



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 : Karina Ika Indasa

## 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 :

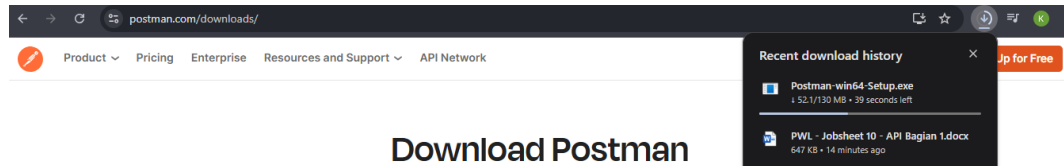
JWT (JSON Web Token) adalah cara untuk mentransfer informasi antara dua pihak secara aman sebagai objek JSON. Ini terdiri dari tiga bagian: header, payload, dan signature. Setelah pengguna berhasil autentikasi, server menghasilkan token JWT yang disematkan dalam permintaan HTTP. Server kemudian memvalidasi 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.



### Download Postman

2. Lakukan instalasi JWT dengan mengetikkan perintah berikut:

```
composer require tymon/jwt-auth:2.1.1
```

Pastikan Anda terkoneksi dengan internet.

```
C:\Windows\System32\cmd.e X + v
Microsoft Windows [Version 10.0.26100.3915]
(c) Microsoft Corporation. All rights reserved.

C:\laragon\www\PWL_2025\Week10\JS10>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)
```

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

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

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

```
C:\laragon\www\PWL_2025\Week10\JS10>php artisan vendor:publish --provider="Tymon\JWTAuth\Providers\LaravelServiceProvider"

INFO Publishing assets.

Copying file [C:\laragon\www\PWL_2025\Week10\JS10\vendor\tymon\jwt-auth\config\config.php] to [C:\laragon\www\PWL_2025\Week10\JS10\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
```

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

```
C:\laragon\www\PWL_2025\Week10\JS10>php artisan jwt:secret
jwt-auth secret [SAo2D6xfVG4ZiHdjIvM7bcSrPf6XOtjOMmvt23vPW4Eny6wnvXN6MPvPgJdB8UPE] 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',  
    ],  
],
```

```
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:

```
<?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';  
}
```

```
4 use Illuminate\Database\Eloquent\Model;  
5 use Illuminate\Database\Eloquent\Factories\HasFactory;  
6 use Illuminate\Database\Eloquent\Relations\BelongsTo;  
7 use Tymon\JWTAuth\Contracts\JWTSubject;  
8 use Illuminate\Foundation\Auth\User as Authenticatable; // im  
9  
35 references | 0 implementations  
10 class UserModel extends Authenticatable  
11 {  
12     0 references | 0 overrides  
13     public function getJWTIdentifier(): mixed  
14     {  
15         return $this->getKey();  
16     }  
17     0 references | 0 overrides  
18     public function getJWTCustomClaims(): array  
19     {  
20         return [];  
21     }  
22 }
```



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.

```
C:\laragon\www\PWL_2025\Week10\JS10>php artisan make:controller Api/RegisterController  
INFO Controller [C:\laragon\www\PWL_2025\Week10\JS10\app\Http\Controllers\Api\RegisterController.php] created successfully.
```

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

```
JS10 > app > Http > Controllers > Api > RegisterController.php > PHP Intelephense > 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 0 references | 0 implementations  
11 class RegisterController extends Controller  
12 0 references | 0 overrides  
13 public function __invoke(Request $request): JsonResponse|mixed  
14 {  
15     //set validation  
16     $validator = Validator::make(data: $request->all(), rules: [  
17         'username' => 'required',  
18         'nama' => 'required',  
19         'password' => 'required min:5|confirmed',  
20         'level_id' => 'required'  
21     ]);  
22  
23     //if validations fails  
24     if ($validator->fails()) {  
25         return response()->json(data: $validator->errors(), status: 422);  
26     }  
27  
28     //create user  
29     $user = UserModel::create(attributes: [  
30         'username' => $request->username,  
31         'nama' => $request->nama,  
32         'password' => bcrypt(value: $request->password),  
33         'level_id' => $request->level_id,  
34     ]);  
35  
36     //return response JSON user is created  
37     if ($user) {  
38         return response()->json(data: [  
39             'success' => true,  
40             'user' => $user,  
41         ], status: 201);  
42     }  
43  
44     //return 350N process insert failed  
45     return response()->json(data: [  
46         'success' => false,  
47     ], status: 409);  
48 }  
49
```

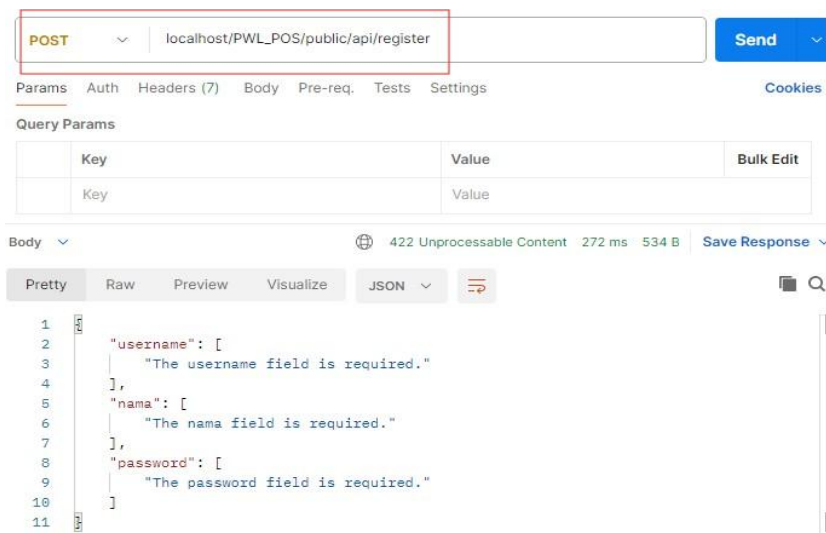


10. Selanjutnya buka routes/api.php, ubah semua kode menjadi seperti berikut.

```
JS10 > routes > api.php
1  <?php
2
3  use Illuminate\Http\Request;
4  use Illuminate\Support\Facades\Route;
5
6  /*
7  |-----
8  | API Routes
9  |-----
10 |
11 | Here is where you can register API routes for your application. These
12 | routes are loaded by the RouteServiceProvider and all of them will
13 | be assigned to the "api" middleware group. Make something great!
14 |
15 |*/
16
17 Route::post(uri: '/register', action: \App\Http\Controllers\Api\RegisterController::class)->name(name: 'register');
18
```

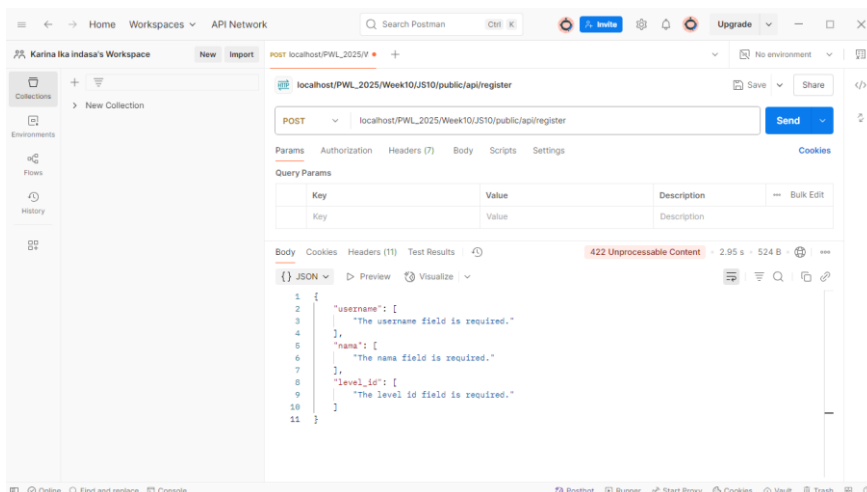
11. Jika sudah, kita akan melakukan uji coba REST API melalui aplikasi Postman.

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



Jika berhasil akan muncul error validasi seperti gambar di atas.

Lakukan percobaan yang sama dan berikan screenshot hasil percobaan Anda.





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

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

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

form-data

| Key   | Value        |
|---|--------------|
| <input checked="" type="checkbox"/> username              | penggunasatu |
| <input checked="" type="checkbox"/> nama                  | Pengguna 1   |
| <input checked="" type="checkbox"/> password              | 12345        |
| <input checked="" type="checkbox"/> password_confirmation | 12345        |
| <input checked="" type="checkbox"/> 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$12$Eb2StV1jsykIllytYGtzH10DVAKcK5p6EgnZnmbChkPicU7SQ3JJu",
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  }
```

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

Lakukan percobaan yang sama dan berikan screenshot hasil percobaan Anda.

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

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

form-data

| Key   | Value        |
|---|--------------|
| <input checked="" type="checkbox"/> username              | penggunasatu |
| <input checked="" type="checkbox"/> nama                  | Pengguna 1   |
| <input checked="" type="checkbox"/> password              | 12345        |
| <input checked="" type="checkbox"/> password_confirmation | 12345        |
| <input checked="" type="checkbox"/> level_id              | 2            |

Body Cookies Headers (11) Test Results 201 Created 1.04 s 101 B

```
1 {
2   "success": true,
3   "user": {
4     "username": "penggunasatu",
5     "nama": "Pengguna 1",
6     "password": "$2y$12$Eb2StV1jsykIllytYGtzH10DVAKcK5p6EgnZnmbChkPicU7SQ3JJu",
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  }
```

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.

```
C:\laragon\www\PWL_2025\Week10\JS10>php artisan make:controller Api/LoginController
```

**INFO** Controller [C:\laragon\www\PWL\_2025\Week10\JS10\app\Http\Controllers\Api>LoginController.php] created successfully.

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

```
JS10 > app > Http > Controllers > Api > LoginController.php > PHP > LoginController
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): JsonResponse|mixed{
12
13         //set validation
14         $validator = Validator::make(data: $request->all(), rules: [
15             'username' => 'required',
16             'password' => 'required'
17         ]);
18
19         //if validation fails
20         if ($validator->fails()) {
21             return response()->json(data: $validator->errors(), status: 422);
22         }
23
24         //get credentials from request
25         $credentials = $request->only(keys: 'username', 'password');
26
27         //if auth failed
28         if (!$token = auth()->guard(name: 'api')->attempt(credentials: $credentials)) {
29             return response()->json(data: [
30                 'success' => false,
31                 'message' => 'Username atau Password Anda salah'
32             ], status: 401);
33         }
34
35         //if auth success
36         return response()->json(data: [
37             'success' => true,
38             'user' => auth()->guard(name: 'api')->user(),
39             'token' => $token
40         ], status: 200);
41     }
42 }
```





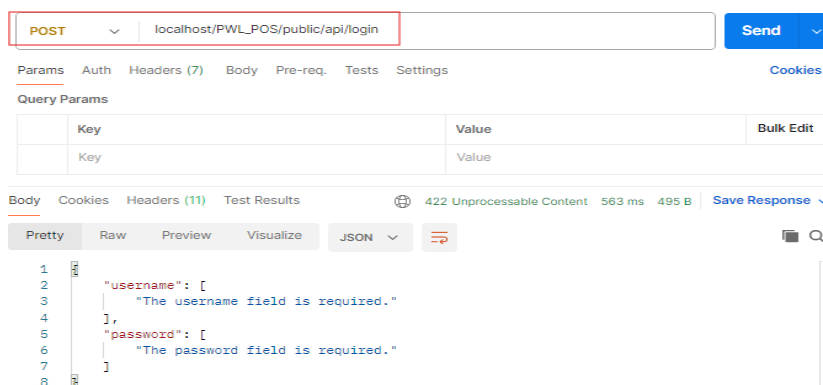
3. Berikutnya tambahkan route baru pada file api.php yaitu /login dan /user.

```
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();
});

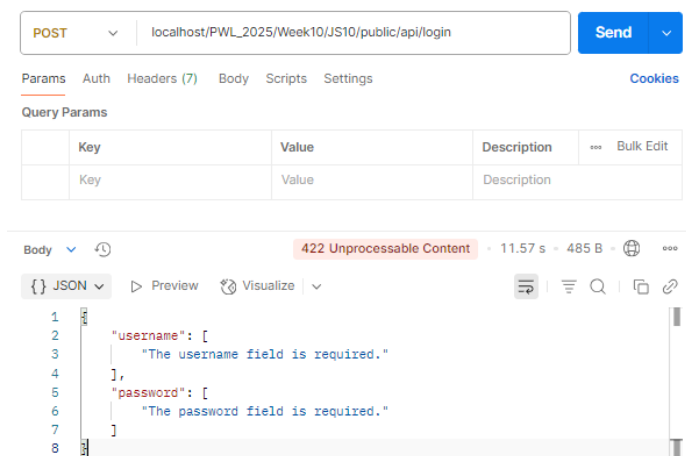
3 use Illuminate\Http\Request;
4 use Illuminate\Support\Facades\Route;
5 use App\Http\Controllers\LoginController;
6
7 /*
8 |-----
9 | API Routes
10 |-----
11 |
12 | Here is where you can register API routes for your application. These
13 | routes are loaded by the RouteServiceProvider and all of them 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'));
19 Route::post(uri: 'login', action: \App\Http\Controllers\Api\LoginController::class->name(name: 'login'));
20 Route::middleware(middleware: 'auth:api')->get(uri: '/user', action: function (Request $request): mixed{
21     return $request->user();
22 });
```

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.

The screenshot shows a Postman interface for a POST request to `localhost/PWL_POS/public/api/login`. The request body is set to `form-data` with two fields: `username` (value: `penggunasatu`) and `password` (value: `12345`). The response status is `200 OK`. The JSON response body is as follows:

```
1 {
2   "success": true,
3   "user": {
4     "user_id": 17,
5     "level_id": 2,
6     "username": "penggunasatu",
7     "nama": "Pengguna 1",
8     "password": "$2y$12$Eb25xVj5ykINytYgtzH10DVAKcK5p6EgnZnmbChkPcIu750Q3Ju",
9     "created_at": "2024-04-22T15:56:04.000000Z",
10    "updated_at": "2024-04-22T15:56:04.000000Z"
11  },
12  "token": "eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiIsInR5cCI6IkpXLTJ1IiwiaWF0Ij0i"
13 }
```

Lakukan percobaan yang sama dan berikan screenshoot hasil percobaan Anda.

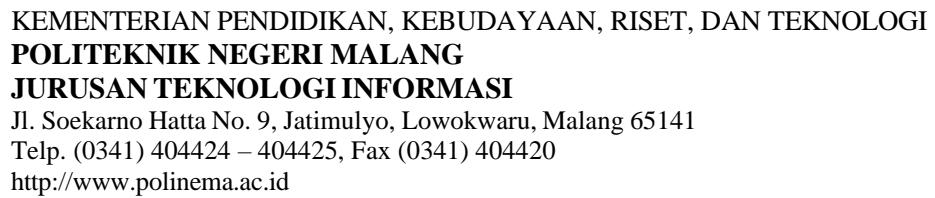
The screenshot shows a Postman interface for a POST request to `localhost/PWL_2025/Week10/JS10/public/api/login`. The request body is set to `form-data` with two fields: `username` (value: `penggunasatu`) and `password` (value: `12345`). The response status is `200 OK`. The JSON response body is as follows:

```
1 {
2   "success": true,
3   "user": {
4     "user_id": 18,
5     "level_id": 2,
6     "profile_picture": null,
7     "username": "penggunasatu",
8     "nama": "Pengguna 1",
9     "created_at": "2025-04-29T16:10:30.000000Z",
10    "updated_at": "2025-04-29T16:10:30.000000Z"
11  },
12  "token": "eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiIsInR5cCI6IkpXLTJ1IiwiaWF0Ij0i"
13 }
```

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

The screenshot shows a Postman interface for a POST request to `localhost/PWL_2025/Week10/JS10/public/api/login`. The request body is set to `form-data` with two fields: `username` (value: `penggunasatu`) and `password` (value: `1234567`). The response status is `401 Unauthorized`. The JSON response body is as follows:

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



- POST

localhost/PWL\_2025/Week10/JSTO/public/api/login

Send

ParamsAuthorizationHeaders (8)BodyScriptsSettings

Cookies

noneform-datax-www-form-urlencodedrawbinaryGraphQL

| Key  | Value        | Description | Bulk Edit |
|--|--------------|-------------|-----------|
| <input checked="" type="checkbox"/> usernameText | penggunasatu |             |           |
| <input checked="" type="checkbox"/> passwordText | 12345        |             |           |

BodyCookiesHeaders (11)Test ResultsVisualize

200 OK · 1.22 s · 936 B · Bulk Edit

{JSONPreviewVisualize}

```
1 {  
2   "success": true,  
3   "user": {  
4     "user_id": 18,  
5     "level_id": 2,  
6     "profile_picture": null,  
7     "username": "penggunasatu",  
8     "nama": "Pengguna 1",  
9     "created_at": "2025-04-29T16:10:30.000000Z",  
10    "updated_at": "2025-04-29T16:10:30.000000Z"  
11  },  
12  "token": "eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.  
      eyJpc3MiOiJodHRwOi8vbG9jYXRob3N0LXBXTF8yMDI1MTAxZWxzWkMCRmEwLnRvbmQ6bnRlcnBvcGVkaW4iLCJpcCqYXQ1dGE3NDVhOTE0OTUzLnRvbmQ6bnRlcjg5RDJldXduIEwiLCJpcCIjoHTglICWenY1oiIOMWRmODcgNGYxYjkzJcWZmZnZhBNHVVkZWN0YjMONTM3MDA2OTB0In.MmGoS5j648f9SdxQtEtYcw6ik3G_i4qtUmppSivM4Y7w"
```

- 

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

```
62
63  JWT_SHOW_BLACKLIST_EXCEPTION=true
64
```

2. Buat Controller baru dengan nama LogoutController.

php artisan make:controller Api/LogoutController

```
C:\laragon\www\PWL_2025\Week10\JS10>php artisan make:controller Api/LogoutController
```

```
INFO Controller [C:\laragon\www\PWL_2025\Week10\JS10\app\Http\Controllers\Api\LogoutController.php]
created successfully.
```

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

```
JS10 > 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
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(data: [
22             'success' => true,
23             'message' => 'Logout Berhasil!',
24         ]);
25     }
26 }
27
```

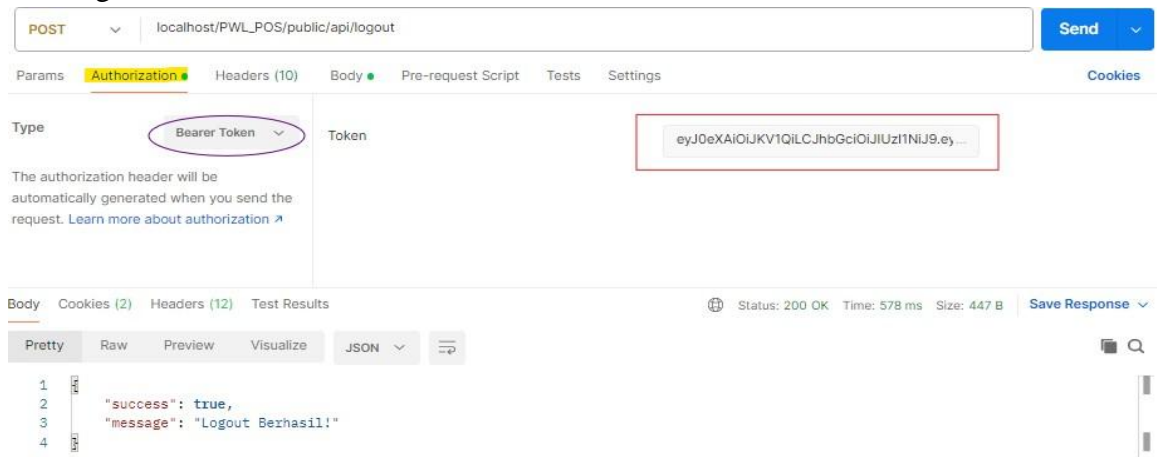
4. Lalu kita tambahkan routes pada api.php

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

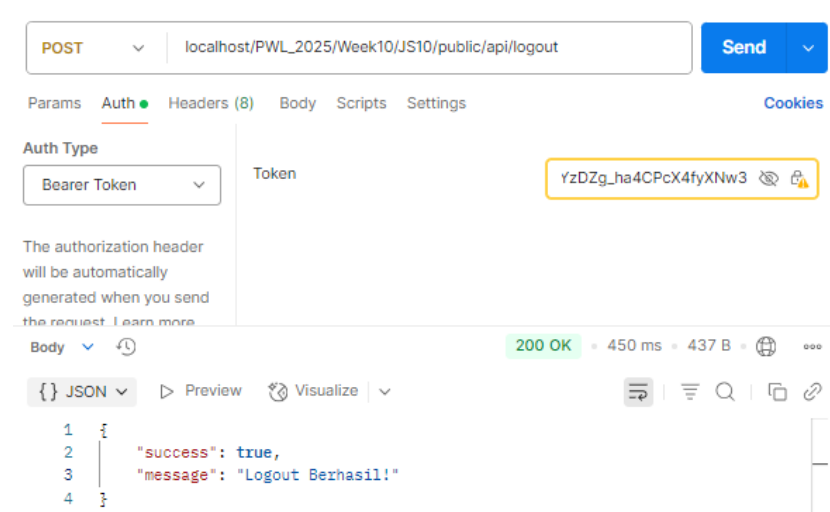
```
3  use Illuminate\Http\Request;
4  use Illuminate\Support\Facades\Route;
5  use App\Http\Controllers\LoginController;
6  use App\Http\Controllers\Api\LogoutController;
7
8  /*
9  |-----
10 | API Routes
11 |-----
12 |
13 | Here is where you can register API routes for your application. These
14 | 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 });
24 Route::post(uri: '/logout', action: App\Http\Controllers\Api\LogoutController::class)->name(name: 'logout');
```



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



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`

```
C:\laragon\www\PWL_2025\Week10\JS10>php artisan make:controller Api/LevelController
```

```
INFO Controller [C:\laragon\www\PWL_2025\Week10\JS10\app\Http\Controllers\Api\LevelController.php] created successfully.
```

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

```
JS10 > 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  0 references | 0 implementations
10 class LevelController extends Controller
11 {
12     0 references | 0 overrides
13     public function index(): Collection
14     {
15         return LevelModel::all();
16     }
17
18     0 references | 0 overrides
19     public function store(Request $request): JsonResponse|mixed
20     {
21         $level = LevelModel::create(attributes: $request->all());
22         return response()->json(data: $level, status: 201);
23     }
24
25     0 references | 0 overrides
26     public function show(LevelModel $level): LevelModel
27     {
28         return $level;
29     }
30
31     0 references | 0 overrides
32     public function update(Request $request, LevelModel $level): LevelModel
33     {
34         $level->update(attributes: $request->all());
35         return $level;
36     }
37
38     0 references | 0 overrides
39     public function destroy(LevelModel $level): JsonResponse|mixed
40     {
41         $level->delete();
42         return response()->json(data: [
43             'success' => true,
44             'message' => 'Data terhapus',
45         ]);
46     }
47 }
```



3. Kemudian kita lengkapi routes pada api.php.

```
7 use App\Http\Controllers\Api\LevelController;  
8  
9 /*  
10 |-----  
11 | API Routes  
12 |-----  
13 |  
14 | Here is where you can register API routes for your application. These  
15 | routes are loaded by the RouteServiceProvider and all of them will  
16 | be assigned to the "api" middleware group. Make something great!  
17 |  
18 |*/  
19  
20 Route::post(uri: '/register', action: App\Http\Controllers\Api\RegisterController::class)->name(name: 'register');  
21 Route::post(uri: 'login', action: App\Http\Controllers\Api\LoginController::class)->name(name: 'login');  
22 Route::middleware('auth:api')->get(uri: '/user', action: function (Request $request): mixed {  
23     return $request->user();  
24 });  
25 Route::post(uri: '/logout', action: App\Http\Controllers\Api\LogoutController::class)->name(name: 'logout');  
26  
27 // Route Level  
28 Route::get(uri: '/levels', action: [LevelController::class, 'index']);  
29 Route::post(uri: '/levels', action: [LevelController::class, 'store']);  
30 Route::get(uri: '/levels/{level}', action: [LevelController::class, 'show']);  
31 Route::put(uri: '/levels/{level}', action: [LevelController::class, 'update']);  
32 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.**

GET localhost/PWL\_POS-main/public/api/levels?level\_kode=ADM&level\_nama=Administrator

Params Authorization Headers (6) Body Scripts Settings Cookies

Query Params

| Key        | Value         | Description |
|------------|---------------|-------------|
| level_kode | ADM           |             |
| level_nama | Administrator |             |

Body Cookies Headers (11) Test Results

200 OK 3.14 s 895 B

```
{  
  "level_id": 1,  
  "level_kode": "ADM",  
  "level_nama": "Administrator",  
  "created_at": null,  
  "updated_at": null  
},  
{  
  "level_id": 2,  
  "level_kode": "MNG",  
  "level_nama": "Manager",  
  "created_at": null,  
  "updated_at": null  
}
```

**Method GET** digunakan untuk mengambil atau menampilkan data dari server.

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-main/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 |
|------------|------------|-------------|
| level_kode | SPV        |             |
| level_nama | Supervisor |             |

Body Cookies Headers (11) Test Results

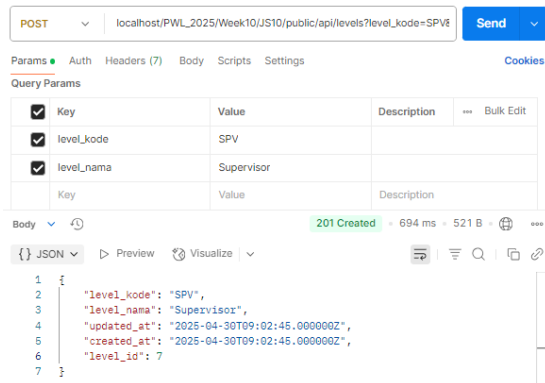
Status: 201 Created Time: 276 ms Size: 531 B Save as example

```
{  
  "level_kode": "SPV",  
  "level_nama": "Supervisor",  
  "updated_at": "2024-04-22T21:48:32.000000Z",  
  "created_at": "2024-04-22T21:48:32.000000Z",  
  "level_id": 4  
}
```



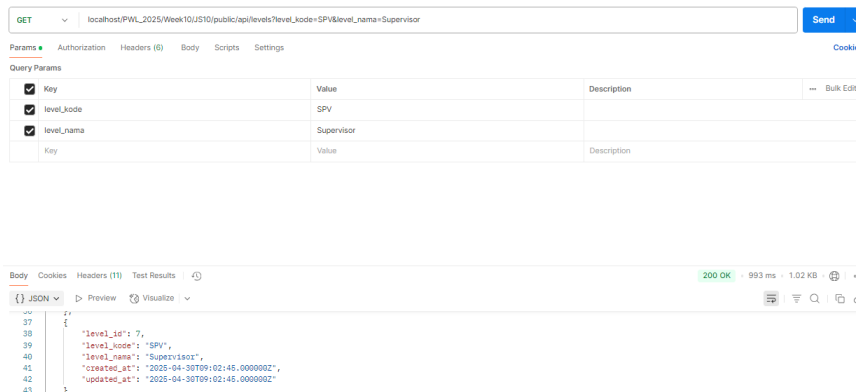


Jelaskan dan berikan screenshot hasil percobaan Anda.



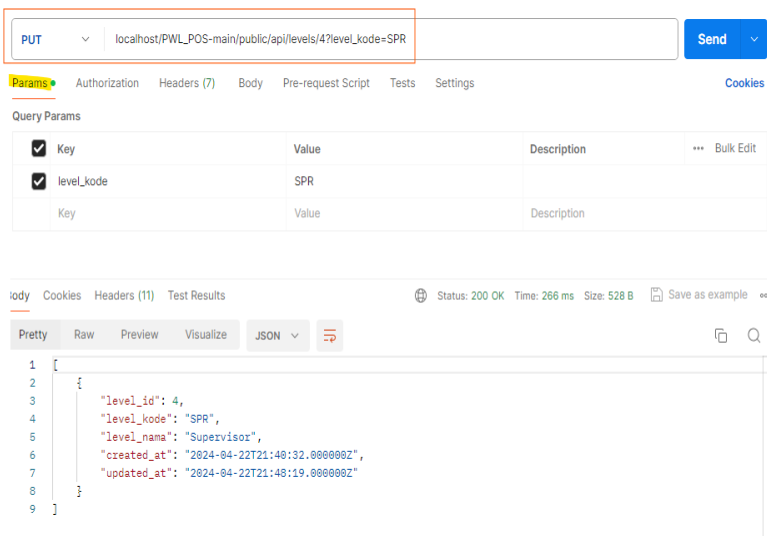
Method POST digunakan untuk mengirim data ke server.

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

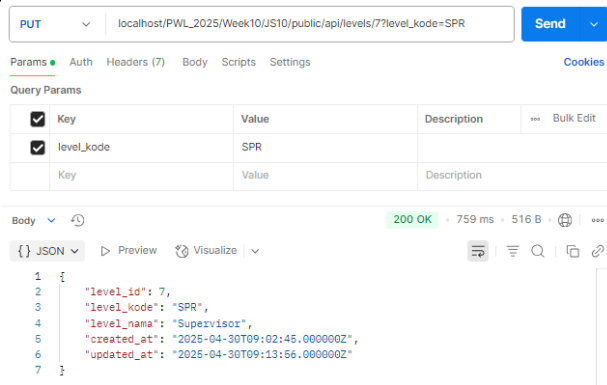


GET untuk menampilkan detail data.

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.

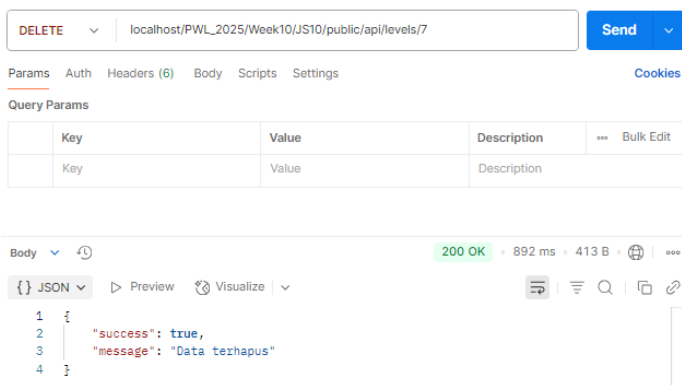


Jelaskan dan berikan screenshot hasil percobaan Anda.



Method **PUT** digunakan saat kita ingin mengupdate data berdasarkan ID.

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



Method **DELETE** dipakai ketika kita ingin menghapus suatu data dari server, berdasarkan ID.

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 CRUD API pada tabel m\_user

- Pertama, buat controller untuk mengolah API pada data user.

`php artisan make:controller Api/UserController`

```
C:\laragon\www\PWL_2025\Week10\JS10>php artisan make:controller Api/UserController
INFO Controller [C:\laragon\www\PWL_2025\Week10\JS10\app\Http\Controllers\Api\UserController.php]
created successfully.
```

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



```
JS10 > app > Http > Controllers > Api > UserController.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\UserModel;
8
9 class UserController extends Controller
10 {
11     1 reference | 0 overrides
12     public function index(): Collection
13     {
14         return UserModel::all();
15     }
16
17     1 reference | 0 overrides
18     public function store(Request $request): JsonResponse|mixed
19     {
20         $user = UserModel::create(attributes: $request->all());
21         return response()->json(data: $user, status: 201);
22     }
23
24     1 reference | 0 overrides
25     public function show(UserModel $user): UserModel
26     {
27         return $user;
28     }
29
30     1 reference | 0 overrides
31     public function update(Request $request, UserModel $user): UserModel
32     {
33         $user->update(attributes: $request->all());
34         return $user;
35     }
36
37     1 reference | 0 overrides
38     public function destroy(UserModel $user): JsonResponse|mixed
39     {
40         $user->delete();
41         return response()->json(data: [
42             'success' => true,
43             'message' => 'Data terhapus',
44         ]);
45     }
46 }
```

- Kemudian kita lengkapi routes pada api.php.

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

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

- Jika sudah. Lakukan uji coba API

- POST

POST localhost/PWL\_2025/Week10/JS10/public/api/users?level\_id=2&username=Mar Send

| Params                                       | Auth | Headers (7) | Body         | Scripts | Settings | Cookies     |
|--|------|-------------|--------------|---------|----------|-------------|
| <input checked="" type="checkbox"/> level_id |      |             | 2            |         |          |             |
| <input checked="" type="checkbox"/> username |      |             | Manager lagi |         |          |             |
| <input checked="" type="checkbox"/> nama     |      |             | Manager lagi |         |          |             |
| <input checked="" type="checkbox"/> password |      |             | 12345        |         |          |             |
| Key  |      |             | Value        |         |          | Description |

Body 201 Created 4.26 s 539 B

JSON Preview Visualize

```
1 {
2   "level_id": "2",
3   "username": "Manager lagi",
4   "nama": "Manager lagi",
5   "updated_at": "2025-04-30T09:38:07.000000Z",
6   "created_at": "2025-04-30T09:38:07.000000Z",
7   "user_id": 19
8 }
```



## ■ GET

GET  Send

Params Auth Headers (6) Body Scripts Settings Cookies

Query Params

| Key | Value | Description | Bulk Edit |
|-----|-------|-------------|-----------|
| Key | Value | Description |           |

Body  727 ms • 2.19 KB

JSON Preview Visualize

```
85 {
86   "level_id": 2,
87   "profile_picture": null,
88   "username": "penggunasatu",
89   "nama": "Pengguna 1",
90   "created_at": "2025-04-29T16:10:30.000000Z",
91   "updated_at": "2025-04-29T16:10:30.000000Z"
92 },
93 {
94   "user_id": 19,
95   "level_id": 2,
96   "profile_picture": null,
97   "username": "Manager lagi",
98   "nama": "Manager lagi",
99   "created_at": "2025-04-30T09:38:07.000000Z",
100   "updated_at": "2025-04-30T09:38:07.000000Z"
101 }
```

## ■ PUT

PUT  Send

Params Auth Headers (7) Body Scripts Settings Cookies

Query Params

| <input checked="" type="checkbox"/> | Key  | Value     | Description | Bulk Edit |
|-------------------------------------|------|-----------|-------------|-----------|
| <input checked="" type="checkbox"/> | nama | icha lagi |             |           |
|                                     | Key  | Value     | Description |           |

Body  398 ms • 552 B

JSON Preview Visualize

```
1 {
2   "user_id": 19,
3   "level_id": 2,
4   "profile_picture": null,
5   "username": "Manager lagi",
6   "nama": "icha lagi",
7   "created_at": "2025-04-30T09:38:07.000000Z",
8   "updated_at": "2025-04-30T09:41:07.000000Z"
9 }
```

## ■ DELETE

DELETE  Send

Params Auth Headers (6) Body Scripts Settings Cookies

Query Params

| Key | Value | Description | Bulk Edit |
|-----|-------|-------------|-----------|
| Key | Value | Description |           |

Body  529 ms • 413 B

JSON Preview Visualize

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



## 2. Implementasi CRUD API pada tabel m\_kategori

- Pertama, buat controller untuk mengolah API pada data kategori.

`php artisan make:controller Api/KategoriController`

```
C:\laragon\www\pwl_2025\Week10\JS10>php artisan make:controller Api/KategoriController
```

```
INFO Controller [C:\laragon\www\pwl_2025\Week10\JS10\app\Http\Controllers\Api\KategoriController.php] created successfully.
```

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

```
JS10 > 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  0 references | 0 implementations
10 class KategoriController extends Controller
11
12     0 references | 0 overrides
13     public function index(): Collection
14     {
15         return KategoriModel::all();
16     }
17
18     0 references | 0 overrides
19     public function store(Request $request): JsonResponse|mixed
20     {
21         $kategori = KategoriModel::create(attributes: $request->all());
22         return response()->json(data: $kategori, status: 201);
23     }
24
25     0 references | 0 overrides
26     public function show(KategoriModel $kategori): KategoriModel
27     {
28         return $kategori;
29     }
30
31     0 references | 0 overrides
32     public function update(Request $request, KategoriModel $kategori): KategoriModel
33     {
34         $kategori->update(attributes: $request->all());
35         return $kategori;
36     }
37
38     0 references | 0 overrides
39     public function destroy(KategoriModel $kategori): JsonResponse|mixed
40     {
41         $kategori->delete();
42         return response()->json(data: [
43             'success' => true,
44             'message' => 'Data kategori terhapus',
45         ]);
46     }
47
48 }
```

- Kemudian kita lengkapi routes pada api.php.

```
9  use App\Http\Controllers\Api\KategoriController;
43 // Route Kategori
44 Route::get(uri: '/kategoris', action: [KategoriController::class, 'index']);
45 Route::post(uri: '/kategoris', action: [KategoriController::class, 'store']);
46 Route::get(uri: '/kategoris/{kategori}', action: [KategoriController::class, 'show']);
47 Route::put(uri: '/kategoris/{kategori}', action: [KategoriController::class, 'update']);
48 Route::delete(uri: '/kategoris/{kategori}', action: [KategoriController::class, 'destroy']);
```

- Jika sudah. Lakukan uji coba API

- POST



POST localhost/PWL\_2025/Week10/JS10/public/api/kategoris?kategori\_kode=SRM&k Send

Params Auth Headers (7) Body Scripts Settings Cookies

Query Params

| <input checked="" type="checkbox"/> | Key           | Value | Description | Bulk Edit |
|-------------------------------------|---------------|-------|-------------|-----------|
| <input checked="" type="checkbox"/> | kategori_kode | SRM   |             |           |
| <input checked="" type="checkbox"/> | kategori_nama | Serum |             |           |
|                                     | Key           | Value | Description |           |

Body 201 Created 4.76 s 526 B

{ } JSON Preview Visualize

```
1 {
2   "kategori_kode": "SRM",
3   "kategori_nama": "Serum",
4   "updated_at": "2025-04-30T09:50:16.000000Z",
5   "created_at": "2025-04-30T09:50:16.000000Z",
6   "kategori_id": 13
7 }
```

## ■ GET

GET localhost/PWL\_2025/Week10/JS10/public/api/kategoris Send

Params Auth Headers (6) Body Scripts Settings Cookies

Query Params

| <input type="checkbox"/> | Key | Value | Description | Bulk Edit |
|--------------------------|-----|-------|-------------|-----------|
|                          | Key | Value | Description |           |

Body 200 OK 832 ms 1.43 KB

{ } JSON Preview Visualize

```
49 {
50   "updated_at": null
51 },
52 {
53   "kategori_id": 9,
54   "kategori_kode": "SPT",
55   "kategori_nama": "Sepatu",
56   "created_at": "2025-03-17T16:07:25.000000Z",
57   "updated_at": "2025-03-17T16:07:25.000000Z"
58 },
59 {
60   "kategori_id": 13,
61   "kategori_kode": "SRM",
62   "kategori_nama": "Serum",
63   "created_at": "2025-04-30T09:50:16.000000Z",
64   "updated_at": "2025-04-30T09:50:16.000000Z"
65 }
```

## ■ PUT

PUT localhost/PWL\_2025/Week10/JS10/public/api/kategoris/13?kategori\_kode=SR Send

Params Auth Headers (7) Body Scripts Settings Cookies

Query Params

| <input checked="" type="checkbox"/> | Key           | Value | Description | Bulk Edit |
|-------------------------------------|---------------|-------|-------------|-----------|
| <input checked="" type="checkbox"/> | kategori_kode | SR    |             |           |
|                                     | Key           | Value | Description |           |

Body 200 OK 537 ms 520 B

{ } JSON Preview Visualize

```
1 {
2   "kategori_id": 13,
3   "kategori_kode": "SR",
4   "kategori_nama": "Serum",
5   "created_at": "2025-04-30T09:50:16.000000Z",
6   "updated_at": "2025-04-30T09:53:04.000000Z"
7 }
```



## DELETE

DELETE

Params Auth Headers (6) Body Scripts Settings Cookies

Query Params

|  | Key | Value | Description | ... | Bulk Edit |
|--|-----|-------|-------------|-----|-----------|
|  | Key | Value | Description |     |           |

Body

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

200 OK • 393 ms • 422 B •

### 3. Implementasi CRUD API pada tabel m\_barang

- Pertama, buat controller untuk mengolah API pada data barang.

`php artisan make:controller Api/BarController`

`C:\laragon\www\PWL_2025\Week10\JS10>php artisan make:controller Api/BarController`

**INFO** Controller `[C:\laragon\www\PWL_2025\Week10\JS10\app\Http\Controllers\Api\BarController.php]` created successfully.

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

```
JS10 > app > Http > Controllers > Api > BarangController.php > PHP > 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 6 references | 0 implementations
10 class BarangController extends Controller
11 {
12     1 reference | 0 overrides
13     public function index(): Collection
14     {
15         return BarangModel::all();
16     }
17
18     1 reference | 0 overrides
19     public function store(Request $request): JsonResponse|mixed
20     {
21         $barang = BarangModel::create(attributes: $request->all());
22         return response()->json(data: $barang, status: 201);
23     }
24
25     1 reference | 0 overrides
26     public function show(BarangModel $barang): BarangModel
27     {
28         return $barang;
29     }
30
31     1 reference | 0 overrides
32     public function update(Request $request, BarangModel $barang): BarangModel
33     {
34         $barang->update(attributes: $request->all());
35         return $barang;
36     }
37
38     1 reference | 0 overrides
39     public function destroy(BarangModel $barang): JsonResponse|mixed
40     {
41         $barang->delete();
42         return response()->json(data: [
43             'success' => true,
44             'message' => 'Data barang terhapus',
45         ]);
46     }
47 }
```





- Kemudian kita lengkapi routes pada api.php.

```
11 use App\Http\Controllers\Api\BarangController;  
  
51 // Route Barang  
52 Route::get(uri: '/barangs', action: [BarangController::class, 'index']);  
53 Route::post(uri: '/barangs', action: [BarangController::class, 'store']);  
54 Route::get(uri: '/barangs/{barang}', action: [BarangController::class, 'show']);  
55 Route::put(uri: '/barangs/{barang}', action: [BarangController::class, 'update']);  
56 Route::delete(uri: '/barangs/{barang}', action: [BarangController::class, 'destroy']);
```

- Jika sudah. Lakukan uji coba API

#### ▪ POST

POST localhost/PWL\_2025/Week10/JS10/public/api/barangs?kategori\_id=1&barang\_n Send

Params Auth Headers (7) Body Scripts Settings Cookies

| Key         | Value | Description |
|-------------|-------|-------------|
| barang_nama | Teh   |             |
| harga_beli  | 6000  |             |
| harga_jual  | 7000  |             |
| barang_kode | B011  |             |

Body 201 Created • 3.90 s • 577 B

JSON Preview Visualize

```
1 {  
2   "kategori_id": "1",  
3   "barang_nama": "Teh",  
4   "harga_beli": "6000",  
5   "harga_jual": "7000",  
6   "barang_kode": "B011",  
7   "updated_at": "2025-04-30T10:06:44.000000Z",  
8   "created_at": "2025-04-30T10:06:44.000000Z",  
9   "barang_id": 13  
10 }
```

#### ▪ GET

GET localhost/PWL\_2025/Week10/JS10/public/api/barangs Send

Params Auth Headers (6) Body Scripts Settings Cookies

Query Params

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

Body 200 OK • 881 ms • 2.05 KB

JSON Preview Visualize

```
103 {  
104   "barang_id": 13,  
105   "kategori_id": 1,  
106   "barang_kode": "B011",  
107   "barang_nama": "Teh",  
108   "harga_beli": 6000,  
109   "harga_jual": 7000,  
110   "created_at": "2025-04-30T10:06:44.000000Z",  
111   "updated_at": "2025-04-30T10:06:44.000000Z"  
112 }
```



## ■ PUT

PUT localhost/PWL\_2025/Week10/JS10/public/api/barangs/13?barang\_nama=Teh ... Send

Params Auth Headers (7) Body Scripts Settings Cookies

Query Params

| <input checked="" type="checkbox"/> | Key         | Value      | Description | Bulk Edit |
|-------------------------------------|-------------|------------|-------------|-----------|
| <input checked="" type="checkbox"/> | barang_nama | Teh Bandol |             |           |
|                                     | Key         | Value      | Description |           |

Body 200 OK 810 ms 573 B

{ } JSON Preview Visualize

```
1 {
2   "barang_id": 13,
3   "kategori_id": 1,
4   "barang_kode": "B011",
5   "barang_nama": "Teh Bandol",
6   "harga_beli": 6000,
7   "harga_jual": 7000,
8   "created_at": "2025-04-30T10:06:44.000000Z",
9   "updated_at": "2025-04-30T10:09:14.000000Z"
10 }
```

## ■ DELETE

DELETE localhost/PWL\_2025/Week10/JS10/public/api/barangs/13 Send

Params Auth Headers (6) Body Scripts Settings Cookies

Query Params

|  | Key | Value | Description | Bulk Edit |
|--|-----|-------|-------------|-----------|
|  | Key | Value | Description |           |

Body 200 OK 403 ms 420 B

{ } JSON Preview Visualize

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

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