

# **LAPORAN PRAKTIKUM**

## **JOBSHEET X: RESTFUL API (*APPLICATION PROGRAMMING INTERFACE*)**

### **MATA KULIAH PEMROGRAMAN WEB LANJUT**

Dosen Pengampu : Dimas Wahyu Wibowo, S.T., M.T.



**Disusun oleh :**

Nama : Vita Eka Saraswati

NIM : 2341760082

No. Absen : 29

**JURUSAN TEKNOLOGI INFORMASI**

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

**POLITEKNIK NEGERI MALANG**

**2025**



Nama : Vita Eka Saraswati  
NIM/Kelas : 2341760082 / SIB 2A  
No. Absen : 29  
Mata Kuliah : Pemrograman Web Lanjut (PWL)

## 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 dipercayai.

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.

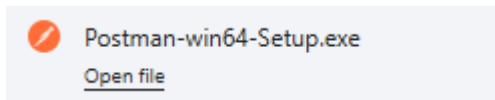
Berikut adalah cara kerja JWT :

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 F:\laragon\www\Semester 4\PWL25\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
Nothing to modify in lock file
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-dumpdf ..... 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

92 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.
```

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

```
php artisan vendor:publish --
```

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

```
PS F:\laragon\www\Semester 4\PWL25\Minggu10\PWL_POS> php artisan vendor:publish --provider="Tymon\JWTAuth\Providers\LaravelServiceProvider"

INFO: Publishing assets.

Copying file [F:\laragon\www\Semester 4\PWL25\Minggu10\PWL_POS\vendor\tymon\jwt-auth\config\config.php] to [F:\laragon\www\Semester 4\PWL25\Minggu10\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.
5. Setelah itu jalankan perintah berikut untuk membuat secret key JWT.



## php artisan jwt:secret

```
PS F:\laragon\www\Semester 4\PWL25\Minggu10\PWL_POS> php artisan jwt:secret  
jwt-auth secret [xXcwTkWBg8g0KVeiac9d5LL49HiGznJ1RCfDJRgZoa9yWUjNPNk5VLnSLcVuw3I] set successfully.
```

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

```
61 JWT_SECRET=xXcwTkWBg8g0KVeiac9d5LL49HiGznJ1RCfDJRgZoa9yWUjNPNk5VLnSLcVuw3I  
62
```

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

```
38 'guards' => [  
39     'web' => [  
40         'driver' => 'session',  
41         'provider' => 'users',  
42     ],  
43     'api' => [  
44         'driver' => 'jwt',  
45         'provider' => 'users',  
46     ],  
47 ],
```

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

```
Minggu10 > PWL_POS > app > Models > UserModel.php > PHP > UserModel > getJWTIdentifier()  
1 <?php  
2  
3 namespace App\Models;  
4  
5 use Illuminate\Database\Eloquent\Factories\HasFactory;  
6 use Illuminate\Database\Eloquent\Model;  
7 use Illuminate\Database\Eloquent\Relations\BelongsTo;  
8 use App\Models\LevelModel;  
9 use Tymon\JWTAuth\Contracts\JWTSubject;  
10 use Illuminate\Foundation\Auth\User as Authenticatable; //implementasi class Authenticatable  
11  
12 145 references | 0 implementations  
13 class UserModel extends Authenticatable implements JWTSubject  
14 {  
15     use HasFactory;  
16  
17 0 references | 0 overrides  
18     public function getJWTIdentifier(): mixed  
19     {  
20         return $this->getKey();  
21     }  
22  
23 0 references | 0 overrides  
24     public function getJWTCustomClaims(): array  
25     {  
26         return [];  
27     }  
28  
29 0 references  
30     protected $table = 'm_user'; // Mendefinisikan nama tabel yang digunakan oleh model ini  
31 }  
32  
33 0 references
```

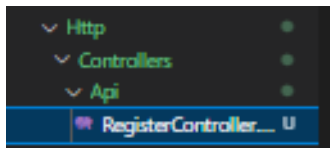


8. Berikutnya kita akan membuat controller untuk register dengan menjalankan perintah berikut.

`php artisan make:controller Api/RegisterController`

```
PS F:\laragon\www\Semester 4\PWL25\Minggu10\PWL_POS> php artisan make:controller Api/RegisterController  
INFO Controller [F:\laragon\www\Semester 4\PWL25\Minggu10\PWL_POS\app\Http\Controllers\Api\RegisterController.php] created successfully.
```

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



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

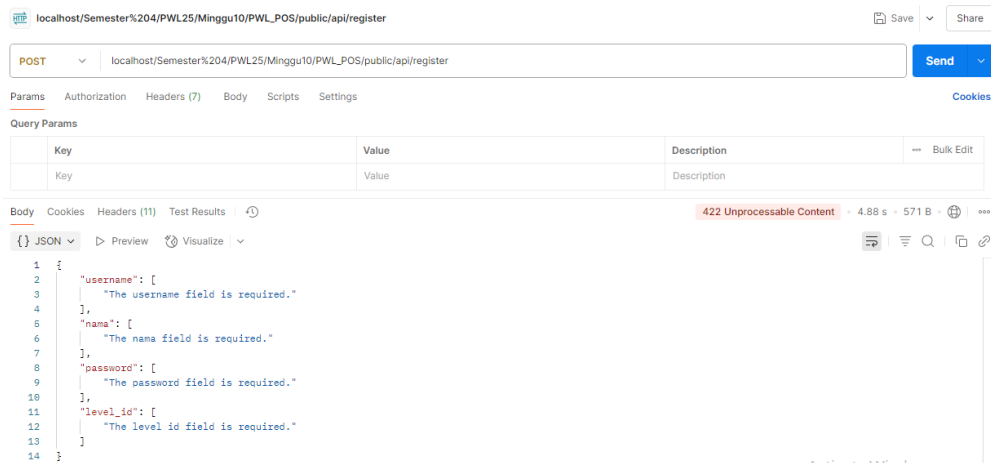
```
Minggu10 > PWL_POS > app > Http > Controllers > Api > RegisterController.php > PHP > RegisterController  
1 <?php  
2  
3 namespace App\Http\Controllers\Api;  
4  
5 use App\Models\UserModel;  
6 use App\Http\Controllers\Controller;  
7 use Illuminate\Http\Request;  
8 use Illuminate\Support\Facades\Validator;  
9  
10 references | 0 implementations  
11 class RegisterController extends Controller  
12  
13 references | 0 overrides  
14 public function __invoke(Request $request): JsonResponse|mixed  
15 {  
16     // Set validation  
17     $validator = Validator::make(data: $request->all(), rules: [  
18         'username' => 'required',  
19         'nama' => 'required',  
20         'password' => 'required|min:5|confirmed',  
21         'level_id' => 'required',  
22     ]);  
23  
24     // If validation fails  
25     if ($validator->fails()) {  
26         return response()->json(data: $validator->errors(), status: 422);  
27     }  
28  
29     // Create user  
30     $user = UserModel::create(attributes: [  
31         'username' => $request->username,  
32         'nama' => $request->nama,  
33         'password' => bcrypt(value: $request->password),  
34         'level_id' => $request->level_id,  
35     ]);  
36  
37     // Return response JSON if user is created  
38     if ($user) {  
39         return response()->json(data: [  
40             'success' => true,  
41             'user' => $user,  
42         ], status: 201);  
43     }  
44  
45     // Return JSON if process insert failed  
46     return response()->json(data: [  
47         'success' => false,  
48     ], status: 409);  
49 }  
50
```



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

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

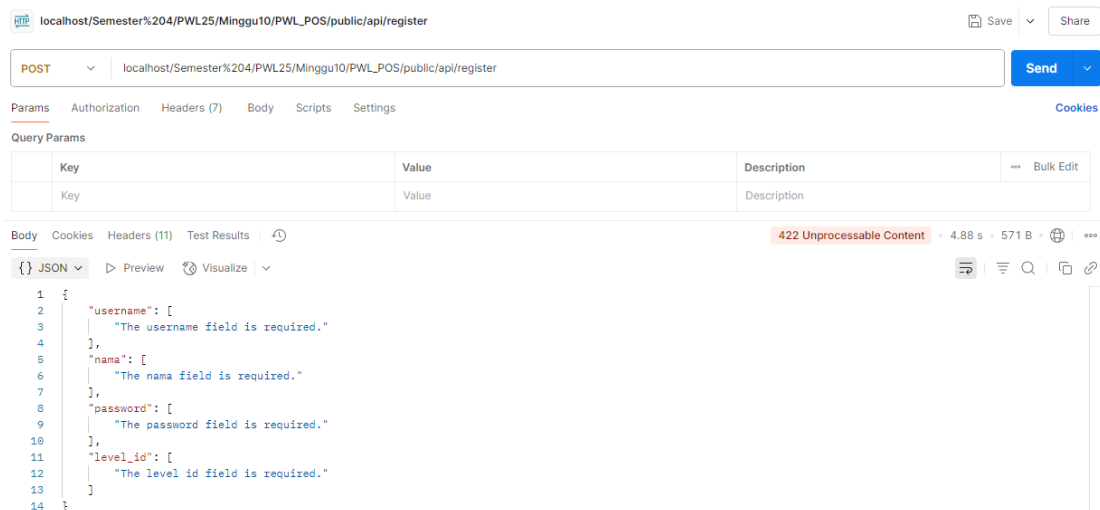
11. Jika sudah, kita akan melakukan uji coba REST API melalui aplikasi Postman.



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.

POST localhost/PWL\_POS/public/api/register

Params Auth Headers (8) **Body** Pre-req. Tests Settings Cookies

form-data

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

Body Cookies Headers (11) Test Results 201 Created 624 ms 645 B Save Response

Pretty Raw Preview Visualize JSON

```
1 {
2   "success": true,
3   "user": {
4     "username": "penggunasatu",
5     "nama": "Pengguna 1",
6     "password": "$2y$12$Eb25rV1jsyKINytYGtzH10VAKcK5pEgnZnmbChkPicIu750QJJu",
7     "level_id": "2",
8     "updated_at": "2024-04-22T15:56:04.000000Z",
9     "created_at": "2024-04-22T15:56:04.000000Z",
10    "user_id": 17
11  }
12 }
```

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.

localhost/Semester%204/PWL25/Minggu10/PWL\_POS/public/api/register

POST localhost/Semester%204/PWL25/Minggu10/PWL\_POS/public/api/register

Params Authorization Headers (8) **Body** Scripts Settings Cookies

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

Key	Value	Description
username	penggunasatu	
nama	Pengguna1	
password	12345	
password_confirmation	12345	
level_id	2	

Body Cookies Headers (11) Test Results 201 Created 9.38 s 560 B

JSON Preview Visualize

```
1 {
2   "success": true,
3   "user": {
4     "username": "penggunasatu",
5     "nama": "Pengguna1",
6     "level_id": "2",
7     "updated_at": "2025-04-27T04:48:21.000000Z",
8     "created_at": "2025-04-27T04:48:21.000000Z",
9     "user_id": 34
10  }
11 }
```

Daftar Pengguna

Filter: Semua

Show 10 entries

No	ID Pengguna	Username	Nama	Level	Aksi
11	31	staff01	Vita	Staff	<a href="#">Detail</a> <a href="#">Edit</a> <a href="#">Hapus</a>
12	32	staff02	Shinta	Staff	<a href="#">Detail</a> <a href="#">Edit</a> <a href="#">Hapus</a>
13	34	penggunasatu	Pengguna1	Manager	<a href="#">Detail</a> <a href="#">Edit</a> <a href="#">Hapus</a>

Showing 11 to 13 of 13 entries

Previous 1 2 Next

13. Lakukan commit perubahan file pada Github.





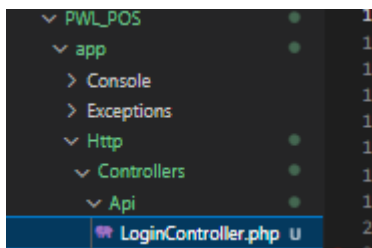
## Praktikum 2 – Membuat RESTful API Login

1. Kita buat file controller dengan nama LoginController.

`php artisan make:controller Api/LoginController`

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

```
PS F:\laragon\www\Semester 4\PWL25\Minggu10\PWL_POS> php artisan make:controller Api/LoginController  
INFO Controller [F:\laragon\www\Semester 4\PWL25\Minggu10\PWL_POS\app\Http\Controllers\Api>LoginController.php] created successfully.
```



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

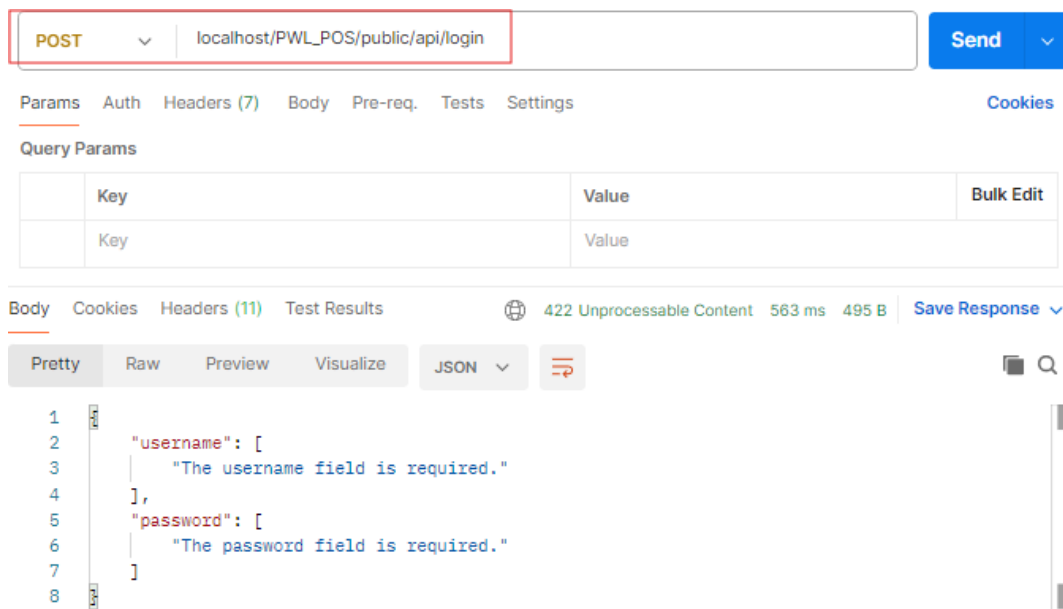
```
Minggu10 > PWL_POS > app > Http > Controllers > Api > LoginController.php > PHP > LoginController > _invoke()
1  <?php
2
3  namespace App\Http\Controllers\Api;
4
5  use App\Http\Controllers\Controller;
6  use Illuminate\Http\Request;
7  use Illuminate\Support\Facades\Validator;
8
9  0 references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



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

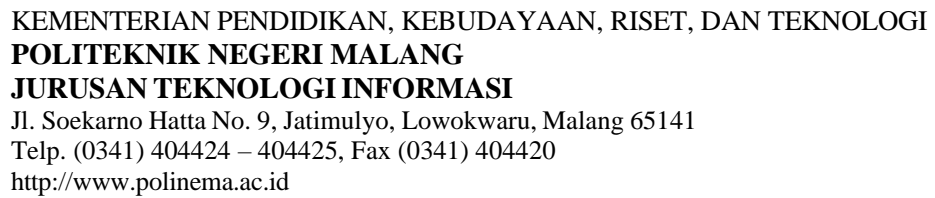
```
Minggu10 > PWL_POS > routes > api.php > _
1  <?php
2
3  use App\Http\Controllers\Api\RegisterController;
4  use App\Http\Controllers\Api>LoginController;
5  use Illuminate\Http\Request;
6  use Illuminate\Support\Facades\Route;
7
8  /*
9  |-----
10 | API Routes
11 |-----
12 |
13 | Here is where you 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(uri: '/register', action: App\Http\Controllers\Api\RegisterController::class)->name(name: 'register');
20 Route::post(uri: '/login', action: App\Http\Controllers\Api>LoginController::class)->name(name: 'login');
21 Route::middleware(middleware: 'auth:api')->get(uri: '/user', action: function (Request $request): mixed {
22     return $request->user();
23 });
```

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 screenshot hasil percobaan Anda.



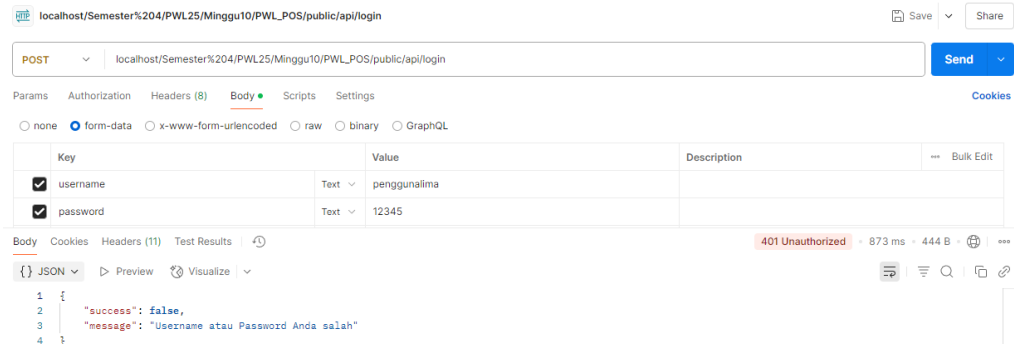
5. Selanjutnya, isikan username dan password sesuai dengan data user yang ada pada database. Klik tab Body dan pilih form-data. Isikan key sesuai dengan kolom data, serta isikan data user. Klik tombol Send, jika berhasil maka akan keluar tampilan seperti berikut. Copy nilai token yang diperoleh pada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

Lakukan percobaan yang sama dan berikan screenshoot hasil percobaan Anda.

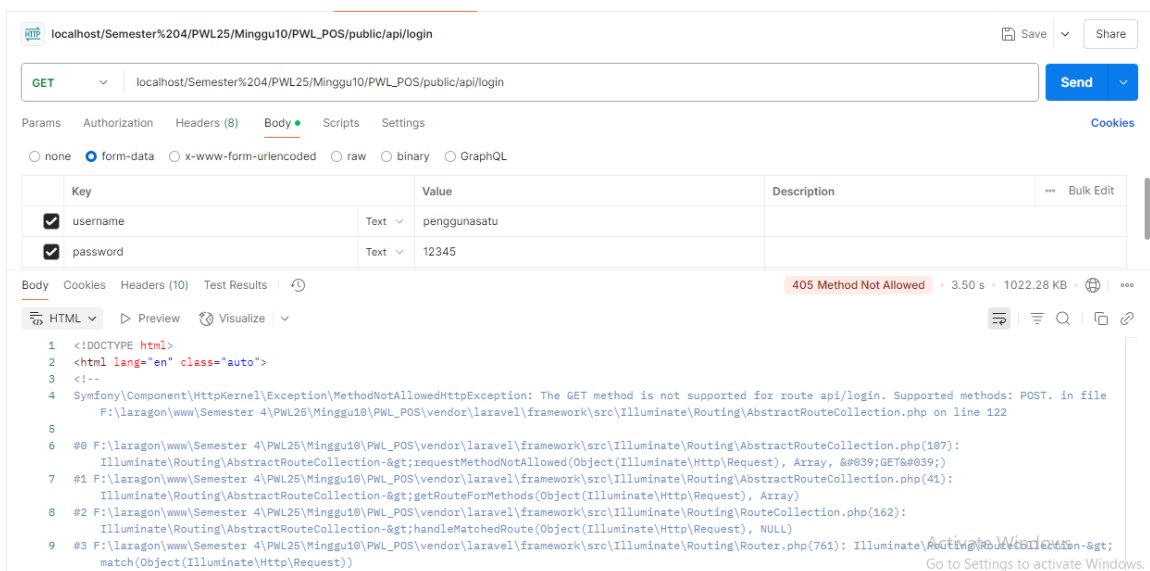
Jobsheet 10 – PWL 2023/2024



6. Lakukan percobaan yang untuk data yang salah dan berikan screenshoot hasil percobaan Anda.



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.



**Jawab :** Method GET tidak dapat digunakan untuk API login ditandai dengan error message 405 Method Not Allowed. Route dalam api.php login menggunakan method post dan hanya mendukung method tersebut.

```
Route::post(uri: '/login', action: App\Http\Controllers\Api\LoginController::class->name(name: 'login'));
```

Sehingga saat kita mencoba menggunakan metode GET, Laravel otomatis mengembalikan error 405 Method Not Allowed, karena method tidak didukung untuk rute api/login

8. Lakukan commit perubahan file pada Github.



### Praktikum 3 – Membuat RESTful API Logout

1. Tambahkan kode berikut pada file .env

`JWT_SHOW_BLACKLIST_EXCEPTION=true`

```
60
61 JWT_SECRET=xXcwTkW8eg8g0KVeIac9d5LL49HiGznJ1RCfDJRgZoa9yWUjNPNk5VLn5LcVuw3I
62 JWT_SHOW_BLACKLIST_EXCEPTION=true
```

2. Buat Controller baru dengan nama LogoutController.

`php artisan make:controller Api/LogoutController`

```
PS F:\laragon\www\Semester 4\PWL25\Minggu10\PWL_POS> php artisan make:controller Api/LogoutController
[INFO] Controller [F:\laragon\www\Semester 4\PWL25\Minggu10\PWL_POS\app\Http\Controllers\Api\LogoutController.php] created successfully.
```

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

```
Minggu10 > PWL_POS > app > Http > Controllers > Api > LogoutController.php > PHP > LogoutController

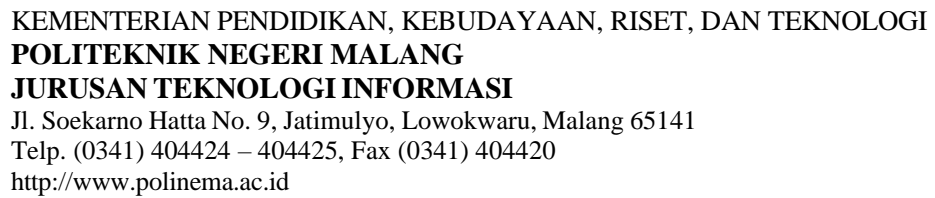
1 <?php
2
3 namespace App\Http\Controllers\Api;
4
5 use Illuminate\Http\Request;
6 use App\Http\Controllers\Controller;
7 use Tymon\JWTAuth\Facades\JWTAuth;
8 use Tymon\JWTAuth\Exceptions\JWTException;
9 use Tymon\JWTAuth\Exceptions\TokenExpiredException;
10 use Tymon\JWTAuth\Exceptions\TokenInvalidException;
11
12 0 references | 0 implementations
13 class LogoutController extends Controller
14
15 0 references | 0 overrides
16 public function __invoke(Request $request): JsonResponse|mixed
17 {
18     // remove token
19     $removeToken = JWTAuth::invalidate(JWTAuth::getToken());
20
21     if ($removeToken) {
22         // return response JSON
23         return response()->json(data: [
24             'success' => true,
25             'message' => 'Logout Berhasil!',
26         ]);
27     }
28 }
```

4. Lalu kita tambahkan routes pada api.php

```
5 use App\Http\Controllers\Api\LogoutController;
6
```

```
21 Route::post(uri: '/logout', action: App\Http\Controllers\Api\LogoutController::class)->name(name: 'logout');
22
```

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 the method 'POST' and the URL 'localhost/PWL\_POS/public/api/logout'. The 'Authorization' tab is selected, showing 'Type' as 'Bearer Token' and a 'Token' field with a Bearer token. Below this, a note states: 'The authorization header will be automatically generated when you send the request. [Learn more about authorization](#)'. The bottom section shows the 'Response' tab with a status of '200 OK', a time of '578 ms', and a size of '447 B'. The response body is a JSON object: `{"success": true, "message": "Logout Berhasil."}`.

- Hal. 13 / 25



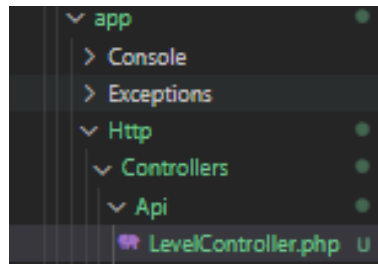
## Praktikum 4 – Implementasi CRUD dalam RESTful API

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`

```
PS F:\laragon\www\Semester 4\PWL25\Minggu10\PWL_POS> php artisan make:controller Api/LevelController
INFO Controller [F:\laragon\www\Semester 4\PWL25\Minggu10\PWL_POS\app\Http\Controllers\Api\LevelController.php] created successfully.
```



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

```
Minggu10 > PWL_POS > app > Http > Controllers > Api > LevelController.php > PHP > LevelController

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     0 references | 0 overrides
12     public function index(): Collection
13     {
14         return LevelModel::all();
15     }
16
17     0 references | 0 overrides
18     public function store(Request $request): JsonResponse|mixed
19     {
20         $level = LevelModel::create(attributes: $request->all());
21         return response()->json(data: $level, status: 201);
22     }
23
24     0 references | 0 overrides
25     public function show(LevelModel $level): LevelModel
26     {
27         return $level;
28     }
29
30     0 references | 0 overrides
31     public function update(Request $request, LevelModel $level): LevelModel
32     {
33         $level->update(attributes: $request->all());
34         return $level;
35     }
36
37     0 references | 0 overrides
38     public function destroy(LevelModel $level): JsonResponse|mixed
39     {
40         $level->delete();
41         return response()->json(data: [
42             'success' => true,
43             'message' => 'Data terhapus',
44         ]);
45     }
46 }
```





3. Kemudian kita lengkapi routes pada api.php.

```
6 use App\Http\Controllers\Api\LevelController;  
  
30 Route::get(uri: '/levels', action: [LevelController::class, 'index']);  
31 Route::post(uri: '/levels', action: [LevelController::class, 'store']);  
32 Route::get(uri: '/levels/{level}', action: [LevelController::class, 'show']);  
33 Route::put(uri: '/levels/{level}', action: [LevelController::class, 'update']);  
34 Route::delete(uri: '/levels/{level}', action: [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.**

The screenshot shows a REST client interface with the following details:

- URL: `localhost/Semester%204/PWL25/Minggu10/PWL_POS/public/api/levels`
- Method: GET
- Status: 200 OK
- Response Body (JSON):

```
[  
  {  
    "level_id": 1,  
    "level_kode": "ADM",  
    "level_nama": "Administrator",  
    "created_at": null,  
    "updated_at": "2025-04-13T00:57:02.000000Z"  
  },  
  {  
    "level_id": 2,  
    "level_kode": "MNG",  
    "level_nama": "Manager",  
    "created_at": null,  
    "updated_at": null  
  },  
  {  
    "level_id": 3,  
    "level_kode": "STF",  
    "level_nama": "Staff",  
    "created_at": null,  
    "updated_at": null  
  }  
]
```

Below the first screenshot, there is another screenshot showing a GET request to `localhost/Semester%204/PWL25/Minggu10/PWL_POS/public/api/levels?level_kode=ADM&level_nama=Administrator`. The response is also 200 OK, but the response body is empty, indicating that the filter is working correctly.

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





POST localhost/PWL\_POS/public/api/levels

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

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

Key	Value	Description	...	Bulk Edit
<input checked="" type="checkbox"/> level_kode	Text SPV			
<input checked="" type="checkbox"/> level_nama	Text Supervisor			
Key	Text Value	Description		

Body Cookies Headers (11) Test Results Status: 201 Created Time: 276 ms Size: 531 B Save as example

Pretty Raw Preview Visualize JSON

```
1 {
2   "level_kode": "SPV",
3   "level_nama": "Supervisor",
4   "updated_at": "2024-04-22T21:40:32.000000Z",
5   "created_at": "2024-04-22T21:40:32.000000Z",
6   "level_id": 4
7 }
```

Jelaskan dan berikan screenshoot hasil percobaan Anda.

localhost/Semester%204/PWL25/Minggu10/PWL\_POS/public/api/levels

POST localhost/Semester%204/PWL25/Minggu10/PWL\_POS/public/api/levels

Params Auth Headers (7) Body Scripts Settings Cookies

Query Params

Key	Value	Description	...	Bulk Edit
<input checked="" type="checkbox"/> level_kode	DIR			
<input checked="" type="checkbox"/> level_nama	Direktur			
Key	Value	Description		

Body JSON Preview Visualize 201 Created 2.66 s 520 B

```
1 {
2   "level_kode": "DIR",
3   "level_nama": "Direktur",
4   "updated_at": "2025-04-27T06:41:05.000000Z",
5   "created_at": "2025-04-27T06:41:05.000000Z",
6   "level_id": 11
7 }
```

Data berhasil ditambahkan

11	DIR	Direktur	<a href="#">Detail</a> <a href="#">Edit</a> <a href="#">Hapus</a>
----	-----	----------	---

Showing 1 to 8 of 8 entries

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

localhost/Semester%204/PWL25/Minggu10/PWL\_POS/public/api/levels?level\_kode=DIR&level\_nama=Direktur

GET localhost/Semester%204/PWL25/Minggu10/PWL\_POS/public/api/levels?level\_kode=DIR&level\_nama=Direktur

Params Authorization Headers (8) Body Scripts Settings Cookies

Query Params

Key	Value	Description	...	Bulk Edit
<input checked="" type="checkbox"/> level_kode	DIR			
<input checked="" type="checkbox"/> level_nama	Direktur			
Key	Value	Description		

Body Cookies Headers (11) Test Results 200 OK 1.02 s 1.3 KB

```
47 [
48   {
49     "level_kode": "IT Support Staff",
50     "level_nama": "IT Support Staff",
51     "created_at": "2025-03-29T06:03:30.000000Z",
52     "updated_at": null
53   },
54   {
55     "level_id": 11,
56     "level_kode": "DIR",
57     "level_nama": "Direktur",
58     "created_at": "2025-04-27T06:41:05.000000Z",
59     "updated_at": "2025-04-27T06:41:05.000000Z"
60   }
61 ]
```



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.

PUT localhost/PWL\_POS-main/public/api/levels/4?level\_kode=SPR

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

Query Params

Key	Value	Description	Bulk Edit
level_kode	SPR		
Key	Value	Description	

body Cookies Headers (11) Test Results Status: 200 OK Time: 266 ms Size: 528 B Save as example

Pretty Raw Preview Visualize JSON

```
1 [
2   {
3     "level_id": 4,
4     "level_kode": "SPR",
5     "level_nama": "Supervisor",
6     "created_at": "2024-04-22T21:40:32.000000Z",
7     "updated_at": "2024-04-22T21:48:19.000000Z"
8   }
9 ]
```

Jelaskan dan berikan screenshoot hasil percobaan Anda.

localhost/Semester%204/PWL25/Minggu10/PWL\_POS/public/api/levels/11?level\_kode=CSV&level\_nama=Customer Service

PUT localhost/Semester%204/PWL25/Minggu10/PWL\_POS/public/api/levels/11?level\_kode=CSV&level\_nama=Customer Service

Params Authorization Headers (7) Body Scripts Settings Cookies

Query Params

Key	Value	Description	Bulk Edit
level_kode	CSV		
level_nama	Customer Service		
Key	Value	Description	

Body Cookies Headers (11) Test Results 200 OK 2.50 s 523 B

{ } JSON Preview Visualize

```
1 {
2   "level_id": 11,
3   "level_kode": "CSV",
4   "level_nama": "Customer Service",
5   "created_at": "2025-04-27T06:41:05.000000Z",
6   "updated_at": "2025-04-27T06:52:07.000000Z"
7 }
```

Data berhasil diedit

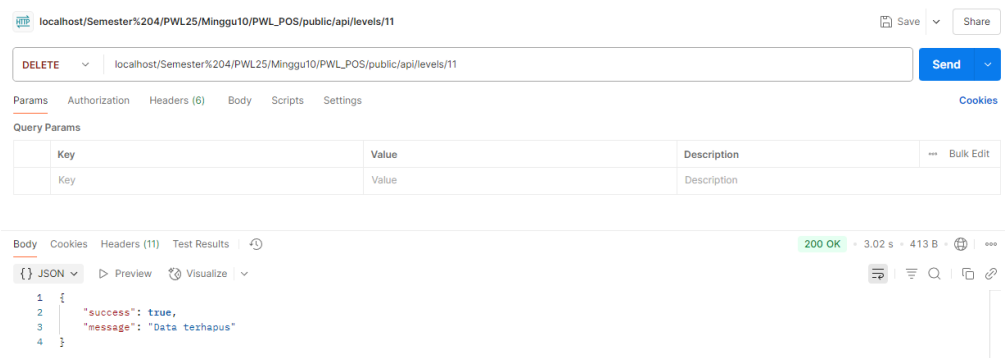
level_id	level_kode	level_nama	Detail	Edit	Hapus
11	CSV	Customer Service			

Showing 1 to 8 of 8 entries

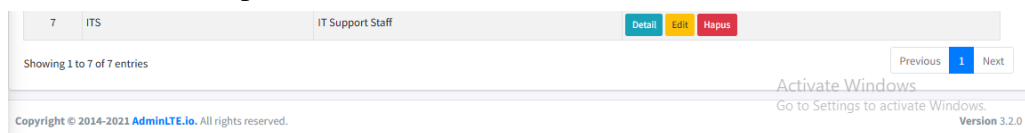
Activate Windows Go to Settings to activate Windows.



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



Data berhasil dihapus



9. Lakukan commit perubahan file pada Github.



## TUGAS

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

### 1. API pada m\_user

- a. Buat controller API pada folder API dengan nama UserController

```
PS F:\laragon\www\Semester 4\PWL25\Minggu10\PWL_POS> php artisan make:controller Api\UserController

INFO Controller [F:\laragon\www\Semester 4\PWL25\Minggu10\PWL_POS\app\Http\Controllers\Api\UserController.php] created successfully.

Minggu10 > PWL_POS > app > Http > Controllers > Api > UserController.php > PHP > UserController > store()

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  0 references | 0 implementations
10 class UserController extends Controller
11 {
12     0 references | 0 overrides
13     public function index(): Collection
14     {
15         return UserModel::all();
16     }
17
18     0 references | 0 overrides
19     public function store(Request $request): JsonResponse|mixed
20     {
21         $user = UserModel::create(attributes: $request->all());
22         return response()->json(data: $user, status: 201);
23     }
24
25     0 references | 0 overrides
26     public function show(UserModel $user): UserModel
27     {
28         return $user;
29     }
30
31     0 references | 0 overrides
32     public function update(Request $request, UserModel $user): UserModel
33     {
34         $user->update(attributes: $request->all());
35         return $user;
36     }
37
38     0 references | 0 overrides
39     public function destroy(UserModel $user): JsonResponse|mixed
40     {
41         $user->delete();
42         return response()->json(data: [
43             'success' => true,
44             'message' => 'Data terhapus',
45         ]);
46     }
47 }
```

- b. Tambahkan route pada routes/api.php

```
7 use App\Http\Controllers\Api\UserController;
8 use Illuminate\Http\Request;

38 //API User
39 Route::get(uri: '/users', action: [UserController::class, 'index']);
40 Route::post(uri: '/users', action: [UserController::class, 'store']);
41 Route::get(uri: '/users/{user}', action: [UserController::class, 'show']);
42 Route::put(uri: '/users/{user}', action: [UserController::class, 'update']);
43 Route::delete(uri: '/users/{user}', action: [UserController::class, 'destroy']);
```



### c. Tes CRUD

- Create user baru dengan method POST

Postman interface showing a POST request to `http://localhost/Semester%204/PWL25/Minggu10/PWL_POS/public/api/users?levelId=3&username=kasir03&nama=Laila&password=12345`. The request body is a JSON object with the following data:

```
{  "level_id": "3",  "username": "kasir03",  "nama": "Laila",  "updated_at": "2025-04-27T07:29:02.000000Z",  "created_at": "2025-04-27T07:29:02.000000Z",  "user_id": 35}
```

The response status is **201 Created** with a response time of 1.55 s and a body size of 527 B. The response body is a JSON object with the same data as the request body.

Data berhasil ditambahkan

Filter						
- Semua - Level Pengguna						
Show 10 entries Search: kasir03						
No	ID Pengguna	Username	Nama	Level	Aksi	
1	35	kasir03	Laila	Staff	<a href="#">Detail</a>	<a href="#">Edit</a> <a href="#">Hapus</a>
Showing 1 to 1 of 1 entries (filtered from 14 total entries)						
Previous		1	Next			

- Read/menampilkan detail user dengan method GET

Postman interface showing a GET request to `http://localhost/Semester%204/PWL25/Minggu10/PWL_POS/public/api/users?username=kasir03&nama=Laila`. The response status is **200 OK** with a response time of 1.16 s and a body size of 2.66 KB. The response body is a JSON object with the following data:

```
{  "user_id": 35,  "level_id": 3,  "profile_photo": null,  "username": "kasir03",  "nama": "Laila",  "created_at": "2025-04-27T07:29:02.000000Z",  "updated_at": "2025-04-27T07:29:02.000000Z"}
```

- Update data user dengan method PUT

Postman interface showing a PUT request to `http://localhost/Semester%204/PWL25/Minggu10/PWL_POS/public/api/users/35?nama=Halley`. The request body is a JSON object with the following data:

```
{  "user_id": 35,  "level_id": 3,  "profile_photo": null,  "username": "kasir03",  "nama": "Halley",  "created_at": "2025-04-27T07:29:02.000000Z",  "updated_at": "2025-04-27T07:36:55.000000Z"}
```

The response status is **200 OK** with a response time of 2.12 s and a body size of 542 B. The response body is a JSON object with the same data as the request body.



## Data berhasil diupdate

Show 10 entries Search: KASIR03

No.	ID Pengguna	Username	Nama	Level	Aksi
1	35	kasir03	Halley	Staff	<a href="#">Detail</a> <a href="#">Edit</a> <a href="#">Hapus</a>

Showing 1 to 1 of 1 entries (filtered from 14 total entries)

Previous 1 Next

- Delete data user dengan method DELETE

PWL-POS User / delete user

DELETE [http://localhost/Semester%204/PWL25/Minggu10/PWL\\_POS/public/api/users/35](http://localhost/Semester%204/PWL25/Minggu10/PWL_POS/public/api/users/35) Send

Params Authorization Headers (6) Body Scripts Settings Cookies

Query Params

Key	Value	Description	Bulk Edit
Key	Value	Description	

Body Cookies Headers (11) Test Results Save Response

200 OK 3.15 s 413 B

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

## Data berhasil dihapus

Filter: Semua Level Pengguna

Show 10 entries Search: KASIR03

No.	ID Pengguna	Username	Nama	Level	Aksi
No matching records found					

Showing 0 to 0 of 0 entries (filtered from 13 total entries)

Previous Next

## 2. API pada m\_kategori

- Buat controller API pada folder API dengan nama KategoriController

```
PS F:\Iaragon\www\Semester 4\PWL25\Minggu10\PWL_POS> php artisan make:controller Api/KategoriController  
INFO Controller [F:\Iaragon\www\Semester 4\PWL25\Minggu10\PWL_POS\app\Http\Controllers\Api\KategoriController.php] created successfully.
```

```
Mingui0 > PWL_POS > app > Http > Controllers > Api > KategoriController.php > PHP > 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 {  
12     public function index(): Collection  
13     {  
14         return KategoriModel::all();  
15     }  
16  
17     public function store(Request $request): JsonResponse|mixed  
18     {  
19         $kategori = KategoriModel::create(attributes: $request->all());  
20         return response()->json(data: $kategori, status: 201);  
21     }  
22  
23     public function show(KategoriModel $kategori): KategoriModel  
24     {  
25         return $kategori;  
26     }  
27  
28     public function update(Request $request, KategoriModel $kategori): KategoriModel  
29     {  
30         $kategori->update(attributes: $request->all());  
31         return $kategori;  
32     }  
33  
34     public function destroy(KategoriModel $kategori): JsonResponse|mixed  
35     {  
36         $kategori->delete();  
37         return response()->json(data: [  
38             'success' => true,  
39             'message' => 'Data kategori terhapus',  
40         ]);  
41     }  
42 }
```



b. Tambahkan route pada routes/api.php

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

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

c. Tes CRUD

- Create kategori baru dengan method POST

Postman interface showing a successful POST request to `http://localhost/Semester%204/PWL25/Minggu10/PWL_POS/public/api/kategoris?kategori_kode=KCW&kategori_nama=Kitchen ware`. The response is a 201 Created status with a JSON body:

```
{
  "kategori_kode": "KCW",
  "kategori_nama": "Kitchen ware",
  "updated_at": "2025-04-27T08:20:56.000000Z",
  "created_at": "2025-04-27T08:20:56.000000Z",
  "kategori_id": 15
}
```

Data berhasil ditambahkan

No	Kode Kategori	Nama Kategori	Aksi
11	KCW	Kitchen ware	<a href="#">Detail</a> <a href="#">Edit</a> <a href="#">Hapus</a>

Showing 11 to 11 of 11 entries

- Read/menampilkan detail kategori dengan method GET

Postman interface showing a successful GET request to `http://localhost/Semester%204/PWL25/Minggu10/PWL_POS/public/api/kategoris?kategori_kode=KCW`. The response is a 200 OK status with a JSON body:

```
{
  "kategori_id": 14,
  "kategori_kode": "SBK",
  "kategori_nama": "Sembako",
  "created_at": "2025-03-29T10:12:13.000000Z",
  "updated_at": null
},
{
  "kategori_id": 15,
  "kategori_kode": "KCW",
  "kategori_nama": "Kitchen ware",
  "created_at": "2025-04-27T08:20:56.000000Z",
  "updated_at": "2025-04-27T08:20:56.000000Z"
}
```



- Update data kategori dengan method PUT

PWL-POS Kategori / New Request

PUT [http://localhost/Semester%204/PWL25/Minggu10/PWL\\_POS/public/api/kategoris/15?kategori\\_nama=Peralatan Dapur](http://localhost/Semester%204/PWL25/Minggu10/PWL_POS/public/api/kategoris/15?kategori_nama=Peralatan Dapur) Send

Params Authorization Headers (7) Body Scripts Settings Cookies

Query Params

Key	Value	Description
kategori_nama	Peralatan Dapur	

Body Cookies Headers (11) Test Results 200 OK 801 ms 531 B Save Response

JSON Preview Visualize

```
1 {
2   "kategori_id": 15,
3   "kategori_kode": "KCM",
4   "kategori_nama": "Peralatan Dapur",
5   "created_at": "2025-04-27T08:28:56.000000Z",
6   "updated_at": "2025-04-27T08:35:47.000000Z"
7 }
```

- Delete data kategori dengan method DELETE

PWL-POS Kategori / New Request

DELETE [http://localhost/Semester%204/PWL25/Minggu10/PWL\\_POS/public/api/kategoris/15](http://localhost/Semester%204/PWL25/Minggu10/PWL_POS/public/api/kategoris/15) Send

Params Authorization Headers (6) Body Scripts Settings Cookies

Query Params

Key	Value	Description
-----	-------	-------------

Body Cookies Headers (11) Test Results 200 OK 2.09 s 422 B Save Response

JSON Preview Visualize

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

### 3. API pada m\_barang

- a. Buat controller API pada folder API dengan nama BarangController

```
PS F:\laragon\www\Semester 4\PWL25\Minggu10\PWL_POS> php artisan make:controller Api\BarangController
INFO Controller [F:\laragon\www\Semester 4\PWL25\Minggu10\PWL_POS\app\Http\Controllers\Api\BarangController.php] created successfully.

Minggu10 > PWL_POS > app > Http > Controllers > Api > BarangController.php > PHP > BarangController > update()

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     0 references | 0 overrides
12     public function index(): Collection
13     {
14         return BarangModel::all();
15     }
16
17     0 references | 0 overrides
18     public function store(Request $request): JsonResponse|mixed
19     {
20         $barang = BarangModel::create(attributes: $request->all());
21         return response()->json(data: $barang, status: 201);
22     }
23 }
```





```
22 public function show(BarangModel $barang): BarangModel
23 {
24     return $barang;
25 }
26
27 0 references | 0 overrides
28 public function update(Request $request, BarangModel $barang): BarangModel
29 {
30     $barang->update(attributes: $request->all());
31     return $barang;
32 }
33
34 0 references | 0 overrides
35 public function destroy(BarangModel $barang): JsonResponse|mixed
36 {
37     $barang->delete();
38     return response()->json(data: [
39         'success' => true,
40         'message' => 'Data terhapus',
41     ]);
42 }
```

b. Tambahkan route pada routes/api.php

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

55 //Route API Barang
56 Route::get(uri: '/barangs', action: [BarangController::class, 'index']);
57 Route::post(uri: '/barangs', action: [BarangController::class, 'store']);
58 Route::get(uri: '/barangs/{barang}', action: [BarangController::class, 'show']);
59 Route::put(uri: '/barangs/{barang}', action: [BarangController::class, 'update']);
60 Route::delete(uri: '/barangs/{barang}', action: [BarangController::class, 'destroy']);
61
```

c. Tes CRUD

- Create data barang baru dengan method POST

Postman interface showing a successful POST request to `http://localhost/Semester%204/PWL25/Minggu10/PWL_POS/public/api/barangs?kategori_id=14&barang_nama=Sun...`. The response status is 201 Created. The JSON response is:

```
{
  "kategori_id": "14",
  "barang_nama": "Sunco Minyak 2 liter",
  "harga_beli": "40000",
  "harga_jual": "45000",
  "barang_kode": "SBK-001",
  "updated_at": "2025-04-27T09:08:44.000000Z",
  "created_at": "2025-04-27T09:08:44.000000Z",
  "barang_id": 38
}
```

Data berhasil ditambahkan

The application's 'Daftar barang' page shows a table with the following data:

No	Kode Barang	Nama Barang	Harga Beli	Harga Jual	Kategori	Aksi
1	SBK-001	Sunco Minyak 2 liter	40.000	45.000	Sembako	<a href="#">Detail</a> <a href="#">Edit</a> <a href="#">Hapus</a>

Showing 1 to 1 of 1 entries (filtered from 27 total entries)



- Read/menampilkan detail barang dengan method GET

Postman interface showing a GET request to the API endpoint `http://localhost/Semester%204/PWL25/Minggu10/PWL_POS/public/api/barangs?38`. The response is a 200 OK status with a JSON body containing details for item 38.

```
{
  "barang_id": 38,
  "kategori_id": 14,
  "barang_kode": "SBK-001",
  "barang_nama": "Sunco Minyak 2 liter",
  "harga_beli": 40000,
  "harga_jual": 45000,
  "created_at": "2025-04-27T09:08:44.000000Z",
  "updated_at": "2025-04-27T09:08:44.000000Z"
}
```

- Update data barang dengan method PUT

Postman interface showing a PUT request to the API endpoint `http://localhost/Semester%204/PWL25/Minggu10/PWL_POS/public/api/barangs/38?barang_nama=Bimoli Minyak Goreng 2 liter`. The response is a 200 OK status with a JSON body containing updated details for item 38.

```
{
  "barang_id": 38,
  "kategori_id": 14,
  "barang_kode": "SBK-001",
  "barang_nama": "Bimoli Minyak Goreng 2 liter",
  "harga_beli": 40000,
  "harga_jual": 45000,
  "created_at": "2025-04-27T09:08:44.000000Z",
  "updated_at": "2025-04-27T09:15:34.000000Z"
}
```

## Data barang berhasil diupdate

Daftar barang

No	Kode Barang	Nama Barang	Harga Beli	Harga Jual	Kategori	Aksi
1	SBK-001	Bimoli Minyak Goreng 2 liter	40.000	45.000	Sembako	<a href="#">Detail</a> <a href="#">Edit</a> <a href="#">Hapus</a>

Showing 1 to 1 of 1 entries (Filtered from 27 total entries)

- Delete data barang dengan method DELETE

Postman interface showing a DELETE request to the API endpoint `http://localhost/Semester%204/PWL25/Minggu10/PWL_POS/public/api/barangs/38`. The response is a 200 OK status with a JSON body indicating success.

```
{
  "success": true,
  "message": "Data terhapus"
}
```