



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 dipercayai.

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

Kita dapat menggunakan JWT untuk:

- **Authentication**

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

- **Pertukaran informasi**

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

Berikut adalah cara kerja JWT :

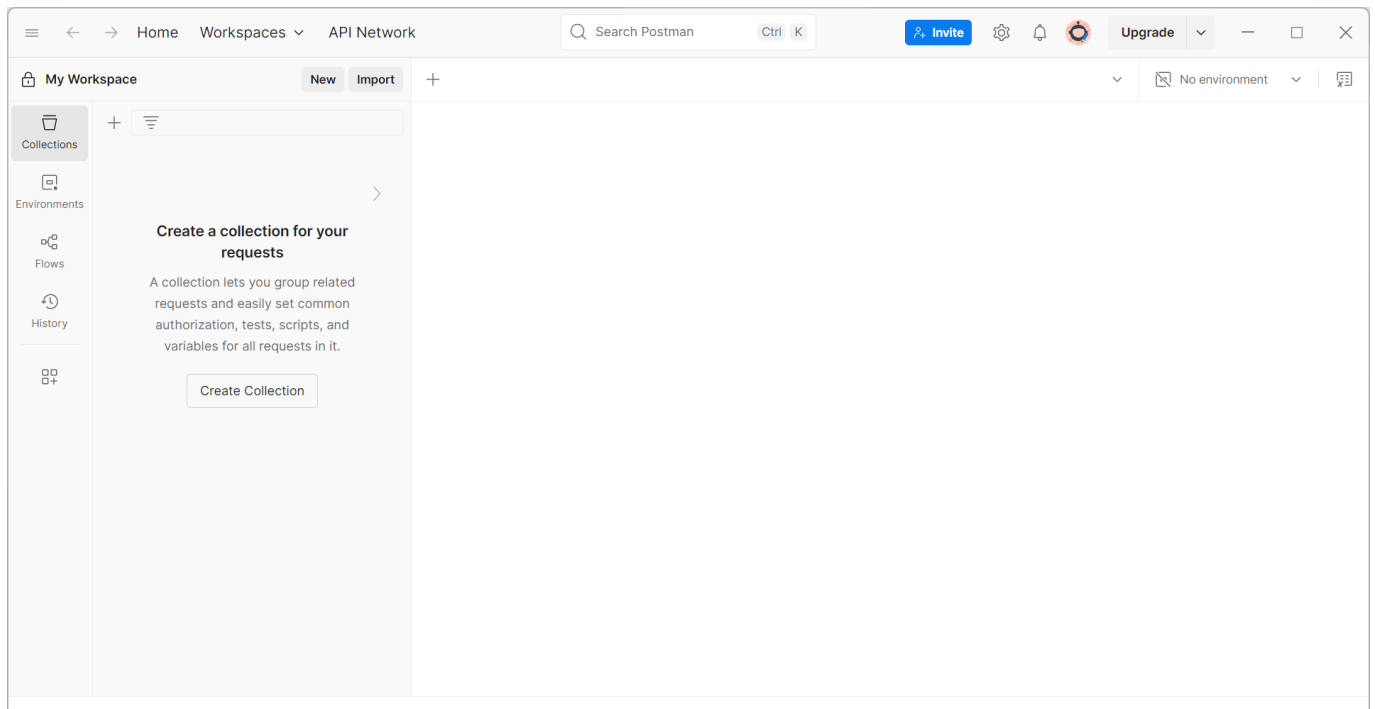
JWT (JSON Web Token) adalah cara untuk mentransfer informasi antara dua pihak secara aman sebagai objek JSON. Ini terdiri dari tiga bagian: header, payload, dan signature. Setelah pengguna berhasil autentikasi, server menghasilkan token JWT yang disematkan dalam permintaan HTTP. Server kemudian memvalidasi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



Praktikum 1 – Membuat RESTful API Register

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

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



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



```
File Edit Selection View ... PWL_2025
EXPLORER: PWL_2025
  > Minggu_1
  > Minggu_2
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS SPELL CHECKER
> illuminate\Foundation\ComposerScripts::postAutoloadDump
> @php artisan package:discover --ansi

[INFO] Discovering packages.

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

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

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

https://repo.packagist.org could not be fully loaded (curl error 28 while downloading https://repo.packagist.org/p2/barryvdh/laravel-dompdf.json: Connection timed out after 10013 milliseconds), package
information was loaded from the local cache and may be out of date
A connection timeout was encountered. If you intend to run Composer without connecting to the internet, run the command again prefixed with COMPOSER_DISABLE_NETWORK=1 to make Composer run in offline mo
de.
Found 1 security vulnerability advisory affecting 1 package.
Run "composer audit" for a full list of advisories.
PS C:\laragon\www\PWL_2025\Minggu_10\Jobsheet10\PWL_POS>
```

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

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

```
[INFO] Publishing assets.

Copying file [C:\laragon\www\PWL_2025\Minggu_10\Jobsheet10\PWL_POS\vendor\tymon\jwt-auth\config\config.php] to [C:\laragon\www\PWL_2025\Minggu_10\Jobsheet10\PWL_POS\config\jwt.php] DONE

PS C:\laragon\www\PWL_2025\Minggu_10\Jobsheet10\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.

```
Minggu_10 > Jobsheet10 > PWL_POS > config > jwt.php
12  return [
19      | Don't forget to set this in your .env file, as it will be used to sign
20      | your tokens. A helper command is provided for this:
21      | 'php artisan jwt:secret'
22      |
23      | Note: This will be used for Symmetric algorithms only (HMAC),
24      | since RSA and ECDSA use a private/public key combo (See below).
25      |
26      | */
27
28      'secret' => env('JWT_SECRET'),
29
30      /*
31      |-----
32      | JWT Authentication Keys
33      |-----
34      |
35      | The algorithm you are using, will determine whether your tokens are
36      | signed with a random string (defined in 'JWT_SECRET') or using the
37      | following public & private keys.
38      |
39      | Symmetric Algorithms:
40      | HS256, HS384 & HS512 will use 'JWT_SECRET'.
41      |
42      | Asymmetric Algorithms:
43      | RS256, RS384 & RS512 / ES256, ES384 & ES512 will use the keys below.
44      |
45      | */
46
47      'keys' => [
48
49          /*
50          |-----
51          | Public Key
52          |-----
53          |
```



5. Setelah itu jalankan perintah berikut untuk membuat secret key JWT.

`php artisan jwt:secret`

```
PS C:\laragon\www\pwl_2025> cd Minggu_10\Jobsheet10\pwl_pos
PS C:\laragon\www\pwl_2025\Minggu_10\Jobsheet10\pwl_pos> php artisan jwt:secret
jwt-auth secret [QjPH2U2XuU4d0ic8Wjro1bm1CznCMQ8y0670SL41hf9cepL12Cs0gvXCAm0VM] set successfully.
PS C:\laragon\www\pwl_2025\Minggu_10\Jobsheet10\pwl_pos>
```

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

```
27
28 'secret' => env('JWT_SECRET'),
29
```

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',
    ],
],
```

```
File Edit Selection View ... < -> Pwl_2025
composer.json M composer.lock M jwt.php U auth.php M X
Minggu_10 > Jobsheet10 > pwl_pos > config > auth.php
3 return [
4     // Of course, a great default configuration has been defined for you
5     // here which uses session storage and the Eloquent user provider.
6
7     // All authentication drivers have a user provider. This defines how the
8     // users are actually retrieved out of your database or other storage
9     // mechanisms used by this application to persist your user's data.
10
11     // Supported: "session"
12     //
13
14     'guards' => [
15         'web' => [
16             'driver' => 'session',
17             'provider' => 'users',
18         ],
19         'api' => [
20             'driver' => 'jwt',
21             'provider' => 'users',
22         ],
23     ],
24
25     /*
26     -----
27     User Providers
28     -----
29
30     // All authentication drivers have a user provider. This defines how the
31     // users are actually retrieved out of your database or other storage
32     // mechanisms used by this application to persist your user's data.
33
34     // If you have multiple user tables or models you may configure multiple
35     // sources which represent each model / table. These sources may then
36     // be assigned to any extra authentication guards you have defined.
37
38     */
39 ]
```



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

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';
}
```

```
5 use Illuminate\Database\Eloquent\Factories\HasFactory;
6 use Illuminate\Database\Eloquent\Relations\BelongsTo;
7 use Illuminate\Foundation\Auth\User as Authenticatable; // implementasi class Authenticatable
8 use Illuminate\Database\Eloquent\Model;
9 use Tymon\JWTAuth\Contracts\JWTSubject;
10
11 class UserModel extends Authenticatable
12 {
13     use HasFactory;
14
15     protected $table = 'm_user'; // Mendefinisikan nama tabel yang digunakan oleh model ini
16     protected $primaryKey = 'user_id'; // Mendefinisikan primary key dari tabel yang digunakan
17     protected $fillable = ['level_id', 'username', 'nama', 'password'];
18
19     protected $hidden = ['password']; // jangan di tampilkan saat select
20
21     protected $casts = ['password' => 'hashed']; // casting password agar otomatis di hash
22
23     public function getJWTIdentifier(){
24         return $this->getKey();
25     }
26
27     public function getJWTCustomClaims(){
28         return [];
29     }
30 }
```

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

`php artisan make:controller Api/RegisterController`

```
PS C:\laragon\www\PwL_2025> cd Minggu_10\Jobsheet10\PwL_POS
PS C:\laragon\www\PwL_2025\Minggu_10\Jobsheet10\PwL_POS> php artisan make:controller Api/RegisterController

[INFO] Controller [C:\laragon\www\PwL_2025\Minggu_10\Jobsheet10\PwL_POS\app\Http\Controllers\Api\RegisterController.php] created successfully.
PS C:\laragon\www\PwL_2025\Minggu_10\Jobsheet10\PwL_POS>
```

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



```
1 <?php
2
3 namespace App\Http\Controllers\Api;
4
5 use App\Http\Controllers\Controller;
6 use Illuminate\Http\Request;
7
8 class RegisterController extends Controller
9 {
10     //
11 }
12
```

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 }
```



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

```
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) {
```

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



```
<?php

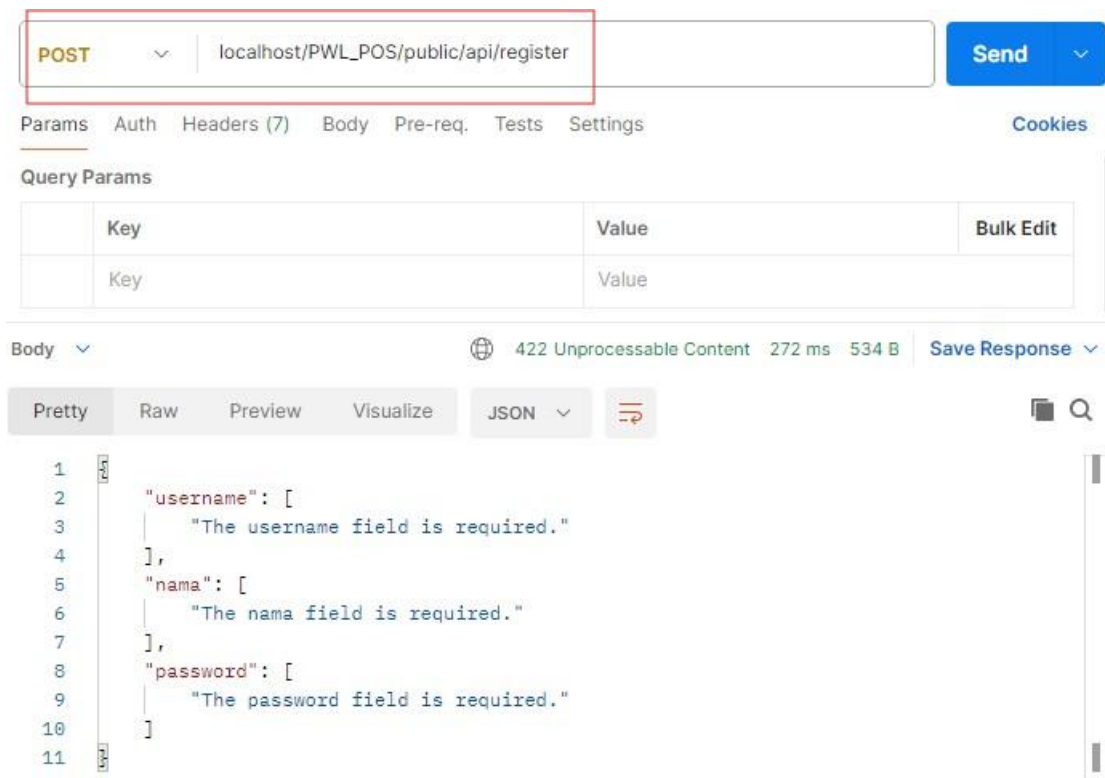
use App\Http\Controllers\Api\RegisterController;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Route;

/*
|-----
| 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.



Jika berhasil akan muncul error validasi seperti gambar di atas.

Lakukan percobaan yang sama dan berikan screenshoot hasil percobaan Anda.



The screenshot shows the Postman interface with a POST request to `localhost/PWL_2025/Minggu_10/Jobsheet10/PWL_POS/public/api/register`. The response is `422 Unprocessable Content` with a JSON body:

```
1 {"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."]}
```

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
<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  {"success": true,  
2    "user": {  
3      "username": "penggunasatu",  
4      "nama": "Pengguna 1",  
5      "password": "$2y$12$Eb2SrV1jsykINytYGtrHi0DVAKcK5p6EgnZnmbChkPicIu7S0QJJU",  
6      "level_id": "2",  
7      "updated_at": "2024-04-22T15:56:04.000000Z",  
8      "created_at": "2024-04-22T15:56:04.000000Z",  
9      "user_id": 17  
10 }
```

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

Home Workspaces API Network Search Postman Ctrl K Invite Upgrade

localhost/PWL_2025/Minggu_10/Jobsheet10/PWL_POS/public/api/register

POST localhost/PWL_2025/Minggu_10/Jobsheet10/PWL_POS/public/api/register Send

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

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

Key	Value
<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 (11) Test Results 201 Created 2.00 s 561 B

JSON Preview Visualize

```
1  {"success":true,"user":{"username":"penggunasatu","nama":"Pengguna 1","level_id":"2","updated_at":"2025-04-26T07:35:11.000000Z",  
    "created_at":"2025-04-26T07:35:11.000000Z","user_id":29}}
```

Lakukan percobaan yang sama dan berikan screenshoot hasil percobaan Anda.



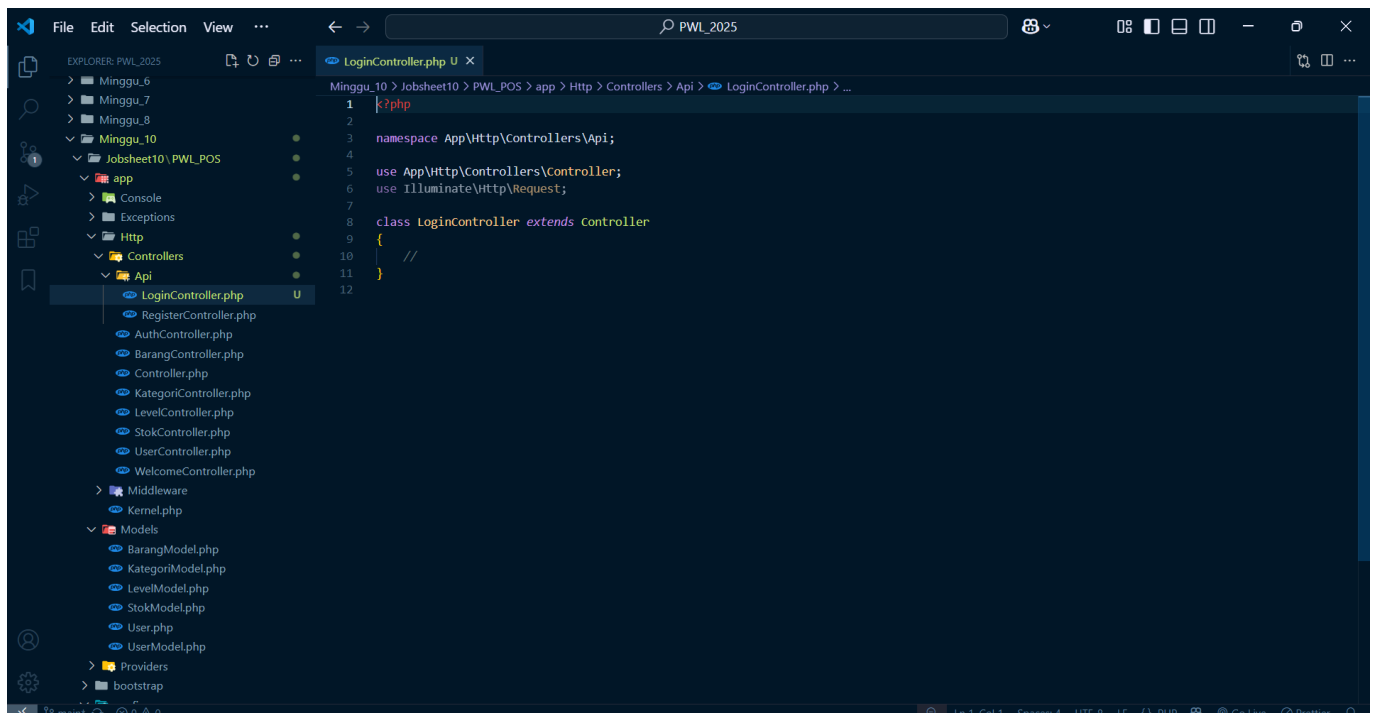
13. Lakukan commit perubahan file pada Github.

Praktikum 2 – Membuat RESTful API Login

1. Kita buat file controller dengan nama LoginController.

`php artisan make:controller Api/LoginController`

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 }
```



```
File Edit Selection View ... PWL_2025
LoginController.php U X
Minggu_10 > Jobsheet10 > PWL_POS > app > Http > Controllers > Api > LoginController.php > LoginController
1 <?php
2
3 namespace App\Http\Controllers\Api;
4
5 use App\Http\Controllers\Controller;
6 use Illuminate\Http\Request;
7 use Illuminate\Support\Facades\Validator;
8
9 class LoginController extends Controller
10 {
11     public function __invoke(Request $request)
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([
```

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

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

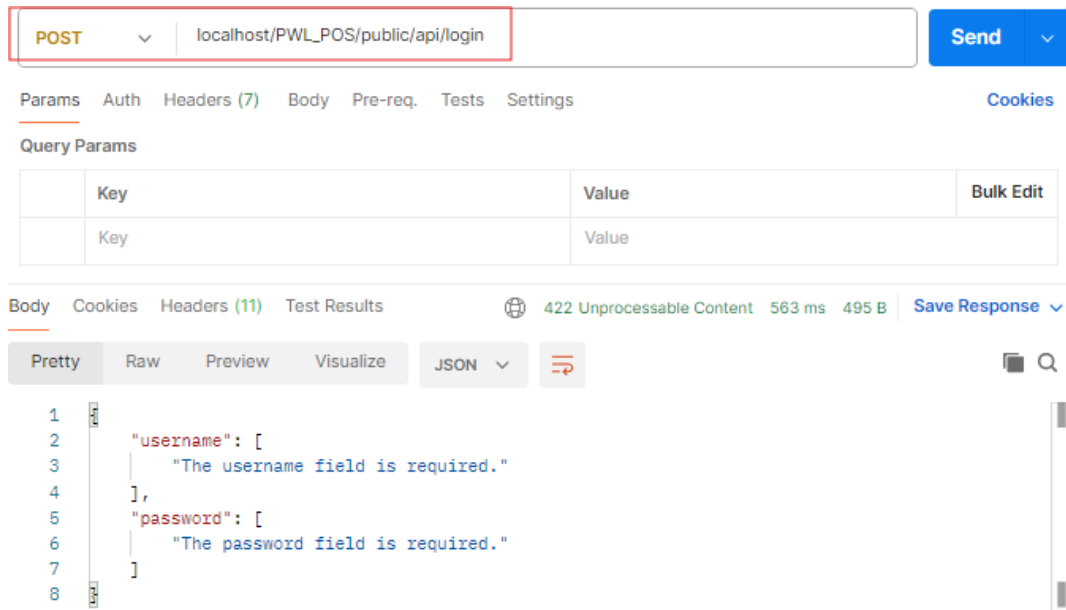
Route::post('/register', App\Http\Controllers\Api\RegisterController::class)->name('register');
Route::post('/login', App\Http\Controllers\Api\LoginController::class)->name('login');
Route::middleware('auth:api')->get('/user', function (Request $request) {
    return $request->user();
});
```

```
File Edit Selection View ... PWL_2025
LoginController.php U api.php M X
Minggu_10 > Jobsheet10 > PWL_POS > routes > api.php > ...
1 <?php
2
3 use Illuminate\Http\Request;
4 use Illuminate\Support\Facades\Route;
5 use App\Http\Controllers\Api\RegisterController;
6 use App\Http\Controllers\Api\LoginController;
7
8 /*
9 |-----
10 | API Routes
11 |-----
12 |
13 | Here is where you can register API routes for your application. These
14 | routes are loaded by the RouteServiceProvider and all of them will
15 | be assigned to the "api" middleware group. Make something great!
16 |
17 |*/
18
19 Route::post('/register', App\Http\Controllers\Api\RegisterController::class)->name('register');
20 Route::post('/login', App\Http\Controllers\Api\LoginController::class)->name('login');
21 Route::middleware('auth:api')->get('/user', function (Request $request) {
22     return $request->user();
23 });
24 Route::middleware('auth:sanctum')->get('/user', function (Request $request) {
25     return $request->user();
26 });
27
```



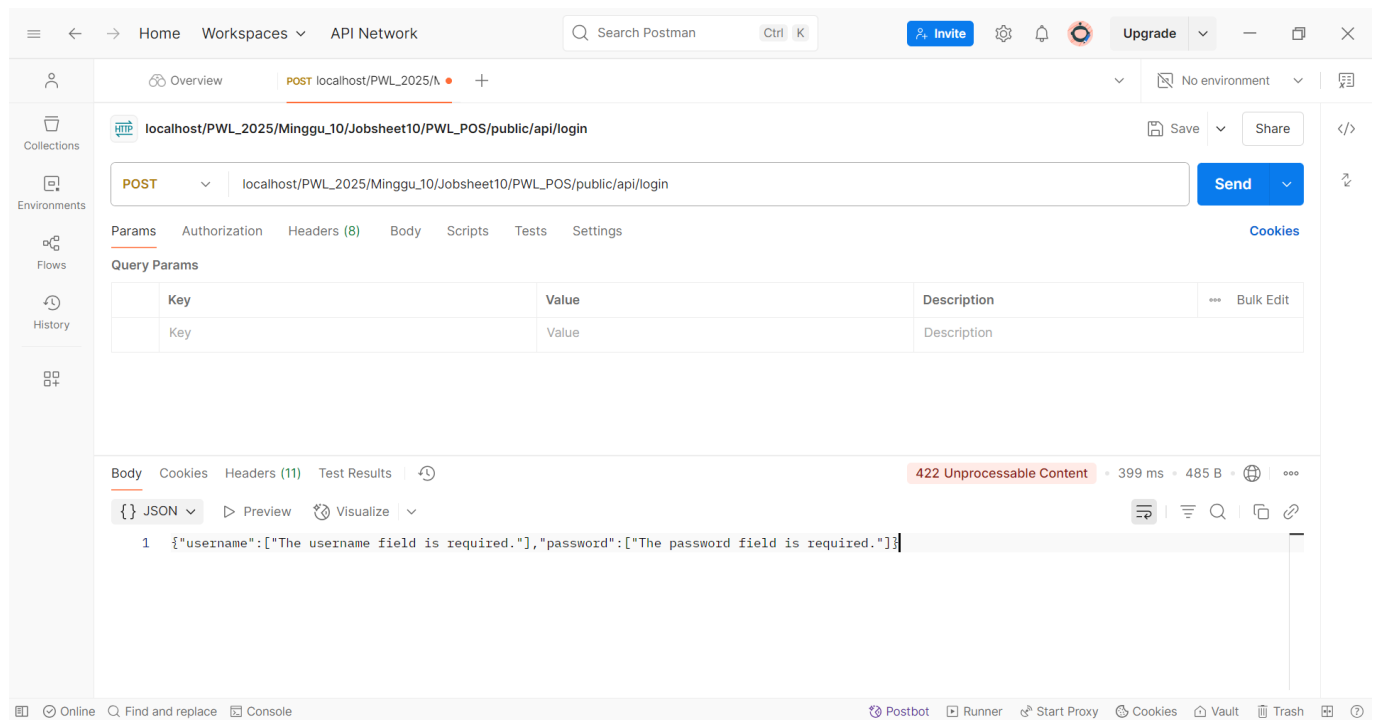

KEMENTERIAN PENDIDIKAN, KEBUDAYAAN, RISET, DAN TEKNOLOGI
POLITEKNIK NEGERI MALANG
JURUSAN TEKNOLOGI INFORMASI
Jl. Soekarno Hatta No. 9, Jatimulyo, Lowokwaru, Malang 65141
Telp. (0341) 404424 – 404425, Fax (0341) 404420
<http://www.polinema.ac.id>

4. Jika sudah, kita akan melakukan uji coba REST API melalui aplikasi Postman. Buka aplikasi Postman, isi URL `localhost/PWL_POS/public/api/login` serta method POST. Klik Send.



Jika berhasil akan muncul error validasi seperti gambar di atas.

Lakukan percobaan yang sama dan berikan screenshot hasil percobaan Anda.



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



POST localhost/PWL_POS/public/api/login Send

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

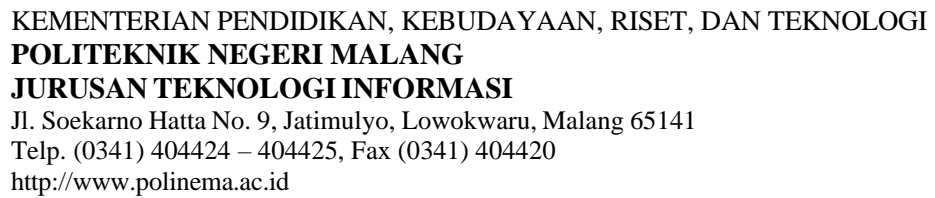
☐ none ☒ form-data ☐ x-www-form-urlencoded ☐ raw ☐ binary

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  {"success": true,
2
3  "user": {
4    "user_id": 17,
5    "level_id": 2,
6    "username": "penggunasatu",
7    "nama": "Pengguna 1",
8    "password": "$2y$12$Eb2SzV1jsykINytYGtzH10DVAKcK5p6EgnZnmbChkPicIu7S0QJJU",
9    "created_at": "2024-04-22T15:56:04.000000Z",
10   "updated_at": "2024-04-22T15:56:04.000000Z"
11  },
12  "token": "eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJpc3MiOiJodHRwOi8vbG9jYXRob3N0L1BXTF9QT1MtbnFpb19wdWJsakMvYXBlL2xvZ2luIiwiaWF0Ij0i"
13 }
```



Hal. 20 / 34



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.**

The screenshot shows the Postman interface with a GET request to `localhost/PWL_2025/Minggu_10/Jobsheet10/PWL_POS/public/api/user`. The request body is form-data with the following parameters:

Key	Value	Description
username	penggunasatu	
password	12345	

The response is a 200 OK status with a response time of 149 ms and a size of 7.28 KB. The body is JSON, showing the following HTML content:

```
1 <!DOCTYPE html>
2 <html lang="en">
3
4 <head>
5   <meta charset="utf-8">
6   <meta name="viewport" content="width=device-width, initial-scale=1">
7   <title>Login Pengguna</title>
8   <meta name="csrf-token" content="nAv4U8hIvNqXAFMN3HMF7DZHEAgCjAfBpo3besaR">
9   <!-- Google Font: Source Sans Pro -->
```

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`



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

```
File Edit Selection View ... PWL_2025
LoginController.php api.php UserModel.php .env x auth.php
Minggu_10 > Jobsheet10 > PWL_POS > .env
42 AWS_DEFAULT_REGION=us-east-1
43 AWS_BUCKET=
44 AWS_USE_PATH_STYLE_ENDPOINT=false
45
46 PUSHER_APP_ID=
47 PUSHER_APP_KEY=
48 PUSHER_APP_SECRET=
49 PUSHER_HOST=
50 PUSHER_PORT=443
51 PUSHER_SCHEME=https
52 PUSHER_APP_CLUSTER=mt1
53
54 VITE_APP_NAME="${APP_NAME}"
55 VITE_PUSHER_APP_KEY="${PUSHER_APP_KEY}"
56 VITE_PUSHER_HOST="${PUSHER_HOST}"
57 VITE_PUSHER_PORT="${PUSHER_PORT}"
58 VITE_PUSHER_SCHEME="${PUSHER_SCHEME}"
59 VITE_PUSHER_APP_CLUSTER="${PUSHER_APP_CLUSTER}"
60
61 JWT_SECRET=QjPMZU2XutA4bNc8Wjr-0Tbmc1cznCMKQsUyo670SL41hf9cePL12Cs6gvXcAmoVM
62 JWT_SHOW_BLACKLIST_EXCEPTION=true
63
```

2. Buat Controller baru dengan nama LogoutController.
php artisan make:controller Api/LogoutController

```
File Edit Selection View ... PWL_2025
LoginController.php api.php UserModel.php LogoutController.php U x .env auth.php
Minggu_10 > Jobsheet10 > PWL_POS > app > Http > Controllers > Api > LogoutController.php > ...
1 <?php
2
3 namespace App\Http\Controllers\Api;
4
5 use App\Http\Controllers\Controller;
6 use Illuminate\Http\Request;
7
8 class LogoutController extends Controller
9 {
10     //
11 }
12
```

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



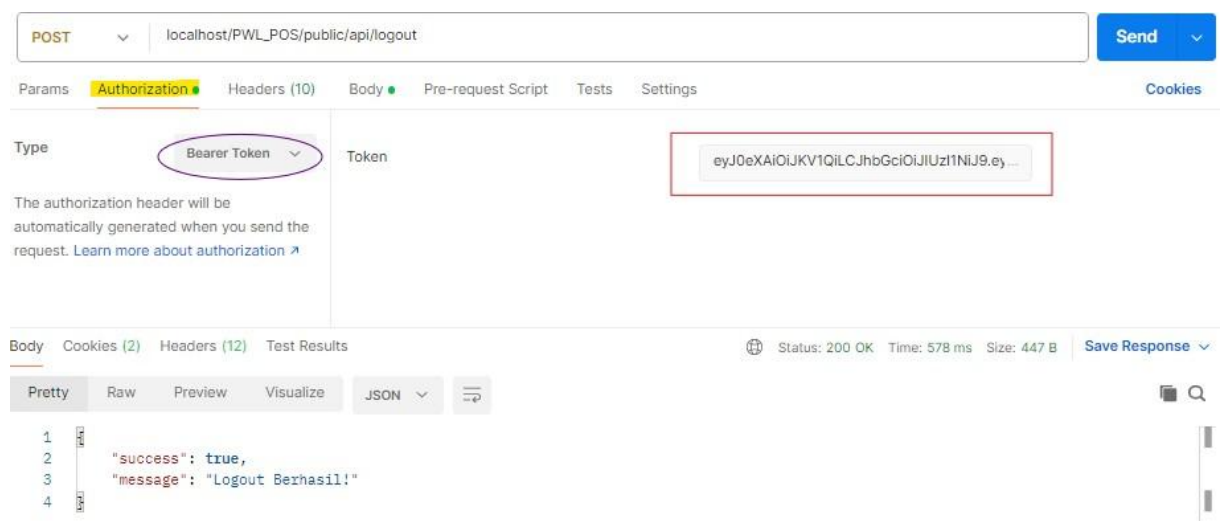
```
1  <?php
2
3  namespace App\Http\Controllers\Api;
4  use Illuminate\Http\Request;
5  use App\Http\Controllers\Controller;
6  use Tymon\JWTAuth\Facades\JWTAuth;
7  use Tymon\JWTAuth\Exceptions\JWTException;
8  use Tymon\JWTAuth\Exceptions\TokenExpiredException;
9  use Tymon\JWTAuth\Exceptions\TokenInvalidException;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



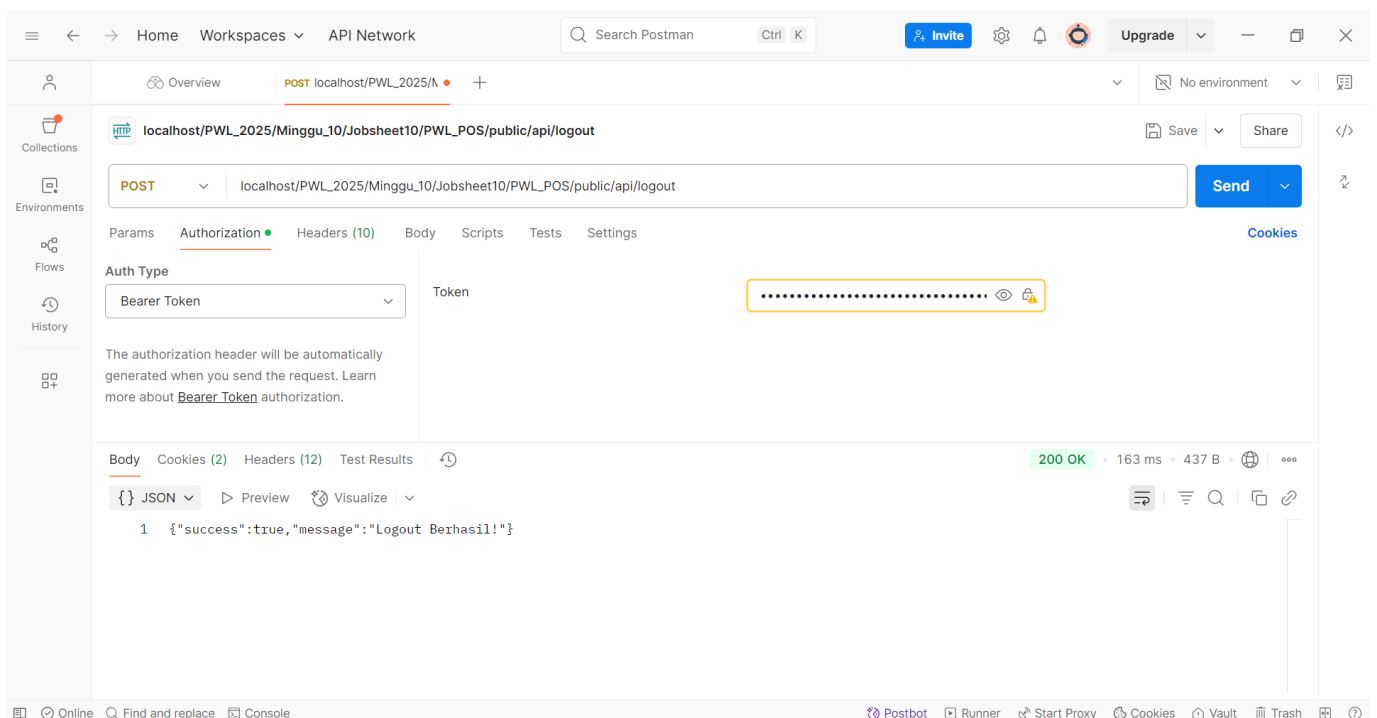
4. Lalu kita tambahkan routes pada api.php

```
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`

```
Minggu_10 > Jobsheet10 > PWL_POS > app > Http > Controllers > Api > LevelController.php > ...
1  <?php
2
3  namespace App\Http\Controllers\Api;
4
5  use App\Http\Controllers\Controller;
6  use Illuminate\Http\Request;
7
8  class LevelController extends Controller
9  {
10     //
11 }
12
```

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',
    ]);
}
```

```
1 <?php
2
3 namespace App\Http\Controllers\Api;
4
5 use App\Http\Controllers\Controller;
6 use Illuminate\Http\Request;
7 use App\Models\LevelModel;
8
9 class LevelController extends Controller
10 {
11     public function index()
12     {
13         return LevelModel::all();
14     }
15
16     public function store(Request $request)
17     {
18         $level = LevelModel::create($request->all());
19         return response()->json($level, 201);
20     }
21
22     public function show(LevelModel $level)
23     {
24         return LevelModel::find($level);
25     }
26
27     public function update(Request $request, LevelModel $level)
28     {
29         $level->update($request->all());
30         return LevelModel::find($level);
31     }
32
33     public function destroy(LevelModel $user)
34     {
35         $user->delete();
36         return response()->json([
```



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']);
```

```
1 <?php
2
3 use Illuminate\Http\Request;
4 use Illuminate\Support\Facades\Route;
5 use App\Http\Controllers\Api\RegisterController;
6 use App\Http\Controllers\Api\LoginController;
7 use App\Http\Controllers\Api\LevelController;
8 use App\Http\Controllers\Api\UserController;
9 use App\Http\Controllers\Api\KategoriController;
10 use App\Http\Controllers\Api\BarangController;
11
12 /*
13 |-----
14 | API Routes
15 |-----
16 |
17 | Here is where you can register API routes for your application. These
18 | routes are loaded by the RouteServiceProvider and all of them will
19 | be assigned to the "api" middleware group. Make something great!
20 |
21 */
22
23 Route::post('/register', App\Http\Controllers\Api\RegisterController::class->name('register'));
24 Route::post('/login', App\Http\Controllers\Api\LoginController::class->name('login'));
25 Route::post('/logout', App\Http\Controllers\Api\LogoutController::class->name('logout'));
26 Route::middleware('auth:api')->get('/user', function (Request $request) {
27     return $request->user();
28 });
29
30 Route::get('levels', [LevelController::class, 'index']);
31 Route::post('levels', [LevelController::class, 'store']);
32 Route::get('levels/{level}', [LevelController::class, 'show']);
33 Route::put('levels/{level}', [LevelController::class, 'update']);
34 Route::delete('levels/{level}', [LevelController::class, 'destroy']);
35 });
```

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



The screenshot shows the Postman interface with a GET request to `localhost/PWL_2025/Minggu_10/Jobsheet10/PWL_POS/public/api/levels`. The response is a 200 OK status with a response time of 418 ms and a body size of 796 B. The response body is a JSON array of level data.

Key	Value	Description
Key	Value	Description

```
1 [{"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","created_at":null,"updated_at":null}, {"level_id":5,"level_kode":"CUS","level_nama":"Pelanggan","created_at":"2025-03-05T02:36:33.000000Z","updated_at":null}]
```

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
<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 screenshot hasil percobaan Anda.

Home Workspaces API Network Search Postman Ctrl K Invite Upgrade

Overview POST localhost/PWL_2025/ Minggu_10/Jobsheet10/PWL_POS/public/api/levels No environment

localhost/PWL_2025/Minggu_10/Jobsheet10/PWL_POS/public/api/levels Save Share

POST localhost/PWL_2025/Minggu_10/Jobsheet10/PWL_POS/public/api/levels Send

Params Authorization Headers (10) Body Scripts 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 (12) Test Results 201 Created 1.04 s 543 B

{ JSON Preview Visualize

```
1 {
2   "level_kode": "SPV",
3   "level_nama": "Supervisor",
4   "updated_at": "2025-04-29T03:10:25.000000Z",
5   "created_at": "2025-04-29T03:10:25.000000Z",
6   "level_id": 10
7 }
```

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



The screenshot shows a Postman interface with a GET request to `localhost/PWL_2025/Minggu_10/Jobsheet10/PWL_POS/public/api/levels`. The response is a 200 OK status with a JSON body containing two user level entries:

```
{
  "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
}
```

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 Postman interface with a PUT request to `localhost/PWL_POS-main/public/api/levels/4?level_kode=SPR`. The Params tab is active, showing a query parameter `level_kode=SPR`. The response is a 200 OK status with a JSON body representing the updated user level:

```
{
  "level_id": 4,
  "level_kode": "SPR",
  "level_nama": "Supervisor",
  "created_at": "2024-04-22T21:40:32.000000Z",
  "updated_at": "2024-04-22T21:40:19.000000Z"
}
```

Jelaskan dan berikan screenshoot hasil percobaan Anda.



Home Workspaces API Network Search Postman Ctrl K Invite Upgrade

Overview PUT localhost/PWL_2025/Mi + No environment

localhost/PWL_2025/Minggu_10/Jobsheet10/PWL_POS/public/api/levels/10?level_kode=SPR Save Share

PUT localhost/PWL_2025/Minggu_10/Jobsheet10/PWL_POS/public/api/levels/10?level_kode=SPR Send

Params Authorization Headers (10) Body Scripts Tests Settings Cookies

Query Params

Key	Value	Description	Bulk Edit
level_kode	SPR		
Key	Value	Description	

Body Cookies Headers (12) Test Results 200 OK 160 ms 540 B

{ } JSON Preview Visualize

```
1 [
2   {
3     "level_id": 10,
4     "level_kode": "SPR",
5     "level_nama": "Supervisor",
6     "created_at": "2025-04-29T03:10:25.000000Z",
7     "updated_at": "2025-04-29T03:15:49.000000Z"
```

Online Find and replace Console Postbot Runner Start Proxy Cookies Vault Trash



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

The screenshot shows the Postman interface for a DELETE request. The URL is `localhost/PWL_2025/Minggu_10/Jobsheet10/PWL_POS/public/api/levels/10?level_kode=SPR`. The request is sent, and the response is a 200 OK status with a JSON body: `{"success": true, "message": "Data terhapus"}`. The interface includes tabs for Params, Authorization, Headers (10), Body, Scripts, Tests, and Settings. The Body tab is selected, showing the JSON response.

9. Lakukan commit perubahan file pada Github.

TUGAS

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

```
35 Route::get('users', [UserController::class, 'index']);
36 Route::post('users', [UserController::class, 'store']);
37 Route::get('users/{user}', [UserController::class, 'show']);
38 Route::put('users/{user}', [UserController::class, 'update']);
39 Route::delete('users/{user}', [UserController::class, 'destroy']);
40
41 Route::get('category', [KategoriController::class, 'index']);
42 Route::post('category', [KategoriController::class, 'store']);
43 Route::get('category/{kategori}', [KategoriController::class, 'show']);
44 Route::put('category/{kategori}', [KategoriController::class, 'update']);
45 Route::delete('category/{kategori}', [KategoriController::class, 'destroy']);
46
47 Route::get('items', [BarangController::class, 'index']);
48 Route::post('items', [BarangController::class, 'store']);
49 Route::get('items/{barang}', [BarangController::class, 'show']);
50 Route::put('items/{barang}', [BarangController::class, 'update']);
51 Route::delete('items/{barang}', [BarangController::class, 'destroy']);
```




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

```
1 <?php
2
3 namespace App\Http\Controllers\Api;
4
5 use App\Http\Controllers\Controller;
6 use App\Models\UserModel;
7 use Illuminate\Http\Request;
8 use Illuminate\Support\Facades\Hash;
9
10 class UserController extends Controller
11 {
12     public function index()
13     {
14         return UserModel::all();
15     }
16
17     public function store(Request $request)
18     {
19         $request['password'] = Hash::make($request['password']);
20         $level = UserModel::create($request->all());
21         return response()->json($level, 201);
22     }
23
24     public function show($user)
25     {
26         return UserModel::find($user);
27     }
28
29     public function update(Request $request, $level)
30     {
31         $data = UserModel::find($level);
32         $data->update($request->all());
33         return UserModel::find($level);
34     }
35
36     public function destroy($user)
37     {
38         $data = UserModel::find($user);
39         $data->delete();
40         return response()->json([
41             'success' => true,
42             'message' => 'Data terhapus',
43         ]);
44     }
45 }
```

```
1 <?php
2
3 namespace App\Http\Controllers\Api;
4
5 use App\Http\Controllers\Controller;
6 use App\Models\KategoriModel;
7 use Illuminate\Http\Request;
8 use Illuminate\Support\Facades\Hash;
9
10 class KategoriController extends Controller
11 {
12     public function index()
13     {
14         return KategoriModel::all();
15     }
16
17     public function store(Request $request)
18     {
19         $kategori = KategoriModel::create($request->all());
20         return response()->json($kategori, 201);
21     }
22
23     public function show($kategori)
24     {
25         return KategoriModel::find($kategori);
26     }
27
28     public function update(Request $request, $kategori)
29     {
30         $data = KategoriModel::find($kategori);
31         $data->update($request->all());
32         return KategoriModel::find($kategori);
33     }
34
35     public function destroy($kategori)
36     {
37         $data = KategoriModel::find($kategori);
38         $data->delete();
39         return response()->json([
40             'success' => true,
41             'message' => 'Data terhapus',
42         ]);
43     }
44 }
```



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

```
<?php
1
2
3 namespace App\Http\Controllers\Api;
4
5 use App\Http\Controllers\Controller;
6 use App\Models\BarangModel;
7 use Illuminate\Http\Request;
8
9 class BarangController extends Controller
10 {
11     public function index()
12     {
13         return BarangModel::all();
14     }
15     public function store(Request $request)
16     {
17         $barang = BarangModel::create($request->all());
18         return response()->json($barang, 201);
19     }
20
21     public function show($barang)
22     {
23         return BarangModel::find($barang);
24     }
25
26     public function update(Request $request, $barang)
27     {
28         $data = BarangModel::find($barang);
29         $data->update($request->all());
30         return BarangModel::find($barang);
31     }
32
33     public function destroy($barang)
34     {
35         $data = BarangModel::find($barang);
36         $data->delete();
37         return response()->json([
38             'success' => true,
39             'message' => 'Data terhapus',
40         ]);
41     }
42 }
```

*** Sekian, dan selamat belajar ***