



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

JOBSHEET 10

RESTFUL API

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

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

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

A. RESTFUL API

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

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



B. JSON Web Token (JWT)

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

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

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

Kita dapat menggunakan JWT untuk:

- Authentication

Ketika pengguna melakukan authentication dan mendapatkan token, maka setiap permintaan berikutnya akan menyertakan token tersebut, dan memungkinkan pengguna untuk mengakses route, service, dan resources yang diizinkan.

- Pertukaran informasi

JSON Web Token adalah cara yang baik untuk mengirimkan informasi antar pihak dengan aman. Dengan token yang sudah ditandatangani dengan algoritma RSA, maka kita bisa tahu siapa yang melakukan request tersebut.

Berikut adalah cara kerja JWT :

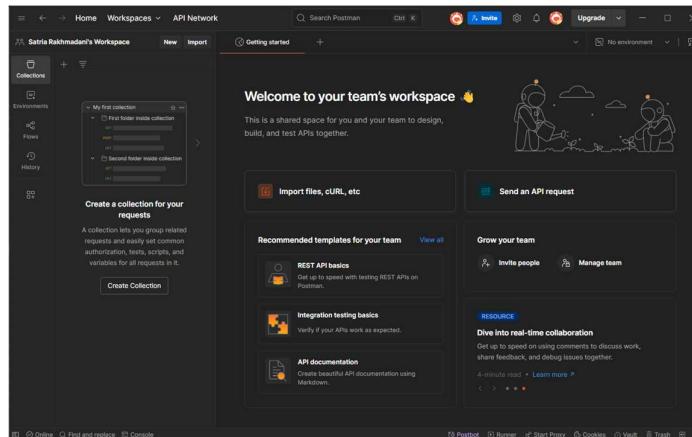
JWT (JSON Web Token) adalah cara untuk mentransfer informasi antara dua pihak secara aman sebagai objek JSON. Ini terdiri dari tiga bagian: header, payload, dan signature. Setelah pengguna berhasil autentikasi, server menghasilkan token JWT yang disematkan dalam permintaan HTTP. Server kemudian memvalidasi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



Praktikum 1 – Membuat RESTful API Register

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

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



Tampilan Aplikasi Postman Ketika Sudah Diinstal.

2. Lakukan instalasi JWT dengan mengetikkan perintah berikut:

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

Pastikan Anda terkoneksi dengan internet.

```
PS D:\Documents\UML\SEMESTER 4\WEB\ML2025\2025sg\UML_PDS\composer require typhon/jwt-auth:2.1.0
composer.json has been updated
Running composer update --no-suggest --no-dev
  Updating dependencies
    Locking composer.json with package information
    Updating dependencies
      lock file operations: 4 installs, 0 updates, 0 removals
        - Installing typhon/jwt-auth (2.1.0)
        - Locking typhon/jwt (4.0.4)
        - Locking stells-marij/lock (6.1.7)
        - Locking typhon/jwt-auth (2.1.0)
      Writing lock file
      Writing lock file
  Installing dependencies from lock file (including require-dev)
    Installing typhon/jwt-auth (2.1.0) from cache
    Removing typhon/jwt (4.0.4)
      Downloading stells-marij/lock (6.1.7)
        - Downloading stells-marij/lock (6.1.7)
          - Downloading typhon/jwt (4.0.4)
            - Downloading typhon/jwt-auth (2.1.0)
              - Downloading typhon/jwt-auth (2.1.0): Extracting archive
              - Installing typhon/jwt (2.1.0): Extracting archive
              - Installing typhon/jwt (4.0.4): Extracting archive
              - Installing typhon/jwt (4.0.4): Extracting archive
              - Installing typhon/jwt (4.0.4): Extracting archive
    Generating optimized autoload files
  Illuminate\Foundation\ComposerScripts::postAutoloadDump
  PS D:\Documents\UML\SEMESTER 4\WEB\ML2025\2025sg\UML_PDS\composer require laravel/asset
  composer.json has been updated
  Running composer update --no-suggest --no-dev
  Discovering packages...
  Barryvdh\Laravel-Doctrine
  Barryvdh\Debugbar
  Laracasts\ Sanctum
  Laracasts\Tinker
  Nette\Bridges
  Nette\Nette
  Nette\Nette\Collision
  Nette\Nette\Terminal
  Nette\Nette\Templid
  Nette\Nette\Utility
  Typhon\Jwt\Auth
  Yajra\ Laravel-Datatables-Buttons
  Yajra\ Laravel-Datatables-Editor
  Yajra\ Laravel-Datatables-Fractal
  Yajra\ Laravel-Datatables-Html
  Yajra\ Laravel-Datatables-Scalable
  92 packages you are using are looking for funding.
  Use the "composer fund" command to find out more!
  > [PHP] artisan vendor:publish --tag=laravel-asset --force

  [INFO] No publishable resources for tag [laravel-asset].
```

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

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

```
PS D:\Documents\Volinema\STUDI_KAMPUS\SEMESTER_4\WEB\VM\2025\Ringgit0\VM\POS> php artisan vendor:publish --provider="Tymon\JWTAuth\Providers\LaravelServiceProvider"
[INFO] Publishing assets.

Copying file [D:\Documents\Volinema\STUDI_KAMPUS\SEMESTER_4\WEB\VM\2025\Ringgit0\VM\POS\vendor\tymon\jwt-auth\config\config.php] to [D:\Documents\Volinema\STUDI_KAMPUS\SEMESTER_4\WEB\VM\2025\Ringgit0\VM\POS\config\config.php]
[nfngjy] 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.

```
PS D:\Documents\Polinema\STUDI KAMPUS\SEMESTER 4\WEB\PWL2025\Minggu10\PWL_POS> php artisan jwt:secret
jwt-auth secret [aH6saEuiVCM0awmrUt2GcWxpQMLwiqZi2E7E1WMeRxQhqdNdEx4j9pMNbvO2foPF] set successfully.
PS D:\Documents\Polinema\STUDI KAMPUS\SEMESTER 4\WEB\PWL2025\Minggu10\PWL_POS> []
```



```
PS D:\Documents\Polinema\STUDI KAMPUS\SEMESTER 4\WEB\PWL2025\Minggu10\PWL_POS> php artisan jwt:secret
jwt-auth secret [aH6saEuiVCM0awmrUt2GcWxpQMLwiqZi2E7E1WMeRxQhqdNdEx4j9pMNbvO2foPF] set successfully.
PS D:\Documents\Polinema\STUDI KAMPUS\SEMESTER 4\WEB\PWL2025\Minggu10\PWL_POS> []
```

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

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

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



```
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Model;
use 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';
```

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.

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

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



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

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



```
<?php

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

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

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

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.

The screenshot shows the Postman interface with a failed API call. The URL is set to `localhost/PWL2025/Minggu10/PWL_POS/public/api/register`. The method is `POST`. The response status is `422 Unprocessable Content`, with a message indicating validation errors for required fields: `username`, `nama`, `password`, and `level_id`.

Key	Value	Description
username	["The username field is required."]	
nama	["The nama field is required."]	
password	["The password field is required."]	
level_id	["The level id field is required."]	

Jika berhasil akan muncul error validasi seperti gambar di atas.

Lakukan percobaan yang sama dan berikan screenshoot hasil percobaan Anda.

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



The screenshot shows a Postman interface with the following details:

- Method: POST
- URL: localhost/PWL2025/Minggu10/PWL_POS/public/api/register
- Body tab selected, showing form-data fields:
 - username: pengguna1
 - nama: Pengguna 1
 - password: 12345
 - password_confirmation: 12345
 - level_id: 2
- Response status: 201 Created
- Response body (JSON):

```
1 {  
2   "success": true,  
3   "user": {  
4     "username": "pengguna1",  
5     "nama": "Pengguna 1",  
6     "level_id": "2",  
7     "updated_at": "2025-04-28T16:11:03.000000Z",  
8     "created_at": "2025-04-28T16:11:03.000000Z",  
9     "user_id": 43  
10   }  
11 }
```

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.

13. Lakukan commit perubahan file pada Github.

Praktikum 2 – Membuat RESTful API Login

1. Kita buat file controller dengan nama LoginController.

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

```
PS D:\Documents\Polinema\STUDI KAMPUS\SEMESTER 4\WEB\PWL2025\Minggu10\PWL_POS> php artisan make:controller Api/LoginController
INFO Controller [D:\Documents\Polinema\STUDI KAMPUS\SEMESTER 4\WEB\PWL2025\Minggu10\PWL_POS\app\Http\Controllers\Api>LoginController.php] created successfully.
```

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

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

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



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

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

```
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();
});
```

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.



The screenshot shows two separate Postman requests for the endpoint `localhost/PWL_2025/Mi/api/login`. Both requests result in a `422 Unprocessable Content` status with a response body indicating validation errors:

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

The first attempt is made from the URL `localhost/PWL_2025/Minggu10/PWL_POS/public/api/login`. The second attempt is made from the URL `localhost/PWL2025/Mi`.

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 successful Postman request for the endpoint `localhost/PWL_POS/public/api/login`. The response status is `200 OK` with a response body containing user information and a token:

```
1 {
2     "success": true,
3     "user": {
4         "user_id": 17,
5         "level_id": 2,
6         "username": "penggunaatu",
7         "nama": "Pengguna 1",
8         "password": "$2b$12$eB2svVijsykINyvYGtrH1ODVAKCK5p6EgnZnmbChkPiclu7S0QJJu",
9         "created_at": "2024-04-22T15:56:04.000000Z",
10        "updated_at": "2024-04-22T15:56:04.000000Z"
11    },
12    "token": "eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJpc3MiOiJodHRwO18vbG9jYWxob3N0L18XTF9QT1MtbWFpb19wdWJsaWMyYXBpL2xzZ2luIiwiaWF0I"
13 }
```



```
POST localhost/PWL2025/M... POST localhost/PWL2025/M...
localhost/PWL2025/Minggu10/PWL_POS/public/api/login
Save Share </>
POST localhost/PWL2025/Minggu10/PWL_POS/public/api/login
Send
Params Authorization Headers (8) Body Scripts Settings Cookies
none form-data x-www-form-urlencoded raw binary GraphQL
Key Value Description Bulk Edit
username pengguna1
password 12345
Key Value Description
Body Cookies Headers (11) Test Results 200 OK 293 ms 938 B
JSON Preview Visualize
1 {
2   "success": true,
3   "user": {
4     "user_id": 43,
5     "level_id": 2,
6     "username": "pengguna1",
7     "nama": "Pengguna 1",
8     "profile_picture": null,
9     "created_at": "2025-04-28T16:11:03.000000Z",
10    "updated_at": "2025-04-28T16:11:03.000000Z"
11  },
12  "token": "eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJpc3MiOiJodHRwOi8vbG9jYWxob3N0L1BX
TDIwMjUvTWluZ2d1MTAvUFdMX1BPUs9wdWJsaWMvYXBpL2xvZ2luIiwiaWF0IjoxNzQ
1ODU4NTAxLCJleHAiOjE3NDU4NjIxMDEsIm5iZiI6MTc0NTg1ODUwMSwanRpIjoia0JHZ
E45cjdkOHZCU3BlbCisInN1YiI6IjQzIiwicHJ2IjoiNDFkZjg4MzRmMWI5OGY3MGVmYT
wYWFlZGVmNDIzNDEzNzAwNjkwyJ9.fdkTjdO1SzBItK97Jy9DDeP6YW3_xRqluj7YD8
k22Y"
13 }
```

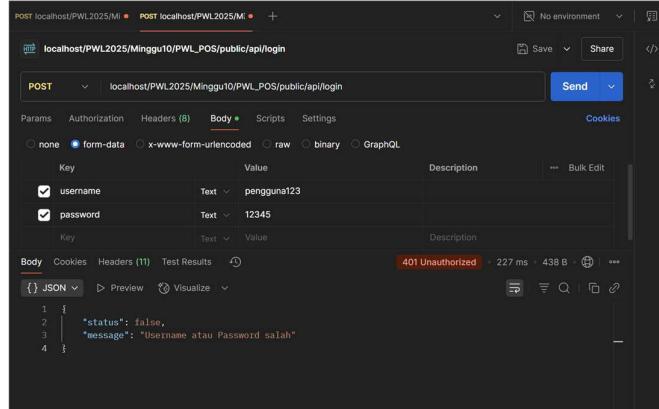
"token":

"eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJpc3MiOiJodHRwOi8vbG9jYWxob3N0L1BX
TDIwMjUvTWluZ2d1MTAvUFdMX1BPUs9wdWJsaWMvYXBpL2xvZ2luIiwiaWF0IjoxNzQ
1ODU4NTAxLCJleHAiOjE3NDU4NjIxMDEsIm5iZiI6MTc0NTg1ODUwMSwanRpIjoia0JHZ
E45cjdkOHZCU3BlbCisInN1YiI6IjQzIiwicHJ2IjoiNDFkZjg4MzRmMWI5OGY3MGVmYT
wYWFlZGVmNDIzNDEzNzAwNjkwyJ9.fdkTjdO1SzBItK97Jy9DDeP6YW3_xRqluj7YD8
k22Y"



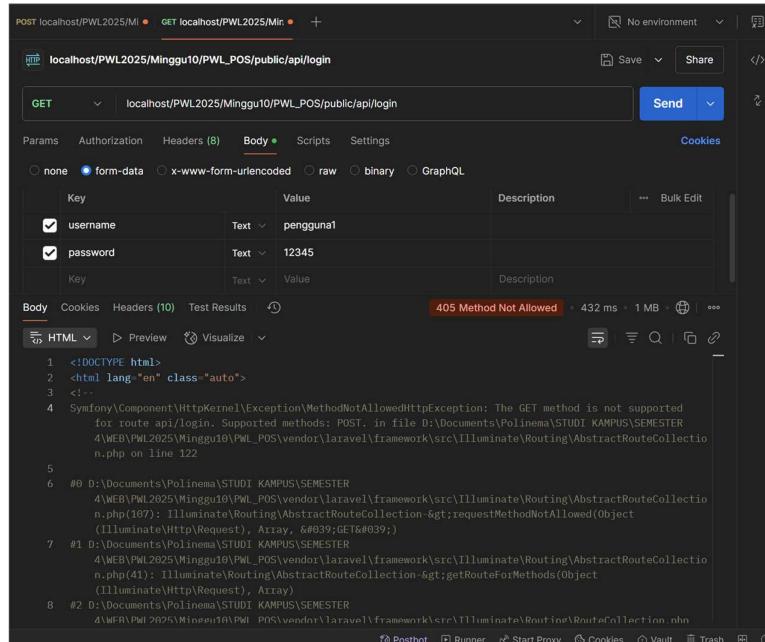
Lakukan percobaan yang sama dan berikan screenshot hasil percobaan Anda.

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



A screenshot of the Postman application interface. The URL is `localhost/PWL2025/Minggu10/PWL_POS/public/api/login`. The method is set to `POST`. The body is in `form-data` format with two fields: `username` (text: `pengguna123`) and `password` (text: `12345`). The response status is `401 Unauthorized` with a response body of `{"status": false, "message": "Username atau Password salah"}`.

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.



A screenshot of the Postman application interface. The URL is `localhost/PWL2025/Minggu10/PWL_POS/public/api/login`. The method is set to `GET`. The body is in `form-data` format with two fields: `username` (text: `pengguna1`) and `password` (text: `12345`). The response status is `405 Method Not Allowed` with a detailed error message in the response body:

```
<!DOCTYPE html>
<html lang="en" class="auto">
<!--
    Symfony\Component\HttpFoundation\Exception\MethodNotAllowedHttpException: The GET method is not supported
    for route api/login. Supported methods: POST.
    in file D:\Documents\Polinema\STUDI KAMPUS\SEMESTER
    4\Web\PWL2025\Minggu10\PWL_POS\vendor\laravel\framework\src\Illuminate\Routing\AbstractRouteCollection
    n.php on line 122
-->
#0 D:\Documents\Polinema\STUDI KAMPUS\SEMESTER
    4\Web\PWL2025\Minggu10\PWL_POS\vendor\laravel\framework\src\Illuminate\Routing\AbstractRouteCollection
    n.php(107): Illuminate\Routing\AbstractRouteCollection->requestMethodNotAllowed(Object
    (Illuminate\Http\Request), Array, &#039;GET&#039;)
#1 D:\Documents\Polinema\STUDI KAMPUS\SEMESTER
    4\Web\PWL2025\Minggu10\PWL_POS\vendor\laravel\framework\src\Illuminate\Routing\AbstractRouteCollection
    n.php(41): Illuminate\Routing\AbstractRouteCollection->getRouteForMethods(Object
    (Illuminate\Http\Request), Array)
#2 D:\Documents\Polinema\STUDI KAMPUS\SEMESTER
    4\Web\PWL2025\Minggu10\PWL_POS\vendor\laravel\framework\src\Illuminate\Routing\RouteCollection.php
```

Dari pengisian data dari database, akan terjadi error dan muncul HTML view. Hal ini disebabkan karena metode yang dipilih adalah ‘GET’. Maka, yang benar dalam metode login ini dengan menggunakan ‘POST’

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
```

```
Minggu10 > PWL_POS > .env
56 VITE_PUSHER_PORT="${PUSHER_PORT}"
57 VITE_PUSHER_SCHEME="${PUSHER_SCHEME}"
58 VITE_PUSHER_APP_CLUSTER="${PUSHER_APP_CLUSTER}"
59
60 JWT_SECRET=aH6saEuiVCM0awmrUt2GcWxpQMLwiqZi2E7E1WMeRxQhqdNxe4j9pMNbv02foPF
61
62 JWT_SHOW_BLACKLIST_EXCEPTION=true
```

2. Buat Controller baru dengan nama LogoutController.

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

```
PS D:\Documents\Polinema\STUDI KAMPUS\SEMESTER 4\WEB\PWL2025\Minggu10\PWL_POS> php artisan make:controller Api/LogoutController
[INFO] Controller [D:\Documents\Polinema\STUDI KAMPUS\SEMESTER 4\WEB\PWL2025\Minggu10\PWL_POS\app\Http\Controllers\Api\LogoutController.php] created successfully.
PS D:\Documents\Polinema\STUDI KAMPUS\SEMESTER 4\WEB\PWL2025\Minggu10\PWL_POS>
```

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

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



4. Lalu kita tambahkan routes pada api.php

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

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

The screenshot shows the Postman interface with a GET request to `localhost/PWL2025/Minggu10/PWL_POS/public/api/logout`. The 'Params' tab is selected, showing a single parameter named 'Key'. The 'Authorization' tab is also visible. A 'Send' button is present at the top right.

6. Isi token pada tab Authorization, pilih Type yaitu Bearer Token. Isikan token yang didapat saat login. Jika sudah klik Send.

The screenshot shows the Postman interface with a POST request to `localhost/PWL_POS/public/api/logout`. The 'Authorization' tab is selected, with 'Type' set to 'Bearer Token'. A token value is entered in the 'Token' field. The response status is 200 OK, and the JSON response body is shown as:

```
1 "success": true,
2 "message": "Logout Berhasil!"
```

Lakukan percobaan yang sama dan berikan screenshoot hasil percobaan Anda.



The screenshot shows a Postman interface with the following details:

- Method: POST
- URL: localhost/PWL2025/Minggu10/PWL_POS/public/api/logout
- Auth Type: Bearer Token
- Token: K97Jy9DDeP6YW3_xRqluj7YD8k22Y
- Body: JSON response (200 OK):

```
1 {  
2   "status": true,  
3   "message": "Logout berhasil"  
4 }
```

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
```

```
PS D:\Documents\Polinema\STUDI KAMPUS\SEMESTER 4\WEB\PWL2025\Minggu10\PWL_POS> php artisan make:controller Api/LevelController
[INFO] Controller [D:\Documents\Polinema\STUDI KAMPUS\SEMESTER 4\WEB\PWL2025\Minggu10\PWL_POS\app\Http\Controllers\Api\LevelController.php] created successfully.
```

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

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

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



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

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

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

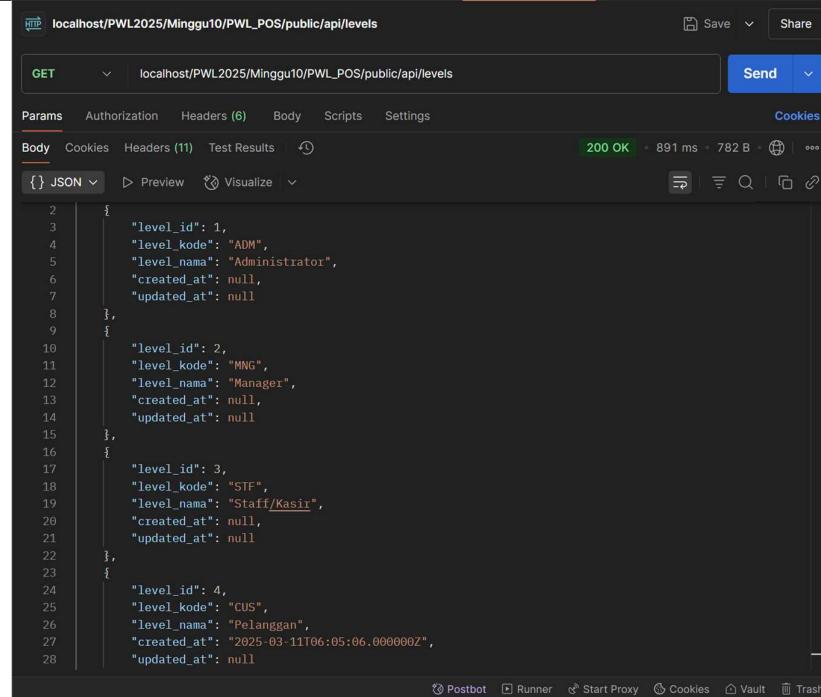
public function destroy(LevelModel $user)
{
    $user->delete();

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

3. Kemudian kita lengkapi routes pada api.php.

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

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



The screenshot shows a Postman interface with a GET request to the URL `localhost/PWL2025/Minggu10/PWL_POS/public/api/levels`. The response status is 200 OK with a response time of 891 ms and a body size of 782 B. The response body is a JSON array containing four objects, each representing a level:

```
[{"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}, {"level_id": 3, "level_kode": "STF", "level_nama": "Staff/Kasir", "created_at": null, "updated_at": null}, {"level_id": 4, "level_kode": "CUS", "level_nama": "Pelanggan", "created_at": "2023-03-11T06:05:06.000000Z", "updated_at": null}]
```

Hasil ini memberikan seluruh data level yang ada di database.

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 | Send

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
level_kode	Text SPV		
level_nama	Text Supervisor		
Key	Value	Description	

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

Body Cookies Headers (11) Test Results

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 screenshot hasil percobaan Anda.

HTTP localhost/PWL2025/Minggu10/PWL_POS/public/api/levels | Save Share

POST | localhost/PWL2025/Minggu10/PWL_POS/public/api/levels | Send

Params Authorization Headers (8) Body Scripts Settings Cookies

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

Key	Value	Description	Bulk Edit
level_kode	Text SPV		
level_nama	Text Supervisor		
Key	Value	Description	

Body Cookies Headers (11) Test Results

{ } JSON Preview Visualize

201 Created 604 ms 521 B

```
1 {  
2   "level_kode": "SPV",  
3   "level_nama": "Supervisor",  
4   "updated_at": "2025-04-29T02:16:05.000000Z",  
5   "created_at": "2025-04-29T02:16:05.000000Z",  
6   "level_id": 6  
7 }
```

	Ubah	Salin	Hapus	level_id	level_kode	level_nama	created_at	updated_at
<input type="checkbox"/>				1	ADM	Administrator	NULL	NULL
<input type="checkbox"/>				2	MNG	Manager	NULL	NULL
<input type="checkbox"/>				3	STF	Staff/Kasir	NULL	NULL
<input type="checkbox"/>				4	CUS	Pelanggan	2025-03-11 06:05:06	NULL
<input type="checkbox"/>				6	SPV	Supervisor	2025-04-29 02:16:05	2025-04-29 02:16:05

Dari metode POST tersebut, akan mengisi data baru di database dengan nama SPV, Supervisor.

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



The screenshot shows a POST request in Postman. The URL is `localhost/PWL2025/Minggu10/PWL_POS/public/api/levels?level_id=6&level_kode=SPV&level_nama=Sup...`. The 'Params' tab has three entries: `level_id` (Value: 6), `level_kode` (Value: SPV), and `level_nama` (Value: Supervisor). The 'Body' tab shows the JSON response:

```
20 "created_at": null,
21 "updated_at": null
22 },
23 {
24   "level_id": 4,
25   "level_kode": "CUS",
26   "level_nama": "Pelanggan",
27   "created_at": "2025-03-11T06:05:06.000000Z",
28   "updated_at": null
29 },
30 [
31   {
32     "level_id": 6,
33     "level_kode": "SPV",
34     "level_nama": "Supervisor",
35     "created_at": "2025-04-29T02:16:05.000000Z",
36     "updated_at": "2025-04-29T02:16:05.000000Z"
37   ]
]
```

Jika saya ingin melakukan metode GET dengan mengisi identity level nya, maka Supervisor akan muncul.

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.

The screenshot shows a PUT request in Postman. The URL is `localhost/PWL_POS-main/public/api/levels/4?level_kode=SPR`. The 'Params' tab has one entry: `level_kode` (Value: SPR). The 'Body' tab shows the JSON response:

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

Jelaskan dan berikan screenshot hasil percobaan Anda.



The screenshot shows a POSTMAN interface with the following details:

- URL:** `localhost/PWL2025/Minggu10/PWL_POS/public/api/levels/6?level_kode=SPR`
- Method:** PUT
- Query Params:** `level_kode` set to `SPR`.
- Body (JSON):**

```
1 [  
2   {  
3     "level_id": 6,  
4     "level_kode": "SPR",  
5     "level_nama": "Supervisor",  
6     "created_at": "2025-04-29T02:16:05.000000Z",  
7     "updated_at": "2025-04-29T02:33:17.000000Z"  
8   }  
9 ]
```
- Response:** 200 OK, 496 ms, 518 B.

Dari Program dengan metode PUT tersebut, akan memperbarui data pada Supervisor.



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

The screenshot shows a Postman interface with a DELETE request to `localhost/PWL2025/Minggu10/PWL_POS/public/api/levels/6?level_kode=SPR`. The response is a 200 OK status with the following JSON body:

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

9. Lakukan commit perubahan file pada Github.

TUGAS

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

m_user

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



```
<?php

namespace App\Http\Controllers\Api;

use App\Http\Controllers\Controller;
use Illuminate\Http\Request;
use App\Models\UserModel;

6 references | 0 implementations
class UserController extends Controller
{
    1 reference | 0 overrides
    public function index(): Collection
    {
        return UserModel::all();
    }

    1 reference | 0 overrides
    public function store(Request $request): JsonResponse|mixed
    {
        $user = UserModel::create($request->all());
        return response()->json($user, 201);
    }

    1 reference | 0 overrides
    public function show(UserModel $user): UserModel
    {
        return $user;
    }

    1 reference | 0 overrides
    public function update(Request $request, UserModel $user): UserModel
    {
        $user->update($request->all());
        return $user;
    }

    1 reference | 0 overrides
    public function destroy(UserModel $user): JsonResponse|mixed
    {
        $user->delete();

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

Index:

localhost/PWL2025/Minggu10/PWL_POS/public/api/users?user_id&level_id&username&password

GET

localhost/PWL2025/Minggu10/PWL_POS/public/api/users?user_id&level_id&username&password

Send

Params

- user_id
- level_id
- username
- password

Body

JSON

200 OK

66, "user_id": 42, "level_id": 1, "username": "admin55", "nama": "Admin 55", "profile_picture": null, "created_at": "2025-04-15T14:54:01.000000Z", "updated_at": "2025-04-15T14:54:01.000000Z" 67, 68, 69, 70, 71, 72, 73, 74, 75, "user_id": 43, "level_id": 2, "username": "pengguna1", "nama": "Pengguna 1", "profile_picture": null, "created_at": "2025-04-28T16:11:03.000000Z", "updated_at": "2025-04-28T16:11:03.000000Z" 76, 77, 78, 79, 80, 81, 82, 83]



Store:

The screenshot shows a POST request to `localhost/PWL2025/Minggu10/PWL_POS/public/api/users?user_id&level_id&username&password`. The body is set to form-data with the following fields:

Key	Value	Description
level_id	3	
username	manager25	
nama	Manager 25	
password	12345	

The response is a **201 Created** status with a time of 23.31 s and a size of 534 B.

```
1 {  
2   "level_id": "3",  
3   "username": "manager25",  
4   "name": "Manager 25",  
5   "updated_at": "2025-04-29T02:56:47.000000Z",  
6   "created_at": "2025-04-29T02:56:47.000000Z",  
7   "user_id": 45  
8 }
```

Show:

The screenshot shows a GET request to `localhost/PWL2025/Minggu10/PWL_POS/public/api/users?user_id=3&level_id=manager25&username=Man...`. The query parameters are:

Key	Value	Description
user_id	3	
level_id	manager25	
username	Manager 25	
password	12345	

The response is a **200 OK** status with a time of 684 ms and a size of 2.01 KB.

```
84 {  
85   "user_id": 45,  
86   "level_id": 3,  
87   "username": "manager25",  
88   "name": "Manager 25",  
89   "profile_picture": null,  
90   "created_at": "2025-04-29T02:56:47.000000Z",  
91   "updated_at": "2025-04-29T02:56:47.000000Z"  
92 }
```



Update:

The screenshot shows a POSTMAN interface. The URL is `localhost/PWL2025/Minggu10/PWL_POS/public/api/users/45?nama=Manager 2525`. The method is set to `PUT`. In the `Params` tab, there are two entries: `Key` and `nama`, both with the value `Manager 2525`. The `Body` tab shows a JSON response with the following data:

```
1 {  
2     "user_id": 45,  
3     "level_id": 3,  
4     "username": "manager25",  
5     "nama": "Manager 2525",  
6     "profile_picture": null,  
7     "created_at": "2025-04-29T02:56:47.000000Z",  
8     "updated_at": "2025-04-29T03:08:13.000000Z"  
9 }
```

The response status is `200 OK` with a time of `479 ms` and a size of `552 B`.

Destroy:

The screenshot shows a POSTMAN interface. The URL is `localhost/PWL2025/Minggu10/PWL_POS/public/api/users/45?nama=Manager 2525`. The method is set to `DELETE`. In the `Params` tab, there are two entries: `Key` and `nama`, both with the value `Manager 2525`. The `Body` tab shows a JSON response with the following data:

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

The response status is `200 OK` with a time of `573 ms` and a size of `418 B`.

m_kategori

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



```
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 overrides
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            $kategori = KategoriModel::create($request->all());
22            return response()->json($kategori, 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            $kategori->update($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
43            return response()->json([
44                'success' => true,
45                'message' => 'Data kategori terhapus',
46            ]);
47        }
48    }
49 }
```

Index:

HTTP New Collection / localhost/PWL2025/Minggu10/PWL_POS/public/api/users/45?nama=Manager 2525

GET localhost/PWL2025/Minggu10/PWL_POS/public/api/kategori

Params Authorization Headers (8) Body Scripts Settings Cookies

Query Params

Key	Value	Description	... Bulk Edit
Key	Value	Description	...

Body Cookies Headers (11) Test Results 200 OK 707 ms 1.17 KB Save Response

{ } JSON ▾ ▶ Preview ⚡ Visualize ▾

```
1 [
2   {
3     "kategori_id": 1,
4     "kategori.kode": "KSM",
5     "kategori.nama": "Kendaraan Sepeda Motor",
6     "created_at": null,
7     "updated_at": null
8   },
9   {
10     "kategori_id": 2,
11     "kategori.kode": "AMMN",
12     "kategori.nama": "Alat Makan",
13     "created_at": null,
14     "updated_at": "2025-03-19T07:46:13.800000Z"
15   },
16   {
17     "kategori_id": 3,
18     "kategori.kode": "PKS",
19     "kategori.nama": "Perkakas",
20     "created_at": null,
21     "updated_at": null
22 ]
```

Postbot Runner Start Proxy Cookies Vault Trash

Store:



HTTP New Collection / localhost/PWL2025/Minggu10/PWL_POS/public/api/users/45?nama=Manager 2525

POST localhost/PWL2025/Minggu10/PWL_POS/public/api/kategoris

Params Authorization Headers (8) Body Scripts Settings Cookies

Body

Key Value Description

kategori_id Text 8

kategori_kode Text IT

kategori_nama Text Produk IT

Key Value Description

Body Cookies Headers (11) Test Results 201 Created 373 ms 528 B Save Response

{ } JSON Preview Visualize

```
1 {  
2   "kategori_kode": "IT",  
3   "kategori_nama": "Produk IT",  
4   "updated_at": "2025-04-29T03:34:32.000000Z",  
5   "created_at": "2025-04-29T03:34:32.000000Z",  
6   "kategori_id": 8  
7 }
```

Show:

HTTP New Collection / localhost/PWL2025/Minggu10/PWL_POS/public/api/users/45?nama=Manager 2525

GET localhost/PWL2025/Minggu10/PWL_POS/public/api/kategoris

Params Authorization Headers (8) Body Scripts Settings Cookies

Query Params

Body

Key Value Description

Key

Body Cookies Headers (11) Test Results 200 OK 221 ms 1.32 KB Save Response

{ } JSON Preview Visualize

```
37 {  
38   "kategori_id": 6,  
39   "kategori_kode": "CML",  
40   "kategori_nama": "Comilan",  
41   "created_at": "2025-03-19T13:16:23.000000Z",  
42   "updated_at": null  
43 },  
44 {  
45   "kategori_id": 7,  
46   "kategori_kode": "MNR",  
47   "kategori_nama": "Minuman Ringan",  
48   "created_at": "2025-03-19T13:17:04.000000Z",  
49   "updated_at": null  
50 },  
51 {  
52   "kategori_id": 8,  
53   "kategori_kode": "IT",  
54   "kategori_nama": "Produk IT",  
55   "created_at": "2025-04-29T03:34:32.000000Z",  
56   "updated_at": "2025-04-29T03:34:32.000000Z"  
57 }  
58 ]
```

Update:



The screenshot shows a POSTMAN interface with the following details:

- Method:** PUT
- URL:** `localhost/PWL2025/Minggu10/PWL_POS/public/api/users/45?nama=Manager 2525`
- Headers:** Authorization, Headers (7), Body, Scripts, Settings
- Params:** `kategori_nama` (Value: `Produk Digital`)
- Body:** JSON response (200 OK):

```
1 {  
2     "kategori_id": 8,  
3     "kategori_kode": "IT",  
4     "kategori_nama": "Produk Digital",  
5     "created_at": "2025-04-29T03:34:32.060000Z",  
6     "updated_at": "2025-04-29T03:38:02.060000Z"  
7 }
```

Destroy:

The screenshot shows a POSTMAN interface with the following details:

- Method:** DELETE
- URL:** `localhost/PWL2025/Minggu10/PWL_POS/public/api/users/45?nama=Manager 2525`
- Headers:** Authorization, Headers (6), Body, Scripts, Settings
- Params:** `Key`
- Body:** JSON response (200 OK):

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

m_barang

```
Route::get('barangs', [BarangController::class, 'index']);  
Route::post('barangs', [BarangController::class, 'store']);  
Route::get('barangs/{barang}', [BarangController::class, 'show']);  
Route::put('barangs/{barang}', [BarangController::class, 'update']);  
Route::delete('barangs/{barang}', [BarangController::class, 'destroy']);
```



Minggu10 > PWL_POS > 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 class BarangController extends Controller
10 {
11     public function index(): Collection
12     {
13         return BarangModel::all();
14     }
15
16     public function store(Request $request): JsonResponse|mixed
17     {
18         $barang = BarangModel::create($request->all());
19         return response()->json($barang, 201);
20     }
21
22     public function show(BarangModel $barang): BarangModel
23     {
24         return $barang;
25     }
26
27     public function update(Request $request, BarangModel $barang): JsonResponse|mixed
28     {
29         $barang->update($request->all());
30         return $barang;
31     }
32
33     public function destroy(BarangModel $barang): JsonResponse|mixed
34     {
35         $barang->delete();
36
37         return response()->json([
38             'success' => true,
39             'message' => 'Data barang terhapus',
40         ]);
41     }
42 }
```

index:

HTTP New Collection / localhost/PWL2025/Minggu10/PWL_POS/public/api/users/45?nama=Manager 2525

GET localhost/PWL2025/Minggu10/PWL_POS/public/api/barangs Send

Params Authorization Headers (6) Body Scripts Settings Cookies

Query Params

Key	Value	Description
Key	Value	Description

Body Cookies Headers (11) Test Results 200 OK · 261 ms · 1.96 KB · Save Response

{ } JSON ▾ Preview Visualize

```
11 {
12     "barang_id": 2,
13     "kategori_id": 1,
14     "barang_kode": "HLM001",
15     "barang_nama": "Helm SNI",
16     "harga_beli": "150000",
17     "harga_jual": "200000",
18     "created_at": null,
19     "updated_at": null
20 },
21 {
22     "barang_id": 3,
23     "kategori_id": 2,
24     "barang_kode": "SDK001",
25     "barang_nama": "Sendok Stainless",
26     "harga_beli": "5000",
27     "harga_jual": "10000",
28     "created_at": null,
29     "updated_at": null
30 }
```



Store:

The screenshot shows a POST request to `localhost/PWL2025/Minggu10/PWL_POS/public/api/users/45?nama=Manager 2525/barangs`. The request body contains the following data:

Key	Value	Description
kategori_id	5	
barang_kode	TPW001	
barang_nama	Tupperware	
harga_beli	45000	
harga_jual	60000	

The response status is **201 Created**.

```
1 {
2   "kategori_id": "5",
3   "barang_kode": "TPW001",
4   "barang_nama": "Tupperware",
5   "harga_beli": "45000",
6   "harga_jual": "60000",
7   "updated_at": "2025-04-29T03:47:31.000000Z",
8   "created_at": "2025-04-29T03:47:31.000000Z",
9   "barang_id": 33
10 }
```

Show:

The screenshot shows a GET request to `localhost/PWL2025/Minggu10/PWL_POS/public/api/users/45?nama=Manager 2525/barangs`. The response status is **200 OK**.

```
101 },
102 [
103   {
104     "barang_id": 33,
105     "kategori_id": 5,
106     "barang_kode": "TPW001",
107     "barang_nama": "Tupperware",
108     "harga_beli": "45000",
109     "harga_jual": "60000",
110     "created_at": "2025-04-29T03:47:31.000000Z",
111     "updated_at": "2025-04-29T03:47:31.000000Z"
112   }
]
```



Update:

The screenshot shows a POSTMAN interface with the following details:

- Method:** PUT
- URL:** localhost/PWL2025/Minggu10/PWL_POS/public/api/users/45?nama=Manager 2525
- Body (JSON):**

```
1 {  
2     "barang_id": 33,  
3     "kategori_id": 5,  
4     "barang_kode": "IPW001",  
5     "barang_nama": "Tupperware Jingga",  
6     "harga_beli": "45000",  
7     "harga_jual": "60000",  
8     "created_at": "2025-04-29T03:47:31.000000Z",  
9     "updated_at": "2025-04-29T03:56:14.000000Z"  
10 }
```
- Response:** 200 OK (202 ms, 588 B)

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

Destroy:

The screenshot shows a POSTMAN interface with the following details:

- Method:** DELETE
- URL:** localhost/PWL2025/Minggu10/PWL_POS/public/api/users/45?nama=Manager 2525
- Body (JSON):**

```
1 {  
2     "barang_id": 33,  
3     "kategori_id": 5,  
4     "barang_kode": "IPW001",  
5     "barang_nama": "Tupperware Jingga",  
6     "harga_beli": "45000",  
7     "harga_jual": "60000",  
8     "created_at": "2025-04-29T03:47:31.000000Z",  
9     "updated_at": "2025-04-29T03:56:14.000000Z"  
10 }
```
- Response:** 200 OK (223 ms, 420 B)

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

*** Sekian, dan selamat belajar ***