



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 : Aqila Nur Azza

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 otentikasi 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 otentikasi 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.

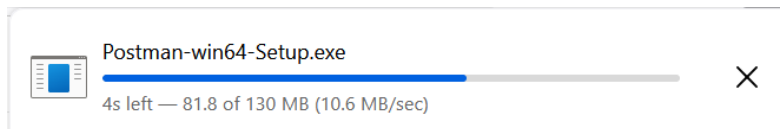
Berikut adalah cara kerja JWT :

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.

```
PS C:\laragon\www\PWL_2025\PWL_WEEK10\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)
- Installing stella-maris/clock (0.1.7): Extracting archive
- Installing lcobucci/clock (2.3.0): Extracting archive
- Installing lcobucci/jwt (4.0.4): Extracting archive
- Installing tymon/jwt-auth (2.1.1): Extracting archive
Generating optimized autoload files
> Illuminate\Foundation\ComposerScripts::postAutoloadDump
> @php artisan package:discover --ansi

INFO: Discovering packages.
barryvdh/laravel-dompdf ..... DONE
laravel/sail ..... DONE
laravel/sanctum ..... DONE
laravel/tinker ..... DONE
nesbot/carbon ..... DONE
nunomaduro/collision ..... DONE
nunomaduro/termwind ..... DONE
spatie/laravel-ignition ..... DONE
tymon/jwt-auth ..... DONE
yajra/laravel-datatables-buttons ..... DONE
yajra/laravel-datatables-editor ..... DONE
yajra/laravel-datatables-fractal ..... DONE
yajra/laravel-datatables-html ..... DONE
yajra/laravel-datatables-oracle ..... DONE

14 packages you are using are looking for funding.
Use the `composer fund` command to find out more!
> @php artisan vendor:publish --tag=laravel-assets --ansi --force
```



```
INFO No publishable resources for tag [laravel-assets].  
Found 1 security vulnerability advisory affecting 1 package.  
Run "composer audit" for a full list of advisories.
```

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

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

```
PS C:\laragon\www\PWL_2025\PWL_WEEK10\pwl_pos> php artisan vendor:publish --provider="Tymon\JWTAuth\Providers\LaravelServiceProvider"
```

```
INFO Publishing assets.  
Copying file [C:\laragon\www\PWL_2025\PWL_WEEK10\pwl_pos\vendor\tymon\jwt-auth\config\config.php] to [C:\laragon\www\PWL_2025\PWL_WEEK10\pwl_pos\config\jwt.php] DONE
```

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.

```
jwt.php PWL_POS\config
```

5. Setelah itu jalankan perintah berikut untuk membuat secret key JWT. `php artisan jwt:secret`

```
PS C:\laragon\www\PWL_2025\PWL_WEEK10\pwl_pos> php artisan jwt:secret
```

```
jwt-auth secret [5JTh1sLrpMBQh5URV6JFNU0uaL9fxZjtVQXmSQR9PY9IbpfvEuPNzF782DN9zXDV] set successfully.
```

Jika berhasil, maka pada file .env akan ditambahkan sebuah baris berisi nilai key JWT_SECRET.

```
1 JWT_SECRET=5JTh1sLrpMBQh5URV6JFNU0uaL9fxZjtVQXmSQR9PY9IbpfvEuPNzF782DN9zXDV
```

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

```
1 'guards' => [  
2     'web' => [  
3         'driver' => 'session',  
4         'provider' => 'users',  
5     ],  
6     'api' => [  
7         'driver' => 'jwt',  
8         'provider' => 'users',  
9     ],  
10 ],
```

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

```
1 use Tymon\JWTAuth\Contracts\JWTSubject;  
2  
3 class UserModel extends Authenticatable implements JWTSubject  
4 {  
5     use HasFactory;  
6  
7     public function getJWTIdentifier() {  
8         return $this->getKey();  
9     }  
10  
11     public function getJWTCustomClaims(){  
12         return [];  
13     }  
14 }
```

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

`php artisan make:controller Api/RegisterController`

```
PS C:\laragon\www\PWL_2025\PWL_WEEK10\pwl_pos> php artisan make:controller Api/RegisterController
```

```
INFO Controller [C:\laragon\www\PWL_2025\PWL_WEEK10\PWL_POS\app\Http\Controllers\Api\RegisterController.php] created successfully.
```

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

```
RegisterController.php PWL_POS\app\H...
```

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 validation 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' => $request->level_id,
33         ]);
34
35         // Return response JSON if user is created
36         if ($user) {
37             return response()->json([
38                 'success' => true,
39                 'user'     => $user,
40             ], 201);
41         }
42
43         // Return JSON if process insert failed
44         return response()->json([
45             'success' => false,
46         ], 409);
47     }
48 }
```

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



```
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'));
19
```

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.

POST localhost/PWL_POS/public/api/register Send

Params Auth Headers (7) Body Pre-req. Tests Settings Cookies

Query Params

Key	Value	Bulk Edit
Key	Value	

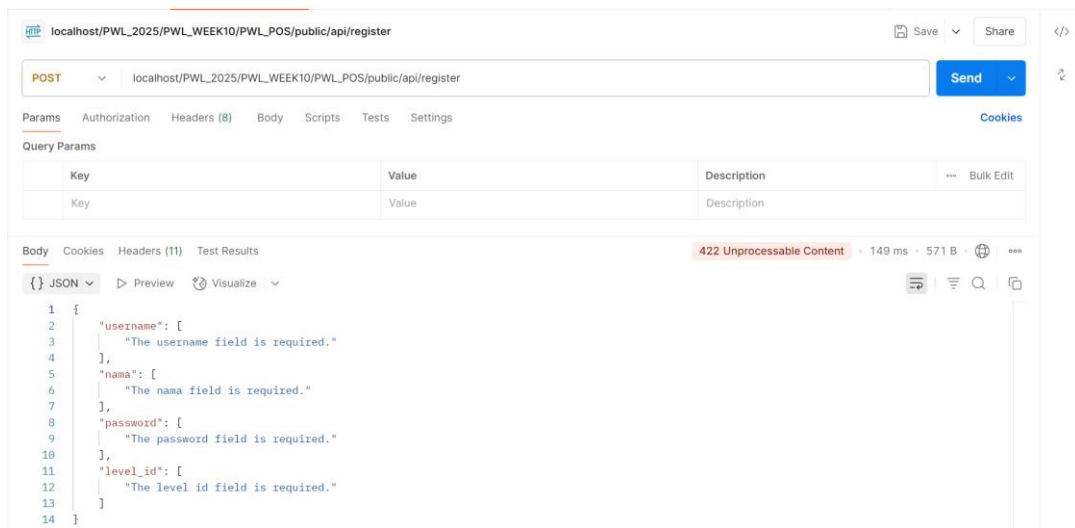
Body 422 Unprocessable Content 272 ms 534 B Save Response

Pretty Raw Preview Visualize JSON

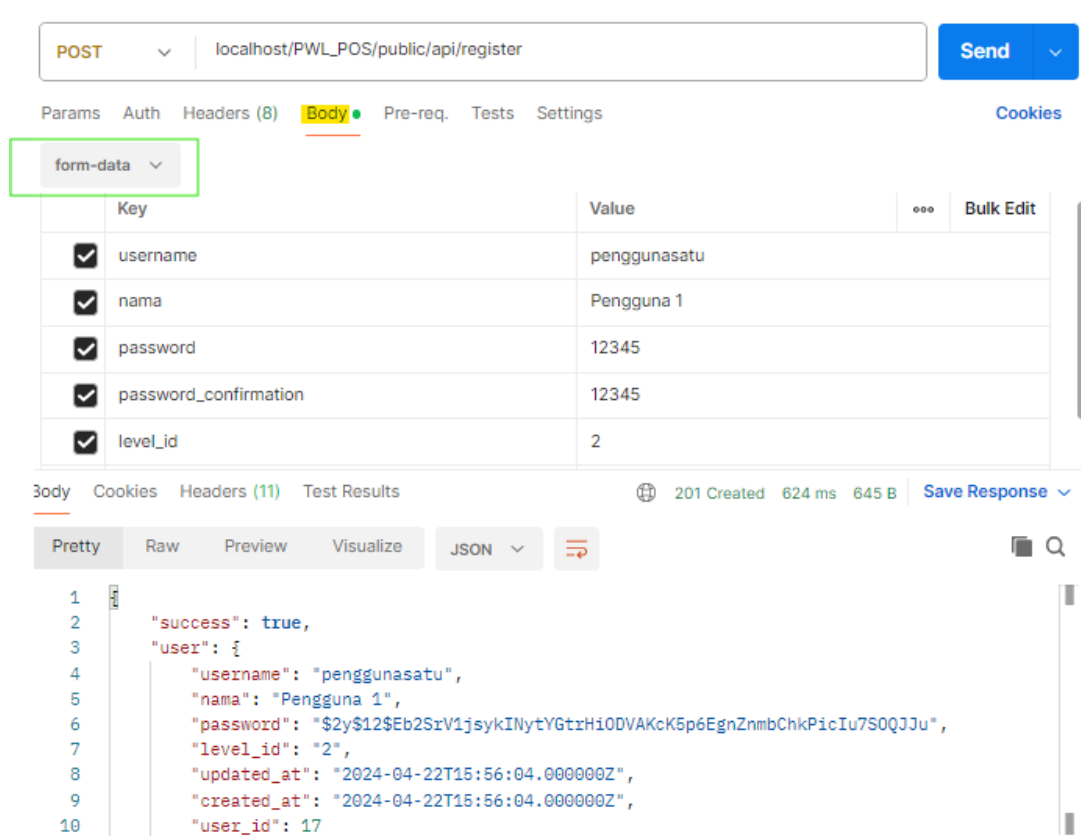
```
1 {
2   "username": [
3     "The username field is required."
4   ],
5   "nama": [
6     "The nama field is required."
7   ],
8   "password": [
9     "The password field is required."
10  ]
11 }
```

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.



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



Lakukan percobaan yang sama dan berikan screenshoot hasil percobaan Anda.

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

form-data ▾

	Key		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_confirm...	Text ▾	12345			
<input checked="" type="checkbox"/>	level_id	Text ▾	2			

Body ▾ 201 Created • 477 ms • 561 B • 🌐 • ⋮

{ } JSON ▾ ▶ Preview 🔗 Visualize ▾

```
1 {
2   "success": true,
3   "user": {
4     "username": "penggunasatu",
5     "nama": "Pengguna 1",
6     "level_id": "2",
7     "updated_at": "2025-04-26T13:35:23.000000Z",
8     "created_at": "2025-04-26T13:35:23.000000Z",
9     "user_id": 18
10  }
11 }
```

Daftar User Home / User

Daftar Pengguna Import User Export Excel Tambah Data Export PDF

Filter: - Semua - ▾
Level Pengguna

Show 10 ▾ entries Search:

No	Username	Nama	Level	Aksi
11	Manager Baru	Aqila	Manager	Detail Edit Hapus
12	CEO	CEO Aqila	Unknown	Detail Edit Hapus
13	penggunasatu	Pengguna 1	Manager	Detail Edit Hapus

Showing 11 to 13 of 13 entries Previous 1 2 Next

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`

```
PS C:\laragon\www\PWL_2025\PWL_WEEK10\pwl_pos> php artisan make:controller Api/LoginController
```



INFO Controller [C:\laragon\www\PWL_2025\PWL_WEEK10\PWL_POS\app\Http\Controllers\Api\LoginController.php] created successfully.

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



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

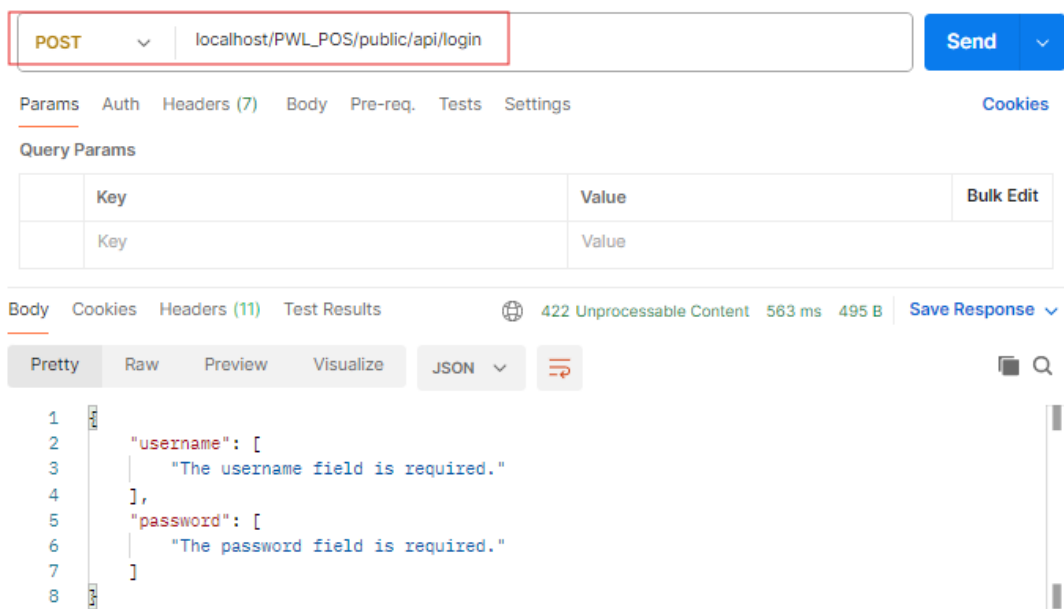
```
1 <?php
2
3 namespace App\Http\Controllers\Api;
4
5 use App\Http\Controllers\Controller;
6 use Illuminate\Http\Request;
7 use Illuminate\Support\Facades\Validator;
8
9 class LoginController extends Controller
10 {
11     public function __invoke(Request $request)
12     {
13         // Set validation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

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



```
1 <?php
2
3 use App\Http\Controllers\Api\RegisterController;
4 use App\Http\Controllers\Api>LoginController;
5 use Illuminate\Http\Request;
6 use Illuminate\Support\Facades\Route;
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 Route::post('/login', LoginController::class)->name('login');
21
22 Route::middleware('auth:api')->get('/user', function (Request $request) {
23     return $request->user();
24 });
25
```

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.



POST localhost/PWL_2025/PWL_WEEK10/PWL_POS/public/api/login Send

Params Auth Headers (8) Body Scripts Tests Settings Cookies

Query Params

Key	Value	Description	Bulk Edit
Key	Value	Description	

Body 422 Unprocessable Content 180 ms 485 B

{ } JSON Preview Visualize

```
1 {
2   "username": [
3     "The username field is required."
4   ],
5   "password": [
6     "The password field is required."
7   ]
8 }
```

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.

POST localhost/PWL_POS/public/api/login Send

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

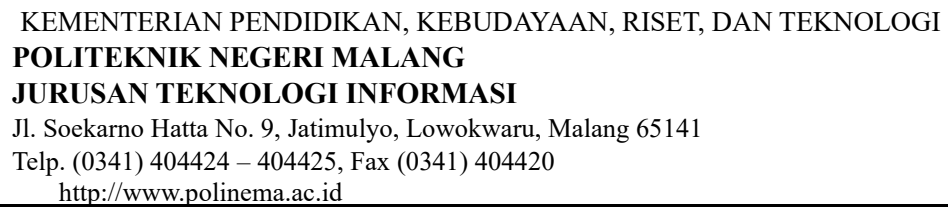
none form-data x-www-form-urlencoded raw binary

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$12$Eb2SxV1jsykINytYGtrHi0DVAKcK5p6EgnZnmbChkPicIu7S0QJJJu",
9     "created_at": "2024-04-22T15:56:04.000000Z",
10    "updated_at": "2024-04-22T15:56:04.000000Z"
11  },
12   "token": "eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJ3c3MiOiJodHRwOi8vbG9jYXRob3N0L1BXTF9QT1MtbnVpbi9wdWJsawMvYXBPL2xvZ21uIiwiaWF0Ij0i"
13 }
```

[illegible]

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

form-data ▾

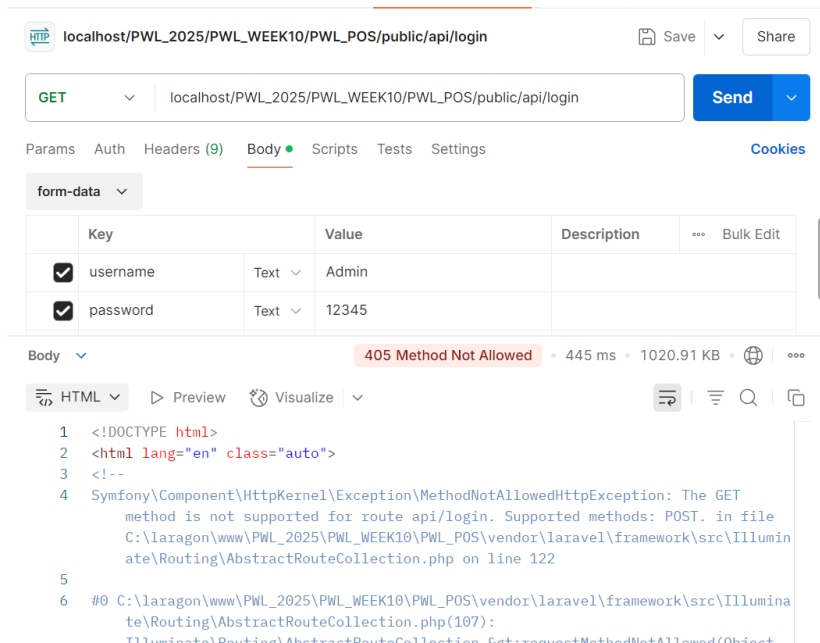
	Key		Value	Description	⋮ Bulk Edit
<input checked="" type="checkbox"/>	username	Text ▾	penggunasatu		
<input checked="" type="checkbox"/>	password	Text ▾	123456		

Body ▾ 401 Unauthorized • 381 ms • 444 B • 🌐 • ⋮

{ } JSON ▾ ▶ Preview 🔗 Visualize ▾

```
1 {
2   "success": false,
3   "message": "Username atau Password Anda salah"
4 }
```

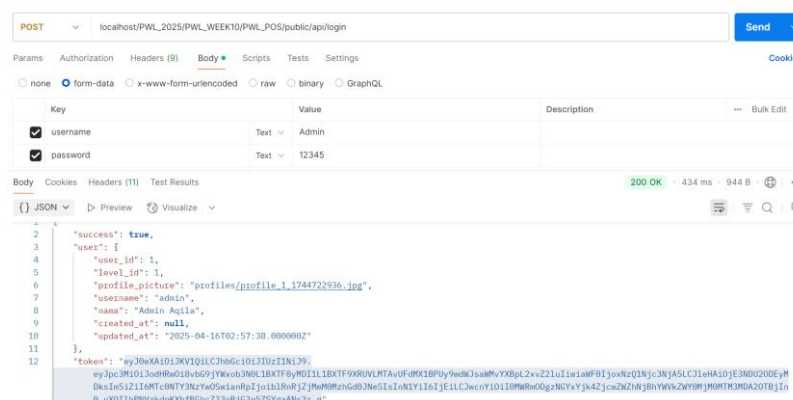
- Jika menggunakan GET



Penjelasan :

Login tidak bisa menggunakan metode GET karena proses login membutuhkan pengiriman data berupa username dan password ke server. Dalam HTTP, semua proses yang mengirimkan data harus menggunakan metode POST, bukan GET. GET hanya digunakan untuk mengambil data, bukan untuk mengirimkan data baru. Setelah login berhasil dan mendapatkan token, barulah kita bisa menggunakan metode GET untuk mengambil data user yang sedang login.

➤ Jika Menggunakan POST



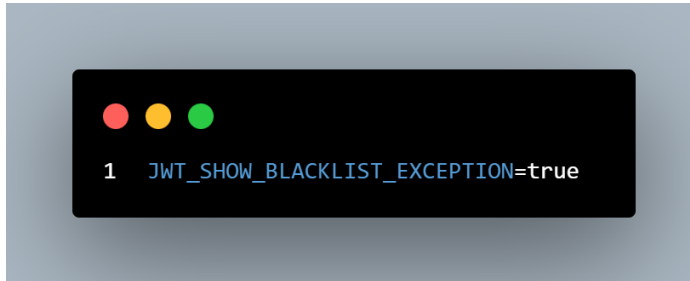
7. Lakukan commit perubahan file pada Github.

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`

```
PS C:\laragon\www\PWL_2025\PWL_WEEK10\pwl_pos> php artisan make:controller Api/LogoutController
```

```
INFO Controller [C:\laragon\www\PWL_2025\PWL_WEEK10\PWL_POS\app\Http\Controllers\Api\LogoutController.php] created successfully.
```

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

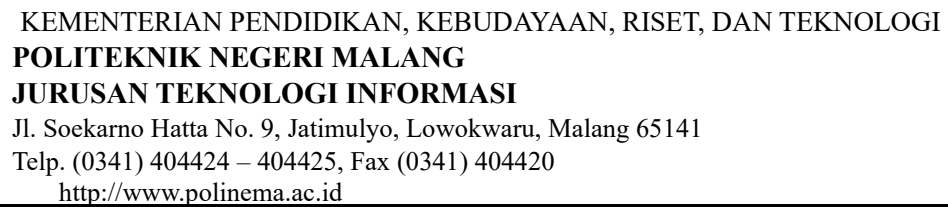


4. Lalu kita tambahkan routes pada api.php

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

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



-
- The screenshot displays the Swagger UI interface for a REST API. The top bar shows a POST method and the endpoint `localhost/PWL_POS/public/api/logout`. The 'Authorization' tab is selected, showing a 'Bearer Token' type and a long alphanumeric token value. Below this, a description states that the authorization header will be automatically generated. The 'Body' tab is also visible, showing a JSON response with 'success: true' and 'message: 'Logout Berhasil!''.

The screenshot displays the Swagger UI for a REST API. The top bar shows the URL 'localhost/PWL_2025/PWL_POS/public/api/logout' and a 'Send' button. The 'Params' tab is selected, showing a 'Bearer Token' field with a dropdown menu. The 'Token' field is highlighted with a yellow box, containing the value 'eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9'. Below the token field, a message states: 'The authorization header will be automatically generated when you send the request. Learn more about [Bearer Token](#) authorization.' The 'Body' tab is selected, showing the response body in JSON format: {"success": true, "message": "Logout Bezhas11!"}. The status bar at the bottom indicates a successful response with '200 OK', a response time of '236 ms', and a content size of '437 B'.

- ```
PS C:\laragon\www\PWL_2025\PWL_WEEK10\pwl_pos> php artisan make:controller Api/LevelController
```
- ```
INFO Controller [C:\laragon\www\PWL_2025\PWL_WEEK10\PWL_POS\app\Http\Controllers\Api\LevelController.php] created successfully.
```

- Hal. 16 / 28



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 index()
12     {
13         return LevelModel::all();
14     }
15
16     public function store(Request $request)
17     {
18         $level = LevelModel::create($request->all());
19         return response()->json($level, 201);
20     }
21
22     public function show(LevelModel $level)
23     {
24         return $level;
25     }
26
27     public function update(Request $request, LevelModel $level)
28     {
29         $level->update($request->all());
30         return $level;
31     }
32
33     public function destroy(LevelModel $level)
34     {
35         $level->delete();
36         return response()->json([
37             'success' => true,
38             'message' => 'Data terhapus',
39         ]);
40     }
41 }
```

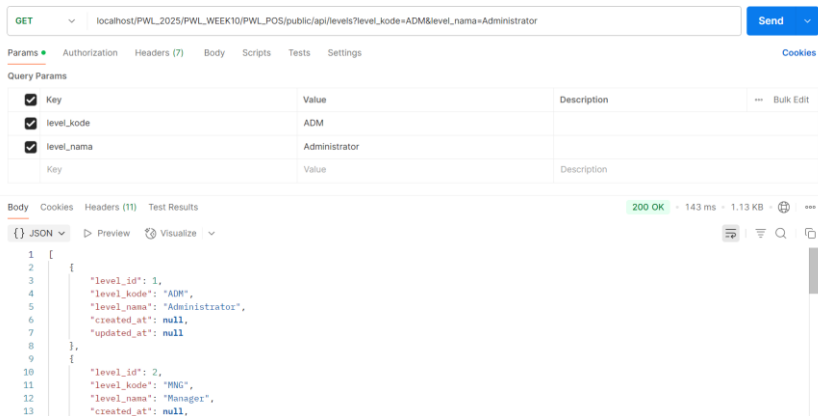
3. Kemudian kita lengkapi routes pada api.php.

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

```
1 // Route Level
2 Route::get('/levels', [LevelController::class, 'index']);
3 Route::post('/levels', [LevelController::class, 'store']);
4 Route::get('/levels/{level}', [LevelController::class, 'show']);
5 Route::put('/levels/{level}', [LevelController::class, 'update']);
6 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.**

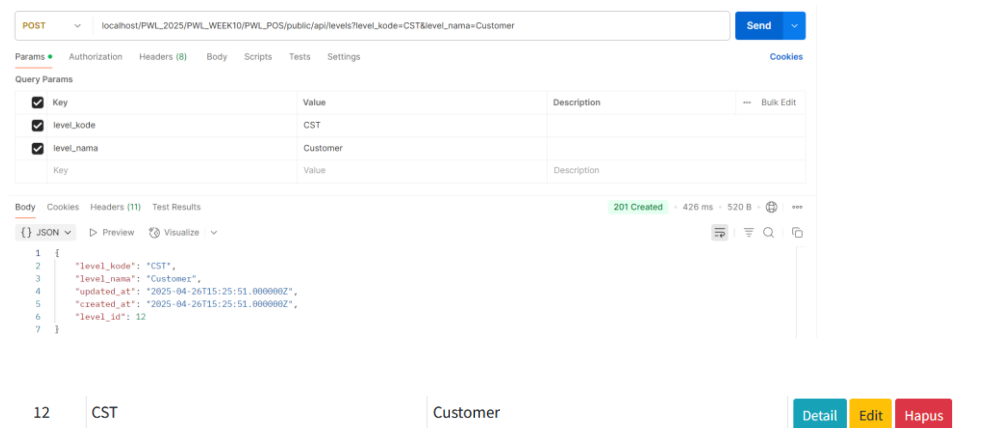


Penjelasan :

GET digunakan hanya untuk mengambil data dari server. Jika hanya untuk menampilkan data levelnya maka menggunakan **GET**.

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

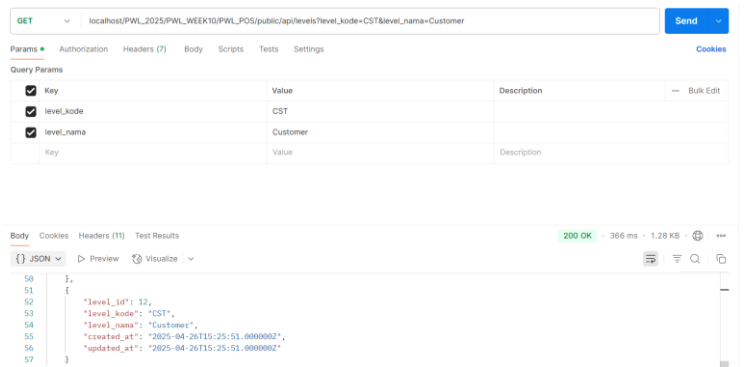
Jelaskan dan berikan screenshot hasil percobaan Anda.



Penjelasan :

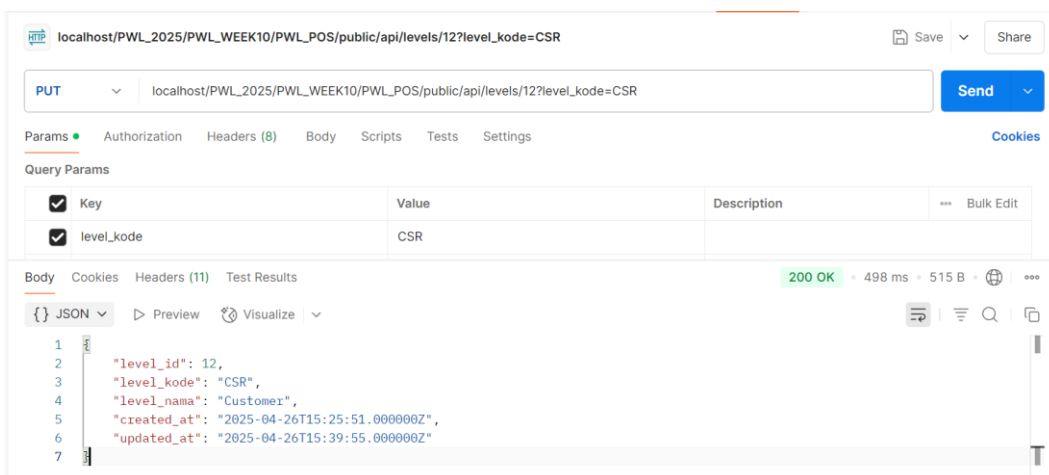
POST dirancang untuk mengirimkan data ke server guna membuat atau memperbarui sumber daya, seperti menambahkan level baru ke database.

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



Penjelasan : Untuk memanggil data sesuai ID maka digunakan **GET**

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 screenshot hasil percobaan Anda.**

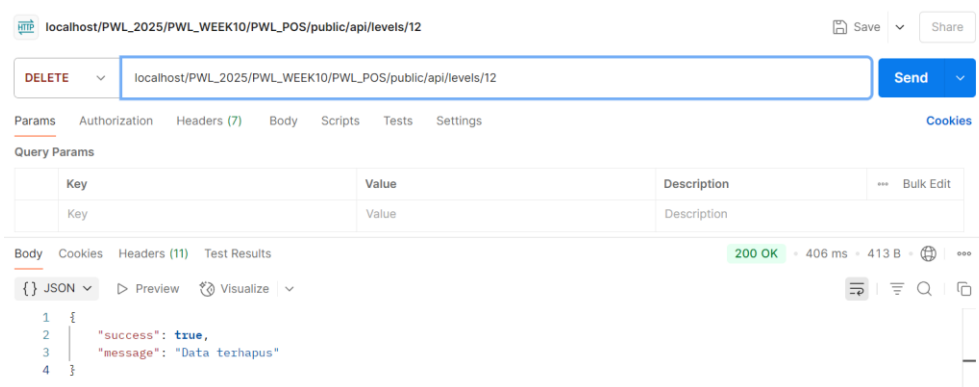


Penjelasan :

PUT digunakan untuk memperbarui data sesuai ID yang sudah ada di server

12	CSR	Customer	Detail Edit Hapus
----	-----	----------	---

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



Penejelasan :

DELETE digunakan untuk menghapus data atau resource (ID) tertentu di server.

9. Lakukan commit perubahan file pada Github.

TUGAS

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

1. Implementasi tabel m_user

- Membuat API/UserController

```
PS C:\laragon\www\PWL_2025\PWL_WEEK10\pwl_pos> php artisan make:controller Api/UserController
```

```
INFO Controller [C:\laragon\www\PWL_2025\PWL_WEEK10\PWL_POS\app\Http\Controllers\Api\UserController.php] created successfully.
```

- Menambahkan pada API/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()
12     {
13         return UserModel::all();
14     }
15
16     public function store(Request $request)
17     {
18         $user = UserModel::create($request->all());
19         return response()->json($user, 201);
20     }
21
22     public function show(UserModel $user)
23     {
24         return $user;
25     }
26
27     public function update(Request $request, UserModel $user)
28     {
29         $user->update($request->all());
30         return $user;
31     }
32
33     public function destroy(UserModel $user)
34     {
35         $user->delete();
36         return response()->json([
37             'success' => true,
38             'message' => 'Data terhapus',
39         ]);
40     }
41 }
```

- Menambahkan pada api.php

```
1 use App\Http\Controllers\Api\UserController;
```

```
1 // Route User
2 Route::get('/users', [UserController::class, 'index']);
3 Route::post('/users', [UserController::class, 'store']);
4 Route::get('/users/{user}', [UserController::class, 'show']);
5 Route::put('/users/{user}', [UserController::class, 'update']);
6 Route::delete('/users/{user}', [UserController::class, 'destroy']);
```

- Implementasi Create - POST



localhost/PWL_2025/PWL_WEEK10/PWL_POS/public/api/users?level_id=2&username=Manager Lagi&nama=Manager Lagi&password=12345

POST

Query Params

Key	Value	Description
level_id	2	
username	Manager Lagi	
nama	Manager Lagi	
password	12345	

Body

```
{
  "level_id": "2",
  "username": "Manager Lagi",
  "nama": "Manager Lagi",
  "updated_at": "2025-04-26T17:28:41.000000Z",
  "created_at": "2025-04-26T17:28:41.000000Z",
  "user_id": 19
}
```

➤ Implementasi Read - GET

localhost/PWL_2025/PWL_WEEK10/PWL_POS/public/api/users?username=manager lagi&nama=manager lagi

GET

Query Params

Key	Value	Description
username	manager lagi	
nama	manager lagi	

Body

```
{
  "nama": "manager lagi",
  "created_at": "2025-04-26T13:35:23.000000Z",
  "updated_at": "2025-04-26T13:35:23.000000Z",
  "user_id": 19,
  "level_id": 2,
  "profile_picture": null,
  "username": "Manager Lagi",
  "nama": "Manager Lagi",
  "created_at": "2025-04-26T17:28:41.000000Z",
  "updated_at": "2025-04-26T17:28:41.000000Z"
}
```

➤ Implementasi Update - PUT

localhost/PWL_2025/PWL_WEEK10/PWL_POS/public/api/users/19?nama=lala

PUT

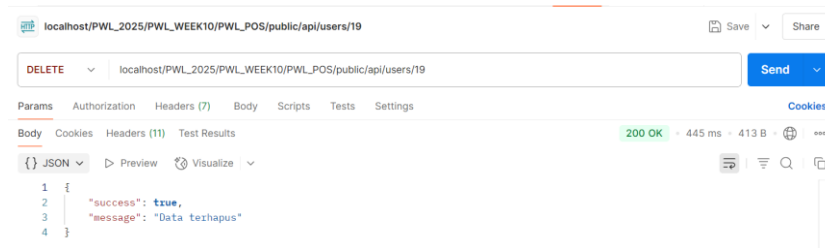
Query Params

Key	Value	Description
nama	lala	

Body

```
{
  "user_id": 19,
  "level_id": 2,
  "profile_picture": null,
  "username": "Manager Lagi",
  "nama": "lala",
  "created_at": "2025-04-26T17:28:41.000000Z",
  "updated_at": "2025-04-26T17:33:07.000000Z"
}
```

➤ Implementasi Delete - DELETE



2. Implementasi tabel m_kategori

- Membuat API/KategoriController

```
PS C:\laragon\www\PWL_2025\PWL_WEEK10\pwl_pos> php artisan make:controller Api/KategoriController
[INFO] Controller [C:\laragon\www\PWL_2025\PWL_WEEK10\PWL_POS\app\Http\Controllers\Api\KategoriController.php] created successfully.
```

- Menambahkan pada API/KategoriController

```
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()
12     {
13         return KategoriModel::all();
14     }
15
16     public function store(Request $request)
17     {
18         $kategori = KategoriModel::create($request->all());
19         return response()->json($kategori, 201);
20     }
21
22     public function show(KategoriModel $kategori)
23     {
24         return $kategori;
25     }
26
27     public function update(Request $request, KategoriModel $kategori)
28     {
29         $kategori->update($request->all());
30         return $kategori;
31     }
32
33     public function destroy(KategoriModel $kategori)
34     {
35         $kategori->delete();
36         return response()->json([
37             'success' => true,
38             'message' => 'Data kategori terhapus',
39         ]);
40     }
41 }
```

- Menambahkan pada api.php

```
1 use App\Http\Controllers\Api\KategoriController;
```



```
1 // Route Kategori
2 Route::get('/kategoris', [KategoriController::class, 'index']);
3 Route::post('/kategoris', [KategoriController::class, 'store']);
4 Route::get('/kategoris/{kategori}', [KategoriController::class, 'show']);
5 Route::put('/kategoris/{kategori}', [KategoriController::class, 'update']);
6 Route::delete('/kategoris/{kategori}', [KategoriController::class, 'destroy']);
```

➤ Implementasi Create - POST

localhost/PWL_2025/PWL_WEEK10/PWL_POS/public/api/kategoris?kategori_kode=LDI&kategori_nama=Ladies

POST localhost/PWL_2025/PWL_WEEK10/PWL_POS/public/api/kategoris?kategori_kode=LDI&kategori_nama=Ladies

Params Authorization Headers (8) Body Scripts Tests Settings Cookies

Key	Value	Description
kategori_kode	LDI	
kategori_nama	Ladies	

Body Cookies Headers (11) Test Results 201 Created 712 ms 527 B

JSON Preview Visualize

```
1 {
2   "kategori_kode": "LDI",
3   "kategori_nama": "Ladies",
4   "updated_at": "2025-04-26T17:43:57.000000Z",
5   "created_at": "2025-04-26T17:43:57.000000Z",
6   "kategori_id": 23
7 }
```

➤ Implementasi Read - GET

localhost/PWL_2025/PWL_WEEK10/PWL_POS/public/api/kategoris?kategori_kode=LDI

GET localhost/PWL_2025/PWL_WEEK10/PWL_POS/public/api/kategoris?kategori_kode=LDI

Params Authorization Headers (7) Body Scripts Tests Settings Cookies

Query Params

Key	Value	Description
kategori_kode	LDI	

Body Cookies Headers (11) Test Results 200 OK 446 ms 1.66 KB

JSON Preview Visualize

```
64 [
65   {
66     "kategori_id": 21,
67     "kategori_kode": "BKJ",
68     "kategori_nama": "Books & Stationery",
69     "created_at": "2025-04-08T04:43:41.000000Z",
70     "updated_at": null
71   },
72   {
73     "kategori_id": 23,
74     "kategori_kode": "LDI",
75     "kategori_nama": "Ladies",
76     "created_at": "2025-04-26T17:43:57.000000Z",
77     "updated_at": "2025-04-26T17:43:57.000000Z"
78   }
79 ]
```

➤ Implementasi Update - PUT



localhost/PWL_2025/PWL_WEEK10/PWL_POS/public/api/kategoris/23?kategori_kode=LDY

PUT localhost/PWL_2025/PWL_WEEK10/PWL_POS/public/api/kategoris/23?kategori_kode=LDY

Params Authorization Headers (8) Body Scripts Tests Settings Cookies

Query Params

Key	Value	Description	Bulk Edit
kategori_kode	LDY		
Key	Value	Description	

Body Cookies Headers (11) Test Results 200 OK 454 ms 522 B

JSON Preview Visualize

```
1 {
2   "kategori_id": 23,
3   "kategori_kode": "LDY",
4   "kategori_nama": "Ladies",
5   "created_at": "2025-04-26T17:43:57.000000Z",
6   "updated_at": "2025-04-26T17:47:08.000000Z"
7 }
```

➤ Implementasi Delete - DELETE

localhost/PWL_2025/PWL_WEEK10/PWL_POS/public/api/kategoris/23

DELETE localhost/PWL_2025/PWL_WEEK10/PWL_POS/public/api/kategoris/23

Params Authorization Headers (7) Body Scripts Tests Settings Cookies

Query Params

Key	Value	Description	Bulk Edit
Key	Value	Description	

Body Cookies Headers (11) Test Results 200 OK 458 ms 422 B

JSON Preview Visualize

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

3. Implementasi tabel m_barang

➤ Membuat API/BarangController

```
PS C:\laragon\www\PWL_2025\PWL_WEEK10\pwl_pos> php artisan make:controller Api/BarangController
```

```
INFO Controller [C:\laragon\www\PWL_2025\PWL_WEEK10\PWL_POS\app\Http\Controllers\Api\BarangController.php] created successfully.
```

➤ Menambahkan pada API/BarangController



```
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()
12     {
13         return BarangModel::all();
14     }
15
16     public function store(Request $request)
17     {
18         $barang = BarangModel::create($request->all());
19         return response()->json($barang, 201);
20     }
21
22     public function show(BarangModel $barang)
23     {
24         return $barang;
25     }
26
27     public function update(Request $request, BarangModel $barang)
28     {
29         $barang->update($request->all());
30         return $barang;
31     }
32
33     public function destroy(BarangModel $barang)
34     {
35         $barang->delete();
36         return response()->json([
37             'success' => true,
38             'message' => 'Data barang terhapus',
39         ]);
40     }
41 }
42
```

- Menambahkan pada api.php

```
1 use App\Http\Controllers\Api\BarangController;
```

```
1 // Route Barang
2 Route::get('/barangs', [BarangController::class, 'index']);
3 Route::post('/barangs', [BarangController::class, 'store']);
4 Route::get('/barangs/{barang}', [BarangController::class, 'show']);
5 Route::put('/barangs/{barang}', [BarangController::class, 'update']);
6 Route::delete('/barangs/{barang}', [BarangController::class, 'destroy']);
7
```

- Implementasi Create – POST



localhost/PWL_2025/PWL_WEEK10/PWL_POS/public/api/barangs?kategori_id=1&barang_nama=Kopi bubuk&harga_beli=4000&ha...

POST localhost/PWL_2025/PWL_WEEK10/PWL_POS/public/api/barangs?kategori_id=1&barang_nama=Kopi bubuk&harga_beli=4000&ha...

Params Authorization Headers (8) Body Scripts Tests Settings Cookies

Key	Value	Description
kategori_id	1	
barang_nama	Kopi bubuk	
harga_beli	4000	
harga_jual	5000	
barang_kode	B018	

Body Cookies Headers (11) Test Results 201 Created · 466 ms · 584 B

```
{
  "kategori_id": "1",
  "barang_nama": "Kopi bubuk",
  "harga_beli": "4000",
  "harga_jual": "5000",
  "barang_kode": "B018",
  "updated_at": "2025-04-26T18:03:53.000000Z",
  "created_at": "2025-04-26T18:03:53.000000Z",
  "barang_id": 35
}
```

➤ Implementasi Read – GET

localhost/PWL_2025/PWL_WEEK10/PWL_POS/public/api/barangs?barang_kode=B018

GET localhost/PWL_2025/PWL_WEEK10/PWL_POS/public/api/barangs?barang_kode=B018

Params Authorization Headers (7) Body Scripts Tests Settings Cookies

Query Params

Key	Value	Description
barang_kode	B018	

Body Cookies Headers (11) Test Results 200 OK · 431 ms · 3.36 KB

```
{
  "barang_id": 35,
  "kategori_id": 1,
  "barang_kode": "B018",
  "barang_nama": "Kopi bubuk",
  "harga_beli": 4000,
  "harga_jual": 5000,
  "created_at": "2025-04-26T18:03:53.000000Z",
  "updated_at": "2025-04-26T18:03:53.000000Z"
}
```

➤ Implementasi Update – PUT

localhost/PWL_2025/PWL_WEEK10/PWL_POS/public/api/barangs/35?barang_kode=B0018

PUT localhost/PWL_2025/PWL_WEEK10/PWL_POS/public/api/barangs/35?barang_kode=B0018

Params Authorization Headers (8) Body Scripts Tests Settings Cookies

Query Params

Key	Value	Description
barang_kode	B0018	

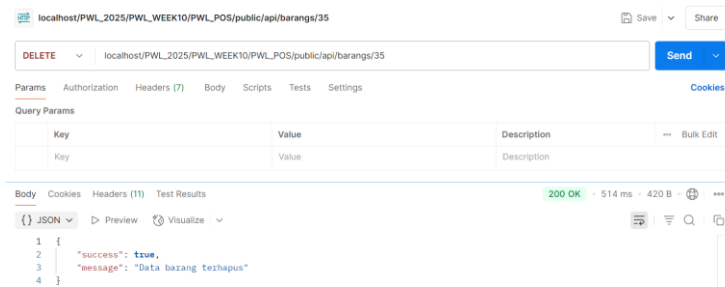
Body Cookies Headers (11) Test Results 200 OK · 466 ms · 574 B

```
{
  "barang_id": 35,
  "kategori_id": 1,
  "barang_kode": "B0018",
  "barang_nama": "Kopi bubuk",
  "harga_beli": 4000,
  "harga_jual": 5000,
  "created_at": "2025-04-26T18:03:53.000000Z",
  "updated_at": "2025-04-26T18:09:44.000000Z"
}
```

➤ Implementasi Delete – DELETE



KEMENTERIAN PENDIDIKAN, KEBUDAYAAN, RISET, DAN TEKNOLOGI
POLITEKNIK NEGERI MALANG
JURUSAN TEKNOLOGI INFORMASI
Jl. Soekarno Hatta No. 9, Jatimulyo, Lowokwaru, Malang 65141
Telp. (0341) 404424 – 404425, Fax (0341) 404420
<http://www.polinema.ac.id>



**** Sekian, dan selamat belajar ****