



Mata Kuliah : Pemrograman Web Lanjut (PWL)
Program Studi : D4 – Teknik Informatika / D4 – Sistem Informasi Bisnis
Semester : 4 (empat) / 6 (enam)
Pertemuan ke- : 10 (tujuh)

Nama	: My Babby Findia R.S
No. Absen	: 16
Kelas	: SIB-2B

JOBSHEET 10

RESTFUL API

Sebelumnya kita sudah membahas mengenai *authentication*, *authorization*, dan *middleware* pada Laravel. Dimana kita telah membuat fungsi login, register, logout, serta pemilihan role dan penerapan session pada halaman web. Pada pertemuan kali ini, kita akan mempelajari penerapan RESTFUL API di dalam project Laravel.

Sebelum kita masuk materi, kita buat dulu project baru yang akan kita gunakan untuk membangun aplikasi sederhana dengan topik *Point of Sales (PoS)*, sesuai dengan **Studi Kasus PWL.pdf**. Jadi kita bikin project Laravel 10 dengan nama **PWL_POS**.

Project PWL_POS akan kita gunakan sampai pertemuan 12 nanti, sebagai project yang akan kita pelajari

A. RESTFUL API

Representational State Transfer (REST) adalah gaya arsitektur perangkat lunak yang mendefinisikan seperangkat prinsip untuk merancang jaringan aplikasi terdistribusi. RESTful API adalah aplikasi pemrograman antarmuka yang mengikuti prinsip-prinsip REST untuk mentransfer data antara klien dan server.

RESTful API adalah salah satu arsitektur dalam API (*Application Program Interface*) yang menggunakan request HTTP untuk mengakses data. Data diakses dengan menggunakan HTTP method GET, PUT, POST dan DELETE yang merujuk pada operasi pembacaan, pembaruan, pembuatan dan penghapusan pada resource. Selain HTTP method, dalam RESTful atau REST digunakan juga HTTP response untuk mendefinisikan respon data yang dikembalikan. Format respon yang umum digunakan berupa JSON (Javascript Object Notation).



B. JSON Web Token (JWT)

JWT adalah singkatan dari JSON Web Token. Ini adalah standar terbuka (RFC 7519) yang mendefinisikan format token yang kompak dan mandiri untuk mentransfer klaim antara dua pihak. JWT sering digunakan dalam autentikasi dan pertukaran informasi yang aman di lingkungan yang tidak terpercaya, seperti internet.

JWT terdiri dari tiga bagian yang dipisahkan oleh titik ("."): header, payload, dan signature. Setiap bagian ini terdiri dari data JSON yang dienkripsi menggunakan algoritma tertentu dan kemudian disatukan untuk membentuk token yang lengkap. Header berisi jenis token dan tipe algoritma yang digunakan untuk enkripsi. Payload berisi klaim atau informasi yang ingin disampaikan. Signature digunakan untuk memverifikasi bahwa token belum berubah dan datanya berasal dari sumber yang dipercaya.

JWT sering digunakan dalam sistem autentikasi dan otorisasi modern, seperti aplikasi web dan layanan web API, karena fleksibilitasnya dalam menyampaikan informasi terenkripsi secara ringkas.

Kita dapat menggunakan JWT untuk:

- **Authentication**
Ketika pengguna melakukan authentication dan mendapatkan token, maka setiap permintaan berikutnya akan menyertakan token tersebut, dan memungkinkan pengguna untuk mengakses route, service, dan resources yang diizinkan.
- **Pertukaran informasi**
JSON Web Token adalah cara yang baik untuk mengirimkan informasi antar pihak dengan aman. Dengan token yang sudah ditandatangani dengan algoritma RSA, maka kita bisa tahu siapa yang melakukan request tersebut.

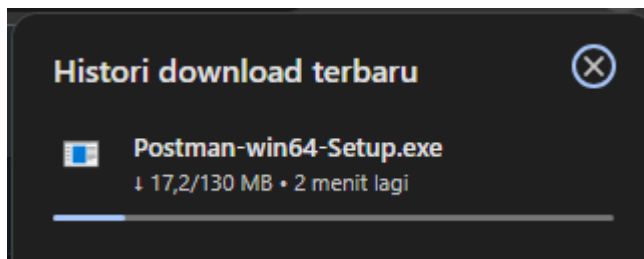
Berikut adalah cara kerja JWT :

JWT (JSON Web Token) adalah cara untuk mentransfer informasi antara dua pihak secara aman sebagai objek JSON. Ini terdiri dari tiga bagian: header, payload, dan signature. Setelah pengguna berhasil autentikasi, server menghasilkan token JWT yang disematkan dalam permintaan HTTP. Server kemudian memvalidasi token untuk memberikan akses ke sumber daya yang diminta. Ini memberikan autentikasi yang aman dan stateless tanpa memerlukan penyimpanan status sesi di server.

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: `composer require tymon/jwt-auth:2.1.1` Pastikan Anda terkoneksi dengan internet.

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
PS E:\laragon\www\PEMOGRAMAN-WEB-LANJUT-2025\Minggu10\PWL_POS> 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, 0 updates, 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)
1/4 [=====>-----] 25%
```

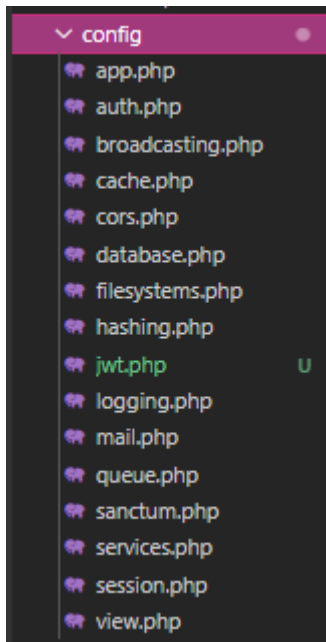
3. Setelah berhasil menginstall JWT, lanjutkan dengan publish konfigurasi file dengan perintah berikut:

```
php artisan vendor:publish --
provider="Tymon\JWTAuth\Providers\LaravelServiceProvider"
```

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
PS E:\laragon\www\PEMOGRAMAN-WEB-LANJUT-2025\Minggu10\PWL_POS> php artisan vendor:publish --provider="Tymon\JWTAuth\Providers\LaravelServiceProvider"
[INFO] Publishing assets.
Copying file [E:\laragon\www\PEMOGRAMAN-WEB-LANJUT-2025\Minggu10\PWL_POS\vendor\tymon\jwt-auth\config\config.php] to [E:\laragon\www\PEMOGRAMAN-WEB-LANJUT-2025\Minggu10\PWL_POS\config\j
.php] DONE
PS E:\laragon\www\PEMOGRAMAN-WEB-LANJUT-2025\Minggu10\PWL_POS>
```



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:\laragon\www\PEMOGRAMAN-WEB-LANJUT-2025\Minggu10\PwL_POS> php artisan jwt:secret
jwt-auth secret [pv89Rn7fJP4TYVMPMA0Fm7qzICaowxKb7JuOwhV18jbhgkG1VGnHuDQAi2iVgK8v] set successfully.
```

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',
    ],
],
```



```
'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 Tymon\JWTAuth\Contracts\JWTSubject;  
use Illuminate\Foundation\Auth\User as Authenticatable;  
  
class UserModel extends Authenticatable implements JWTSubject  
{  
  
    public function getJWTIdentifier(){  
        return $this->getKey();  
    }  
  
    public function getJWTCustomClaims(){  
        return [];  
    }  
  
    protected $table = 'm_user';  
    protected $primaryKey = 'user_id';  
}
```



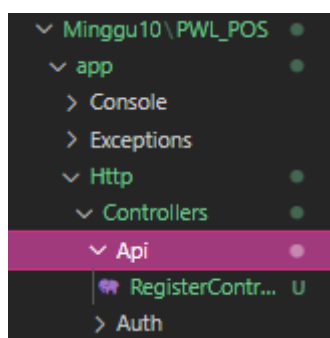
```
Minggu10 > PWL_POS > app > Models > UserModel.php > PHP Intelephense > UserModel > level
1  <?php
2
3  namespace App\Models;
4
5  use Illuminate\Database\Eloquent\Model;
6  use Illuminate\Database\Eloquent\Factories\HasFactory;
7  use Illuminate\Database\Eloquent\Relations\BelongsTo;
8  use Tymon\JWTAuth\Contracts\JWTSubject;
9  use Illuminate\Foundation\Auth\User as Authenticatable; // implementasi class Authenticatable
10
11
12  132 references | 0 implementations
13  class UserModel extends Authenticatable implements JWTSubject
14
15      0 references | 0 overrides
16      public function getJWTIdentifier(): mixed
17      {
18          return $this->getKey();
19      }
20
21      0 references | 0 overrides
22      public function getJWTCustomClaims(): array
23      {
24          return [];
25      }
26
27      0 references
28      protected $table = 'm_user';
29
30      0 references
31      protected $primaryKey = 'user_id';
32
33      0 references
34      protected $fillable = ['username', 'password', 'nama', 'level_id', 'user_profile_picture'];
35
36      0 references
37      protected $hidden = ['password']; // jangan di tampilkan saat select
38
39      0 references
40      protected $casts = ['password' => 'hashed']; // casting password agar otomatis di hash
```

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

```
php artisan make:controller Api/RegisterController
```

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

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
PS E:\laragon\www\PEMOGRAMAN-WEB-LAYOUT-2025\Minggu10\PWL_POS> php artisan make:controller Api/RegisterController
INFO Controller [E:\laragon\www\PEMOGRAMAN-WEB-LAYOUT-2025\Minggu10\PWL_POS\app\Http\Controllers\Api\RegisterController.php] created successfully.
PS E:\laragon\www\PEMOGRAMAN-WEB-LAYOUT-2025\Minggu10\PWL_POS>
```





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

```
1  <?php
2
3  namespace App\Http\Controllers\Api;
4
5  use App\Models\UserModel;
6  use App\Http\Controllers\Controller;
7  use Illuminate\Http\Request;
8  use Illuminate\Support\Facades\Validator;
9
10 class RegisterController extends Controller
11 {
12     public function __invoke(Request $request)
13     {
```




```
14 //set validation
15 $validator = Validator::make($request->all(), [
16     'username' => 'required',
17     'nama' => 'required',
18     'password' => 'required|min:5|confirmed',
19     'level_id' => 'required'
20 ]);
21
22 //if validations fails
23 if($validator->fails()){
24     return response()->json($validator->errors(), 422);
25 }
26
27 //create user
28 $user = UserModel::create([
29     'username' => $request->username,
30     'nama' => $request->nama,
31     'password' => bcrypt($request->password),
32     'level_id' => $request->level_id,
33 ]);
34
35 //return response JSON user is created
36 if($user){
37     return response()->json([
38         'success' => true,
39         'user' => $user,
40     ], 201);
41 }
42
43 //return JSON process insert failed
44 return response()->json([
45     'success' => false,
46 ], 409);
47 }
48 }
```




```
Minggu10 > PWL_POS > app > Http > Controllers > Api > RegisterController.php > PHP > RegisterController > __invoke()
1  <?php
2
3  namespace App\Http\Controllers\Api;
4
5  use App\Models\UserModel;
6  use App\Http\Controllers\Controller;
7  use Illuminate\Http\Request;
8  use Illuminate\Support\Facades\Validator;
9
10 0 references | 0 implementations
11 class RegisterController extends Controller
12 {
13     0 references | 0 overrides
14     public function __invoke(Request $request): JsonResponse|mixed
15     {
16         //set validation
17         $validator = Validator::make($request->all(), [
18             'username' => 'required',
19             'nama' => 'required',
20             'password' => 'required|min:5|confirmed',
21             'level_id' => 'required'
22         ]);
23
24         //if validations fails
25         if ($validator->fails()) {
26             return response()->json($validator->errors(), 422);
27         }
28
29         //create user
30         $user = UserModel::create([
31             'username' => $request->username,
32             'nama' => $request->nama,
33             'password' => bcrypt(value: $request->password),
34             'level_id' => $request->level_id,
35         ]);
36
37         //return response JSON user is created
38         if ($user) {
39             return response()->json([
40                 'success' => true,
```

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



```
<?php

use App\Http\Controllers\Api\RegisterController;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Route;

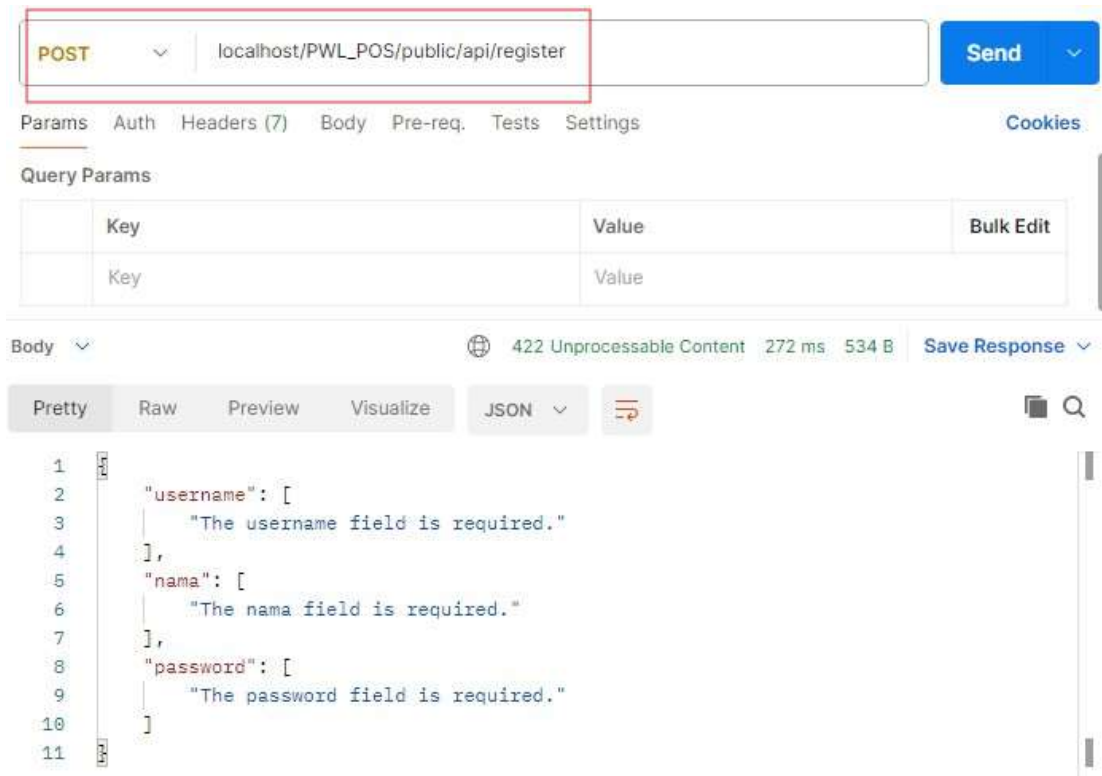
/*
|-----
| API Routes
|-----
|
| Here is where you can register API routes for your application. These
| routes are loaded by the RouteServiceProvider and all of them will
| be assigned to the "api" middleware group. Make something great!
|
*/

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

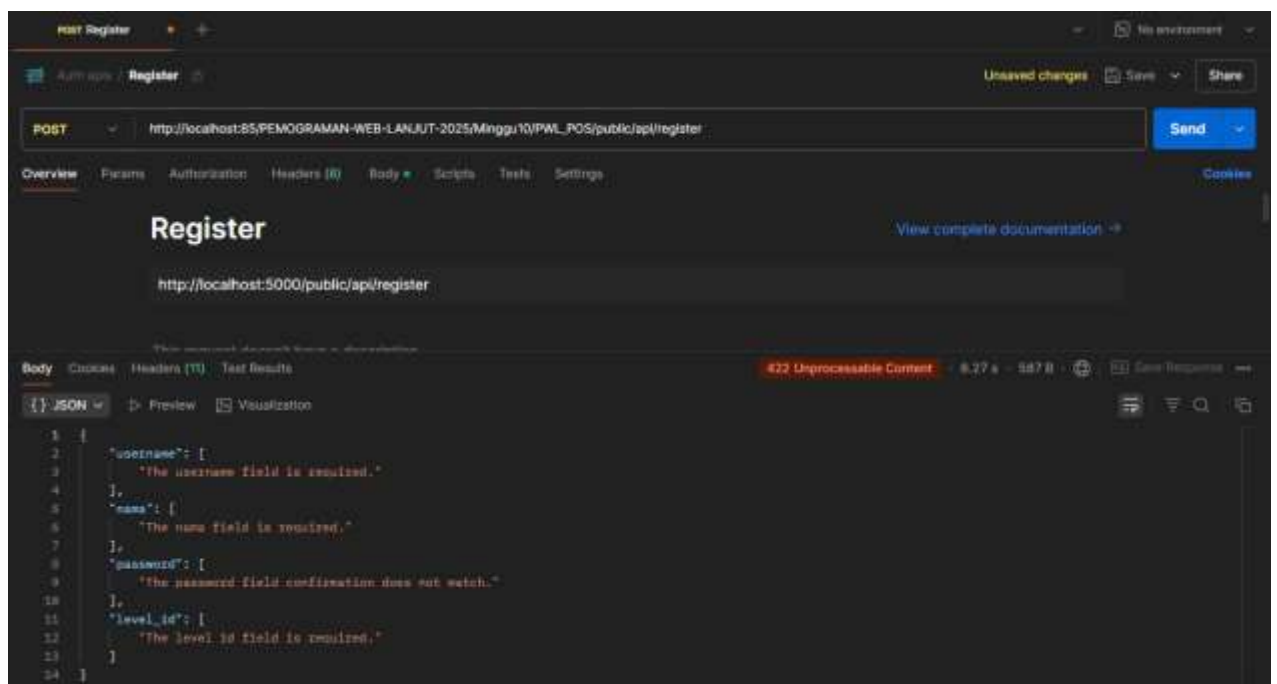
Minggu10 > PWL_POS > routes > api.php

```
1  <?php
2
3  use App\Http\Controllers\Api\RegisterController;
4  use Illuminate\Http\Request;
5  use Illuminate\Support\Facades\Route;
6
7  /*
8  |-----
9  | API Routes
10 |-----
11 |
12 | Here is where you can register API routes for your application. These
13 | routes are loaded by the RouteServiceProvider and all of them will
14 | be assigned to the "api" middleware group. Make something great!
15 |
16 */
17
18 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 screenshoot 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.

The screenshot shows the Postman interface for a POST request to `localhost/PWL_POS/public/api/register`. The 'Body' tab is selected, and 'form-data' is chosen as the body type. The following table represents the data entered in the form:

Key	Value
username	penggunasatu
nama	Pengguna 1
password	12345
password_confirmation	12345
level_id	2

Below the table, the 'Test Results' section shows a 201 Created status with a response time of 624 ms and a body size of 645 B. The response is displayed in JSON format:

```
{
  "success": true,
  "user": {
    "username": "penggunasatu",
    "nama": "Pengguna 1",
    "password": "$2y$12$Eb2SrV1jsykINytYGtrHi0DVAKcK5p6EgnZnmbChkPicIu7S0QJJU",
    "level_id": "2",
    "updated_at": "2024-04-22T15:56:04.000000Z",
    "created_at": "2024-04-22T15:56:04.000000Z",
    "user_id": 17
  }
}
```

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



POST <http://localhost:85/PEMOGRAMAN-WEB-LANJUT-2025/> **Send**

Overview Params Auth Headers (8) **Body** Scripts Tests Settings Cookies

form-data

	Key	Type	Value	Description	...	Bulk Edit
<input checked="" type="checkbox"/>	username	Text	penggunasatu			
<input checked="" type="checkbox"/>	nama	Text	Pengguna 1			
<input checked="" type="checkbox"/>	password	Text	12345			
<input checked="" type="checkbox"/>	password_confirmation	Text	12345			
<input checked="" type="checkbox"/>	level_id	Text	2			

Body **201 Created** • 2.59 s • 561 B • Save Response

JSON Preview Visualization

```
1 {
2   "success": true,
3   "user": {
4     "username": "penggunasatu",
5     "nama": "Pengguna 1",
6     "level_id": "2",
7     "updated_at": "2025-04-21T13:19:17.000000Z",
8     "created_at": "2025-04-21T13:19:17.000000Z",
9     "user_id": 55
10  }
```

Lakukan percobaan yang sama dan berikan screenshoot hasil percobaan Anda.

13. Lakukan commit perubahan file pada Github.

Praktikum 2 – Membuat RESTful API Login

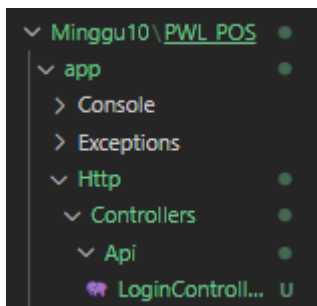
1. Kita buat file controller dengan nama LoginController. `php artisan make:controller Api/LoginController`



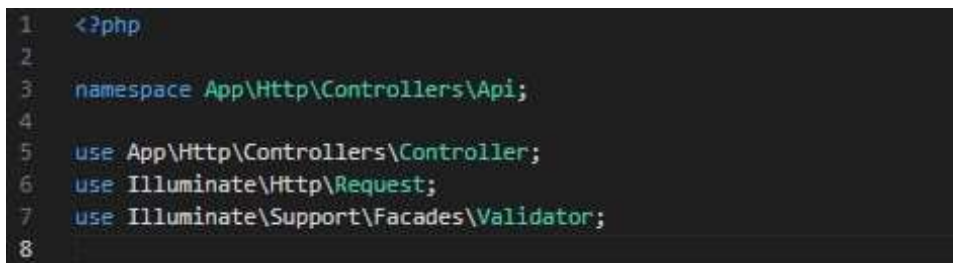
Jika berhasil maka akan ada tambahan controller pada folder Api dengan nama LoginController.



```
PS E:\laragon\www\PEMROGRAMAN-NEB-LANOUT-2025\Minggu10\VM_POS> php artisan make:controller Api/LoginController
[INFO] Controller [E:\laragon\www\PEMROGRAMAN-NEB-LANOUT-2025\Minggu10\VM_POS\app\Http\Controllers\Api>LoginController.php] created successfully.
PS E:\laragon\www\PEMROGRAMAN-NEB-LANOUT-2025\Minggu10\VM_POS>
```



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



```
1 <?php
2
3 namespace App\Http\Controllers\Api;
4
5 use App\Http\Controllers\Controller;
6 use Illuminate\Http\Request;
7 use Illuminate\Support\Facades\Validator;
8
```




```
9  class LoginController extends Controller
10 {
11     public function __invoke(Request $request)
12     {
13         //set validation
14         $validator = Validator::make($request->all(), [
15             'username' => 'required',
16             'password' => 'required'
17         ]);
18
19         //if validation fails
20         if ($validator->fails()) {
21             return response()->json($validator->errors(), 422);
22         }
23
24         //get credentials from request
25         $credentials = $request->only('username', 'password');
26
27         //if auth failed
28         if (!$token = auth()->guard('api')->attempt($credentials)) {
29             return response()->json([
30                 'success' => false,
31                 'message' => 'Username atau Password Anda salah'
32             ], 401);
33         }
34
35         //if auth success
36         return response()->json([
37             'success' => true,
38             'user' => auth()->guard('api')->user(),
39             'token' => $token
40         ], 200);
41     }
42 }
```




```
Minggu10 > PWL_POS > app > Http > Controllers > Api > LoginController.php > PHP Intelephense > LoginController > __invoke
1  <?php
2
3  namespace App\Http\Controllers\Api;
4
5  use App\Http\Controllers\Controller;
6  use Illuminate\Http\Request;
7  use Illuminate\Support\Facades\Validator;
8
9  0 references | 0 implementations
10 class LoginController extends Controller
11 {
12     0 references | 0 overrides
13     public function __invoke(Request $request): JsonResponse|mixed
14     {
15         //set validation
16         $validator = Validator::make($request->all(), [
17             'username' => 'required',
18             'password' => 'required'
19         ]);
20
21         //if validation fails
22         if ($validator->fails()) {
23             return response()->json($validator->errors(), 422);
24         }
25
26         //get credentials from request
27         $credentials = $request->only('username', 'password');
28
29         //if auth failed
30         if (!$token = auth()->guard('api')->attempt(credentials: $credentials)) {
31             return response()->json([
32                 'success' => false,
33                 'message' => 'Username atau Password Anda salah',
34             ], 401);
35         }
36
37         //if auth success
38         return response()->json([
39             'success' => true,
40             'user' => auth()->guard('api')->user(),
41             'token' => $token
42         ], 200);
43     }
44 }
```

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

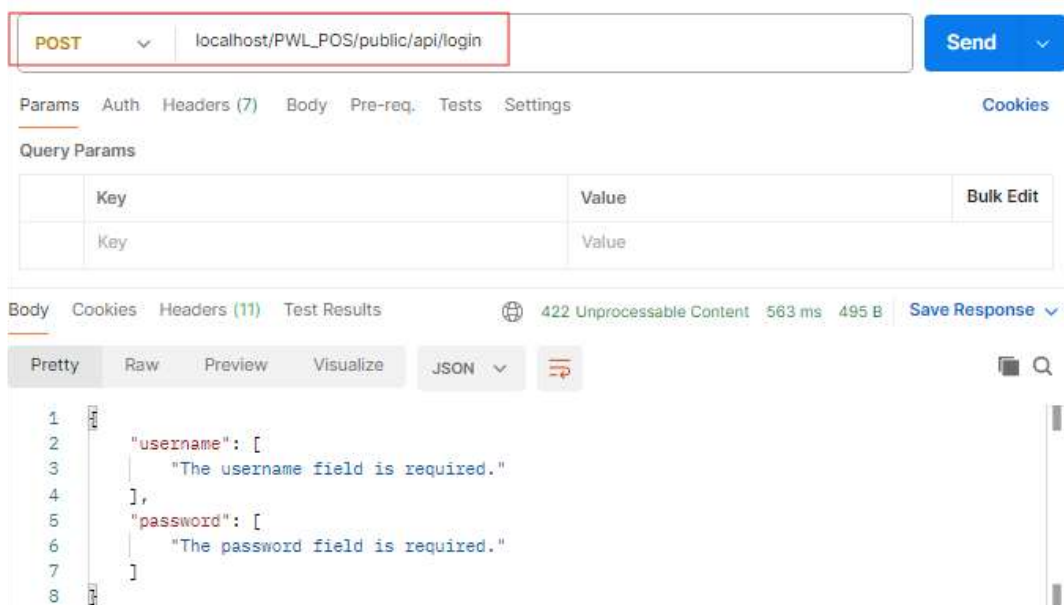
```
use App\Http\Controllers\Api\LoginController;

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();
});
```



```
Minggu10 > PWL_POS > routes > api.php > ...
1  <?php
2
3  use App\Http\Controllers\Api\RegisterController;
4  use Illuminate\Http\Request;
5  use Illuminate\Support\Facades\Route;
6  use App\Http\Controllers\Api>LoginController;
7
8  /*
9  |-----
10 | API Routes
11 |-----
12 |
13 | Here is where you can register API routes for your application. These
14 | routes are loaded by the RouteServiceProvider and all of them will
15 | be assigned to the "api" middleware group. Make something great!
16 |
17 */
18
19 Route::post('/register', App\Http\Controllers\Api\RegisterController::class)->name('register');
20
21 Route::post('/register', App\Http\Controllers\Api\RegisterController::class)->name('register');
22 Route::post('/login', App\Http\Controllers\Api>LoginController::class)->name('login');
23 Route::middleware('auth:api')->get('/user', function (Request $request): mixed {
24     return $request->user();
25 });
26
```

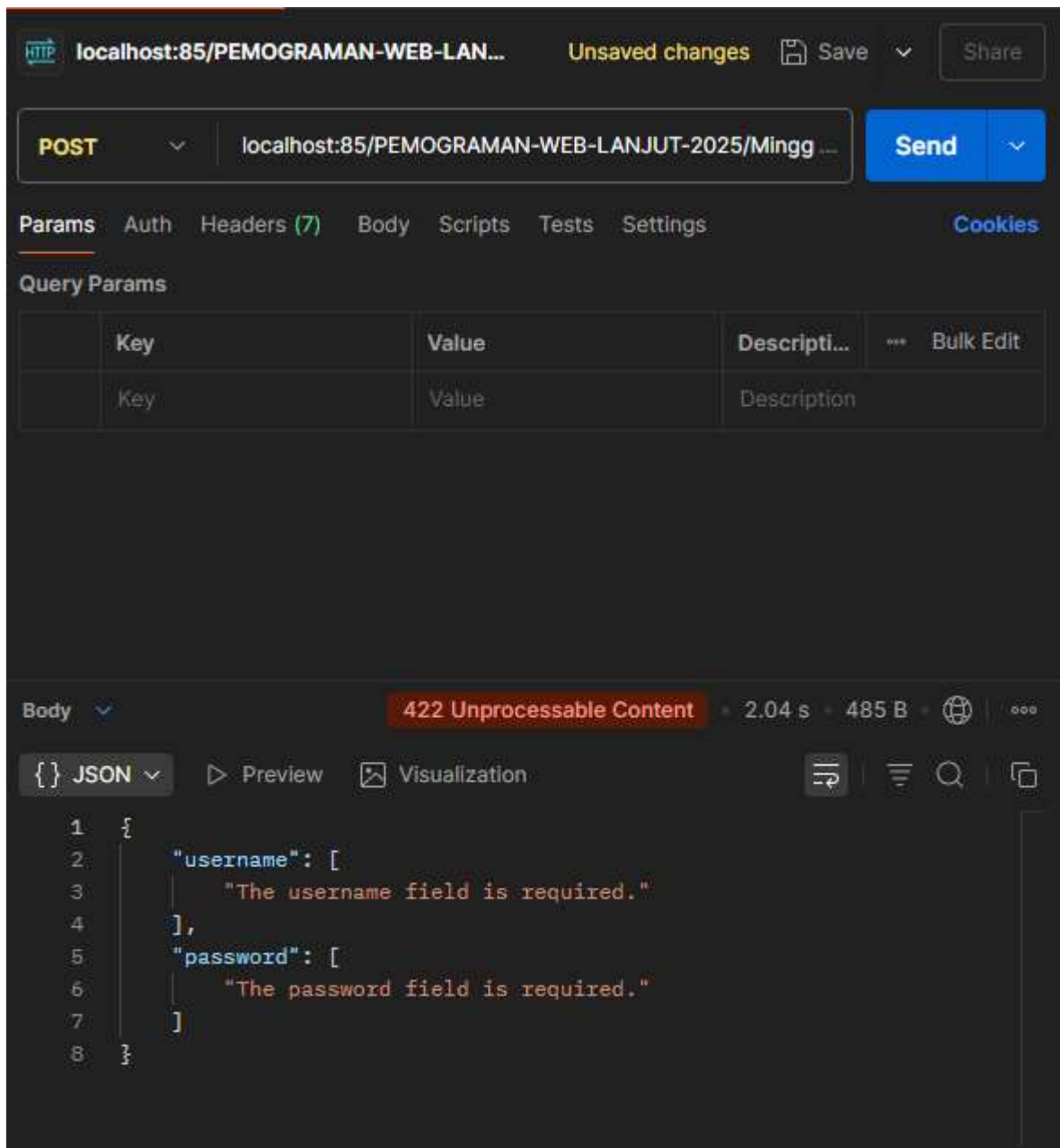
4. Jika sudah, kita akan melakukan uji coba REST API melalui aplikasi Postman. Buka aplikasi Postman, isi URL localhost/PWL_POS/public/api/login serta method POST. Klik Send.



Jika berhasil akan muncul error validasi seperti gambar di atas.



Lakukan percobaan yang sama dan berikan screenshoot hasil percobaan Anda.



- 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.



POST localhost/PWL_POS/public/api/login Send

Params Authorization Headers (8) Body Pre-request Script Tests Settings Cookies

none form-data x-www-form-urlencoded raw binary

Key	Value	...	Bulk Edit
<input checked="" type="checkbox"/> username	penggunasatu		
<input checked="" type="checkbox"/> password	12345		
Key	Value		

Body Cookies Headers (11) Test Results Status: 200 OK Time: 1501 ms Size: 986 B Save Response

Pretty Raw Preview Visualize JSON

```
1 {
2   "success": true,
3   "user": {
4     "user_id": 17,
5     "level_id": 2,
6     "username": "penggunasatu",
7     "nama": "Pengguna 1",
8     "password": "$2y$12$Eb2SxV1jsykINytY6tzHi0DVAKcK5p6EgnZnmbChkPciIu7S0QJJJu",
9     "created_at": "2024-04-22T15:56:04.000000Z",
10    "updated_at": "2024-04-22T15:56:04.000000Z"
11  },
12  "token": "eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJ3c3MiOiJodHRwOi8vbG9jYXRob3N0L1BXTF9QT1MtbnVpb19wdWJsawMvYXNpL2xvZ2luIiwiaWF0Ij0i"
13 }
```

POST localhost:8515/PEMOORAMAN-WEB-LANJUT-2025/Minggu10/PWL_POS/public/api/login Send

Params Authorization Headers (8) Body Scripts Tests Settings Cookies

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

Key	Value	Description	Bulk Edit
<input checked="" type="checkbox"/> username	Test - penggunasatu		
<input checked="" type="checkbox"/> password	Test - 12345		
Key	Value	Description	

Body Cookies Headers (11) Test Results Status: 200 OK 3.37 s 1007 B

JSON Preview Visualization

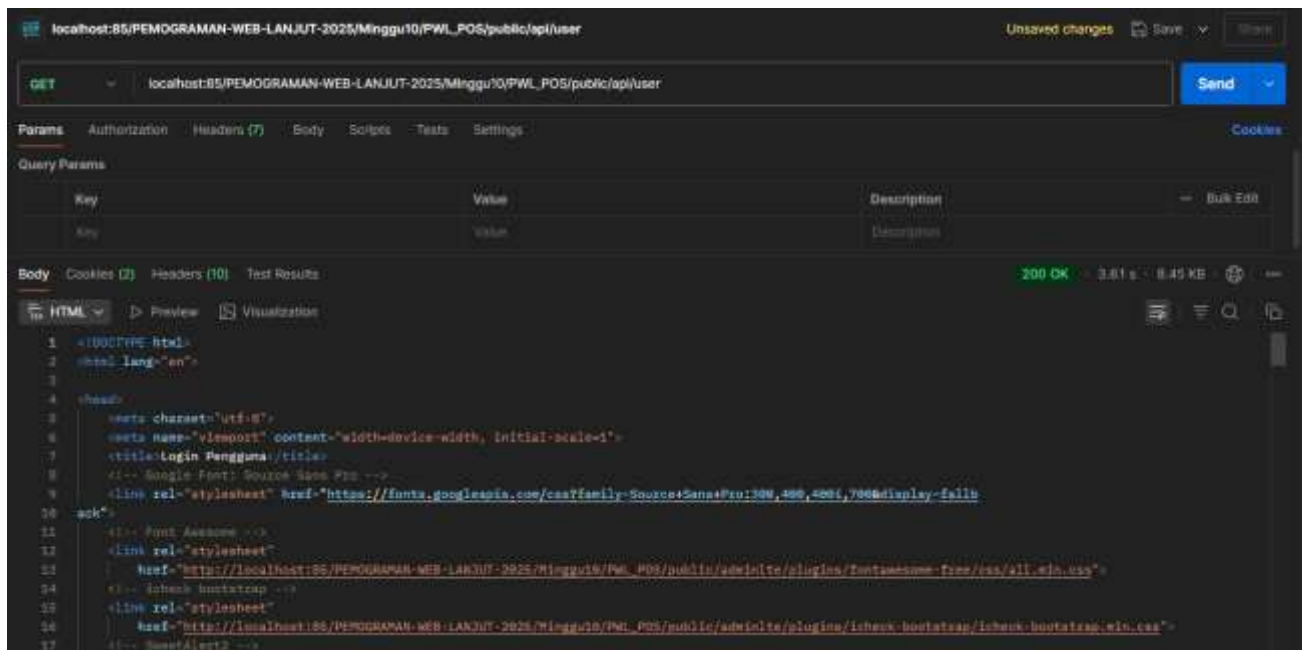
```
1 {
2   "success": true,
3   "user": {
4     "user_id": 15,
5     "level_id": 3,
6     "foto_profile": null,
7     "username": "penggunasatu",
8     "nama": "Pengguna 1",
9     "user_profile_picture": null,
10    "created_at": "2023-04-21T13:19:17.000000Z",
11    "updated_at": "2023-04-21T13:19:17.000000Z",
12    "foto": null
13  },
14  "token": "eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJ3c3MiOiJodHRwOi8vbG9jYXRob3N0L1BXTF9QT1MtbnVpb19wdWJsawMvYXNpL2xvZ2luIiwiaWF0Ij0i"
15 }
```



Lakukan percobaan yang sama dan berikan screenshoot hasil percobaan Anda.

6. 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.

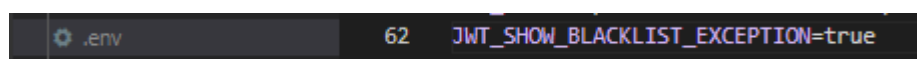
Jelaskan hasil dari percobaan tersebut.



7. Lakukan commit perubahan file pada Github.

Praktikum 3 – Membuat RESTful API Logout

1. Tambahkan kode berikut pada file `.env`
`JWT_SHOW_BLACKLIST_EXCEPTION=true`



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



```
Minggu10 > PWL_POS > app > Http > Controllers > Api > LogoutController.php > PHP Intelephense > LogoutController
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 0 references | 0 implementations
12 class LogoutController extends Controller
13 {
14     0 references | 0 overrides
14     public function __invoke(Request $request): JsonResponse|mixed
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 }
```

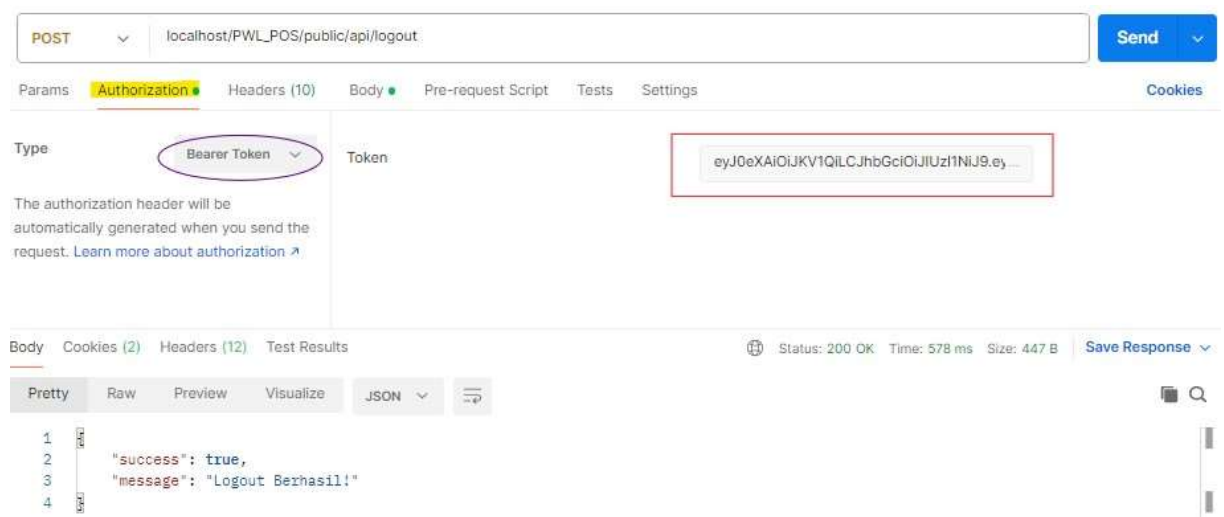
4. Lalu kita tambahkan routes pada api.php

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

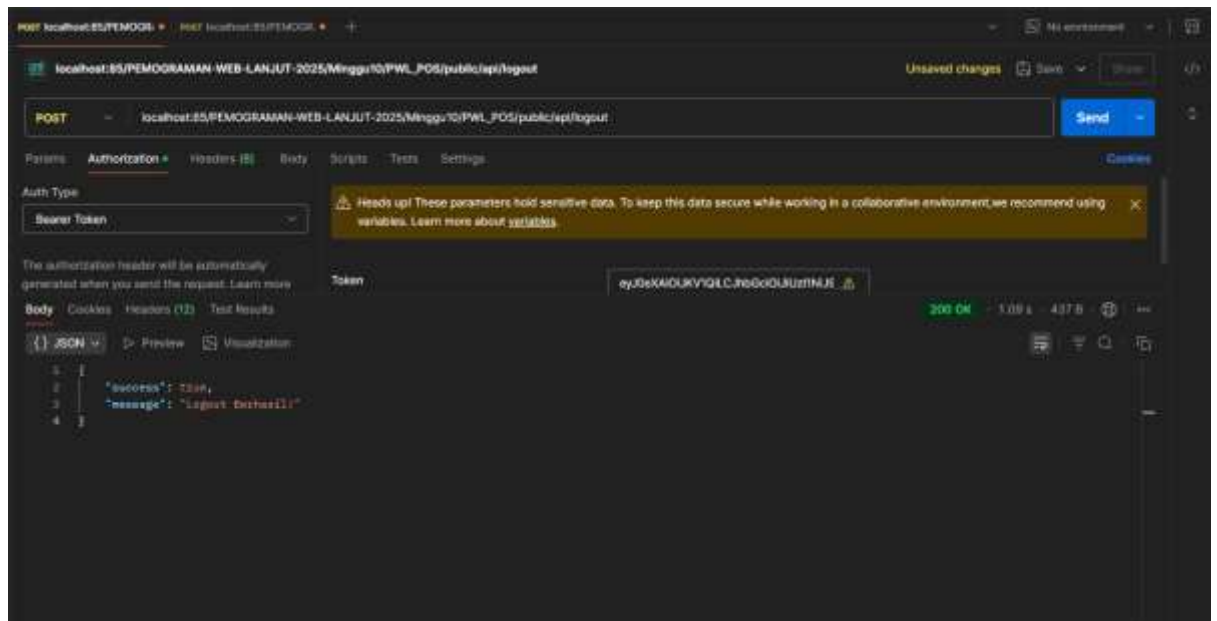



```
Minggu10 > PWL_POS > routes > api.php > ...
1  <?php
2
3  use App\Http\Controllers\Api\RegisterController;
4  use Illuminate\Http\Request;
5  use Illuminate\Support\Facades\Route;
6  use App\Http\Controllers\Api>LoginController;
7
8  /*
9  |-----
10 | API Routes
11 |-----
12 |
13 | Here is where you can register API routes for your application. These
14 | routes are loaded by the RouteServiceProvider and all of them will
15 | be assigned to the "api" middleware group. Make something great!
16 |
17 */
18
19 Route::post('/register', App\Http\Controllers\Api\RegisterController::class)->name('register');
20
21 Route::post('/register', App\Http\Controllers\Api\RegisterController::class)->name('register');
22 Route::post('/login', App\Http\Controllers\Api>LoginController::class)->name('login');
23 Route::middleware('auth:api')->get('/user', function (Request $request): mixed {
24     return $request->user();
25 });
26 Route::post('/logout', App\Http\Controllers\Api\LogoutController::class)->name('logout');
27
```

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.



Lakukan percobaan yang sama dan berikan screenshoot hasil percobaan Anda.



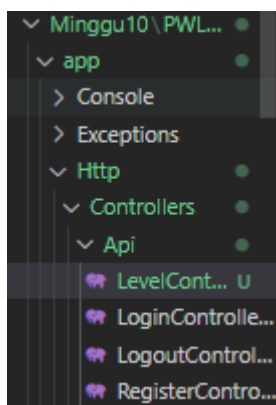
7. Lakukan commit perubahan file pada Github.

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.

`php artisan make:controller Api/LevelController`



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



```
namespace App\Http\Controllers\Api;
use App\Http\Controllers\Controller;
use Illuminate\Http\Request;
use App\Models\LevelModel;

class LevelController extends Controller
{
    public function index()
    {
        return LevelModel::all();
    }
}
```

```
public function store(Request $request)
{
    $level = LevelModel::create($request->all());
    return response()->json($level, 201);
}

public function show(LevelModel $level)
{
    return LevelModel::find($level);
}

public function update(Request $request, LevelModel $level)
{
    $level->update($request->all());
    return LevelModel::find($level);
}

public function destroy(LevelModel $user)
{
    $user->delete();

    return response()->json([
        'success' => true,
        'message' => 'Data terhapus',
    ]);
}
}
```



```
Minggu10 > PWL_POS > app > Http > Controllers > Api > LevelController.php > PHP > LevelController > destroy()
1  <?php
2
3  namespace App\Http\Controllers\Api;
4  use App\Http\Controllers\Controller;
5  use Illuminate\Http\Request;
6  use App\Models\LevelModel;
7
8  0 references | 0 implementations
9  class LevelController extends Controller
10 {
11     0 references | 0 overrides
12     public function index(): Collection{
13         return LevelModel::all();
14     }
15     0 references | 0 overrides
16     public function store(Request $request): JsonResponse|mixed
17     {
18         $level = LevelModel::create($request->all());
19         return response()->json($level, 201);
20     }
21     0 references | 0 overrides
22     public function show(LevelModel $level): array|Collection|LevelModel|Model|null
23     {
24         return LevelModel::find($level);
25     }
26     0 references | 0 overrides
27     public function update(Request $request, LevelModel $level): array|Collection|LevelModel|Model|null
28     {
29         $level->update($request->all());
30         return LevelModel::find($level);
31     }
32     0 references | 0 overrides
33     public function destroy(LevelModel $user): JsonResponse|mixed
34     {
35         $user->delete();
36         return response()->json([
37             'success' => true,
38             'message' => 'Data terhapus',
39         ]);
40     }
41 }
```

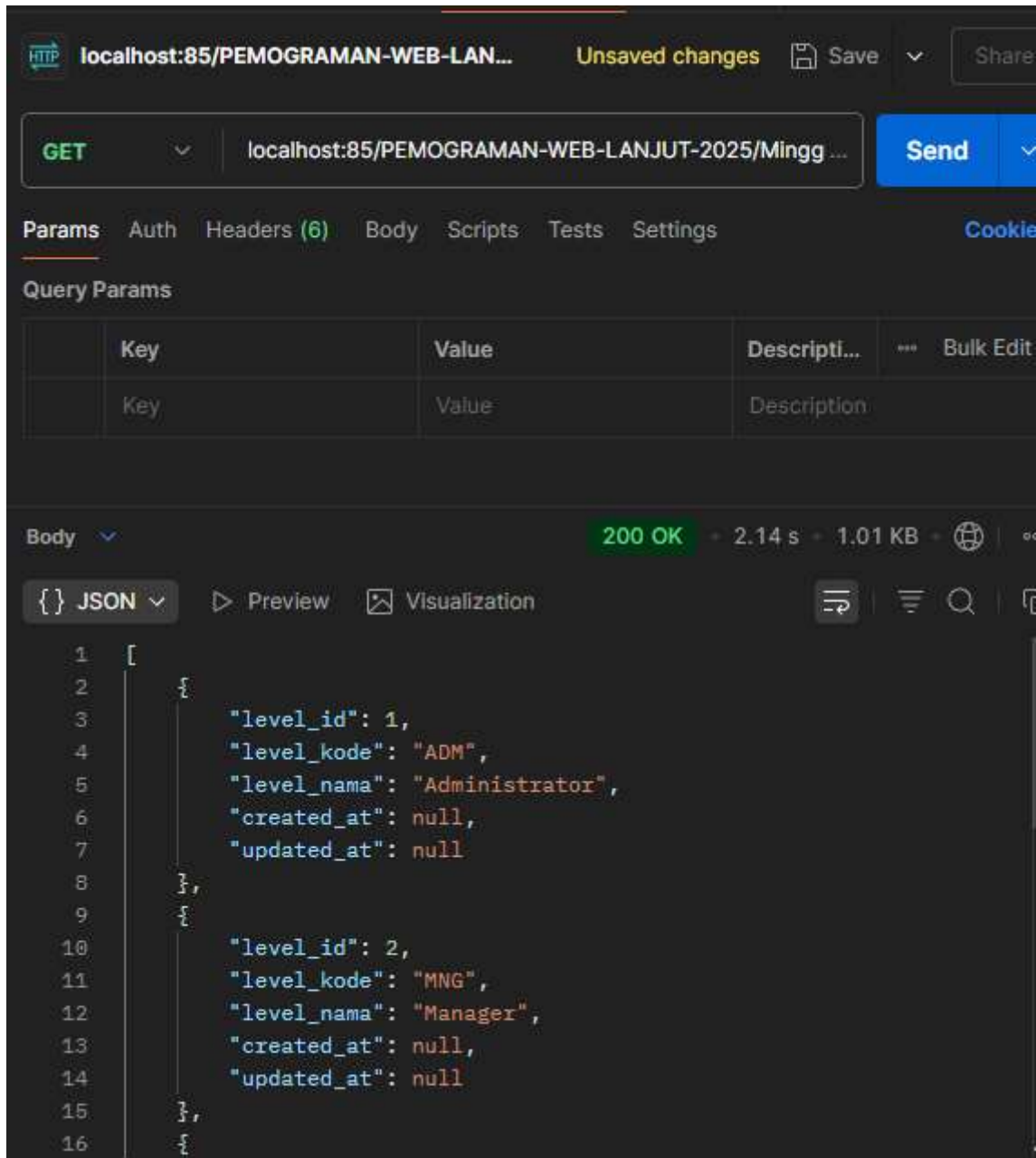
3. Kemudian kita lengkapi routes pada api.php.

```
use App\Http\Controllers\Api\LevelController;

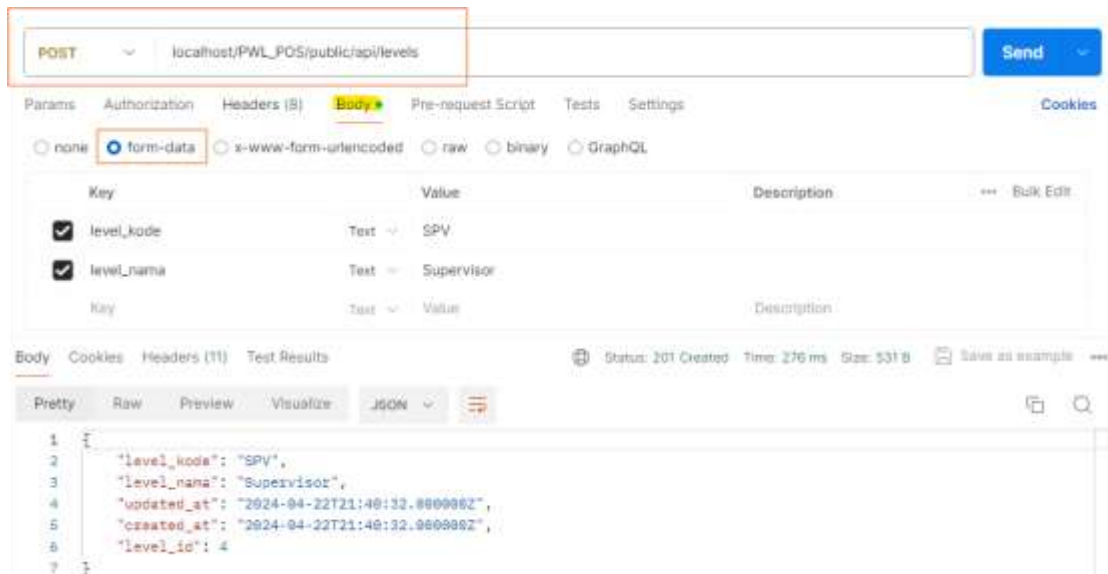
Route::get('levels', [LevelController::class, 'index']);
Route::post('levels', [LevelController::class, 'store']);
Route::get('levels/{level}', [LevelController::class, 'show']);
Route::put('levels/{level}', [LevelController::class, 'update']);
Route::delete('levels/{level}', [LevelController::class,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



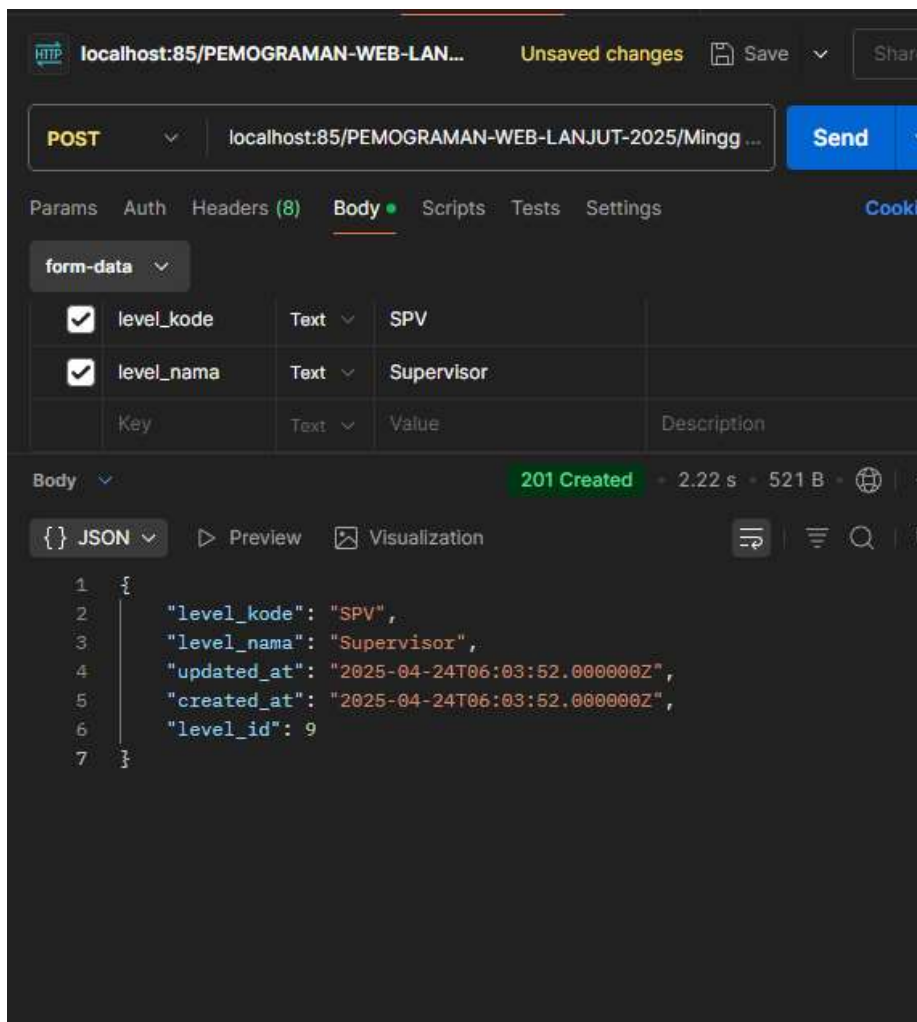
4. Jika sudah. Lakukan uji coba API mulai dari fungsi untuk menampilkan data. Gunakan URL: localhost/PWL_POS-main/public/api/levels dan method GET. **Jelaskan dan berikan screenshot hasil percobaan Anda.**



5. Kemudian, lakukan percobaan penambahan data dengan URL : localhost/PWL_POSmain/public/api/levels dan method POST seperti di bawah ini.

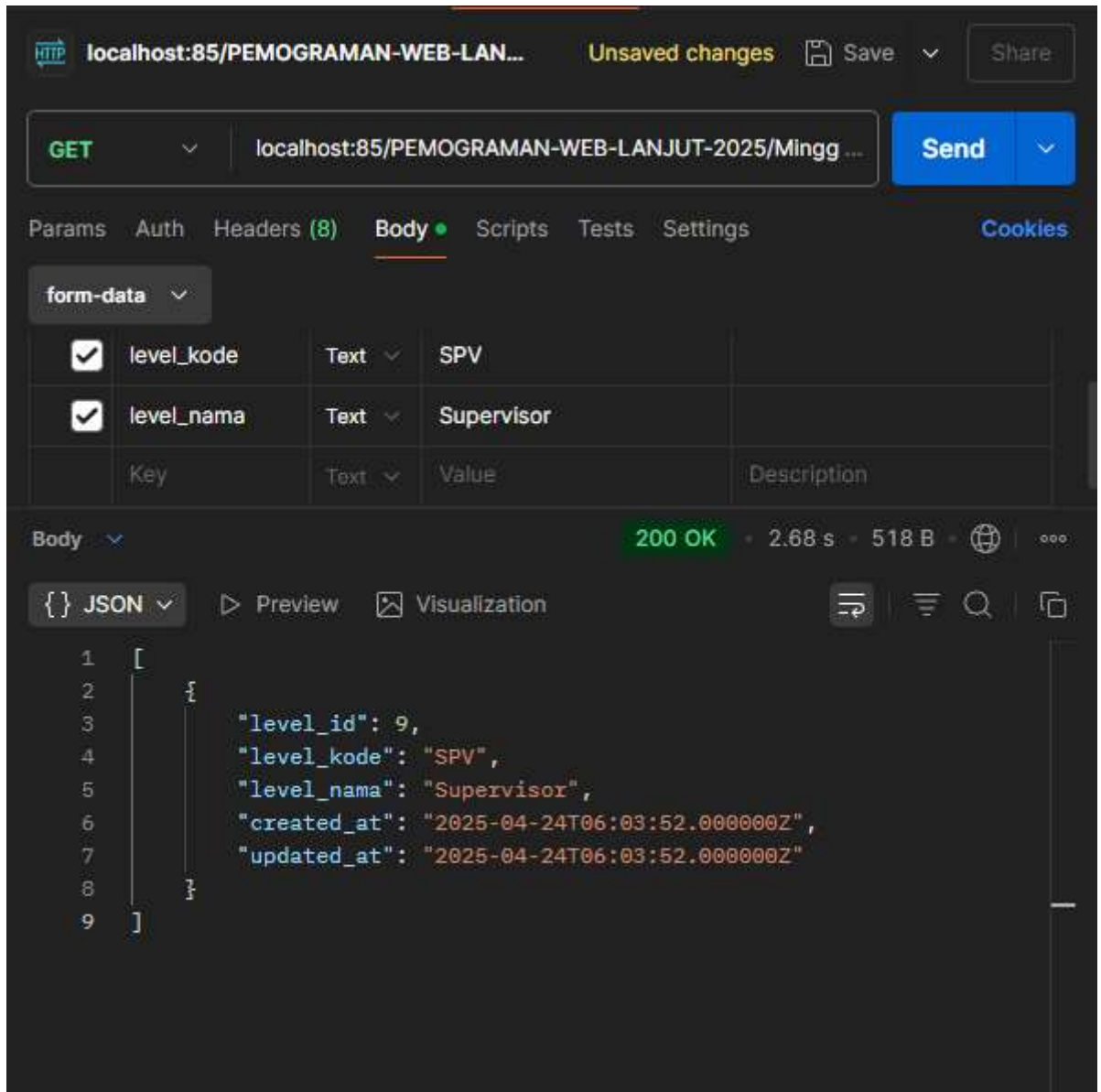


Jelaskan dan berikan screenshoot hasil percobaan Anda.





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





localhost:85/PEMOGRAMAN-WEB-L... Unsaved changes Save Share

PUT localhost:85/PEMOGRAMAN-WEB-LANJUT-2025/Min... Send

Params Auth Headers (8) Body Scripts Tests Settings Cookies

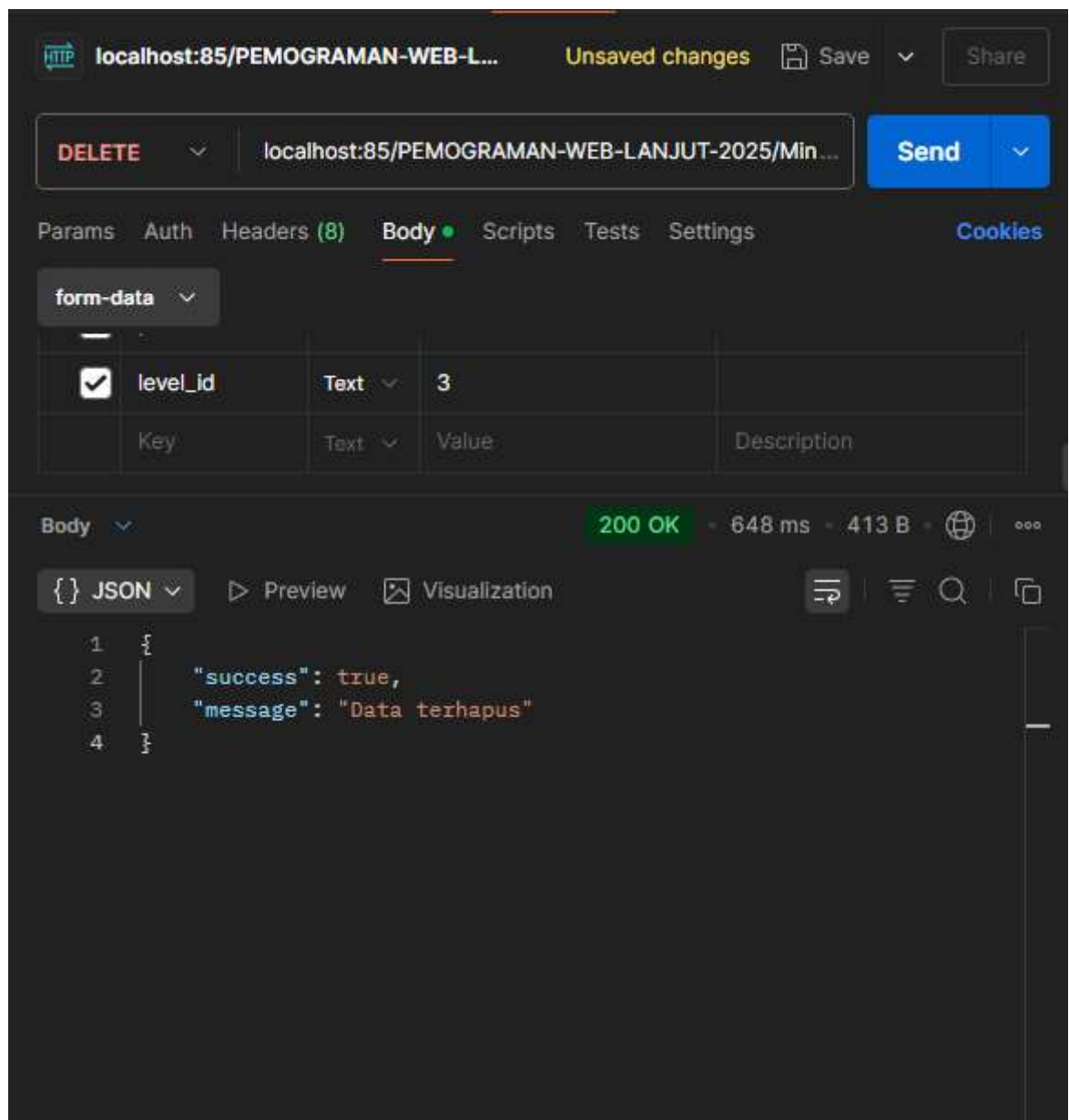
form-data

<input checked="" type="checkbox"/>	password	Text	bebi15
<input checked="" type="checkbox"/>	password_co...	Text	bebi15
<input checked="" type="checkbox"/>	level_id	Text	3

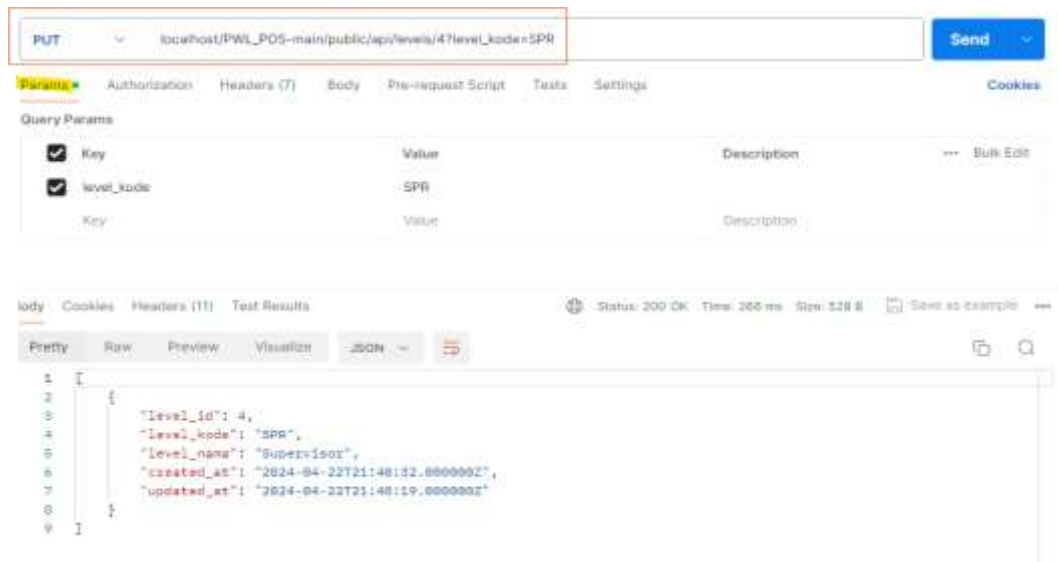
Body 200 OK • 1.17 s • 587 B

{ } JSON Preview Visualization

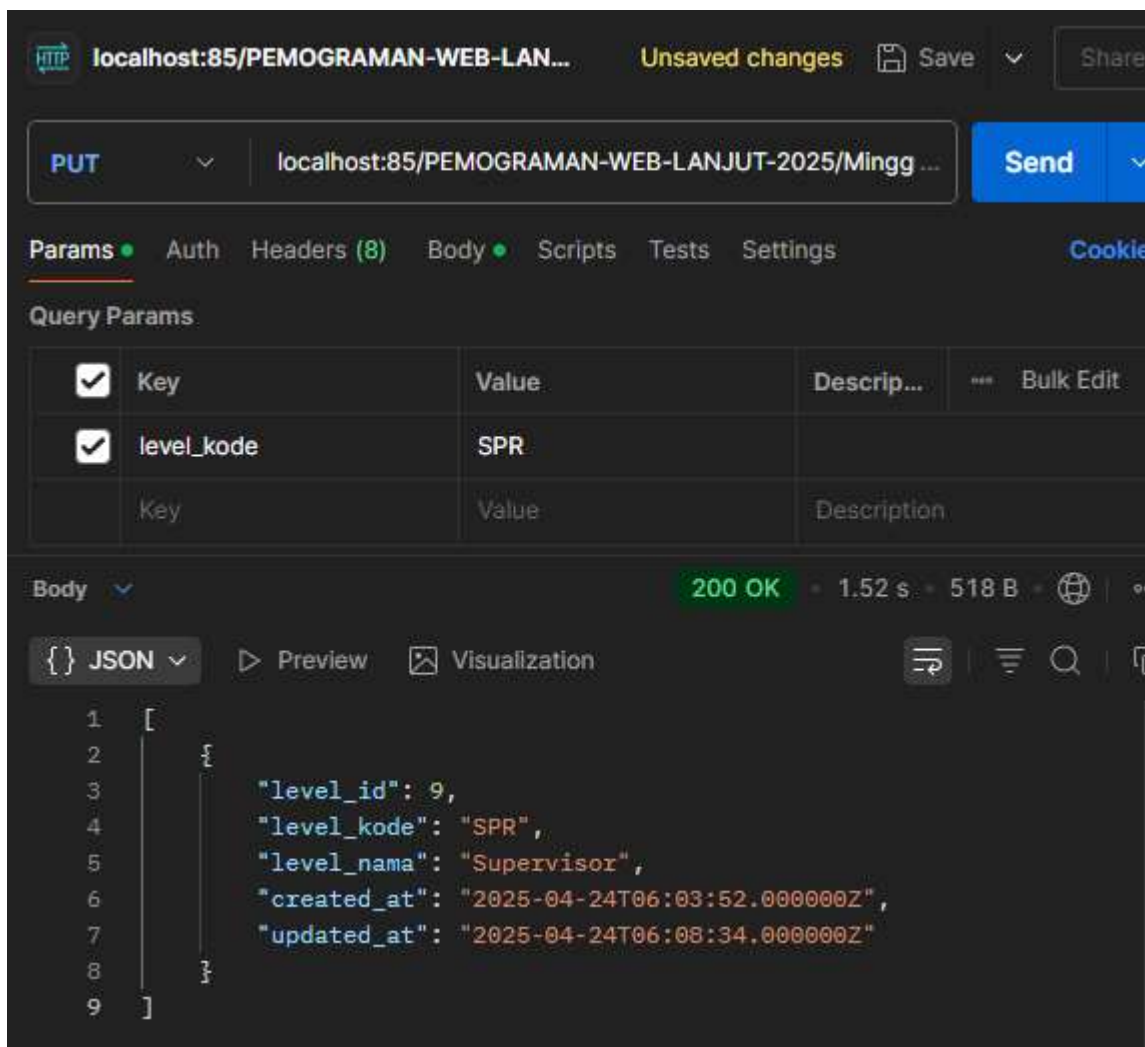
```
1 {
2   "user_id": 56,
3   "level_id": 3,
4   "foto_profil": null,
5   "username": "bebicantik",
6   "nama": "bebifindia",
7   "user_profile_picture": null,
8   "created_at": "2025-04-28T10:04:44.000000Z",
9   "updated_at": "2025-04-28T10:04:44.000000Z",
10  "foto": null
11 }
```



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

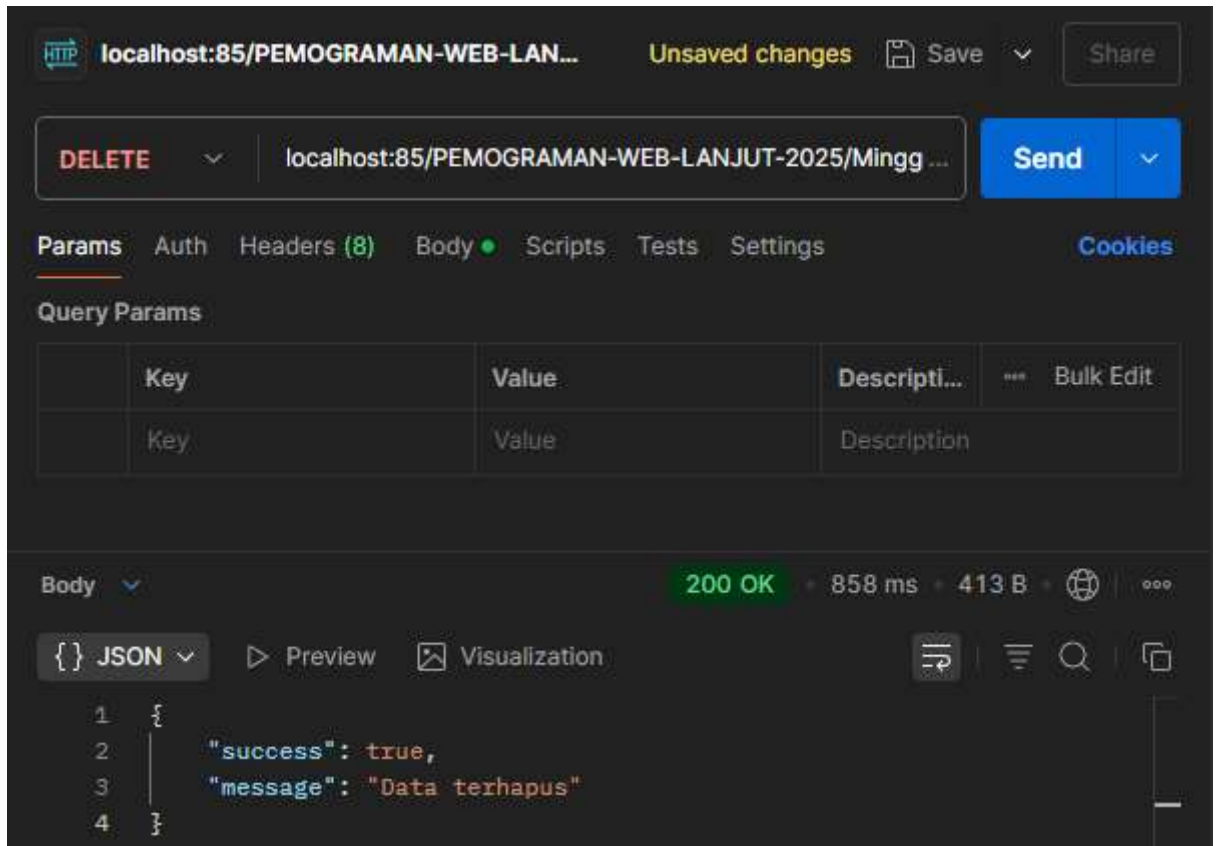


Jelaskan dan berikan screenshoot hasil percobaan Anda.





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



9. Lakukan commit perubahan file pada Github.



TUGAS

Implementasikan CRUD API pada tabel lainnya yaitu tabel m_user, m_kategori, dan m_barang

M_user

UserController.php

```
Minggu10 > PWL_POS > app > Http > Controllers > Api > UserController.php > PHP Intelephense > UserController
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\UserModel;
8
9  class UserController extends Controller
10 {
11     public function index(): Collection
12     {
13         return UserModel::all();
14     }
15
16     public function store(Request $request): JsonResponse|mixed
17     {
18         $user = UserModel::create($request->all());
19         return response()->json($user, 201);
20     }
21
22     public function show($id): array|Collection|Model|UserModel
23     {
24         return UserModel::findOrFail($id);
25     }
26
27     public function update(Request $request, $id): JsonResponse|mixed
28     {
29         $user = UserModel::findOrFail($id);
30         $user->update($request->all());
31         return response()->json($user);
32     }
33
34     public function destroy($id): JsonResponse|mixed
35     {
36         $user = UserModel::findOrFail($id);
```



Route User

```
// user
Route::get('users', [UserController::class, 'index']);
Route::post('users', [UserController::class, 'store']);
Route::get('users/{user}', [UserController::class, 'show']);
Route::put('users/{user}', [UserController::class, 'update']);
Route::delete('users/{user}', [UserController::class, 'destroy']);
// kategori
```

localhost:85/PEMOGRAMAN-WEB-L... Unsaved changes Save Share

GET localhost:85/PEMOGRAMAN-WEB-LANJUT-2025/Min... Send

Params Auth Headers (6) Body Scripts Tests Settings Cookies

Query Params

Key	Value	Descrip...	Bulk Edit
Key	Value	Description	

Body 200 OK 19.47 s 4.24 KB

{ } JSON Preview Visualization

```
1 [
2   {
3     "user_id": 1,
4     "level_id": 1,
5     "foto_profil": null,
6     "username": "admin",
7     "nama": "admin",
8     "user_profile_picture": null,
9     "created_at": null,
10    "updated_at": "2025-03-11T02:10:51.000000Z",
11    "foto": null
12  },
13  {
14    "user_id": 2,
15    "level_id": 2,
16    "foto_profil": null
```




The screenshot shows a web browser window with a REST client interface. The address bar displays `localhost:85/PEMOGRAMAN-WEB-L...`. The interface includes a **POST** method selector, a URL field with `localhost:85/PEMOGRAMAN-WEB-LANJUT-2025/Min...`, and a **Send** button. Below the URL bar, tabs for **Params**, **Auth**, **Headers (8)**, **Body**, **Scripts**, **Tests**, and **Settings** are visible. The **Body** tab is active, showing a **form-data** type. It contains three fields: **password** (Text, value: `beb15`), **password_co...** (Text, value: `beb15`), and **level_id** (Text, value: `3`). Below the form fields, the **Body** tab shows a **201 Created** status with a response time of `3.05 s` and a size of `535 B`. The response is displayed in **JSON** format, showing a successful creation of a user record.

```
{
  "username": "bebicantik",
  "nama": "bebifindia",
  "level_id": "3",
  "updated_at": "2025-04-28T10:04:44.000000Z",
  "created_at": "2025-04-28T10:04:44.000000Z",
  "user_id": 56
}
```




localhost:85/PEMOGRAMAN-WEB-L... Unsaved changes Save Share

DELETE localhost:85/PEMOGRAMAN-WEB-LANJUT-2025/Min... Send

Params Auth Headers (8) Body Scripts Tests Settings Cookies

form-data

<input checked="" type="checkbox"/>	password_co...	Text	bebi15
<input checked="" type="checkbox"/>	level_Id	Text	3

Body 200 OK • 1.05 s • 413 B

{ } JSON Preview Visualization

```
1 {
2   "success": true,
3   "message": "Data terhapus"
4 }
```



M_Kategori

KategoriController.php

```
Minggu10 > PWL_POS > app > Http > Controllers > Api > KategoriController.php > ...
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\KategoriModel;
8
9  class KategoriController extends Controller
10 {
11     public function index(): Collection
12     {
13         return KategoriModel::all();
14     }
15
16     public function store(Request $request): JsonResponse|mixed
17     {
18         $kategori = KategoriModel::create($request->all());
19         return response()->json($kategori, 201);
20     }
21
22     public function show($id): array|Collection|KategoriModel|Model
23     {
24         return KategoriModel::findOrFail($id);
25     }
26
27     public function update(Request $request, $id): JsonResponse|mixed
28     {
29         $kategori = KategoriModel::findOrFail($id);
30         $kategori->update($request->all());
31         return response()->json($kategori);
32     }
33
34     public function destroy($id): JsonResponse|mixed
35     {
36         $kategori = KategoriModel::findOrFail($id);
37         $kategori->delete();
38         return response()->json([
39             'success' => true,
40             'message' => 'Data terhapus',
41         ]);
42     }
43 }
```

Route Kategori

```
// kategori
Route::get('kategoris', [KategoriController::class, 'index']);
Route::post('kategoris', [KategoriController::class, 'store']);
Route::get('kategoris/{kategori}', [KategoriController::class, 'show']);
Route::put('kategoris/{kategori}', [KategoriController::class, 'update']);
Route::delete('kategoris/{kategori}', [KategoriController::class, 'destroy']);
// kategori
```



localhost:85/PEMOGRAMAN-WEB-L... Unsaved changes Save Share

GET localhost:85/PEMOGRAMAN-WEB-LANJUT-2025/Minggu10/PWL_POS/public/api/kategori

Params Auth Headers (8) Body Scripts Tests Settings Cookies

form-data

	Key		Value	Descri...	Bulk Edit
<input checked="" type="checkbox"/>	username	Text	bebicantik		
<input checked="" type="checkbox"/>	nama	Text	bebifindia		

Body 200 OK 5.56 s 1.85 KB

{ } JSON Preview Visualization

```
1  [
2    {
3      "kategori_id": 1,
4      "kategori_kode": "ELK001",
5      "kategori_nama": "Elektronik",
6      "created_at": "2025-03-04T04:25:33.000000Z",
7      "updated_at": "2025-03-04T04:25:33.000000Z"
8    },
9    {
10     "kategori_id": 2,
11     "kategori_kode": "PMS002",
12     "kategori_nama": "Pakaian & Mode",
13     "created_at": "2025-03-04T04:25:33.000000Z",
14     "updated_at": "2025-03-04T04:25:33.000000Z"
15   },
16 ]
```



localhost:85/PEMOGRAMAN-WEB-L... Unsaved changes Save Share

POST localhost:85/PEMOGRAMAN-WEB-LANJUT-2025/Min... Send

Params Auth Headers (8) Body Scripts Tests Settings Cookies

form-data

	Key	Type	Value	Description
<input checked="" type="checkbox"/>	kategori_nama	Text	Skincare	
<input checked="" type="checkbox"/>	kategori_kode	Text	SKN	

Body 201 Created 1.72 s 529 B

{ } JSON Preview Visualization

```
1 {  
2   "kategori_nama": "Skincare",  
3   "kategori_kode": "SKN",  
4   "updated_at": "2025-04-28T11:21:51.000000Z",  
5   "created_at": "2025-04-28T11:21:51.000000Z",  
6   "kategori_id": 53  
7 }
```



localhost:85/PEMOGRAMAN-WEB-L... Unsaved changes Save Share

DELETE localhost:85/PEMOGRAMAN-WEB-LANJUT-2025/Minggu10/PWL_POS/public/api/kategori/53 Send

Params Auth Headers (8) Body Scripts Tests Settings Cookies

form-data

	Key	Type	Value	Description
<input checked="" type="checkbox"/>	kategori_nama	Text	Skincare	
<input checked="" type="checkbox"/>	kategori_kode	Text	SKN	

Body 200 OK • 1.12 s • 413 B

{ } JSON Preview Visualization

```
1 {
2   "success": true,
3   "message": "Data terhapus"
4 }
```



M_Barang

BarangController.php

```
Minggu10 > PWL_POS > app > Http > Controllers > Api > BarangController.php > ...
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\BarangModel;
8
9  class BarangController extends Controller
10 {
11     public function index(): Collection
12     {
13         return BarangModel::all();
14     }
15
16     public function store(Request $request): JsonResponse|mixed
17     {
18         $barang = BarangModel::create($request->all());
19         return response()->json($barang, 201);
20     }
21
22     public function show($id): array|BarangModel|Collection|Model
23     {
24         return BarangModel::findOrFail($id);
25     }
26
27     public function update(Request $request, $id): JsonResponse|mixed
28     {
29         $barang = BarangModel::findOrFail($id);
30         $barang->update($request->all());
31         return response()->json($barang);
32     }
33
34     public function destroy($id): JsonResponse|mixed
35     {
36         $barang = BarangModel::findOrFail($id);
37         $barang->delete();
38         return response()->json([
39             'success' => true,
40             'message' => 'Data terhapus',
```




Route Barang

```
51 // barang
52 Route::get('barangs', [BarangController::class, 'index']);
53 Route::post('barangs', [BarangController::class, 'store']);
54 Route::get('barangs/{barang}', [BarangController::class, 'show']);
55 Route::put('barangs/{barang}', [BarangController::class, 'update']);
56 Route::delete('barangs/{barang}', [BarangController::class, 'destroy']);
57
```

localhost:85/PEMOGRAMAN-WEB-L... Unsaved changes Save Share

GET localhost:85/PEMOGRAMAN-WEB-LANJUT-2025/Minggu10/PWL_POS/public/api/barangs Send

Params Auth Headers (8) Body Scripts Tests Settings Cookies

form-data

Key	Type	Value	Description
<input checked="" type="checkbox"/> kategori_nama	Text	Skincare	
<input checked="" type="checkbox"/> kategori_kode	Text	SKN	

Body 200 OK 1.45 s 3.43 KB

{ JSON Preview Visualization

```
1 [
2   {
3     "barang_id": 1,
4     "kategori_id": 1,
5     "barang_kode": "BRG001",
6     "barang_nama": "Laptop ASUS ROG",
7     "harga_beli": 15000000,
8     "harga_jual": 18000000,
9     "created_at": "2025-03-04T04:43:17.000000Z",
10    "updated_at": "2025-03-04T04:43:17.000000Z"
11  },
12  {
13    "barang_id": 2,
14    "kategori_id": 1,
15    "barang_kode": "BRG002",
16    "barang_nama": "Samsung Galaxy S23"
```



localhost:85/PEMOGRAMAN-WEB-L... Unsaved changes Save Share

POST localhost:85/PEMOGRAMAN-WEB-LANJUT-2025/Min... Send

Params Auth Headers (8) Body Scripts Tests Settings Cookies

form-data

	Key		Value	Descri...	Bulk Edit
<input checked="" type="checkbox"/>	barang_kode	Text	HN23		
<input checked="" type="checkbox"/>	barang_nama	Text	Handuk		

Body 201 Created 2.59 s 582 B

{ } JSON Preview Visualization

```
1 {
2   "barang_kode": "HN23",
3   "barang_nama": "Handuk",
4   "kategori_id": "2",
5   "harga_beli": "12000",
6   "harga_jual": "14000",
7   "updated_at": "2025-04-28T11:29:09.000000Z",
8   "created_at": "2025-04-28T11:29:09.000000Z",
9   "barang_id": 42
10 }
```



localhost:85/PEMOGRAMAN-WEB-L... Unsaved changes Save Share

DELETE localhost:85/PEMOGRAMAN-WEB-LANJUT-2025/Min... Send

Params Auth Headers (8) Body Scripts Tests Settings Cookies

form-data

	Key		Value	Descri...	Bulk Edit
<input checked="" type="checkbox"/>	barang_kode	Text	HN23		
<input checked="" type="checkbox"/>	barangq_nama	Text	Handuk		

Body 200 OK 1.30 s 413 B

{ } JSON Preview Visualization

```
1 {
2   "success": true,
3   "message": "Data terhapus"
4 }
```

*** Sekian, dan selamat belajar ***