

PRAKTIKUM PEMROGRAMAN WEB LANJUT

TUGAS PERTEMUAN 10

RESTFUL API

DISUSUN OLEH :

GILANG PURNOMO

NIM:2341720042



PROGRAM STUDI D-IV TEKNIK INFORMATIKA

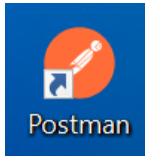
JURUSAN TEKNOLOGI INFORMASI

POLITEKNIK NEGERI MALANG

APRIL 2025

Praktikum 1 – Membuat RESTful API Register

1. Sebelum memulai membuat REST API, terlebih dahulu download aplikasi Postman di <https://www.postman.com/downloads>.
Aplikasi ini akan digunakan untuk mengerjakan semua tahap praktikum pada Jobsheet ini.



2. Lakukan instalasi JWT dengan mengetikkan perintah berikut:
[composer require tymon/jwt-auth:2.1.1](#)
Pastikan Anda terkoneksi dengan internet.

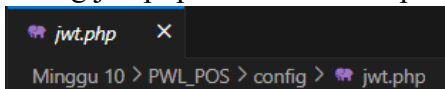
```
PS C:\laragon\www\Web_Lanjut\Minggu 10\PWL_POS> composer require tymon/jwt-auth:2.1.1
./composer.json has been updated
Running composer update tymon/jwt-auth
Loading composer repositories with package information
Updating dependencies
Lock file operations: 4 installs, 0 updates, 0 removals
- Locking lcobucci/clock (2.3.0)
- Locking lcobucci/jwt (4.0.4)
- Locking stella-maris/clock (0.1.7)
- Locking tymon/jwt-auth (2.1.1)
Writing lock file
Installing dependencies from lock file (including require-dev)
```

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

[php artisan vendor:publish --provider="Tymon\JWTAuth\Providers\LaravelServiceProvider"](#)

```
INFO Publishing assets.
Copying file [C:\laragon\www\Web_Lanjut\Minggu 10\PWL_POS\vendor\tymon\jwt-auth\config\config.php] to [C:\laragon\www\Web_Lanjut\Minggu 10\PWL_POS\config\jwt.php] DONE
PS C:\laragon\www\Web_Lanjut\Minggu 10\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](#)

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

```
PS C:\laragon\www\Web_Lanjut\Minggu 10\PWL_POS> php artisan jwt:secret
jwt-auth secret [413rCq73xe54RJ5YDnYwjIRYaQdgqdMsbwP4eyeXM9r40QNwX6xLf0dALiJTRIT3] set successfully.
```

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

```

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

```

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

```

6 use Tymon\JWTAuth\Contracts\JWTSubject;
7 use Illuminate\Database\Eloquent\Relations\BelongsTo;
8 use Illuminate\Foundation\Auth\User as Authenticatable;
9
10 class UserModel extends Authenticatable implements JWTSubject
11 {
12     public function getJWTIdentifier()
13     {
14         return $this->getKey();
15     }
16
17     public function getJWTCustomClaims()
18     {
19         return [];
20     }
21
22     protected $table = 'm_user';
23     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.

```

PS C:\laragon\www\Web_Lanjut\Minggu 10\PWL_POS> php artisan make:controller Api/RegisterController
INFO Controller [C:\laragon\www\Web_Lanjut\Minggu 10\PWL_POS\app\Http\Controllers\Api\RegisterController.php]
created successfully.

```

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

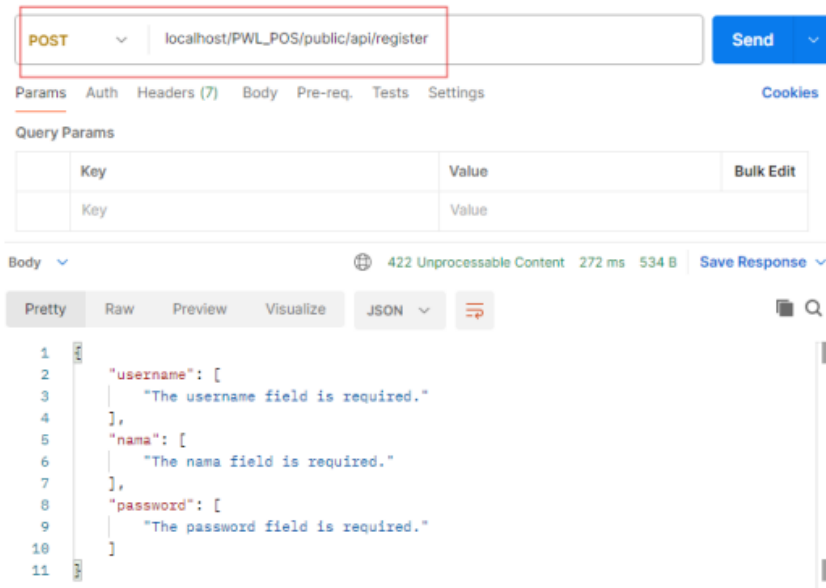
use App\Http\Controllers\Api\RegisterController;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Route;

/*
|-----
| 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');

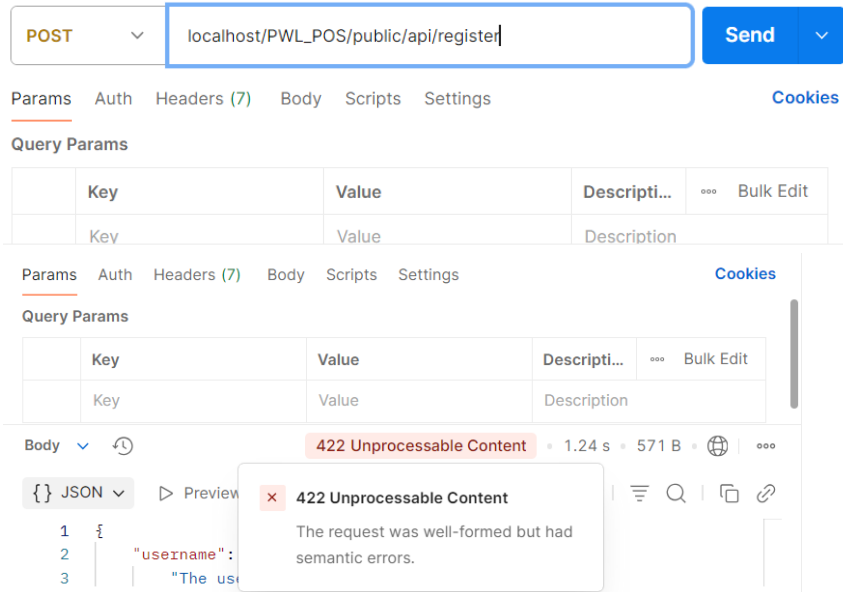
```

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.

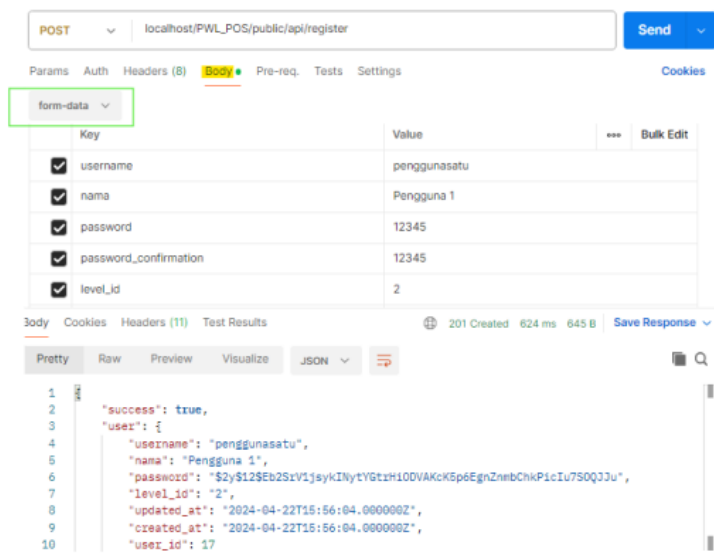


Jika berhasil akan muncul error validasi seperti gambar di atas.

Lakukan percobaan yang sama dan berikan screenshoot hasil percobaan Anda.

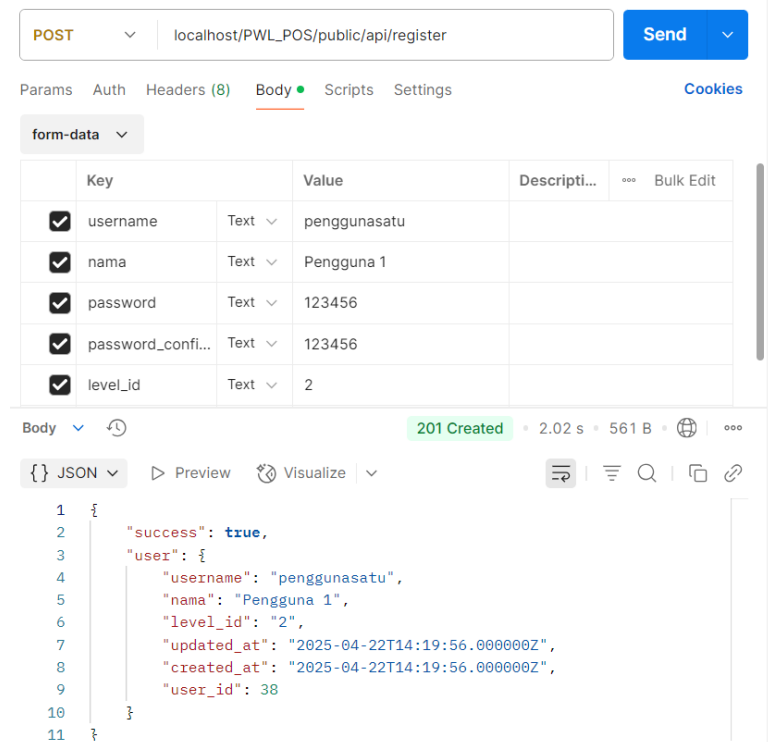


12. Sekarang kita coba masukkan data. Klik tab Body dan pilih form-data. Isikan key sesuai dengan kolom data, serta isikan data registrasi menggunakan nilai yang Anda inginkan.



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

Lakukan percobaan yang sama dan berikan screenshoot hasil percobaan Anda.



13. Lakukan commit perubahan file pada Github.

Link commit :

https://github.com/Gilangp/Pemrograman_Web_Lanjut/commit/1c0de2bb944cceda6b9abbf1a19fc5f122dba9c0

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 C:\laragon\www\Web_Lanjut\Minggu 10\PWL_POS> php artisan make:controller Api/LoginController  
  
INFO Controller [C:\laragon\www\Web_Lanjut\Minggu 10\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.

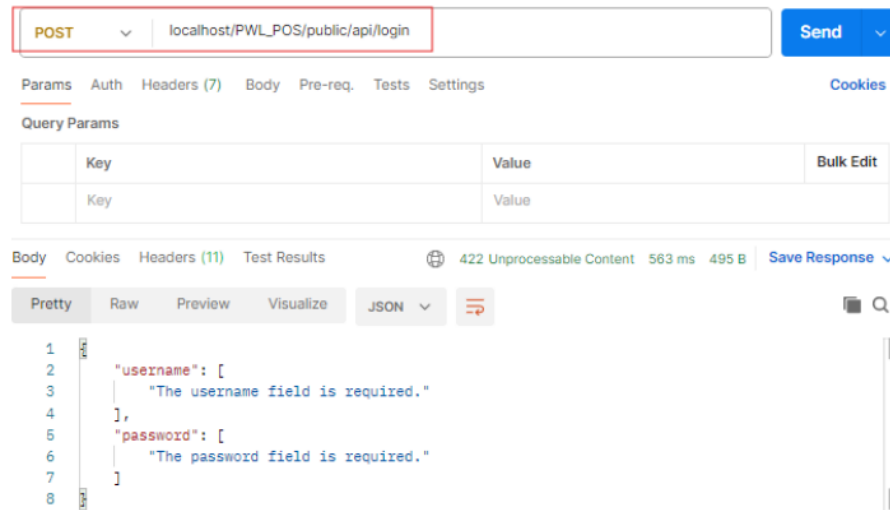
```

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();
});

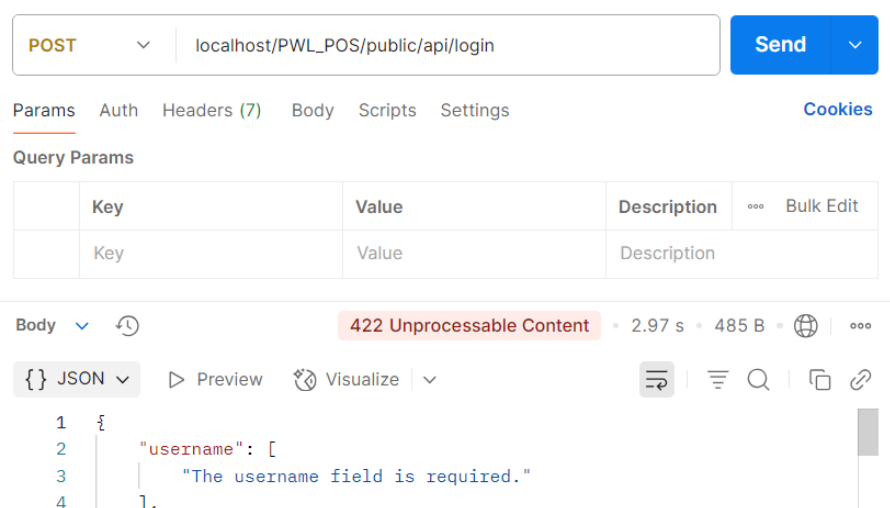
```

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.



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 Auth Headers (8) Body Scripts Settings Cookies

form-data

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

Body 401 Unauthorized • 2.87 s • 444 B

JSON Preview Visualize

```

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

Harusnya passwordnya ada 6 digit dan diatas menginputkan 5 digit maka terjadi error karena password tidak sesuai data.

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.

GET localhost/PWL_POS/public/api/user Send

Params Auth Headers (8) Body Scripts Settings Cookies

Auth Type

Bearer Token

The authorization header will be automatically generated when you send the request. Learn more about Bearer Token

Heads up! These parameters hold sensitive data. To keep this data secure while working in a collaborative environment, we recommend using variables. Learn more about [variables](#).

Token

eyJ0eXAiOiJKV1QiLCJhbG...

Body 200 OK • 1.54 s • 564 B

JSON Preview Visualize

```

1 {
2   "user_id": 38,
3   "level_id": 2,
4   "username": "penggunasatu",
5   "nama": "Pengguna 1",
6   "photo": null,
7   "created_at": "2025-04-22T14:19:56.000000Z",
8   "updated_at": "2025-04-22T14:19:56.000000Z"
9 }
```

Jadi pertama membuat request get dengan endpoint api/user, lalu menggunakan metode Auth dengan tipe Bearer Token dan memasukkan token yang sudah didapat saat login untuk mengetahui informasi login.

8. Lakukan commit perubahan file pada Github.

Link Commit :

https://github.com/Gilangp/Pemrograman_Web_Lanjut/commit/0cb32efe2fb1615d9789a0d0700a36918f5074bb

Praktikum 3 – Membuat RESTful API Logout

1. Tambahkan kode berikut pada file .env

[JWT_SHOW_BLACKLIST_EXCEPTION=true](#)

```
61 JWT_SECRET=413rCq73xe54RJ5YDnYwjIRYaQdgqdMsbWP4eyeXM9r40QNwX6xLf0dALiJTrit3
62 JWT_SHOW_BLACKLIST_EXCEPTION=true
```

2. Buat Controller baru dengan nama LogoutController.

[php artisan make:controller Api/LogoutController](#)

```
PS C:\laragon\www\Web_Lanjut\Minggu 10\PWL_POS> php artisan make:controller Api/LogoutController
INFO Controller [C:\laragon\www\Web_Lanjut\Minggu 10\PWL_POS\app\Http\Controllers\Api\LogoutController.php]
created successfully.
```

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

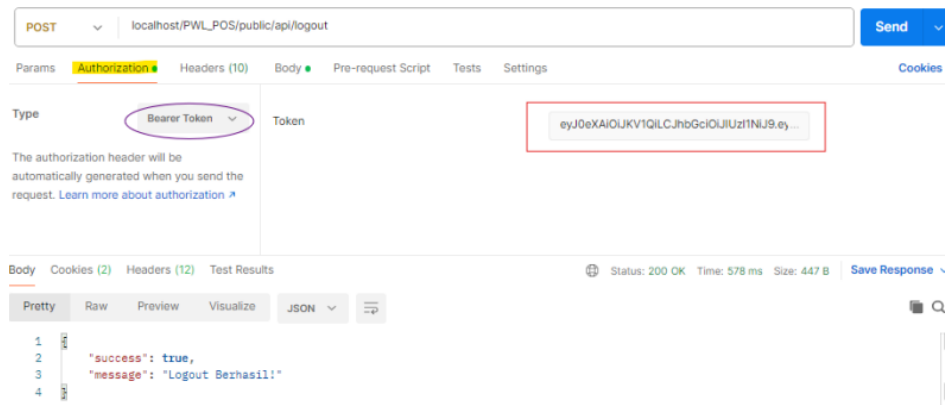
```
1 <?php
2
3 namespace App\Http\Controllers\Api;
4 use Illuminate\Http\Request;
5 use App\Http\Controllers\Controller;
6 use Tymon\JWTAuth\Facades\JWTAuth;
7 use Tymon\JWTAuth\Exceptions\JWTException;
8 use Tymon\JWTAuth\Exceptions\TokenExpiredException;
9 use Tymon\JWTAuth\Exceptions\TokenInvalidException;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

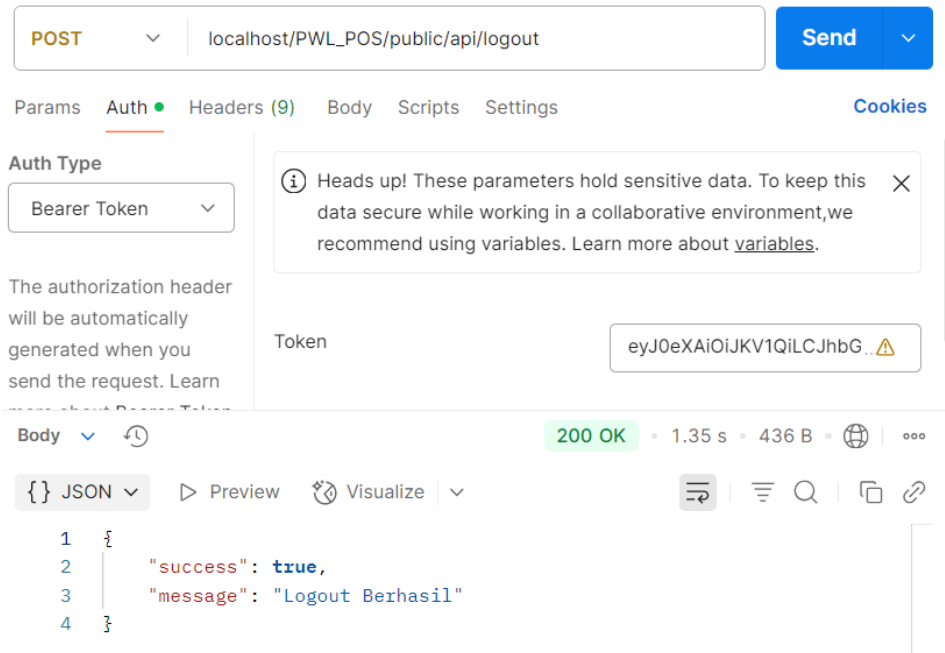
```
Route::post('/logout', App\Http\Controllers\Api\LogoutController::class)->name('logout');
```

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.

Link Commit :

https://github.com/Gilangp/Pemrograman_Web_Lanjut/commit/742c5cd472535b8a93fdc901f7e9981903d2ed1f

Praktikum 4 – Implementasi CRUD dalam RESTful API

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

[php artisan make:controller Api/LevelController](#)

```
PS C:\laragon\www\Web_Lanjut\Minggu 10\PWL_POS> php artisan make:controller Api/LevelController
INFO Controller [C:\laragon\www\Web_Lanjut\Minggu 10\PWL_POS\app\Http\Controllers\Api\LevelController.php]
created successfully.
```

- 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',
        ]);
    }
}
```

- 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']);
```

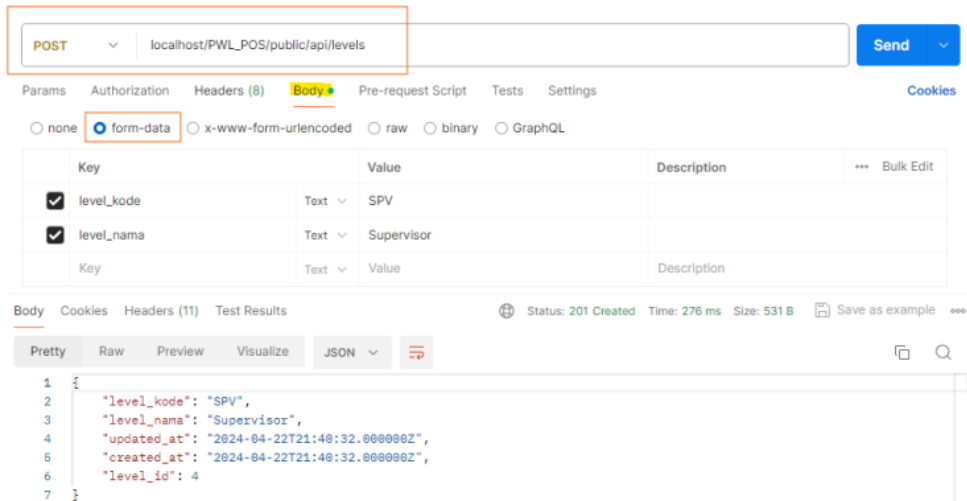
- Jika sudah. Lakukan uji coba API mulai dari fungsi untuk menampilkan data. Gunakan URL: localhost/PWL_POS-main/public/api/levels dan method GET. **Jelaskan dan berikan screenshot hasil percobaan Anda.**

The screenshot shows a web browser's developer tools interface. At the top, the method is set to 'GET' and the URL is 'localhost/PWL_POS/public/api/levels'. The 'Send' button is visible. Below the URL bar, the 'Body' tab is selected, showing a '200 OK' status, a response time of 42.72 s, and a body size of 925 B. The response is displayed in JSON format, showing an array of four objects representing user levels.

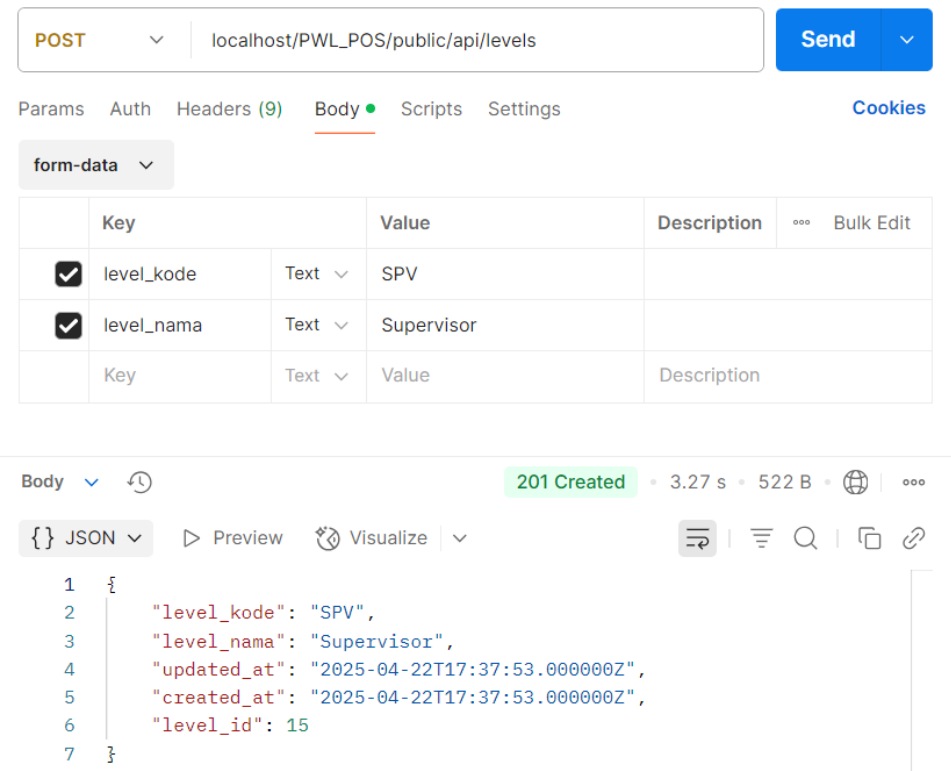
```
1  [
2    {
3      "level_id": 1,
4      "level_kode": "ADM",
5      "level_nama": "Administrator",
6      "created_at": null,
7      "updated_at": null
8    },
9    {
10     "level_id": 2,
11     "level_kode": "MNG",
12     "level_nama": "Manager",
13     "created_at": null,
14     "updated_at": null
15   },
16   {
17     "level_id": 3,
18     "level_kode": "STF",
19     "level_nama": "Staff/Kasir",
20     "created_at": null,
21     "updated_at": null
22   },
23   {
24     "level_id": 4,
25     "level_kode": "CUS",
26     "level_nama": "Customer",
```

Jadi menggunakan get untuk mengambil data dengan api/levels dan ditampilkan isi datanya.

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



Jelaskan dan berikan screenshoot hasil percobaan Anda.



Jadi menggunakan post untuk membuat data baru pada api/levels dengan isi form tersebut.

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

GET localhost/PWL_POS/public/api/levels/4 Send

Params Auth Headers (8) Body ● Scripts Settings Cookies

Query Params

	Key	Value	Description	...	Bulk Edit
	Key	Value	Description		

Body 200 OK • 40.67 s • 491 B

{ } JSON Preview Visualize

```
1 [
2   {
3     "level_id": 4,
4     "level_kode": "CUS",
5     "level_nama": "Customer",
6     "created_at": "2025-02-28T04:38:47.000000Z",
7     "updated_at": null
8   }
9 ]
```

Menggunakan get untuk mengambil data berdasarkan api/levels/{id} agar bisa melihat detail data berdasarkan idnya.

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": "2024-04-22T21:48:32.000000Z",
7     "updated_at": "2024-04-22T21:48:19.000000Z"
8   }
9 ]
```

Jelaskan dan berikan screenshoot hasil percobaan Anda.

PUT localhost/PWL_POS/public/api/levels/15?level_kode=SPR Send

Params Auth Headers (8) Body Scripts Settings Cookies

Query Params

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

Body 200 OK • 891 ms • 519 B

{ } JSON Preview Visualize

```
1 [
2   {
3     "level_id": 15,
4     "level_kode": "SPR",
5     "level_nama": "Supervisor",
6     "created_at": "2025-04-22T17:37:53.000000Z",
7     "updated_at": "2025-04-22T17:48:01.000000Z"
8   }
9 ]
```

Jadi menggunakan put untuk memperbarui data berdasarkan api/levels/{id} dan parameter level_kode untuk inputan data baru lalu akan ditampilkan dengan data yang barunya.

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

DELETE localhost/PWL_POS/public/api/levels/15 Send

Params Auth Headers (8) Body Scripts Settings Cookies

Query Params

	Key	Value	Description	...	Bulk Edit
	Key	Value	Description		

Body 200 OK • 2.93 s • 413 B

{ } JSON Preview Visualize

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

Menggunakan delete untuk menghapus data dengan api/levels/{id}, jadi berdasarkan level_idnya.

9. Lakukan commit perubahan file pada Github.

Link Commit :

https://github.com/Gilangp/Pemrograman_Web_Lanjut/commit/4a85899adb38dc3e7005e12db020ad571bcd3914

TUGAS

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

Link Commit :

https://github.com/Gilangp/Pemrograman_Web_Lanjut/commit/83575608c4d94575b7181ba06d392a7f9ea513f1