



Nama : Arimbi Putri Hapsari
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 :

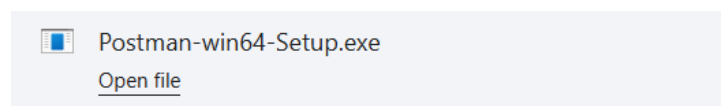
JWT (JSON Web Token) adalah cara untuk mentransfer informasi antara dua pihak secara aman sebagai objek JSON. Ini terdiri dari tiga bagian: header, payload, dan signature. Setelah pengguna berhasil autentikasi, server menghasilkan token JWT yang disematkan dalam permintaan HTTP. Server kemudian memvalidasi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



2. Lakukan instalasi JWT dengan mengetikkan perintah berikut: `composer require tyson/jwt-auth:2.1.1` Pastikan Anda terkoneksi dengan internet.

```
tyson/jwt-auth ..... DONE
yajra/laravel-datatables-buttons ..... DONE
yajra/laravel-datatables-editor ..... DONE
yajra/laravel-datatables-fractal ..... DONE
yajra/laravel-datatables-html ..... DONE
yajra/laravel-datatables-oracle ..... DONE

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

[INFO] No publishable resources for tag [laravel-assets].

Found 1 security vulnerability advisory affecting 1 package.
Run "composer audit" for a full list of advisories.
PS D:\laragon\www\pwl_25\week10\jobsheet\pwl_pos>
```

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

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

```
PS D:\laragon\www\pwl_25\week10\jobsheet\pwl_pos> php artisan vendor:publish --provider="Tyson\JWTAuth\Providers\LaravelServiceProvider"

[INFO] Publishing assets.

Copying file [D:\laragon\www\pwl_25\week10\jobsheet\pwl_pos\vendor\tyson\jwt-auth\config\config.php] to [D:\laragon\www\pwl_25\week10\jobsheet\pwl_pos\config\
jwt.php] DONE
PS D:\laragon\www\pwl_25\week10\jobsheet\pwl_pos>
```

4. Jika perintah di atas berhasil, maka kita akan mendapatkan 1 file baru yaitu `config/jwt.php`. Pada file ini dapat dilakukan konfigurasi jika memang diperlukan.

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

```
PS D:\laragon\www\pwl_25\week10\jobsheet\pwl_pos> php artisan jwt:secret
jwt-auth secret [9xXPqPe2sWdhFr5efWufsyApfZX3oJ3gPgaoV0qa47gJNb15yVMr07Vp1RNk84Ea] set successfully.
PS D:\laragon\www\pwl_25\week10\jobsheet\pwl_pos>
```



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

```
61 JWT_SECRET=9xXPqPe2sWdhFr5efWufsyApfZX3oJ3gPgaoV0qa47gjNb15yVMrO7Vp1RNk84Ea
62
```

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 [];
    }

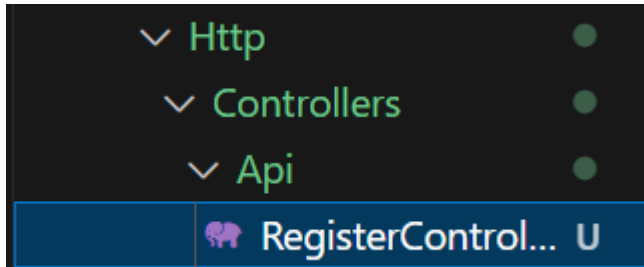
    protected $table = 'm_user';
    protected $primaryKey = 'user_id';
}
```



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.

POST localhost/PWL_POS/public/api/register Send

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

Query Params

Key	Value	Bulk Edit
Key	Value	

Body 422 Unprocessable Content 272 ms 534 B Save Response

Pretty Raw Preview Visualize JSON

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

Jika berhasil akan muncul error validasi seperti gambar di atas.

Lakukan percobaan yang sama dan berikan screenshoot hasil percobaan Anda.



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

- Method:** POST
- URL:** http://127.0.0.1:8000/api/register
- Response Status:** 422 Unprocessable Content
- Response Time:** 543 ms
- Response Size:** 461 B
- Response Body (JSON):**

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

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 Send

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

form-data

	Key	Value	...	Bulk Edit
<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$12$Eb2SrV1jsykINytYGtrHi0DVAKcK5p6EgnZnmbChkPicIu7S0QJJJu",
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
```

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.

HTTP http://127.0.0.1:8000/api/register Save Share

POST http://127.0.0.1:8000/api/register Send

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

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

<input checked="" type="checkbox"/>	username	Text	penggunasatu
<input checked="" type="checkbox"/>	nama	Text	Pengguna1
<input checked="" type="checkbox"/>	password	Text	12345
<input checked="" type="checkbox"/>	password_confirmation	Text	12345
<input checked="" type="checkbox"/>	level_id	Text	2



```
Body Cookies Headers (10) Test Results 201 Created 2.42 s 527 B
{} JSON Preview Visualize
1 {
2   "status": true,
3   "message": "Registrasi berhasil!",
4   "data": {
5     "username": "penggunasatu",
6     "nama": "Pengguna1",
7     "level_id": 2,
8     "updated_at": "2025-04-28T07:33:47.000000Z",
9     "created_at": "2025-04-28T07:33:47.000000Z",
10    "user_id": 29
11  }
12 }
```

7	managerbaru	Manager Baru	Manager	Detail	Edit	Hapus
8	staffkeren	Staff Keren	Staff/Kasir	Detail	Edit	Hapus
9	penggunasatu	Pengguna1	Manager	Detail	Edit	Hapus

Showing 1 to 9 of 9 entries

Previous 1 Next

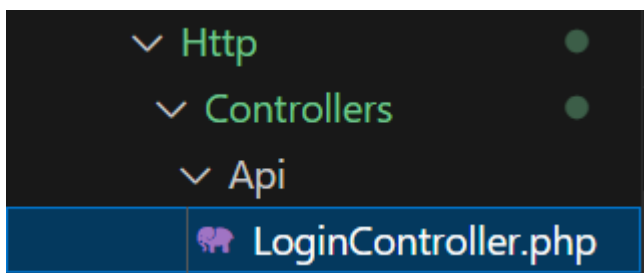
13. Lakukan commit perubahan file pada Github.

Praktikum 2 – Membuat RESTful API Login

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

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

```
PS D:\laragon\www\PWL_25\week10\jobsheet\PWL_POS> php artisan make:controller Api/LoginController
INFO Controller [D:\laragon\www\PWL_25\week10\jobsheet\PWL_POS\app\Http\Controllers\Api>LoginController.php] created successfully.
```



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



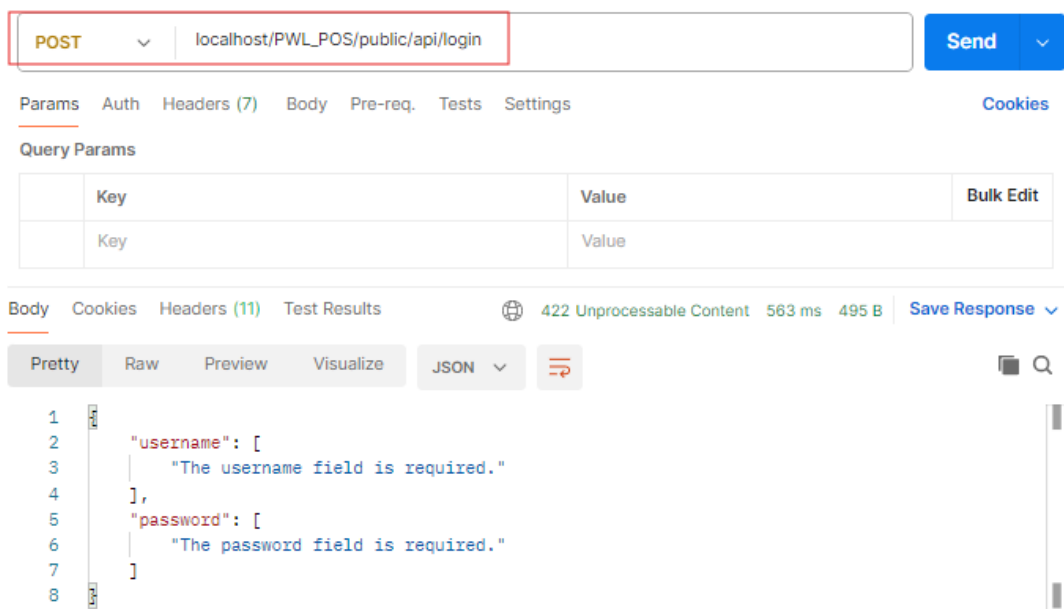
```
1  <?php
2
3  namespace App\Http\Controllers\Api;
4
5  use App\Http\Controllers\Controller;
6  use Illuminate\Http\Request;
7  use Illuminate\Support\Facades\Validator;
8
9  class LoginController extends Controller
10 {
11     public function __invoke(Request $request)
12     {
13         //set validation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



Jika berhasil akan muncul error validasi seperti gambar di atas.

Lakukan percobaan yang sama dan berikan screenshot hasil percobaan Anda.



HTTP <http://127.0.0.1:8000/api/login> Save Share

POST <http://127.0.0.1:8000/api/login> Send

Params Authorization Headers (7) Body Scripts Tests Settings Cookies

Query Params

Key	Value	Description	Bulk Edit
Key	Value	Description	

Body Cookies Headers (10) Test Results 422 Unprocessable Content • 494 ms • 422 B •

{ } JSON Preview Visualize

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

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



POST localhost/PWL_POS/public/api/login Send

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

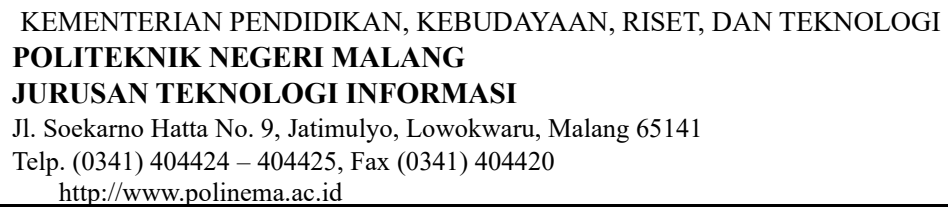
none **form-data** x-www-form-urlencoded raw binary

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

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

Pretty Raw Preview Visualize JSON

```
1  {
2    "success": true,
3    "user": {
4      "user_id": 17,
5      "level_id": 2,
6      "username": "penggunasatu",
7      "nama": "Pengguna 1",
8      "password": "$2y$12$Eb2SxV1jsykINytYGtrHi0DVAKcK5p6EgnZnmbChkPicIu7S0QJJJu",
9      "created_at": "2024-04-22T15:56:04.000000Z",
10     "updated_at": "2024-04-22T15:56:04.000000Z"
11   },
12   "token": "eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJpc3MiOiJodHRwOi8vbG9jYXRob3N0L1BXTF9QT1MtbnVpbi9wdWJsaWVYb2xvZ21uIiwiaWF0Ij01"
13 }
```

HTTP

http://127.0.0.1:8000/api/login

Save

Share

POT

http://127.0.0.1:8000/api/login

Send

Params

Authorization

Headers (8)

Body •

Scripts

Tests

Settings

Cookies

☐ none

☒ form-data

☐ x-www-form-urlencoded

☐ raw

☐ binary

☐ GraphQL

	Key		Value	Description	⋮ Bulk Edit
<input checked="" type="checkbox"/>	username	Text ▾	penggunasatu		
<input checked="" type="checkbox"/>	password	Text ▾	12345		
	Key	Text ▾	Value	Description	

Body

Cookies

Headers (10)

Test Results

↺

200 OK • 420 ms • 846 B • 🌐 ⋮

{ } JSON ▾

▶ Preview

🔗 Visualize ▾

➡ ☰ 🔍 📄 🔗

```
1 {  
2   "success": true,  
3   "user": {  
4     "user_id": 29,  
5     "level_id": 2,  
6     "username": "penggunasatu",  
7     "nama": "Pengguna1",  
8     "user_profile_picture": null,  
9     "created_at": "2025-04-28T07:33:47.000000Z",  
10    "updated_at": "2025-04-28T07:33:47.000000Z"  
11  },  
12  "token": "eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.  
      eyJpc3MiOiJodHRwOi8vMTI3LjAuMC4xOjgwMDAvYXBPL2xvZ2luIiwiaWF0IjoxNzQ1ODIzMzkzLCJleHAiOiE3NDU0MzA  
      5OTMsIm5iZiI6MTc0NTgyNm5MyWianRpIjoiriRVNzb3R3S2hxWjcwQWlwdSIInN1YiI6IjI5IiwicHJ2IjoiriNDVkZjg4Mz  
      RmMWI1OGY3MGVmyTYyYWFlZGVmNDIzNDEzNDEzNDEzAwNjkwYyJ9.ymxZu-BeIXU2Lk-smkdYh--deCs8IjjCs0oA5aWI-FM"
```

-
- 10 - PWL 2023/2024
Hal. 15 / 39
- Jobsheet



HTTP <http://127.0.0.1:8000/api/login> Save Share

POST <http://127.0.0.1:8000/api/login> Send

Params Authorization Headers (8) Body Scripts Tests Settings Cookies

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

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

Body Cookies Headers (10) Test Results 401 Unauthorized 96 ms 381 B

{ } JSON Preview Visualize

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

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.**

HTTP <http://127.0.0.1:8000/api/login> Save Share

GET <http://127.0.0.1:8000/api/login> Send

Params Authorization Headers (8) Body Scripts Tests Settings Cookies

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

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

Body Cookies Headers (9) Test Results 405 Method Not Allowed 337 ms 1022.46 KB

HTML Preview Visualize

```
1 <!DOCTYPE html>
2 <html lang="en" class="auto">
3 <!--
4 Symfony\Component\HttpKernel\Exception\MethodNotAllowedHttpException: The GET method is not supported
  for route api/login. Supported methods: POST. in file
  D:\laragon\www\PWL_25\week10\jobsheet\PWL_POS\vendor\laravel\framework\src\Illuminate\Routing\Abstr
  actRouteCollection.php on line 122
5
6 #0 D:\laragon\www\PWL_25\week10\jobsheet\PWL_POS\vendor\laravel\framework\src\Illuminate\Routing\Abstra
```



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

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

8. Lakukan commit perubahan file pada Github.

Praktikum 3 – Membuat RESTful API Logout

1. Tambahkan kode berikut pada file .env

```
JWT_SHOW_BLACKLIST_EXCEPTION=true
```

```
61 JWT_SECRET=9xXPqPe2sWdhFr5efWufsyApfZX3oJ3gPgaoV0qa47gjNb15yVMr07Vp1RNk84Ea
62 JWT_SHOW_BLACKLIST_EXCEPTION=true
```

2. Buat Controller baru dengan nama LogoutController.

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

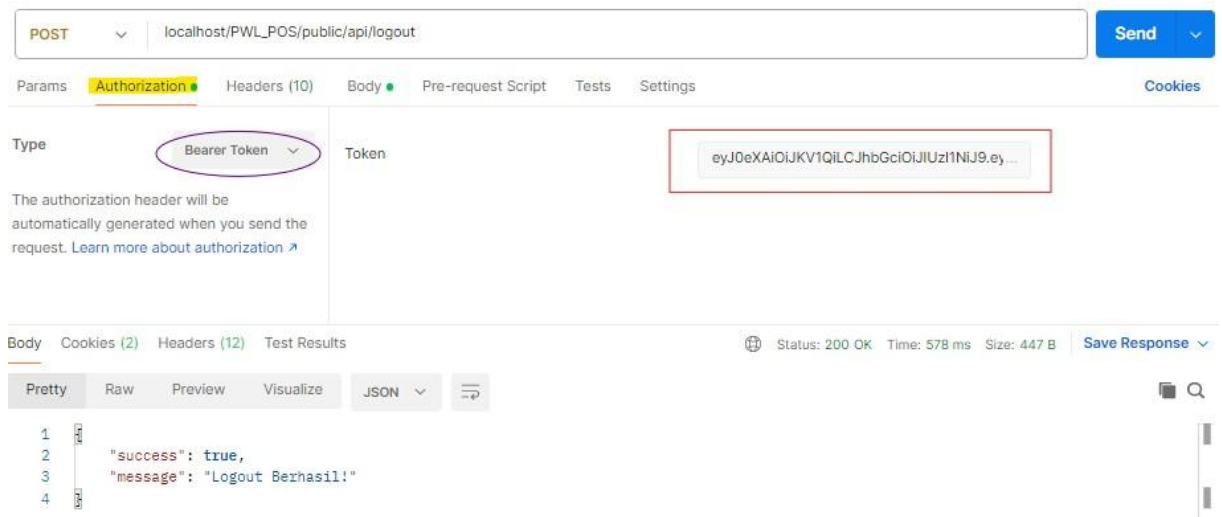
```
INFO Controller [D:\laragon\www\PWL_25\week10\jobsheet\PWL_POS\app\Http\Controllers\Api\LogoutController.php] created successfully.
PS D:\laragon\www\PWL_25\week10\jobsheet\PWL_POS>
```

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

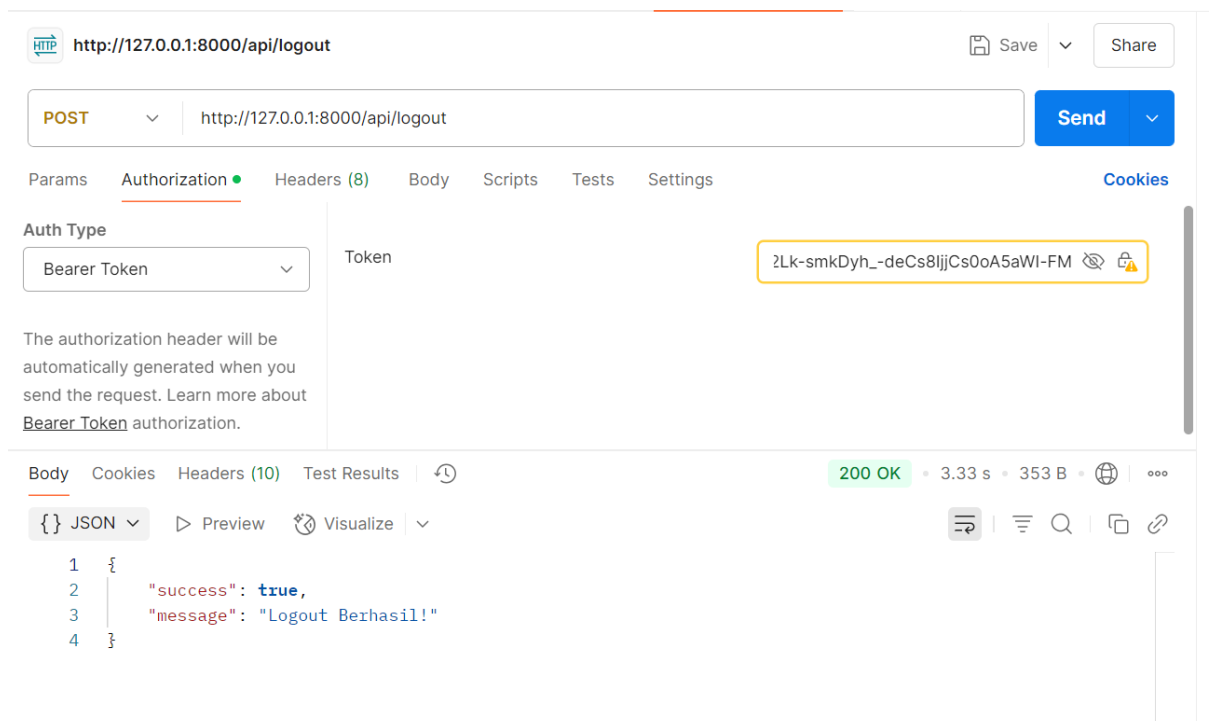


```
1  <?php
2
3  namespace App\Http\Controllers\Api;
4  use Illuminate\Http\Request;
5  use App\Http\Controllers\Controller;
6  use Tymon\JWTAuth\Facades\JWTAuth;
7  use Tymon\JWTAuth\Exceptions\JWTException;
8  use Tymon\JWTAuth\Exceptions\TokenExpiredException;
9  use Tymon\JWTAuth\Exceptions\TokenInvalidException;
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.
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

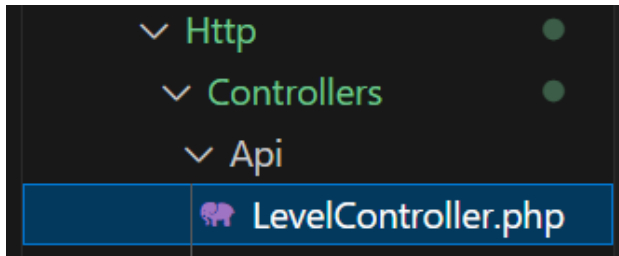
Pada praktikum ini kita akan menggunakan tabel `m_level` untuk dimodifikasi menggunakan RESTful API.



1. Pertama, buat controller untuk mengolah API pada data level.

`php artisan make:controller Api/LevelController`

```
INFO Controller [D:\laragon\www\PWL_25\week10\jobsheet\PWL_POS\app\Http\Controllers\Api\LevelController.php] created successfully.  
PS D:\laragon\www\PWL_25\week10\jobsheet\PWL_POS>
```



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

```
namespace App\Http\Controllers\Api;  
use App\Http\Controllers\Controller;  
use Illuminate\Http\Request;  
use App\Models\LevelModel;  
  
class LevelController extends Controller  
{  
    public function index()  
    {  
        return LevelModel::all();  
    }  
}
```




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

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

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

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

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

3. Kemudian kita lengkapi routes pada api.php.

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

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

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



Request 1: `http://127.0.0.1:8000/api/levels`

Response 1: 200 OK, 1.19 s, 718 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}]
```

Request 2: `http://127.0.0.1:8000/api/levels?level_kode=ADM&level_nama=Administrator`

Response 2: 200 OK, 64 ms, 718 B

```
[{"level_id": 1, "level_kode": "ADM", "level_nama": "Administrator", "created_at": null, "updated_at": null}]
```

5. Kemudian, lakukan percobaan penambahan data dengan URL : `localhost/PWL_POSmain/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
<input checked="" type="checkbox"/> level_kode	Text SPV			
<input checked="" type="checkbox"/> level_nama	Text Supervisor			
Key	Text Value	Description		

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

Pretty Raw Preview Visualize JSON

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

Jelaskan dan berikan screenshoot hasil percobaan Anda.

HTTP http://127.0.0.1:8000/api/levels?level_kode=SPV&level_nama=Supervisor Save Share

POST http://127.0.0.1:8000/api/levels?level_kode=SPV&level_nama=Supervisor Send

Params Authorization Headers (7) Body Scripts Tests Settings Cookies

Query Params

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

Body Cookies Headers (10) Test Results 201 Created 65 ms 458 B

{ } JSON Preview Visualize

```
1 {
2   "level_kode": "SPV",
3   "level_nama": "Supervisor",
4   "updated_at": "2025-04-28T08:43:28.000000Z",
5   "created_at": "2025-04-28T08:43:28.000000Z",
6   "level_id": 5
7 }
```

Data berhasil ditambahkan

5	SPV	Supervisor	Detail	Edit	Hapus
---	-----	------------	------------------------	----------------------	-----------------------

Showing 1 to 5 of 5 entries

Previous 1 Next



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

HTTP http://127.0.0.1:8000/api/levels?level_kode=SPV&level_nama=Supervisor Save Share </>

GET http://127.0.0.1:8000/api/levels?level_kode=SPV&level_nama=Supervisor Send

Params Authorization Headers (6) Body Scripts Tests Settings Cookies

Query Params

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

Body Cookies Headers (10) Test Results 200 OK • 69 ms • 864 B •

{ } JSON Preview Visualize

```
29   },
30   {
31     "level_id": 5,
32     "level_kode": "SPV",
33     "level_nama": "Supervisor",
34     "created_at": "2025-04-28T08:43:28.000000Z",
35     "updated_at": "2025-04-28T08:43:28.000000Z"
36   }
37 ]
```

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



PUT localhost/PWL_POS-main/public/api/levels/4?level_kode=SPR Send

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

Query Params

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

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

Pretty Raw Preview Visualize JSON

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

Jelaskan dan berikan screenshoot hasil percobaan Anda.

HTTP http://127.0.0.1:8000/api/levels/5?level_kode=SPR&level_nama=Supervisor Save Share

PUT http://127.0.0.1:8000/api/levels/5?level_kode=SPR&level_nama=Supervisor Send

Params Authorization Headers (7) Body Scripts Tests Settings Cookies

Query Params

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

Body Cookies Headers (10) Test Results 200 OK 114 ms 455 B

{ } JSON Preview Visualize

```
1 [
2   {
3     "level_id": 5,
4     "level_kode": "SPR",
5     "level_nama": "Supervisor",
6     "created_at": "2025-04-28T08:43:28.000000Z",
7     "updated_at": "2025-04-28T08:50:15.000000Z"
8   }
9 ]
```

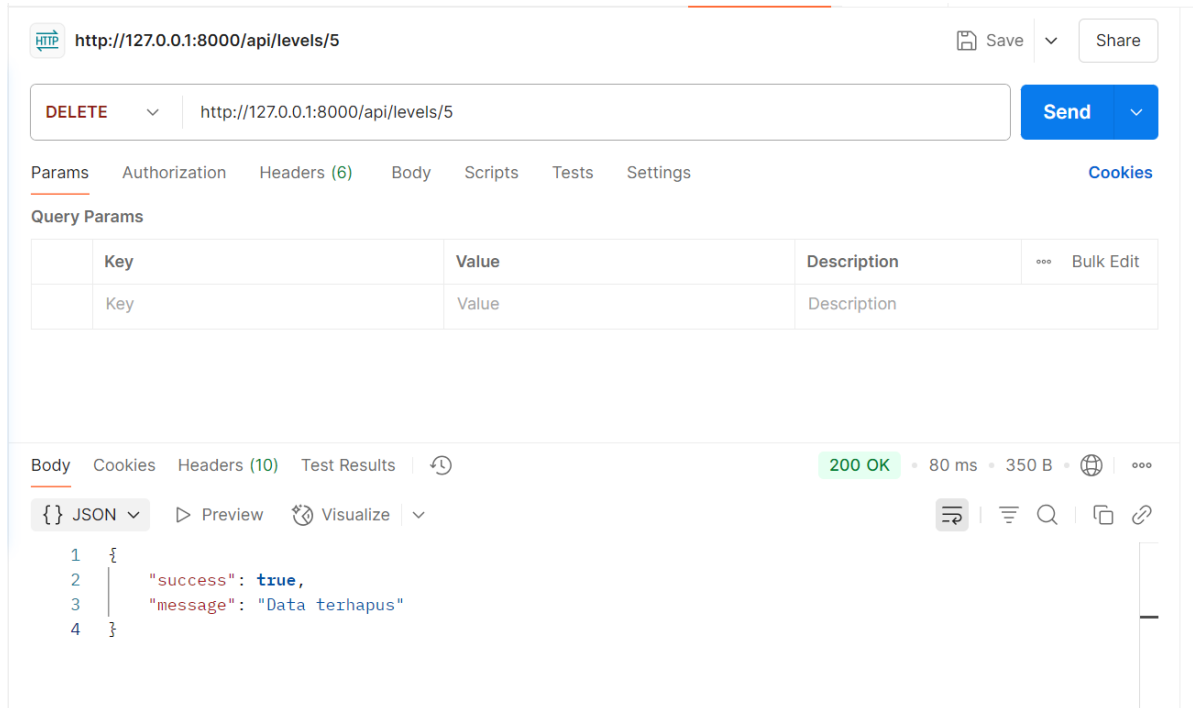
Data berhasil diedit

5	SPR	Supervisor	Detail	Edit	Hapus
---	-----	------------	--------	------	-------

Showing 1 to 5 of 5 entries Previous



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



9. Lakukan commit perubahan file pada Github.

TUGAS

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

1. API pada m_user

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



```
PS D:\laragon\www\PWL_25\week10\jobsheet\PWL_POS> php artisan make:controller Api/UserController
```

```
INFO Controller [D:\laragon\www\PWL_25\w.
```

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

b. Tambahkan route pada routes/api.php



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

```
32 // API User
33 Route::get('/users', [UserController::class, 'index']);
34 Route::post('/users', [UserController::class, 'store']);
35 Route::get('/users/{user}', [UserController::class, 'show']);
36 Route::put('/users/{user}', [UserController::class, 'update']);
37 Route::delete('/users/{user}', [UserController::class, 'destroy']);
```

c. Test CURD

⇒ Create user baru dengan method POST

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

- URL:** `http://127.0.0.1:8000/api/users?level_id=3&username=kasir03&nama=putri&password=12345`
- Method:** POST
- Params:**

Key	Value	Description
level_id	3	
username	kasir03	
nama	putri	
password	12345	
- Response:** 200 OK, 956 ms, 481 B. The response body is a JSON object:

```
{  "data": {    "level_id": "3",    "username": "kasir03",    "nama": "putri",    "updated_at": "2025-04-28T09:20:19.000000Z",    "created_at": "2025-04-28T09:20:19.000000Z",    "user_id": 31  },  "status": 201}
```

Data berhasil ditambahkan

10	kasir03	putri	Staff/Kasir	Detail Edit Hapus
----	---------	-------	-------------	---

Showing 1 to 10 of 10 entries

⇒ Read/menampilkan detail user dengan method GET



HTTP http://127.0.0.1:8000/api/users?level_id=3&username=kasir03&nama=putri&password=12345 Save Share

GET http://127.0.0.1:8000/api/users?level_id=3&username=kasir03&nama=putri&password=12345 Send

Params Authorization Headers (6) Body Scripts Tests Settings Cookies

Key	Value	Description
level_id	3	
username	kasir03	
nama	putri	
password	12345	

Body Cookies Headers (10) Test Results 200 OK • 179 ms • 1.96 KB

{ } JSON Preview Visualize

```
83 {
84   "user_id": 31,
85   "level_id": 3,
86   "username": "kasir03",
87   "nama": "putri",
88   "user_profile_picture": null,
89   "created_at": "2025-04-28T09:20:19.000000Z",
90   "updated_at": "2025-04-28T09:20:19.000000Z"
91 }
```

⇒ **Update data user dengan method PUT**

Params Authorization Headers (7) Body Scripts Settings Cookies

Query Params

Key	Value	Description
nama	Hailey	

Body Cookies Headers (11) Test Results 200 OK • 2.12 s • 542 B Save Response

{ } JSON Preview Visualize

```
1 {
2   "user_id": 35,
3   "level_id": 3,
4   "profile_photo": null,
5   "username": "kasir03",
6   "nama": "Hailey",
7   "created_at": "2025-04-27T07:29:02.000000Z",
8   "updated_at": "2025-04-27T07:36:55.000000Z"
9 }
```

⇒ **Delete data user dengan method DELETE**



<http://127.0.0.1:8000/api/users/3> Save Share

DELETE

<http://127.0.0.1:8000/api/users/3>

Send

Params Authorization Headers (6) Body Scripts Tests Settings Cookies

Query Params

	Key	Value	Description	...	Bulk Edit
	Key	Value	Description		

Body Cookies Headers (10) Test Results

200 OK • 244 ms • 350 B •

{ } JSON

Preview

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



2. API pada m_kategori

⇒ Buat controller API pada folder API dengan nama KategoriController

```
▼ TERMINAL
PS D:\laragon\www\PWL_25\week10\jobsheet\PWL_POS> php artisan make:controller Api/KategoriController

INFO Controller [D:\laragon\www\PWL_25\week10\jobsheet\PWL_POS\app\Http\Controllers\Api\KategoriController.php] created successfully.
```

```
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 use Illuminate\Database\Eloquent\Collection;
9 use Illuminate\Http\JsonResponse;
10
11 class KategoriController extends Controller
12 {
13     public function index()
14     {
15         return KategoriModel::all();
16     }
17
18     public function store(Request $request)
19     {
20         $kategori = KategoriModel::create($request->all());
21         return response()->json($kategori, 201);
22     }
23
24     public function show(KategoriModel $kategori)
25     {
26         return $kategori;
27     }
28
29     public function update(Request $request, KategoriModel $kategori)
30     {
31         $kategori->update($request->all());
32         return $kategori;
33     }
34
35     public function destroy(KategoriModel $kategori)
36     {
37         $kategori->delete();
38
39         return response()->json([
40             'success' => true,
41             'message' => 'Data kategori terhapus',
42         ]);
43     }
44 }
```



b. Tambahkan route pada routes/api.php

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

```
39 // API Kategori
40 Route::get('/kategoris', [KategoriController::class, 'index']);
41 Route::post('/kategoris', [KategoriController::class, 'store']);
42 Route::get('/kategoris/{kategori}', [KategoriController::class, 'show']);
43 Route::put('/kategoris/{kategori}', [KategoriController::class, 'update']);
44 Route::delete('/kategoris/{kategori}', [KategoriController::class, 'destroy']);
45
```

C. Test Curd

⇒ **Create kategori baru dengan method POST**

HTTP http://127.0.0.1:8000/api/kategoris?kategori_kode=KCW&kategori_nama=Kitchen ware Save Share

POST http://127.0.0.1:8000/api/kategoris?kategori_kode=KCW&kategori_nama=Kitchen ware Send

Params Authorization Headers (7) Body Scripts Tests Settings Cookies

Query Params

<input checked="" type="checkbox"/>	Key	Value	Description	...	Bulk Edit
<input checked="" type="checkbox"/>	kategori_kode	KCW			
<input checked="" type="checkbox"/>	kategori_nama	Kitchen ware			
	Key	Value	Description		

Body Cookies Headers (10) Test Results 201 Created • 127 ms • 470 B •

{ } JSON Preview Visualize

```
1 {
2   "kategori_kode": "KCW",
3   "kategori_nama": "Kitchen ware",
4   "updated_at": "2025-04-28T09:00:40.000000Z",
5   "created_at": "2025-04-28T09:00:40.000000Z",
6   "kategori_id": 20
7 }
```

Data berhasil ditambahkan

6	KAT006	Camilan	Detail Edit Hapus
7	KCW	Kitchen ware	Detail Edit Hapus

Showing 1 to 7 of 7 entries Previous 1

⇒ **Menampilkan detail kategori dengan method GET**



HTTP http://127.0.0.1:8000/api/kategoris?kategori_kode=KCW&kategori_nama=Kitchen ware Save Share

GET http://127.0.0.1:8000/api/kategoris?kategori_kode=KCW&kategori_nama=Kitchen ware Send

Params Authorization Headers (6) Body Scripts Tests Settings Cookies

Query Params

<input checked="" type="checkbox"/>	Key	Value	Description	...	Bulk Edit
<input checked="" type="checkbox"/>	kategori_kode	KCW			
<input checked="" type="checkbox"/>	kategori_nama	Kitchen ware			
	Key	Value	Description		

Body Cookies Headers (10) Test Results 200 OK • 71 ms • 1.1 KB • ...

{ } JSON Preview Visualize

```
43  },
44  {
45      "kategori_id": 20,
46      "kategori_kode": "KCW",
47      "kategori_nama": "Kitchen ware",
48      "created_at": "2025-04-28T09:00:40.000000Z",
49      "updated_at": "2025-04-28T09:00:40.000000Z"
50  }
51  ]
```

Postbot Runner Start Proxy Cookies Vault Trash

⇒ Update data kategori dengan method PUT

HTTP http://127.0.0.1:8000/api/kategoris/20?kategori_nama=Peralatan dapur Save Share

PUT http://127.0.0.1:8000/api/kategoris/20?kategori_nama=Peralatan dapur Send

Params Authorization Headers (7) Body Scripts Tests Settings Cookies

Query Params

<input checked="" type="checkbox"/>	Key	Value	Description	...	Bulk Edit
<input checked="" type="checkbox"/>	kategori_nama	Peralatan dapur			
	Key	Value	Description		

Body Cookies Headers (10) Test Results 200 OK • 96 ms • 468 B • ...

{ } JSON Preview Visualize

```
1  {
2      "kategori_id": 20,
3      "kategori_kode": "KCW",
4      "kategori_nama": "Peralatan dapur",
5      "created_at": "2025-04-28T09:00:40.000000Z",
6      "updated_at": "2025-04-28T09:05:19.000000Z"
7  }
```



⇒ Delete data kategori dengan method DELETE

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

- URL:** `http://127.0.0.1:8000/api/kategoris/20?kategori_nama=Peralatan dapur`
- Method:** DELETE
- Send Button:** A blue button labeled "Send".
- Params:** A tab labeled "Params" is active, showing a table of query parameters.
- Query Params Table:**

<input checked="" type="checkbox"/>	Key	Value	Description	...	Bulk Edit
<input checked="" type="checkbox"/>	kategori_nama	Peralatan dapur			
	Key	Value	Description		
- Body:** A tab labeled "Body" is active, showing the response in JSON format.
- Response Status:** 200 OK, 210 ms, 359 B.
- Response Body (JSON):**

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

2. API pada m_barang

- Buat controller API pada folder API dengan nama BarangController

```
PS D:\laragon\www\PWL_25\week10\jobsheet\PWL_POS> php artisan make:controller Api\BarangController

INFO Controller [D:\laragon\www\PWL_25\week10\jobsheet\PWL_POS\app\Http\Controllers\Api\BarangController.php] created successfully.
```



```
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 use Illuminate\Database\Eloquent\Collection;
9 use Illuminate\Http\JsonResponse;
10
11 class BarangController extends Controller
12 {
13     public function index()
14     {
15         return BarangModel::all();
16     }
17
18     public function store(Request $request)
19     {
20         $barang = BarangModel::create($request->all());
21         return response()->json([
22             'data' => $barang,
23             'status' => 201
24         ]);
25     }
26
27     public function show(BarangModel $barang)
28     {
29         return $barang;
30     }
31
32     public function update(Request $request, BarangModel $barang)
33     {
34         $barang->update($request->all());
35         return $barang;
36     }
37
38     public function destroy(BarangModel $barang)
39     {
40         $barang->delete();
41         return response()->json([
42             'message' => 'Data terhapus.'
43         ], 200);
44     }
45 }
46
```

- b. Tambahkan route pada routes/api.php

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



```
46 // Route API Barang
47 Route::get('/barangs', [BarangController::class, 'index']);
48 Route::post('/barangs', [BarangController::class, 'store']);
49 Route::get('/barangs/{barang}', [BarangController::class, 'show']);
50 Route::put('/barangs/{barang}', [BarangController::class, 'update']);
51 Route::delete('/barangs/{barang}', [BarangController::class, 'destroy']);
```

C. Test CRUD

⇒ Create data barang baru dengan method POST

HTTP http://127.0.0.1:8000/api/barangs?Barang_id=20&kategori_id=1&barang_kode=BRG020&barang_na... Save Share

POST http://127.0.0.1:8000/api/barangs?Barang_id=20&kategori_id=1&barang_kode=BRG020&barang_nam Send

Params Authorization Headers (7) Body Scripts Tests Settings Cookies

Query Params

<input checked="" type="checkbox"/>	Key	Value	Description	...	Bulk Edit
<input checked="" type="checkbox"/>	Barang_id	20			
<input checked="" type="checkbox"/>	kategori_id	1			
<input checked="" type="checkbox"/>	barang_kode	BRG020			
<input checked="" type="checkbox"/>	barang_nama	Sunscreen			
<input checked="" type="checkbox"/>	harga_beli	35000			
<input checked="" type="checkbox"/>	harga_jual	40000			



HTTP http://127.0.0.1:8000/api/barangs?Barang_id=20&kategori_id=1&barang_kode=BRG020&barang_na... Save Share </>

GET http://127.0.0.1:8000/api/barangs?Barang_id=20&kategori_id=1&barang_kode=BRG020&barang_nam Send

Params Authorization Headers (6) Body Scripts Tests Settings Cookies

Key	Value	Description
<input checked="" type="checkbox"/> barang_nama	Sunscreen	
<input checked="" type="checkbox"/> harga_beli	35000	
<input checked="" type="checkbox"/> harga_jual	40000	

Body Cookies Headers (10) Test Results 200 OK • 102 ms • 3.71 KB

{ } JSON Preview Visualize

```
200 {
201   "updated_at": null
202 },
203 {
204   "barang_id": 22,
205   "kategori_id": 1,
206   "barang_kode": "BRG020",
207   "barang_nama": "Sunscreen",
208   "harga_beli": 35000,
209   "harga_jual": 40000,
210   "created_at": "2025-04-28T09:44:56.000000Z",
211   "updated_at": "2025-04-28T09:44:56.000000Z"
212 }
```

⇒ Data Berhasil ditambah

Daftar barang Import Barang Export Barang Export Barang (PDF) Tambah Data (Ajax)

Filter - Semua -
Kategori Barang

Show 10 entries Search:

No	Kode Barang	Nama Barang	Harga Beli	Harga Jual	Kategori	Aksi
21	BRG020	Sunscreen	35.000	40.000	Pakaian	Detail Edit Hapus

Showing 21 to 21 of 21 entries Previous 1 2 3 Next



⇒ Read/menampilkan detail barang dengan method GET

HTTP http://127.0.0.1:8000/api/barangs?Barang_id=20&kategori_id=1&barang_kode=BRG020&barang_na... Save Share

GET http://127.0.0.1:8000/api/barangs?Barang_id=20&kategori_id=1&barang_kode=BRG020&barang_na... Send

Params Authorization Headers (6) Body Scripts Tests Settings Cookies

Key	Value	Description
barang_nama	Sunscreen	
harga_beli	35000	
harga_jual	40000	

Body Cookies Headers (10) Test Results 200 OK • 102 ms • 3.71 KB

JSON Preview Visualize

```
200 {
201   "updated_at": null
202 },
203 {
204   "barang_id": 22,
205   "kategori_id": 1,
206   "barang_kode": "BRG020",
207   "barang_nama": "Sunscreen",
208   "harga_beli": 35000,
209   "harga_jual": 40000,
210   "created_at": "2025-04-28T09:44:56.000000Z",
211   "updated_at": "2025-04-28T09:44:56.000000Z"
212 }
```

⇒ Update data barang dengan method PUT

HTTP http://127.0.0.1:8000/api/barangs/22?Barang_id=20&kategori_id=1&barang_kode=BRG020&barang... Save Share

PUT http://127.0.0.1:8000/api/barangs/22?Barang_id=20&kategori_id=1&barang_kode=BRG020&barang... Send

Params Authorization Headers (7) Body Scripts Tests Settings Cookies

barang_kode	BRG020
barang_nama	Sunscreen SPF 30
harga_beli	35000

Body Cookies Headers (10) Test Results 200 OK • 106 ms • 526 B

JSON Preview Visualize

```
1 {
2   "barang_id": 22,
3   "kategori_id": "1",
4   "barang_kode": "BRG020",
5   "barang_nama": "Sunscreen SPF 30",
6   "harga_beli": "35000",
7   "harga_jual": "40000",
8   "created_at": "2025-04-28T09:44:56.000000Z",
9   "updated_at": "2025-04-28T09:51:25.000000Z"
10 }
```

Data barang berhasil diupdate



Daftar barang

Filter: - Semua -

Search:

No	Kode Barang	Nama Barang	Harga Beli	Harga Jual	Kategori	Aksi
21	BRG020	Sunscreen SPF 30	35.000	40.000	Pakaian	Detail Edit Hapus

Showing 21 to 21 of 21 entries

Previous 1 2 3 Next

⇒ Delete data barang dengan method DELETE

HTTP http://127.0.0.1:8000/api/barangs/22?Barang_id=20&kategori_id=1&barang_kode=BRG020&barang... Save Share

DELETE http://127.0.0.1:8000/api/barangs/22?Barang_id=20&kategori_id=1&barang_kode=BRG020&barang... Send

Params Authorization Headers (6) Body Scripts Tests Settings Cookies

Body Cookies Headers (10) Test Results

200 OK • 100 ms • 336 B

```
{
  "message": "Data terhapus."
}
```