

TUGAS
WEB Lanjut

PERTEMUAN 9

API Bagian 1

OLEH

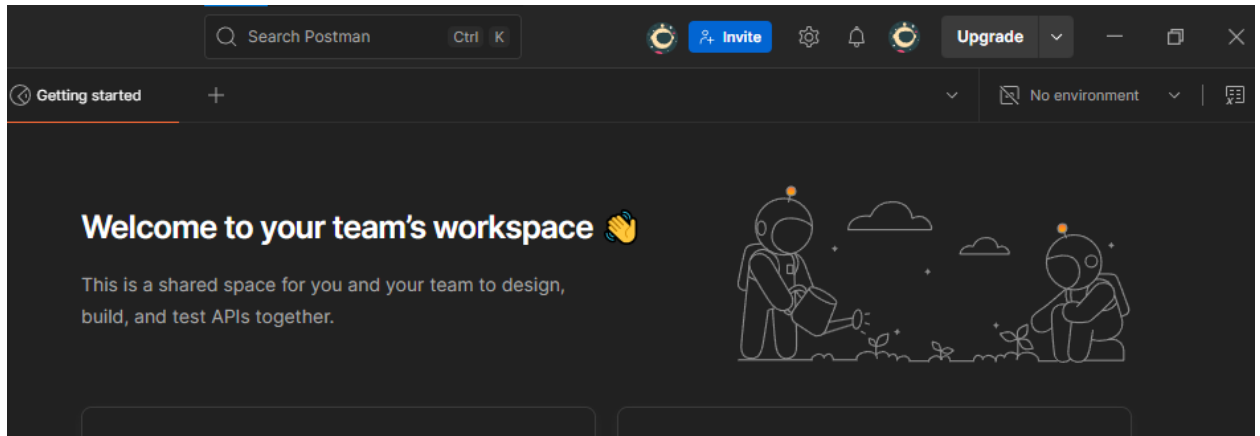
MUHAMMAD NUR AZIZ NIM. 2341720237



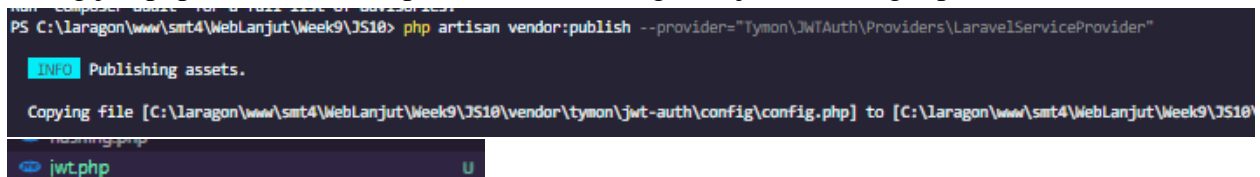
PROGRAM STUDI D-IV TEKNIK INFORMATIKA
JURUSAN TEKNOLOGI INFORMASI
POLITEKNIK NEGERI MALANG
21 April 2025

Praktikum 1 – Membuat RESTful API Register

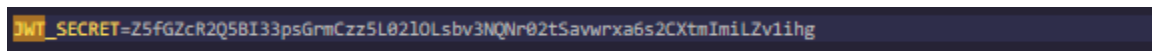
1. Sebelum memulai membuat REST API, terlebih dahulu download aplikasi Postman di <https://www.postman.com/downloads>. Aplikasi ini akan digunakan untuk mengerjakan semua tahap praktikum pada Jobsheet ini.



2. Lakukan instalasi JWT dengan mengetikkan perintah berikut: `composer require tymon/jwt-auth:2.1.1` Pastikan Anda terkoneksi dengan internet.
3. Setelah berhasil menginstall JWT, lanjutkan dengan publish konfigurasi file dengan perintah berikut: `php artisan vendor:publish --provider="Tymon\JWTAuth\Providers\LaravelServiceProvider"`
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`.



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

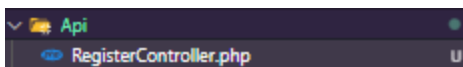
```
'guards' => [
    'web' => [
        'driver' => 'session',
        'provider' => 'users',
    ],
    'api' => [
        'driver' => 'jwt',
        'provider' => 'users',
    ],
],
```

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

```
use Tymon\JWTAuth\Contracts\JWTSubject;
use Illuminate\Database\Eloquent\Relations\BelongsTo;
use Illuminate\Foundation\Auth\User as Authenticatable;

class UserModel extends Authenticatable implements JWTSubject
{
    public function getJWTIdentifier(){
        return $this->getKey();
    }
    public function getJWTCustomClaims(){
        return [];
    }
}
```

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.



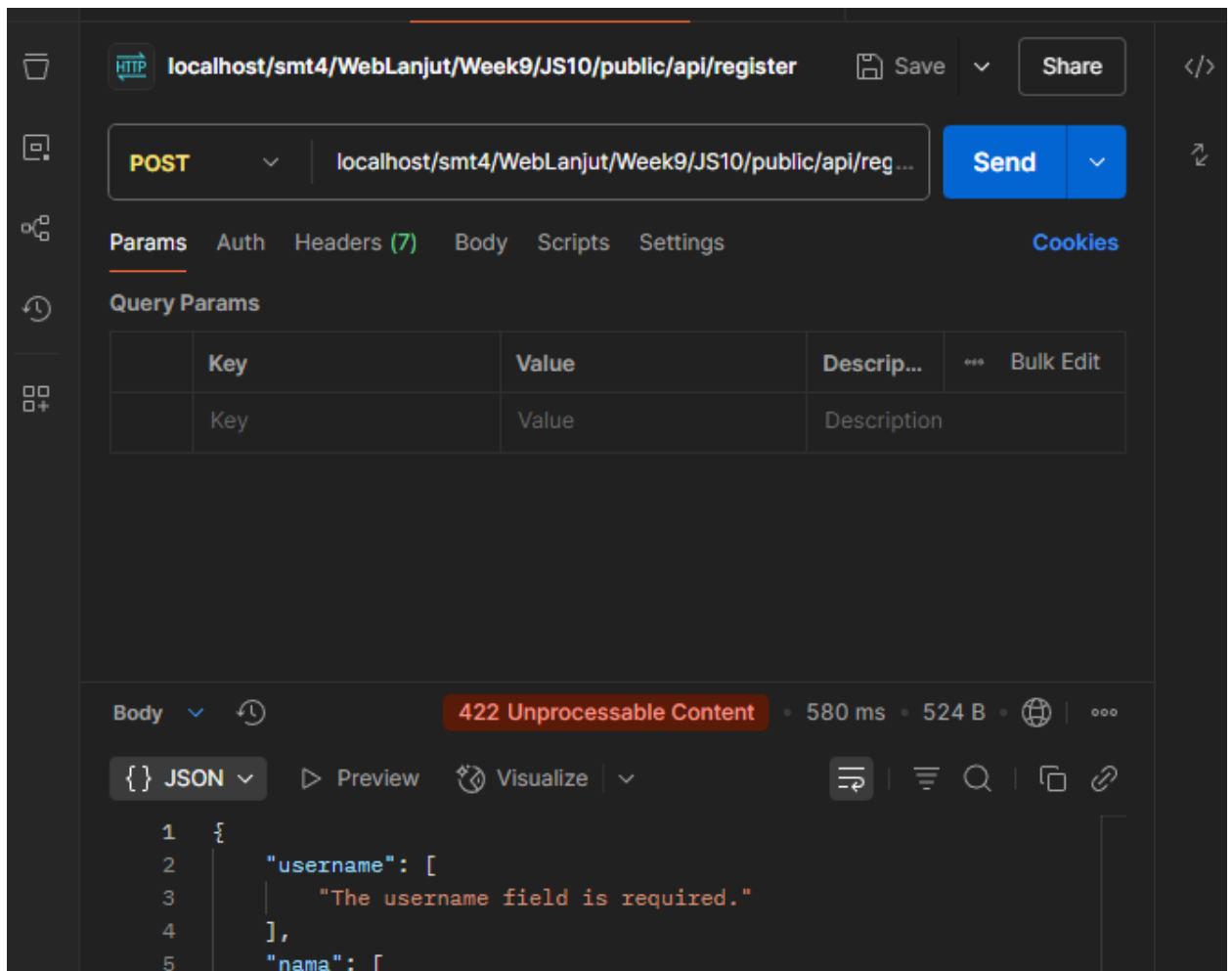
9. Buka file tersebut, dan ubah kode menjadi seperti berikut.
10. Selanjutnya buka routes/api.php, ubah semua kode menjadi seperti berikut.

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

/*
|-----
| 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', 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



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 screenshot hasil percobaan Anda.

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

Key	Value	Description
username	penggunasatu	
nama	Penggunana1	
password	123456	
level_id	2	
password_con...	123456	

The response status is **201 Created** (418 ms, 562 B). The JSON response is:

```

{
  "success": true,
  "user": {
    "username": "penggunasatu",
    "nama": "Penggunana1",
    "level_id": "2",
    "updated_at": "2025-04-23T03:36:50.000000Z",
    "created_at": "2025-04-23T03:36:50.000000Z",
    "user_id": 24
  }
}

```

Below the JSON response, a table from a web application is shown:

ID	Username	Name	Role	Actions
16	penggunasatu	Penggunana1	Manager	Detail Edit Hapus

Showing 11 to 16 of 16 entries. Page 2 of 2.

13. Lakukan commit perubahan file pada Github.

<https://github.com/ren5602/smt4/commit/0813f2a097afbd435cb81f6842247244dae3ff>

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\smt4\WebLanjut\Week9\JS10> php artisan make:controller Api/LoginController

INFO Controller [C:\laragon\www\smt4\WebLanjut\Week9\JS10\app\Http\Controllers\Api>LoginController.php] created successfully

```

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

```

class LoginController extends Controller
{
    //set validation
    $validator = Validator::make($request->all(), [
        'username' => 'required',
        'password' => 'required'
    ]);

    //if validation fails
    if ($validator->fails()) {
        return response()->json($validator->errors(), 422);
    }

    //get credentials from request
    $credentials = $request->only('username', 'password');

    //if auth failed
    if (!$token = auth()->guard('api')->attempt($credentials)) {
        return response()->json([
            'success' => false,
            'message' => 'Username atau Password Anda salah'
        ], 401);
    }

    //if auth success
    return response()->json([
        'success' => true,
        'user' => auth()->guard('api')->user(),
        'token' => $token
    ], 200);
}

```

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

```

19 Route::post('/register', RegisterController::class)->name('register');
20 Route::post('/login', LoginController::class)->name('login');
21 Route::middleware('auth:api')->get('/user', function (Request $request) {
22     return $request->user();
23 });
24

```

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

The screenshot shows the Postman interface with a POST request to `127.0.0.1:8000/api/login`. The response status is `422 Unprocessable Content` with a response time of 145 ms and a body size of 422 B. The response body is in JSON format:

```

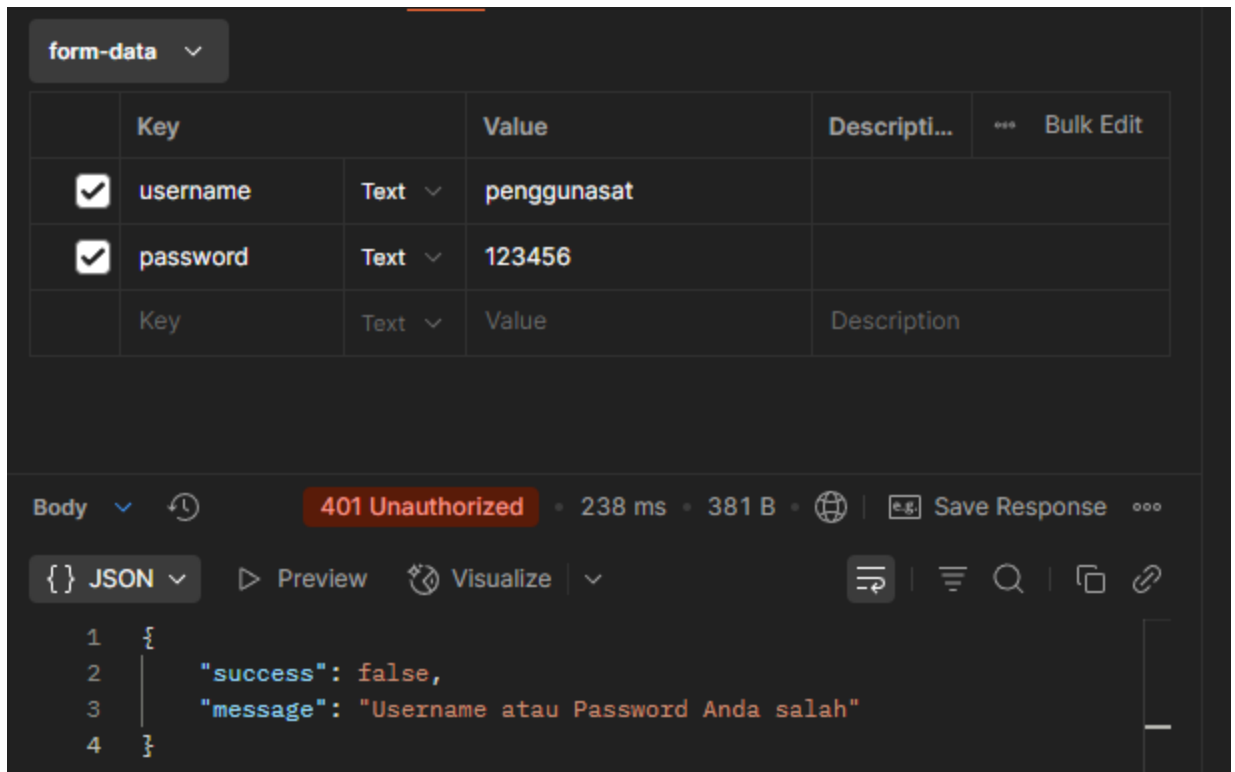
{
  "username": [
    "The username field is required."
  ],
  "password": [
    "The password field is required."
  ]
}

```

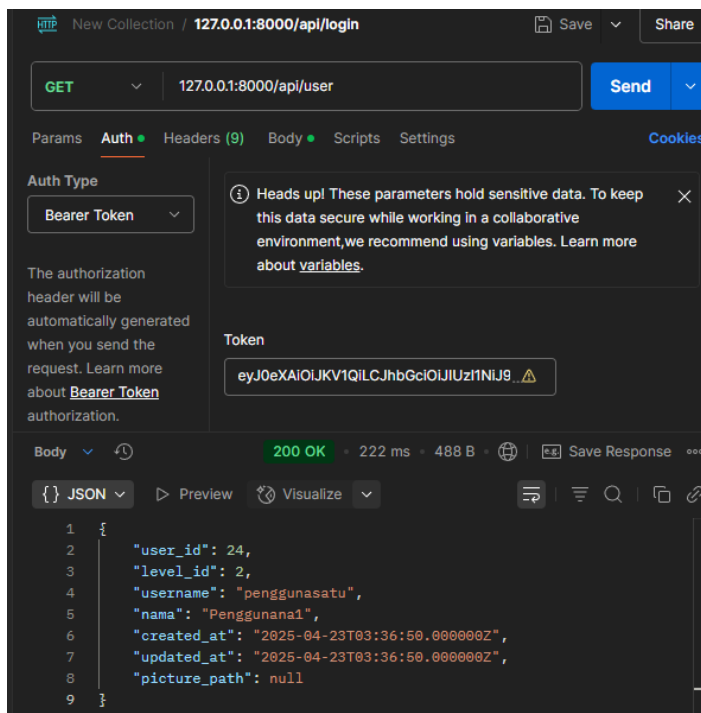
- Selanjutnya, isikan username dan password sesuai dengan data user yang ada pada database. Klik tab Body dan pilih form-data. Isikan key sesuai dengan kolom data, serta isikan data user. Klik tombol Send, jika berhasil maka akan keluar tampilan seperti berikut. Copy nilai token yang diperoleh pada saat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



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



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.



- Membuat logincontroller API

- Membuat route pada api
 - Kemudian melakukan post login, simpan token
 - Selanjutnya melakukan request get dari user yang login dengan auth token dan memperlihatkan user yang sedang 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

```
JWT_SECRET=Z5fGZcR2Q5BI33psGrnCzz5L0210Lsbv3NQNr02tSavwrx6s2CXtmImiLZv1ihg
JWT_SHOW_BLACKLIST_EXCEPTION=true
```

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

```
PS C:\laragon\www\smt4\WebLanjut\Week9\JS10> php artisan make:controller Api/LogoutController

INFO Controller [C:\laragon\www\smt4\WebLanjut\Week9\JS10\app\Http\Controllers\Api/LogoutController.php] created successfully.
```

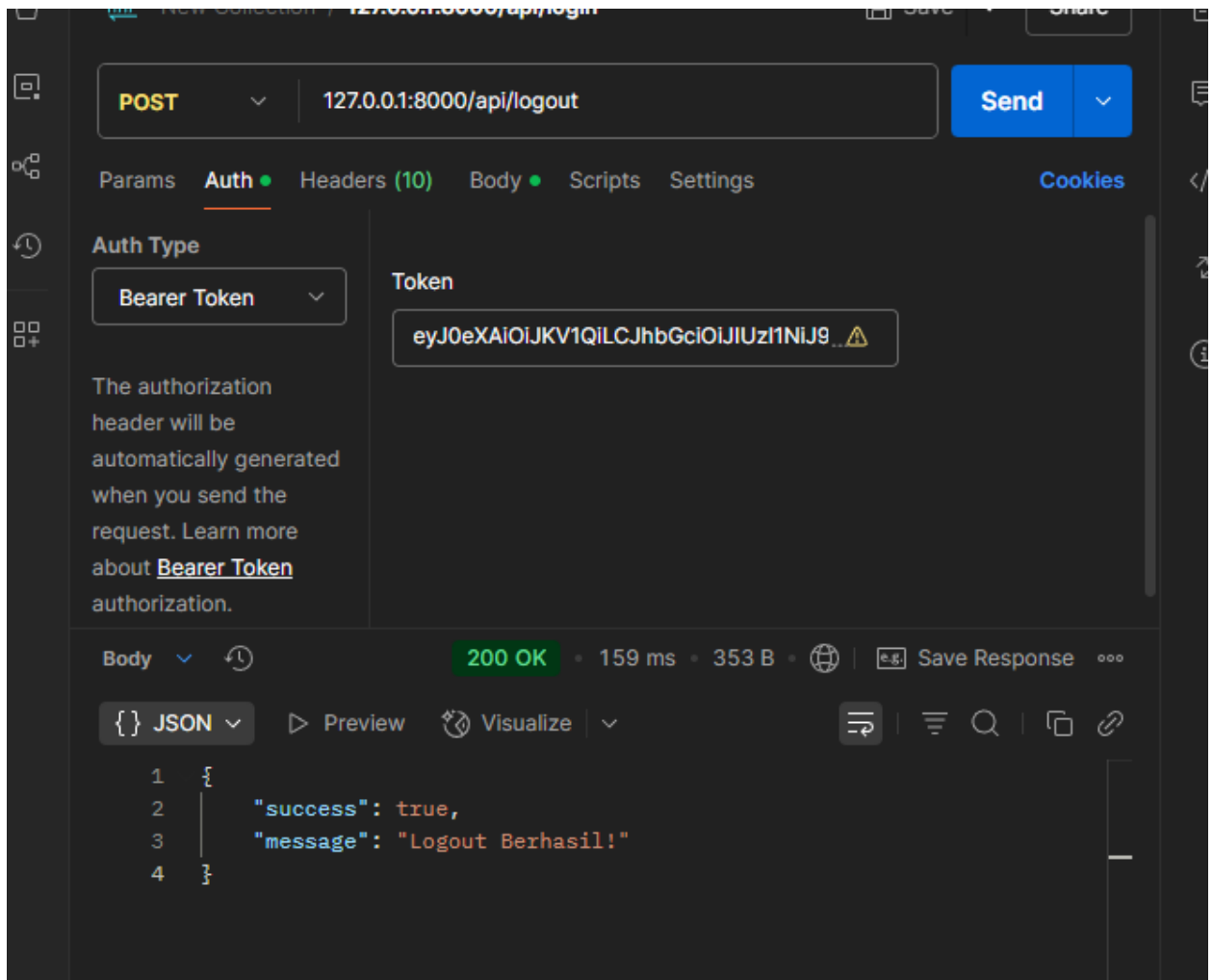
3. Buka file tersebut dan ubah kode menjadi seperti berikut.

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

4. Lalu kita tambahkan routes pada api.php

```
Route::post('/login', LoginController::class)->name('login');
Route::post('/logout', LogoutController::class)->name('logout');
Route::middleware('auth:api')->get('/user', function (Request $request) {
```

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.



7. Lakukan commit perubahan file pada Github.

<https://github.com/ren5602/smt4/commit/e2929109186c156710ce42a0260355724f022fcd>

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\smt4\WebLanjut\Week9\JS10> php artisan make:controller Api/LevelController
INFO: Controller [C:\laragon\www\smt4\WebLanjut\Week9\JS10\app\Http\Controllers\Api\LevelController.php] created successfully.
```

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

```

Windsurf: Refactor | Explain
class LevelController extends Controller
{
    Windsurf: Refactor | Explain | Generate Function Comment | X
    public function index()
    {
        return LevelModel::all();
    }

    Windsurf: Refactor | Explain | Generate Function Comment | X
    public function store(Request $request)
    {
        $level = LevelModel::create($request->all());
        return response()->json($level, 201);
    }

    Windsurf: Refactor | Explain | Generate Function Comment | X
    public function show(LevelModel $level)
    {
        return LevelModel::find($level);
    }

    Windsurf: Refactor | Explain | Generate Function Comment | X
    public function update(Request $request, LevelModel $level)
    {
        $level->update($request->all());
        return LevelModel::find($level);
    }
}

```

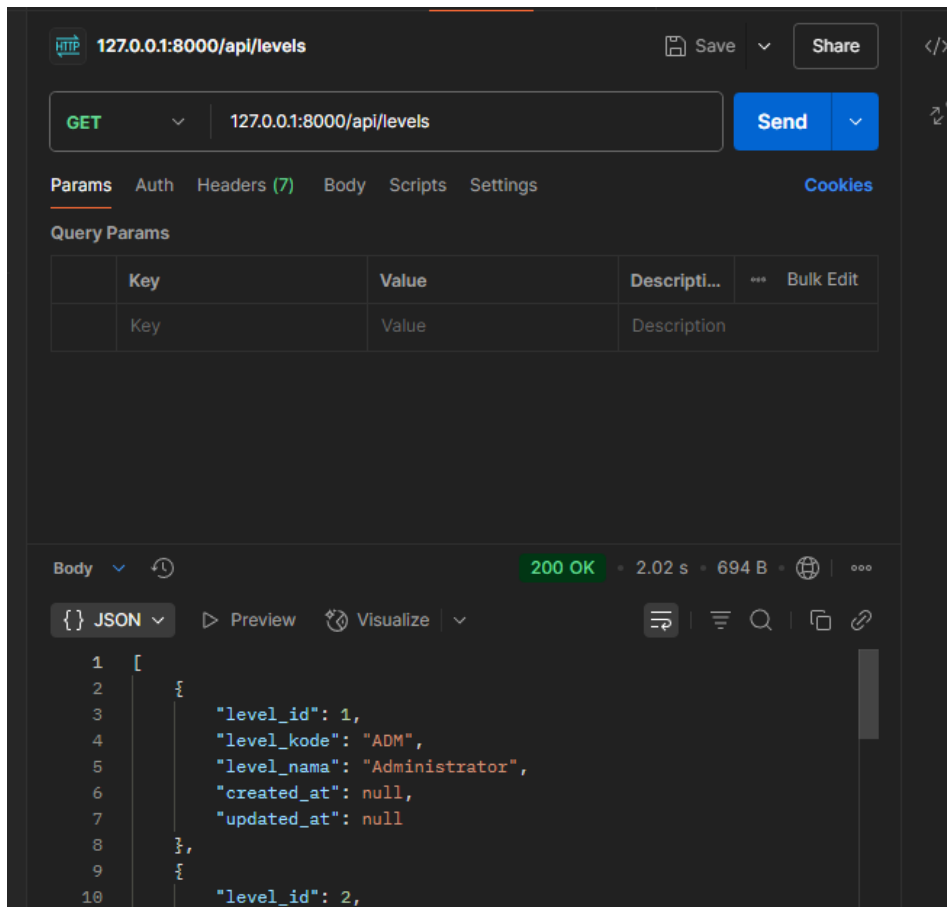
3. Kemudian kita lengkapi routes pada api.php.

```

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



- Mengambil data informasi dari setiap level yang ada di dalam database
5. Kemudian, lakukan percobaan penambahan data dengan URL :
localhost/PWL_POSmain/public/api/levels dan method POST seperti di bawah ini.
Jelaskan dan berikan screenshot hasil percobaan Anda.

POST 127.0.0.1:8000/api/levels Send

Params Auth Headers (9) **Body** Scripts Settings Cookies

form-data

	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 201 Created • 546 ms • 458 B •

{ } JSON Preview Visualize

```
1 {
2   "level_kode": "SPV",
3   "level_nama": "Supervisor",
4   "updated_at": "2025-04-23T06:34:43.000000Z",
5   "created_at": "2025-04-23T06:34:43.000000Z",
6   "level_id": 7
7 }
```

Membuat post levels dengan menginput level_kode SPV dan level_nama Supervisor

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

GET 127.0.0.1:8000/api/levels/7 Send

Params Auth Headers (9) **Body** Scripts Settings Cookies

form-data

	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	

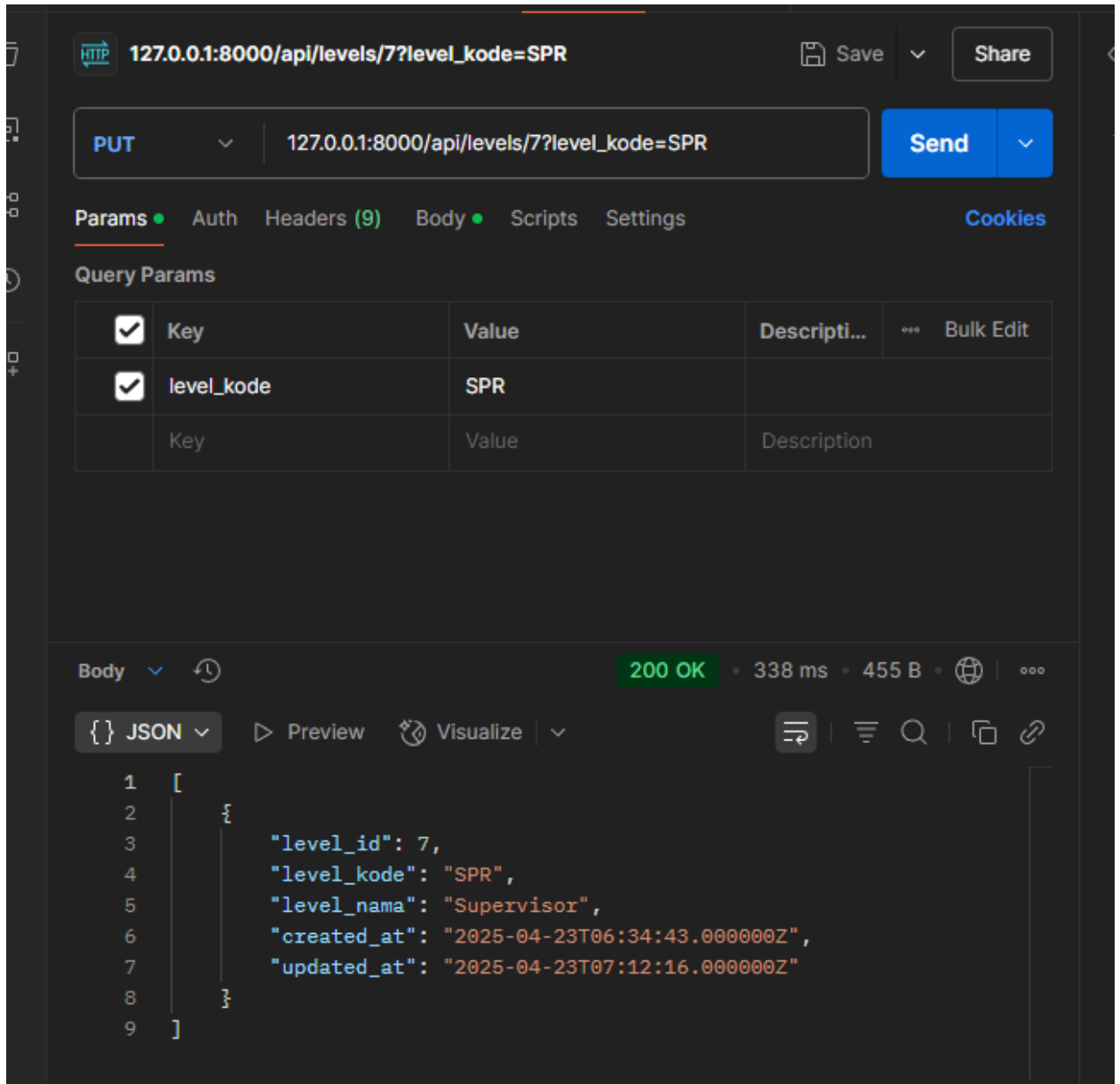
Body 200 OK 179 ms 455 B

{ } JSON Preview Visualize

```
1 [
2   {
3     "level_id": 7,
4     "level_kode": "SPV",
5     "level_nama": "Supervisor",
6     "created_at": "2025-04-23T06:34:43.000000Z",
7     "updated_at": "2025-04-23T06:34:43.000000Z"
8   }
9 ]
```

Menngambil informasi dari level dengan level id 7

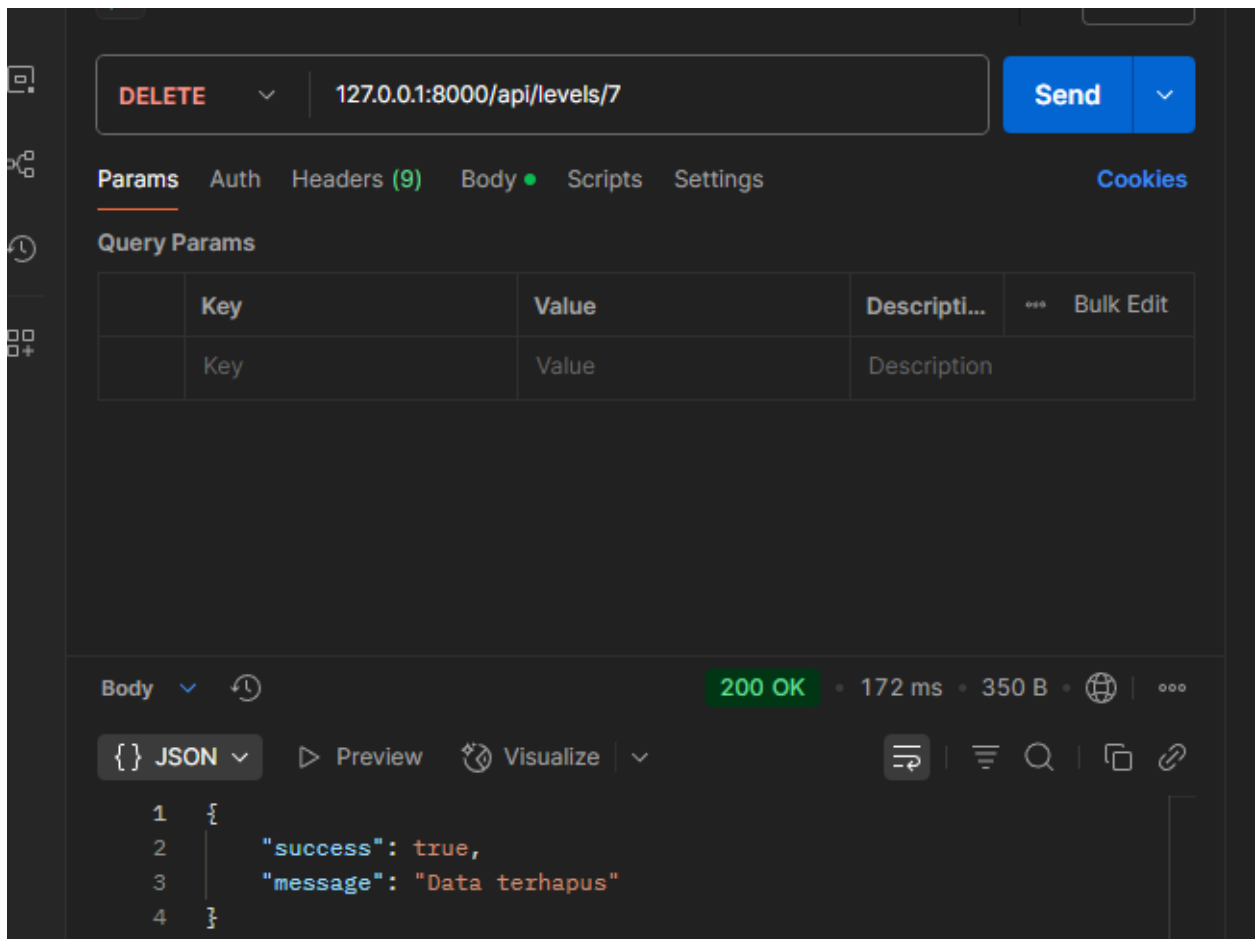
7. Jika sudah, kita coba untuk melakukan edit data menggunakan `localhost/PWL_POSmain/public/api/levels/{id}` dan method PUT. Isikan data yang ingin diubah pada tab Param



Jelaskan dan berikan screenshoot hasil percobaan Anda.

Melakukan edit data dengan parameter level_kode dari SPV ke SPR dengan request put

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



Melakukan request delete untuk menghapus data Level Supervisor

9. Lakukan commit perubahan file pada Github.

TUGAS

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

<https://github.com/ren5602/smt4/commit/ead35bfe95fc7432aded7efa66f197377fa15052>