->fails()){
            return response()->json($validator->errors(), 422);
        }

        //create user
        $user = UserModel::create([
            'username' => $request->username,
            'nama' => $request->nama,
            'password' => bcrypt($request->password),
            'level_id' => $request->level_id,
        ]);

        //return response JSON user is created
        if ($user){
            return response()->json([
                'success' => true,
                'user' => $user,
            ], 201);
        }

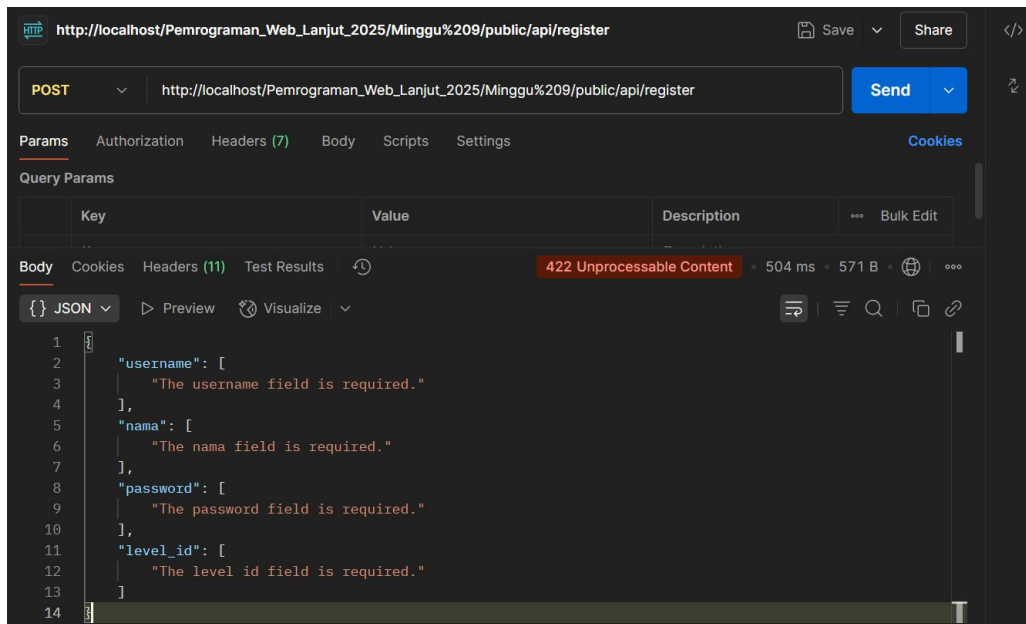
        //return JSON process insert failed
        return response()->json([
            'success' => false,
        ], 409);
    }
}
```

10. Selanjutnya buka routes/api.php, ubah semua kode menjadi seperti berikut.

```
*/
Route::post('/register', App\Http\Controllers\Api\RegisterController::class)->name('register');
```

11. Jika sudah, kita akan melakukan uji coba REST API melalui aplikasi Postman.

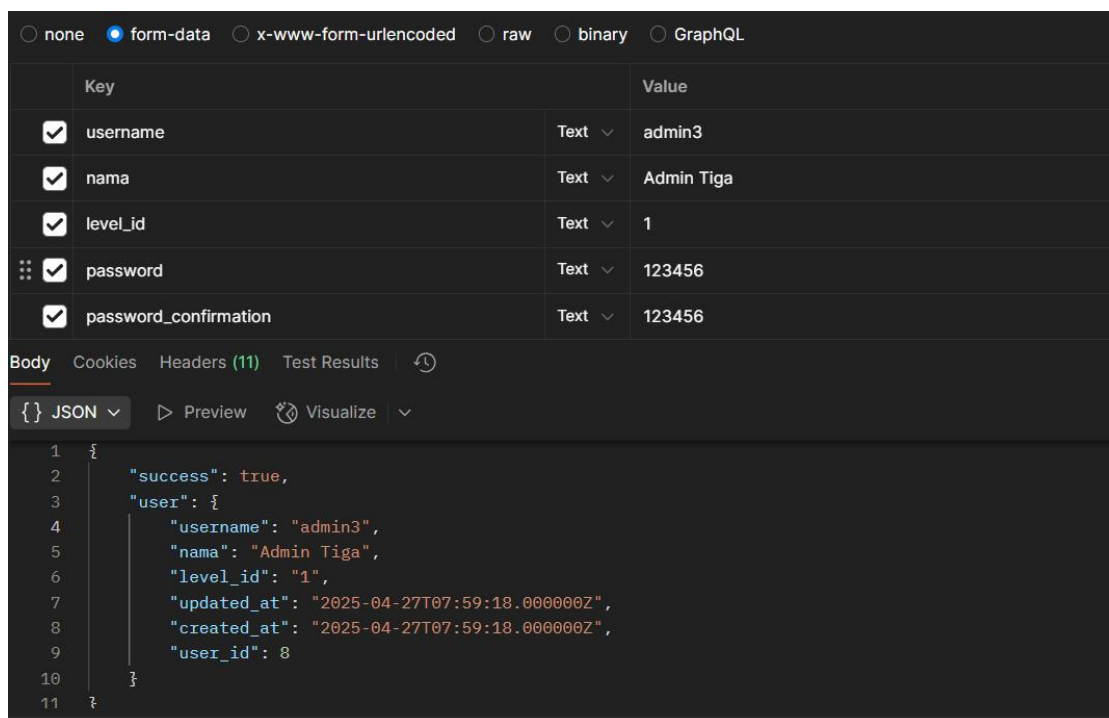
Buka aplikasi Postman, isi URL localhost/PWL\_POS/public/api/register serta method POST. Klik Send.



Jika berhasil akan muncul error validasi seperti gambar di atas.

Lakukan percobaan yang sama dan berikan screenshot hasil percobaan Anda.

12. Sekarang kita coba masukkan data. Klik tab Body dan pilih form-data. Isikan key sesuai dengan kolom data, serta isikan data registrasi menggunakan nilai yang Anda inginkan.



Setelah klik tombol Send, jika berhasil maka akan keluar pesan sukses seperti gambar di atas.

Lakukan percobaan yang sama dan berikan screenshot hasil percobaan Anda.

13. Lakukan commit perubahan file pada Github.



## Praktikum 2 - Membuat RESTful API Login

### 1. Kita buat file controller dengan nama LoginController

```
PS E:\Aplik\laragon\www\Pemrograman_Web_Lanjut_2025\Minggu 9> php artisan make:controller Api/LoginController

INFO Controller [E:\Aplik\laragon\www\Pemrograman_Web_Lanjut_2025\Minggu 9\app\Http\Controllers\Api>LoginController.php] created successfully.
```

`php artisan make:controller Api/LoginController`

Jika berhasil maka akan ada tambahan controller pada folder

### 2. Buka file tersebut, dan ubah kode menjadi seperti berikut

```
use App\Http\Controllers\Controller;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Validator;

class LoginController extends Controller
{
    public function __invoke(Request $request)
    {
        //set validation
        $validator = Validator::make($request->all(), [
            'username' => 'required',
            'password' => 'required'
        ]);

        //if validation fails
        if ($validator->fails()) {
            return response()->json($validator->errors(), 422);
        }

        //get credentials from request
        $credentials = $request->only('username', 'password');

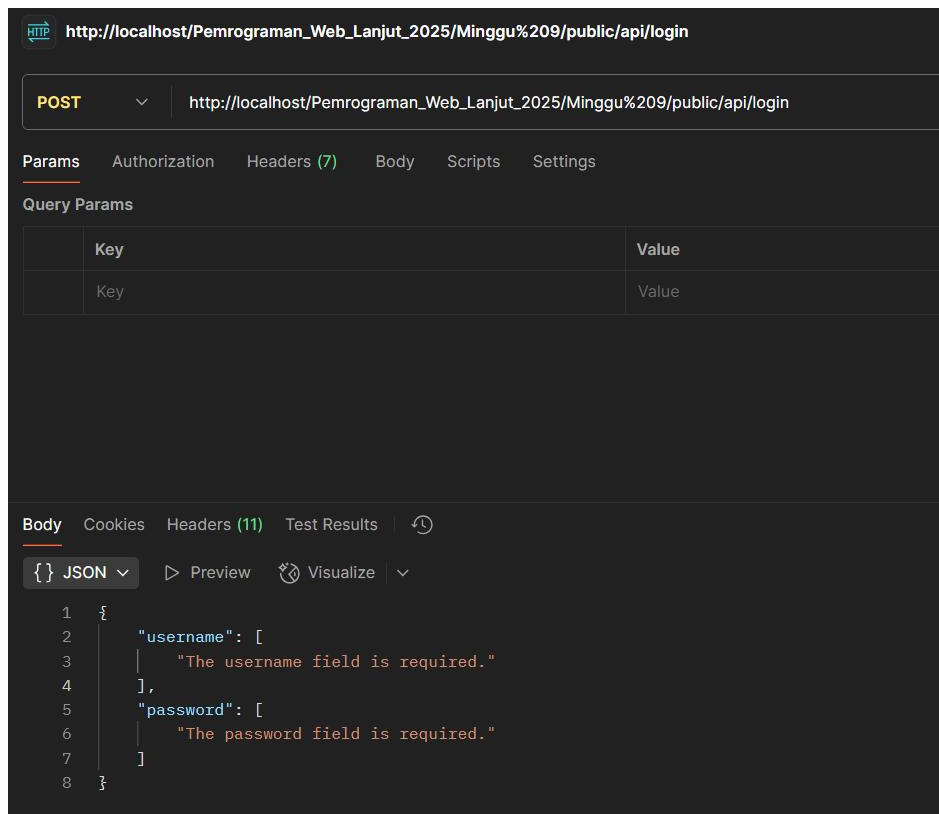
        //if auth failed
        if (!$token = auth()->guard('api')->attempt($credentials)) {
            return response()->json([
                'success' => false,
                'message' => 'Username atau Password Anda salah',
            ], 401);
        }

        //if auth success
        return response()->json([
            'success' => true,
            'user' => auth()->guard('api')->user(),
            'token' => $token,
        ], 200);
    }
}
```

### 3. Berikutnya tambahkan route baru pada file api.php yaitu /login dan /user

```
Route::post('/register', App\Http\Controllers\Api\RegisterController::class)->name('register');
Route::post('/login', App\Http\Controllers\Api>LoginController::class)->name('login');
Route::middleware('auth:api')->get('/user', function (Request $request){
    return $request->user();
});
```

4. Jika sudah, kita akan melakukan uji coba REST API melalui aplikasi Postman. Buka aplikasi Postman, isi URL localhost/PWL\_POS/public/api/login serta method POST. Klik Send.



Jika berhasil akan muncul error validasi seperti gambar di atas.

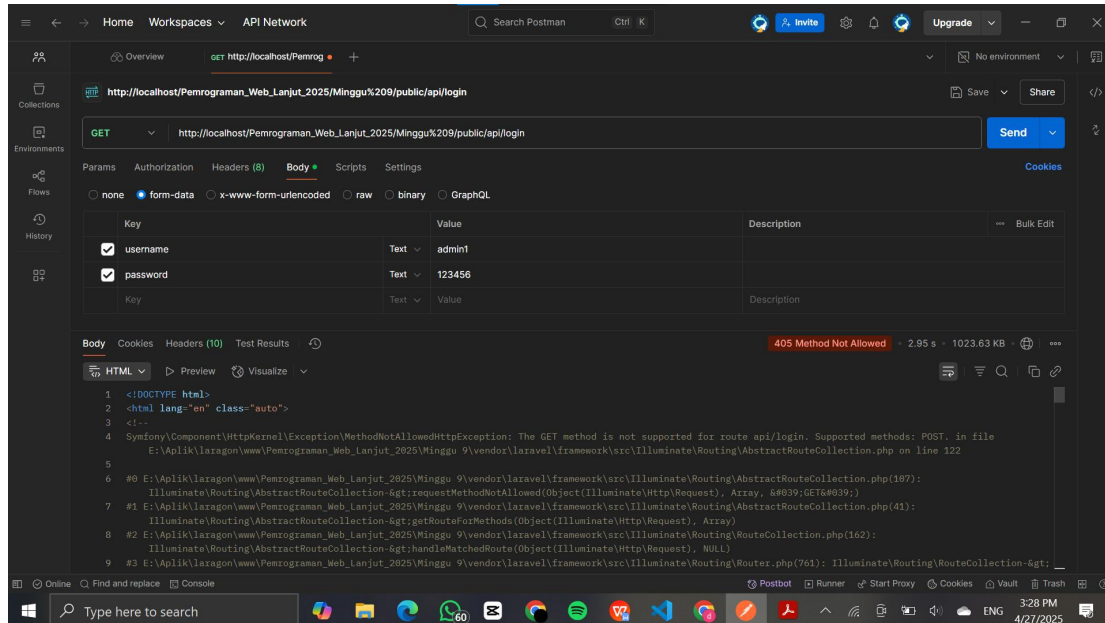
**Lakukan percobaan yang sama dan berikan screenshoot hasil percobaan Anda.**

5. Selanjutnya, isikan username dan password sesuai dengan data user yang ada pada database. Klik tab Body dan pilih form-data. Isikan key sesuai dengan kolom data, serta isikan data user. Klik tombol Send, jika berhasil maka akan keluar tampilan seperti berikut. Copy nilai token yang diperoleh pada saat login karena akan diperlukan pada saat logout.





7. Coba kembali melakukan login dengan data yang benar. Sekarang mari kita coba menampilkan data user yang sedang login menggunakan URL `localhost/PWL_POS/public/api/user` dan method GET. Jelaskan hasil dari percobaan tersebut.



Error karena HTTP GET tidak didukung untuk rute ini, dan route api.php menggunakan POST bukan GET

8. Lakukan commit perubahan file pada Github.

### Praktikum 3 – Membuat RESTful API Logout

1. Tambahkan kode berikut pada file .env

```
JWT_SECRET=LpAgFCf0vRwLGtyJTADZV8a5ihrt2GE5YnaBJr5ZPrqiddAcpzqGXhtybOwmh1RR
JWT_SHOW_BLACKLIST_EXCEPTION=true
```

JWT\_SHOW\_BLACKLIST\_EXCEPTION=true

2. Buat Controller baru dengan nama LogoutController.

```
PS E:\Aplik\laragon\www\Pemrograman_Web_Lanjut_2025\Minggu 9> php artisan make:controller Api/LogoutController

INFO Controller [E:\Aplik\laragon\www\Pemrograman_Web_Lanjut_2025\Minggu 9\app\Http\Controllers\Api\LogoutController.php] created successfully.
```

php artisan make:controller Api/LogoutController

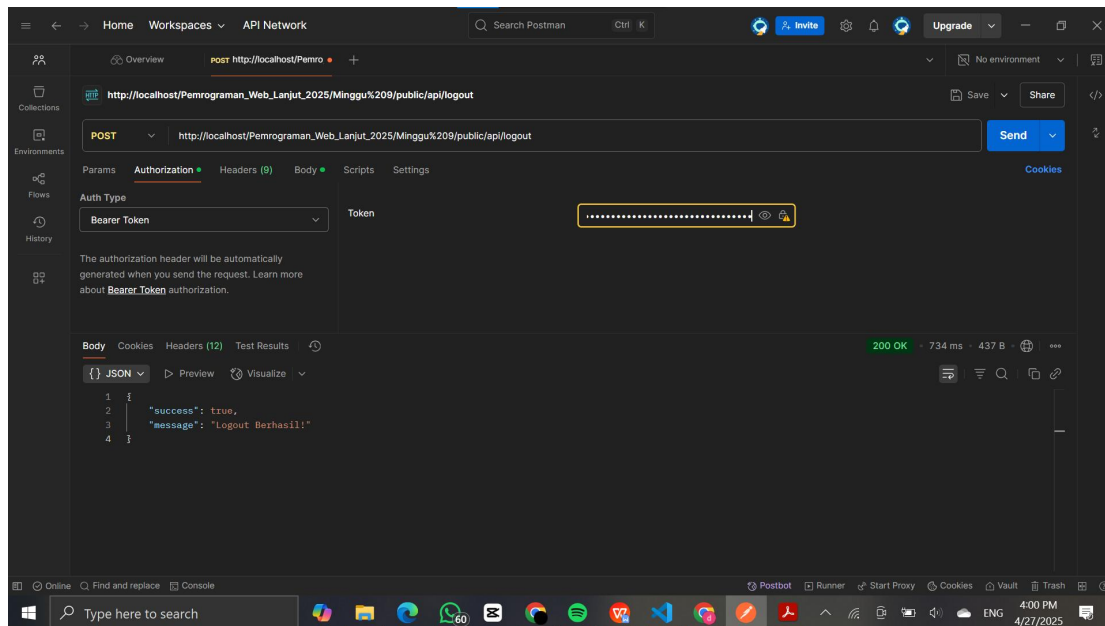
3. Buka file tersebut dan ubah kode menjadi seperti berikut.

```
1 <?php
2
3 namespace App\Http\Controllers\Api;
4
5 use Illuminate\Http\Request;
6 use App\Http\Controllers\Controller;
7 use Tymon\JWTAuth\Facades\JWTAuth;
8 use Tymon\JWTAuth\Exceptions\JWTException;
9 use Tymon\JWTAuth\Exceptions\TokenExpiredException;
10 use Tymon\JWTAuth\Exceptions\TokenInvalidException;
11
12 class LogoutController extends Controller
13 {
14     public function __invoke(Request $request)
15     {
16         //remove token
17         $removeToken = JWTAuth::invalidate(JWTAuth::getToken());
18
19         if ($removeToken) {
20             //return response JSON
21             return response()->json([
22                 'success' => true,
23                 'message' => 'Logout Berhasil!',
24             ]);
25         }
26     }
27 }
```

4. Lalu kita tambahkan routes pada api.php

```
});
Route::post('/logout', App\Http\Controllers\Api\LogoutController::class)->name('logout');
```

5. Jika sudah, kita akan melakukan uji coba REST API melalui aplikasi Postman. Buka aplikasi Postman, isi URL localhost/PWL\_POS/public/api/logout serta method POST.
6. Isi token pada tab Authorization, pilih Type yaitu Bearer Token. Isikan token yang didapat saat login. Jika sudah klik Send.



7. Lakukan commit perubahan file pada Github.

## Praktikum 4 – Implementasi CRUD dalam RESTful API

Pada praktikum ini kita akan menggunakan tabel `m_level` untuk dimodifikasi menggunakan RESTful API.

1. Pertama, buat controller untuk mengolah API pada data level.

```
78309da..6b80f00 main -> main
PS E:\Aplik\laragon\www\Pemrograman_Web_Lanjut_2025\Minggu 9> php artisan make:controller Api/LevelController

INFO Controller [E:\Aplik\laragon\www\Pemrograman_Web_Lanjut_2025\Minggu 9\app\Http\Controllers\Api\LevelController.php] creat
```

php artisan make:controller Api/LevelController

2. Setelah berhasil, buka file tersebut dan tuliskan kode seperti berikut yang berisi fungsi CRUDnya.

```
1  <?php
2
3  namespace App\Http\Controllers\Api;
4
5  use App\Http\Controllers\Controller;
6  use Illuminate\Http\Request;
7  use App\Models\LevelModel;
8
9  class LevelController extends Controller
10 {
11     public function store(Request $request)
12     {
13         $level = LevelModel::create($request->all());
14         return response()->json($level, 201);
15     }
16
17     public function show(LevelModel $level)
18     {
19         return LevelModel::find($level);
20     }
21
22     public function update(Request $request, LevelModel $level)
23     {
24         $level->update($request->all());
25         return LevelModel::find($level);
26     }
27
28     public function destroy(LevelModel $user)
29     {
30         $user->delete();
31         return response()->json([
32             'success' => true,
33             'message' => 'Data terhapus',
34         ]);
35     }
36 }
37
```

3. Kemudian kita lengkapi routes pada api.php.

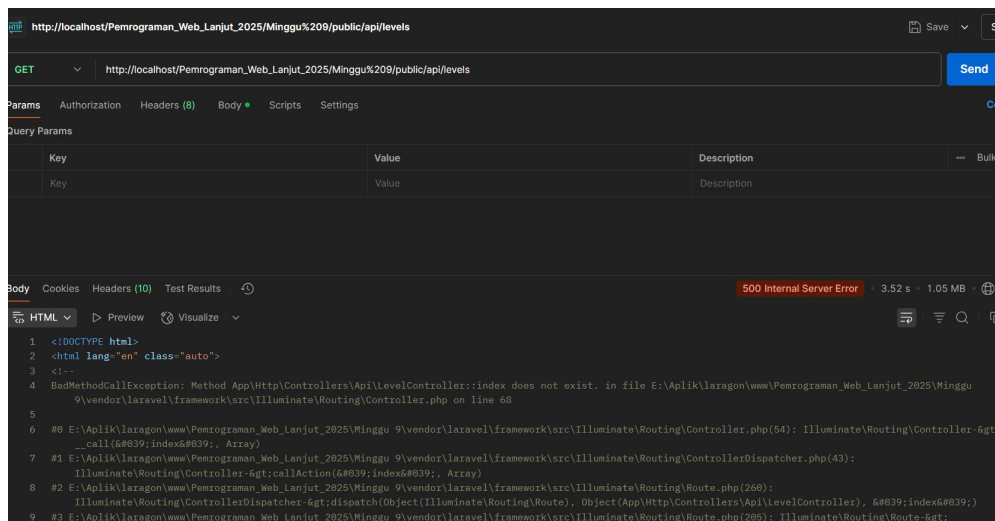
```

use App\Http\Controllers\Api\LevelController;

Route::get('levels', [LevelController::class, 'index']);
Route::post('levels', [LevelController::class, 'store']);
Route::get('levels/{level}', [LevelController::class, 'show']);
Route::put('levels/{level}', [LevelController::class, 'update']);
Route::delete('levels/{level}', [LevelController::class, 'destroy']);

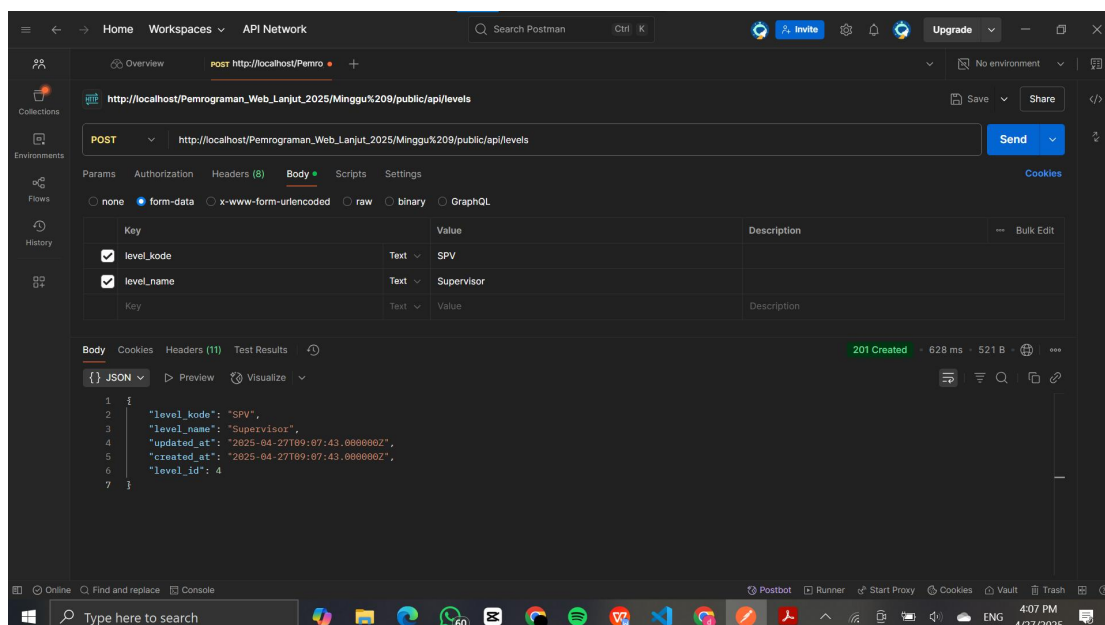
```

4. Jika sudah. Lakukan uji coba API mulai dari fungsi untuk menampilkan data. Gunakan URL: localhost/PWL\_POS-main/public/api/levels dan method GET. Jelaskan dan berikan screenshoot hasil percobaan Anda.



Saat mencoba mengakses rute /api/levels dengan metode GET, seharusnya mengarah ke method index di LevelController. Tetapi, method index tidak ditemukan di dalam class App\Http\Controllers\Api\LevelController.

5. Kemudian, lakukan percobaan penambahan data dengan URL : localhost/PWL\_POS-main/public/api/levels dan method POST seperti di bawah ini.





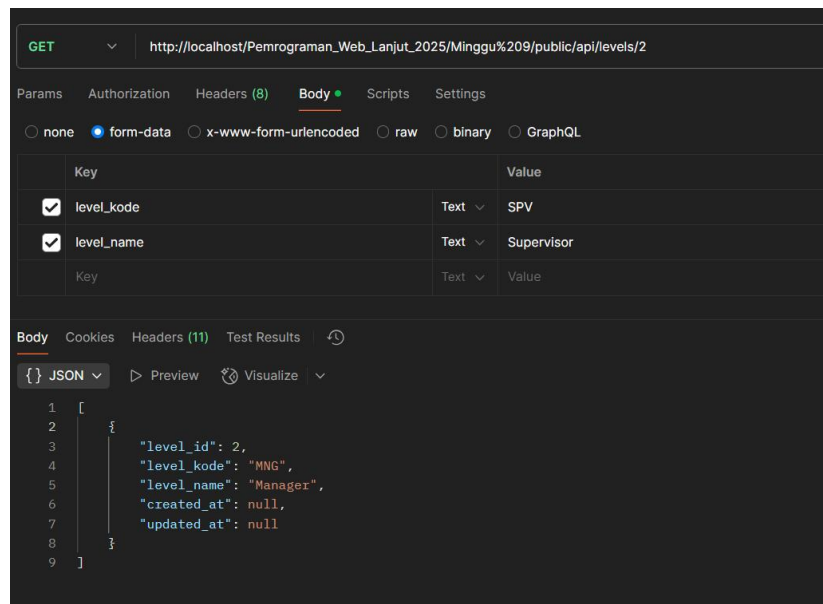
- Body Permintaan: Di bagian "Body" dengan format x-www-form-urlencoded, Anda mengirimkan dua buah key-value pair:

- level\_kode: SPV
- level\_nama: Supervisor

Ini adalah data yang Anda kirimkan ke server untuk membuat level baru.

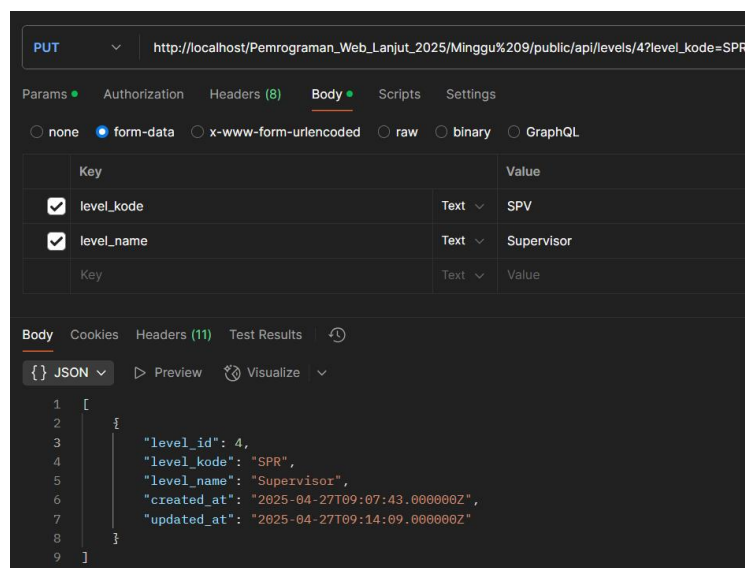
- Status Code 201 Created: Kode status 201 Created menunjukkan bahwa permintaan POST Anda berhasil diproses dan sumber daya baru telah berhasil dibuat di server. Ini adalah respons yang diharapkan setelah operasi store yang sukses.

6. Berikutnya lakukan percobaan menampilkan detail data. Jelaskan dan berikan screenshoot hasil percobaan Anda.



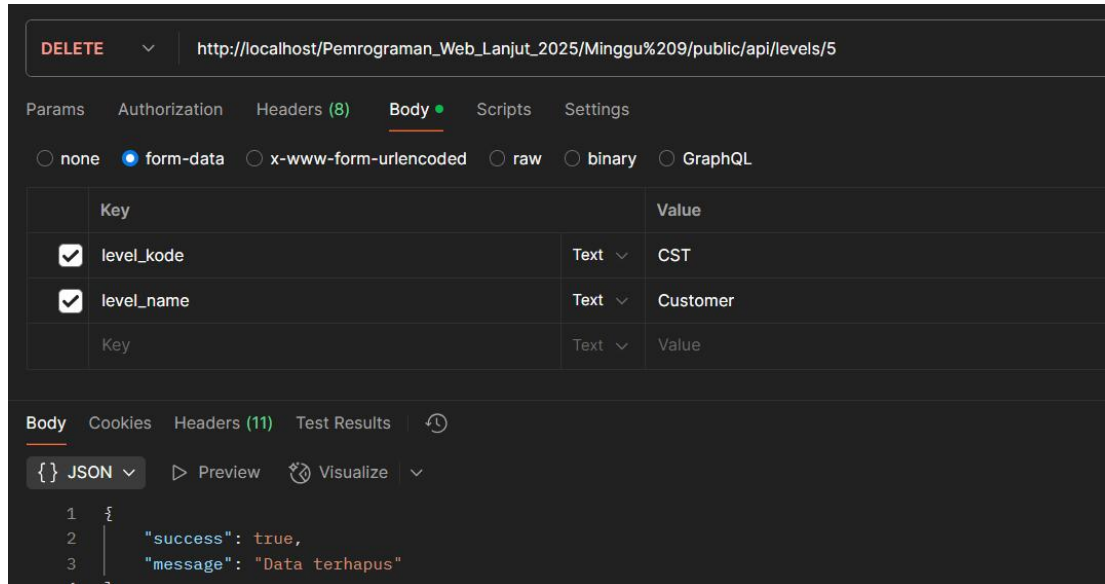
Di inputkan level berapa yang dicari maka akan keluar hasil pada body.

7. Jika sudah, kita coba untuk melakukan edit data menggunakan localhost/PWL\_POS-main/public/api/levels/{id} dan method PUT. Isikan data yang ingin diubah pada tab Param.



Rute dengan metode PUT di URL `/api/levels/{level}` akan diarahkan ke method update pada LevelController. Variabel `{level}` pada URL akan mengidentifikasi data level mana yang ingin diubah. Data perubahan akan dikirimkan melalui body permintaan.

8. Terakhir lakukan percobaan hapus data. Jelaskan dan berikan screenshoot hasil percobaan Anda.



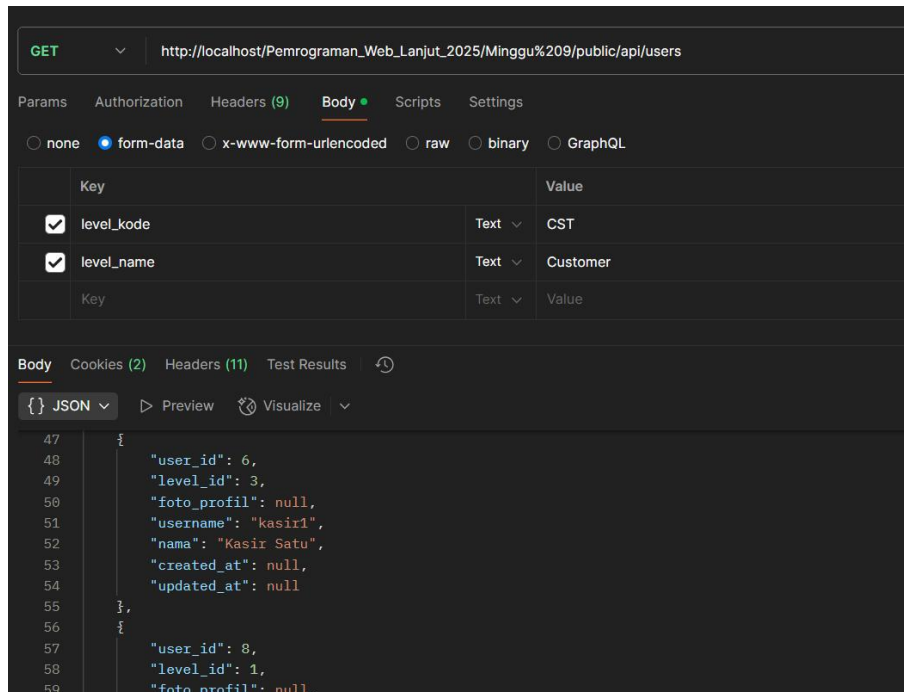
Data ke-5 dapat terhapus karena menggunakan DELETE dan pada route `api.php` sudah ada.

9. Lakukan commit perubahan file pada Github.

## TUGAS

Implementasikan CRUD API pada tabel lainnya yaitu tabel m\_user, m\_kategori, dan m\_barang

Users :



GET `http://localhost/Pemrograman_Web_Lanjut_2025/Minggu%209/public/api/users`

Params Authorization Headers (9) **Body** Scripts Settings

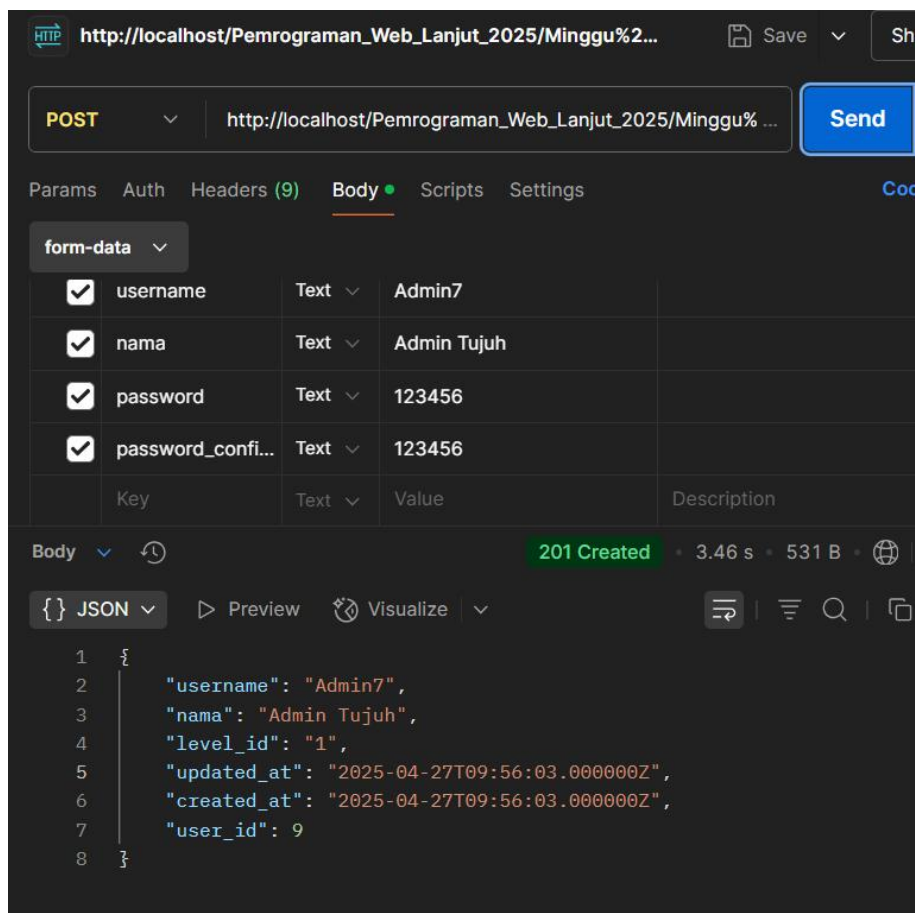
☐ none ☒ form-data ☐ x-www-form-urlencoded ☐ raw ☐ binary ☐ GraphQL

Key	Value
<input checked="" type="checkbox"/> level_kode	Text CST
<input checked="" type="checkbox"/> level_name	Text Customer
Key	Text Value

Body Cookies (2) Headers (11) Test Results

**JSON** Preview Visualize

```
47 {
48   "user_id": 6,
49   "level_id": 3,
50   "foto_profil": null,
51   "username": "kasir1",
52   "nama": "Kasir Satu",
53   "created_at": null,
54   "updated_at": null
55 },
56 {
57   "user_id": 8,
58   "level_id": 1,
59   "foto_profil": null,
```



HTTP `http://localhost/Pemrograman_Web_Lanjut_2025/Minggu%209/public/api/users` Save Send

POST `http://localhost/Pemrograman_Web_Lanjut_2025/Minggu%209/public/api/users`

Params Auth Headers (9) **Body** Scripts Settings

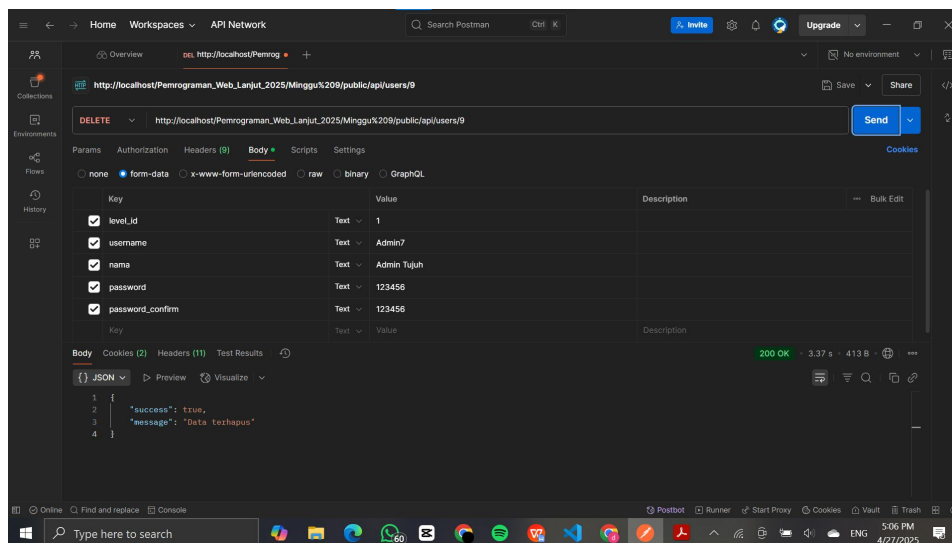
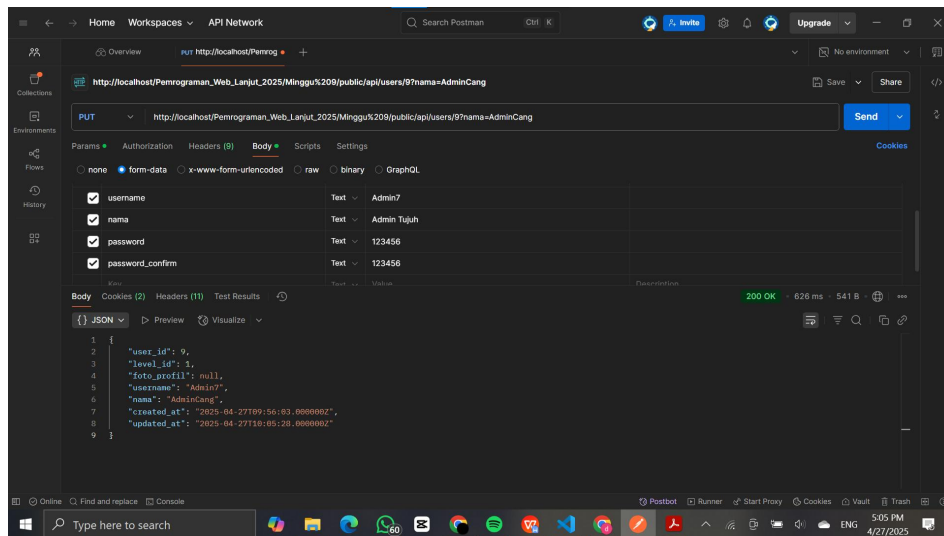
**form-data**

<input checked="" type="checkbox"/> username	Text	Admin7	
<input checked="" type="checkbox"/> nama	Text	Admin Tujuh	
<input checked="" type="checkbox"/> password	Text	123456	
<input checked="" type="checkbox"/> password_confirmation	Text	123456	
Key	Text	Value	Description

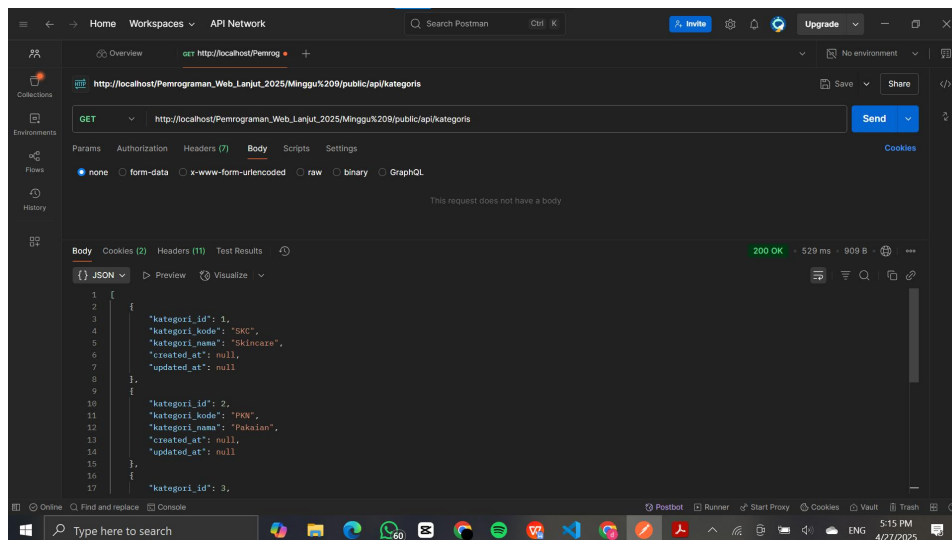
Body 201 Created • 3.46 s • 531 B •

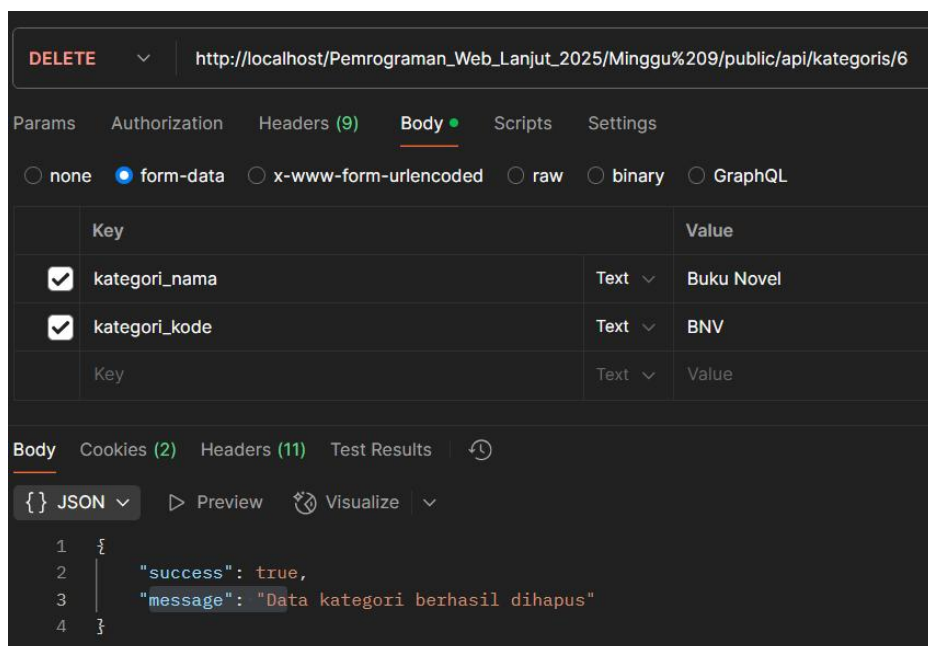
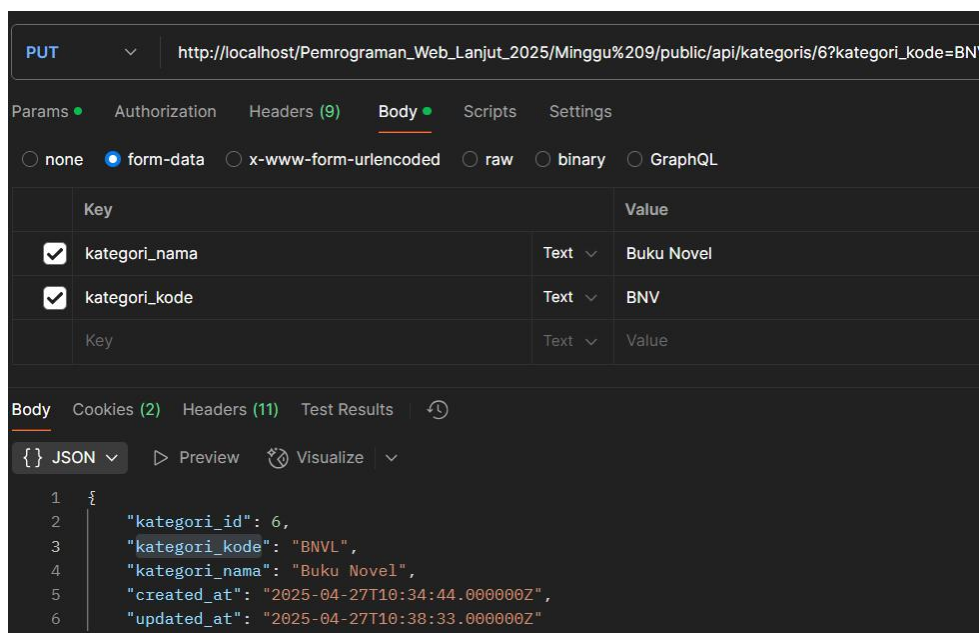
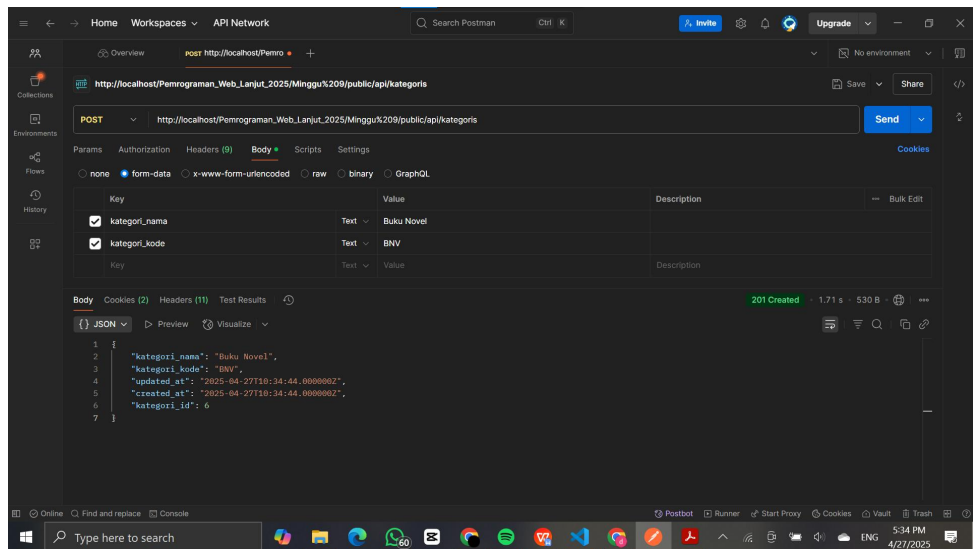
**JSON** Preview Visualize

```
1 {
2   "username": "Admin7",
3   "nama": "Admin Tujuh",
4   "level_id": "1",
5   "updated_at": "2025-04-27T09:56:03.000000Z",
6   "created_at": "2025-04-27T09:56:03.000000Z",
7   "user_id": 9
8 }
```



Kategori :







Barang:

Postman interface showing a GET request to `http://localhost/Pemrograman_Web_Lanjut_2025/Minggu%209/public/api/barangs`. The response is a 200 OK status with a JSON body containing an array of items. The first item is:

```
{  "barang_id": 1,  "kategori_id": 1,  "barang_kode": "BNG001",  "barang_nama": "Toner",  "harga_beli": 150000,  "harga_jual": 180000,  "created_at": null,  "updated_at": null}
```

Postman interface showing a POST request to `http://localhost/Pemrograman_Web_Lanjut_2025/Minggu%209/public/api/barangs`. The request is using form-data with the following fields:

Key	Value
<input checked="" type="checkbox"/> kategori_id	3
<input checked="" type="checkbox"/> harga_beli	125000
<input checked="" type="checkbox"/> harga_jual	150000
<input checked="" type="checkbox"/> stok	14

The response is a 201 Created status with a JSON body:

```
{  "barang_kode": "BNV",  "barang_nama": "Buku Novel",  "kategori_id": "3",  "harga_beli": "125000",  "harga_jual": "150000",  "updated_at": "2025-04-27T10:45:21.000000Z",  "created_at": "2025-04-27T10:45:21.000000Z",  "barang_id": 11}
```

PUT

http://localhost/Pemrograman\_Web\_Lanjut\_2025/Minggu%20...

Send

Params

Auth

Headers (9)

Body

Scripts

Settings

Cook

form-data

	Key		Value	Description	...	Bulk Edit
<input checked="" type="checkbox"/>	barang_kode	Text	BNV			
<input checked="" type="checkbox"/>	barang_nama	Text	Buku Novel			
<input checked="" type="checkbox"/>	kategori_id	Text	3			
<input checked="" type="checkbox"/>	harga_beli	Text	125000			

Body

200 OK

503 ms

576 B

{ } JSON

Preview

Visualize

```
1  {
2    "barang_id": 11,
3    "kategori_id": 3,
4    "barang_kode": "BKNV",
5    "barang_nama": "Buku Novel",
6    "harga_beli": 125000,
7    "harga_jual": 150000,
8    "created_at": "2025-04-27T10:45:21.000000Z",
9    "updated_at": "2025-04-27T10:46:45.000000Z"
10 }
```

DELETE

http://localhost/Pemrograman\_Web\_Lanjut\_2025/Minggu%20!

S

Params

Auth

Headers (9)

Body

Scripts

Settings

form-data

	Key		Value	Description	...
<input checked="" type="checkbox"/>	barang_kode	Text	BNV		
<input checked="" type="checkbox"/>	barang_nama	Text	Buku Novel		
<input checked="" type="checkbox"/>	kategori_id	Text	3		
<input checked="" type="checkbox"/>	harga_beli	Text	125000		

Body

200 OK

549 ms

428 B

{}

JSON

Preview

Visualize

1

{

2

"success": true,

3

"message": "Data barang berhasil dihapus"

4

}