

Jobsheet 10: RESTFUL API

Dosen Pengampu Mata Kuliah :

M. Hasyim Ratsanjani, S.Kom., M.Kom.



Dibuat Oleh :

Daffa Yudisa Akbar : TI-2H / 10 : 2241720008

Politeknik Negeri Malang

Jurusan Teknologi Informasi

D-IV Teknik Informatika

2024

1. Praktikum 1 - Membuat RESTful API Register

No.	Langkah Praktikum
1.	<p>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.</p>
2.	<p>Lakukan instalasi JWT dengan mengetikkan perintah berikut: composer require tymon/jwt-auth:2.1.1 Pastikan Anda terkoneksi dengan internet.</p>

3.	<p>Setelah berhasil menginstall JWT, lanjutkan dengan publish konfigurasi file dengan perintah berikut:</p> <p>php artisan vendor: publish --provider="Tymon\JWTAuth\Providers\LaravelServiceProvider"</p> <pre>PS C:\laragon\www\PWL_POS> php artisan vendor:publish --provider="Tymon\JWTAuth\Providers\LaravelServiceProvider" [INFO] Publishing assets. Copying file [C:\laragon\www\PWL_POS\vendor\tymon\jwt-auth\config\config.php] to [C:\laragon\www\PWL_POS\config\jwt.php] DONE PS C:\laragon\www\PWL_POS></pre>
4.	<p>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.</p>
5.	<p>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.</p> <pre>PS C:\laragon\www\PWL_POS> php artisan jwt:secret jwt-auth secret [0Q5ghur63TBoXFY89z4JjVvCp40x2WmJzy7JTRfTpBAj8sP99hAr1UzFgghupDUq] set successfully. PS C:\laragon\www\PWL_POS></pre>
6.	<p>Selanjutnya lakukan konfigurasi guard API. Buka config/auth.php. Ubah bagian 'guards' menjadi seperti berikut.</p> <pre>38 'guards' => [39 'web' => [40 'driver' => 'session', 41 'provider' => 'users', 42], 43 'api' => [44 'driver' => 'jwt', 45 'provider' => 'users', 46], 47],</pre>
7.	<p>Kita akan menambahkan kode di model UserModel, ubah kode seperti berikut:</p> <pre>auth.php M UserModel.php M app > Models > UserModel.php > ... 1 <?php 2 3 namespace App\Models; 4 5 use Illuminate\Database\Eloquent\Factories\HasFactory; 6 use Illuminate\Database\Eloquent\Model; 7 use Illuminate\Database\Eloquent\Relations\BelongsTo; 8 use App\Models\LevelModel; 9 use Tymon\JWTAuth\Contracts\JWTSubject; 10 use Illuminate\Foundation\Auth\User as Authenticatable; 11 12 13 class UserModel extends Authenticatable implements JWTSubject 14 { 15 16 public function getJWTIdentifier(){ 17 return \$this->getKey(); 18 } 19 20 public function getJWTCustomClaims(){ 21 return []; 22 } 23 24 use HasFactory; 25 26 protected \$table = 'm_user'; 27 protected \$primaryKey = 'user_id'; 28 29 protected \$fillable = ['level_id', 'username', 'name', 'password']; 30 31 public function level(): BelongsTo 32 { 33 return \$this->belongsTo(LevelModel::class, 'level_id', 'level_id'); 34 } 35 36 }</pre>

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\PWL_POS> php artisan make:controller Api/RegisterController

INFO Controller [C:\laragon\www\PWL_POS\app\Http\Controllers\Api\RegisterController.php] created successfully.

PS C:\laragon\www\PWL_POS>
```

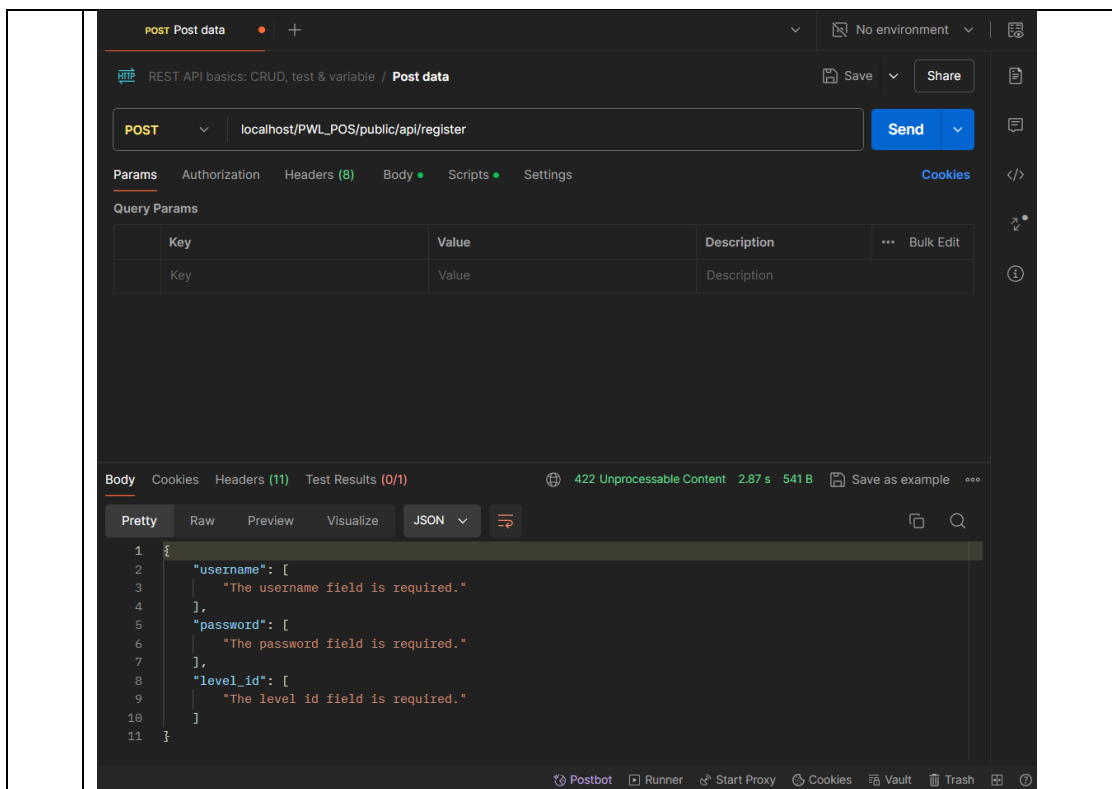
9. Buka file tersebut, dan ubah kode menjadi seperti berikut.

```
auth.php M UserModel.php M RegisterController.php U X
app > Http > Controllers > Api > RegisterController.php > PHP Intelephense > RegisterController > __invoke
7 use App\Models\UserModel;
8 use Illuminate\Support\Facades\Validator;
9
10 class RegisterController extends Controller
11 {
12     public function __invoke(Request $request)
13     {
14         $validator = Validator::make($request->all(), [
15             'username' => 'required',
16             'name' => 'required',
17             'password' => 'required|min:5|confirmed',
18             'level_id' => 'required'
19         ]);
20
21         if($validator->fails()){
22             return response()->json($validator->errors(), 422);
23         }
24
25         $user = UserModel::create([
26             'username' => $request->username,
27             'name' => $request->name,
28             'password' => bcrypt($request->password),
29             'level_id' => $request->level_id
30         ]);
31
32         if($user){
33             return response()->json([
34                 'success' => true,
35                 'user' => $user,
36             ], 201);
37         }
38
39         return response()->json([
40             'success' => false,
41         ], 409);
42     }
43 }
44
```

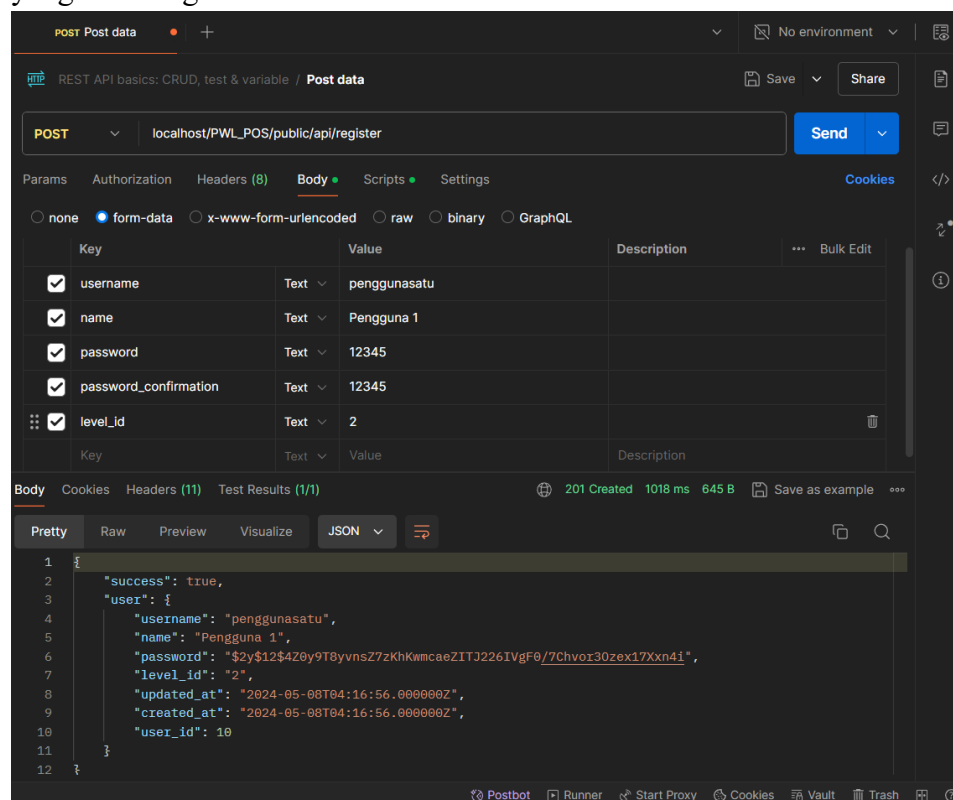
10. Selanjutnya buka routes/api.php, ubah semua kode menjadi seperti berikut.

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

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.

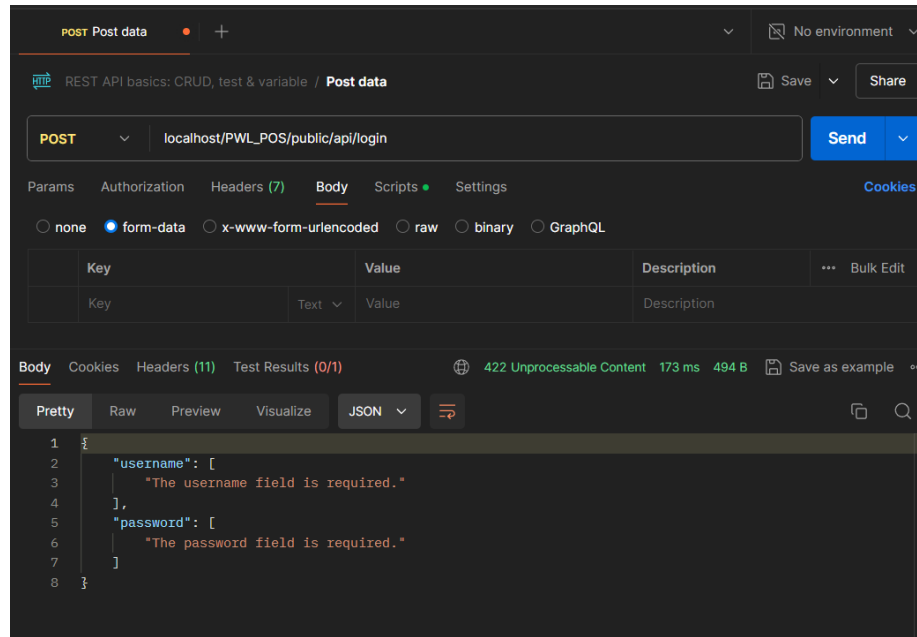


13. Lakukan commit perubahan file pada Github.

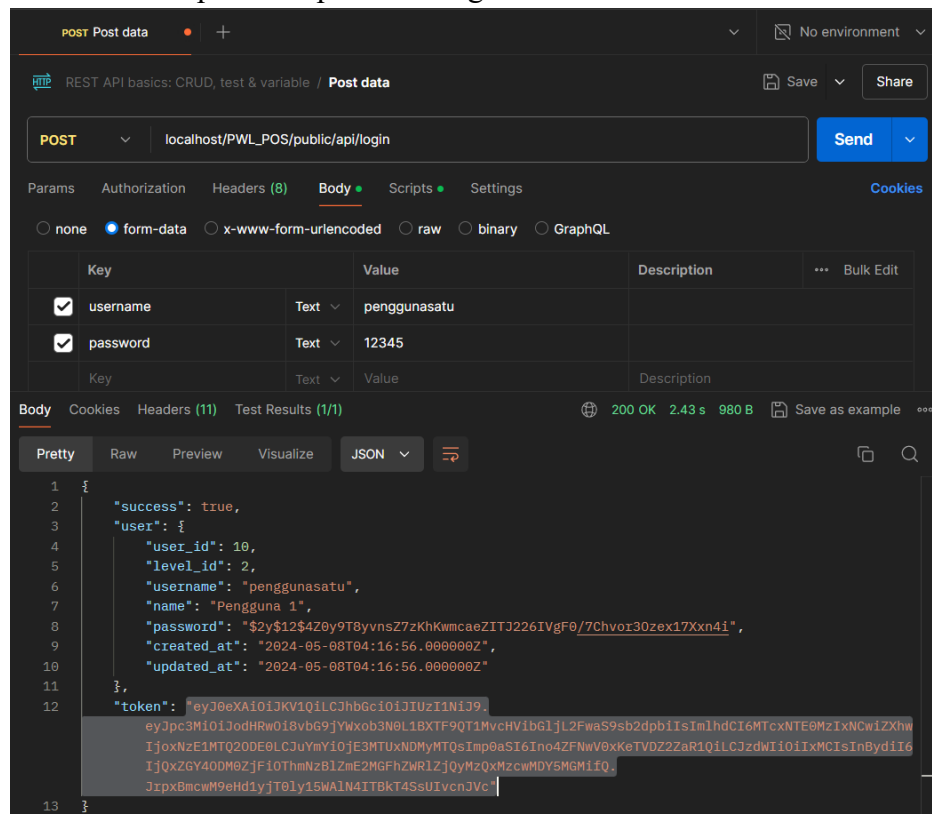
2. Praktikum 2 - Membuat RESTful API Login

No.	Langkah Praktikum
1.	<p>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.</p> <pre> PS C:\laragon\www\PWL_POS> php artisan make:controller Api/LoginController [INFO] Controller [C:\laragon\www\PWL_POS\app\Http\Controllers\Api>LoginController.php] created successfully. PS C:\laragon\www\PWL_POS> </pre>
2.	<p>Buka file tersebut, dan ubah kode menjadi seperti berikut.</p> <pre> LoginController.php U X app > Http > Controllers > Api > LoginController.php > PHP Intelephense > LoginController > __invoke 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 \$validator = Validator::make(\$request->all(), [14 'username' => 'required', 15 'password' => 'required' 16]); 17 18 if (\$validator->fails()){ 19 return response()->json(\$validator->errors(), 422); 20 } 21 22 \$credentials = \$request->only('username', 'password'); 23 24 if (!\$token = auth()->guard('api')->attempt(\$credentials)){ 25 return response()->json([26 'success' => false, 27 'message' => 'Username atau Password Salah' 28], 410); 29 } 30 31 return response()->json([32 'success' => true, 33 'user' => auth()->user(), 34 'token' => \$token 35]); 36 } 37 } 38 </pre>
3.	<p>Berikutnya tambahkan route baru pada file api.php yaitu /login dan /user</p> <pre> LoginController.php U api.php M X routes > api.php > ... 1 <?php 2 3 use Illuminate\Http\Request; 4 use Illuminate\Support\Facades\Route; 5 6 /* 7 * API Routes 8 * 9 * Here is where you can register API routes for your application. These 10 * routes are loaded by the RouteServiceProvider and all of them will 11 * be assigned to the "api" middleware group. Make something great! 12 */ 13 14 Route::post('/register', App\Http\Controllers\Api\RegisterController::class)->name('register'); 15 Route::post('/login', App\Http\Controllers\Api>LoginController::class)->name('login'); 16 Route::middleware('auth:api')->get('/user', function (Request \$request) { 17 return \$request->user(); 18 }); 19 </pre>

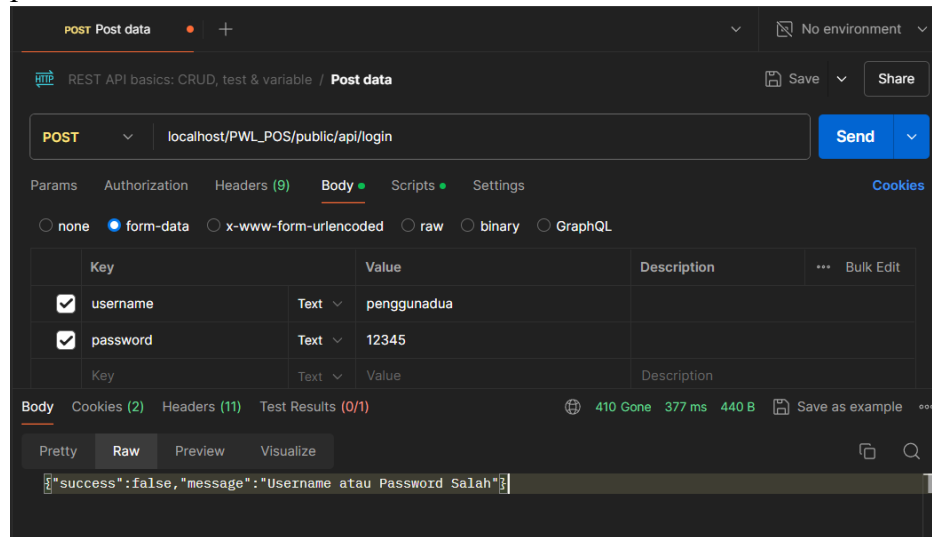
- | | |
|----|---|
| 4. | Jika sudah, kita akan melakukan uji coba REST API melalui aplikasi Postman. Buka aplikasi Postman, isi URL localhost/PWL_POS/public/api/login serta method POST. Klik Send. |
|----|---|



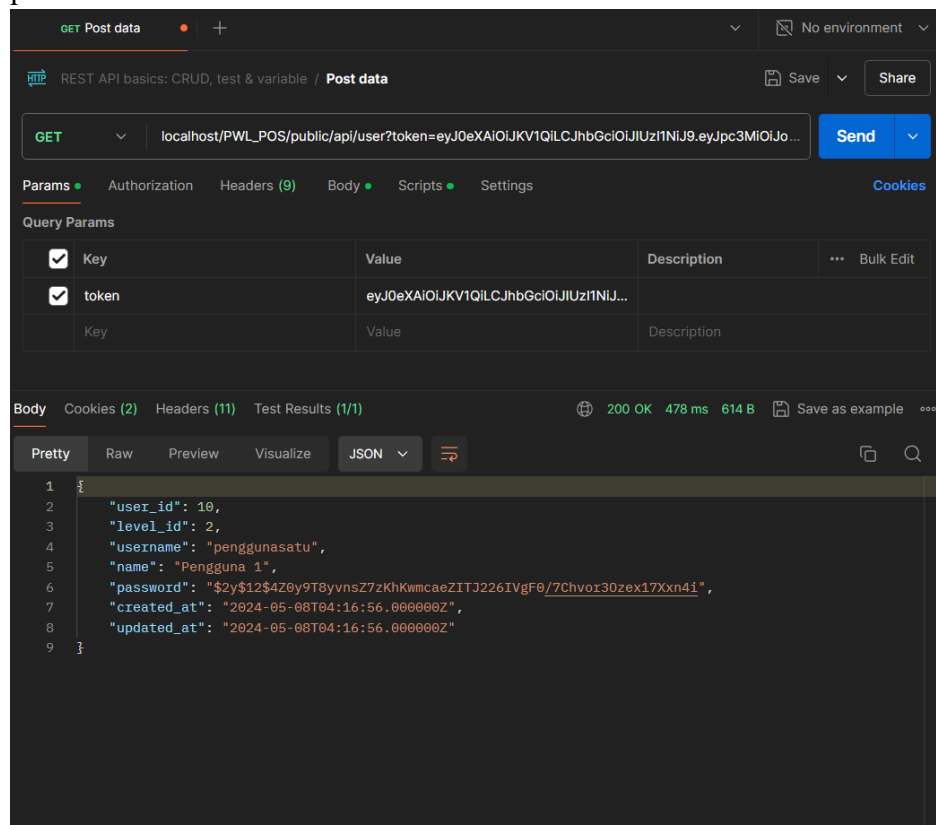
- | | |
|----|---|
| 5. | Selanjutnya, isikan username dan password sesuai dengan data user yang ada pada database. Klik tab Body dan pilih form-data. Isikan key sesuai dengan kolom data, serta isikan data user. Klik tombol Send, jika berhasil maka akan keluar tampilan seperti berikut. Copy nilai token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



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



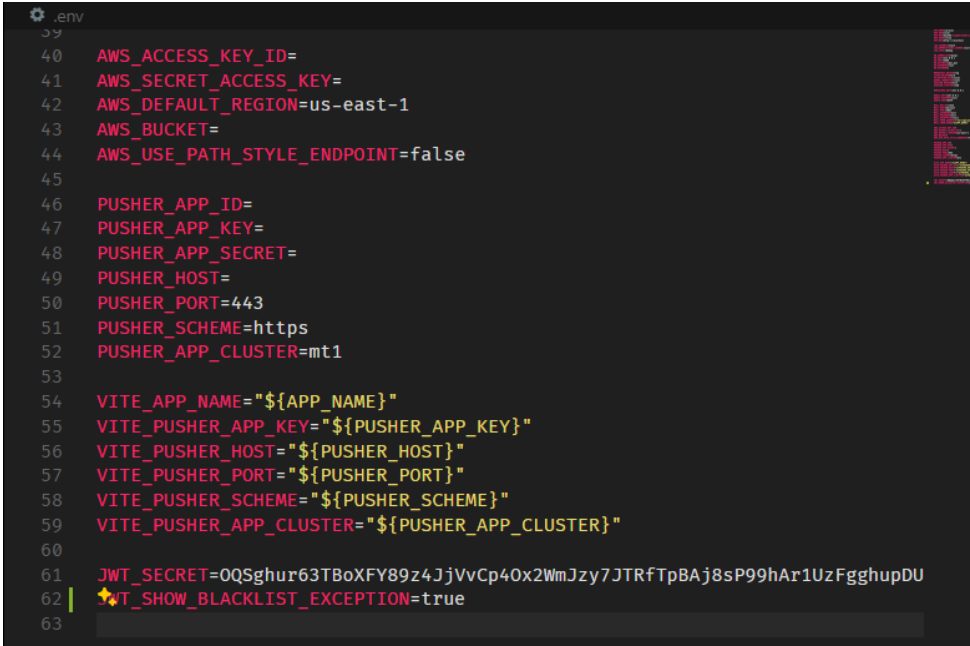
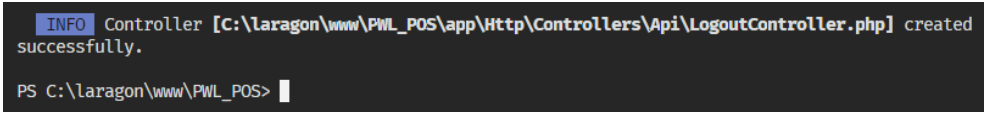
7. Coba kembali melakukan login dengan data yang benar. Sekarang mari kita coba menampilkan data user yang sedang login menggunakan URL `localhost/PWL_POS/public/api/user` dan method GET. Jelaskan hasil dari percobaan tersebut.

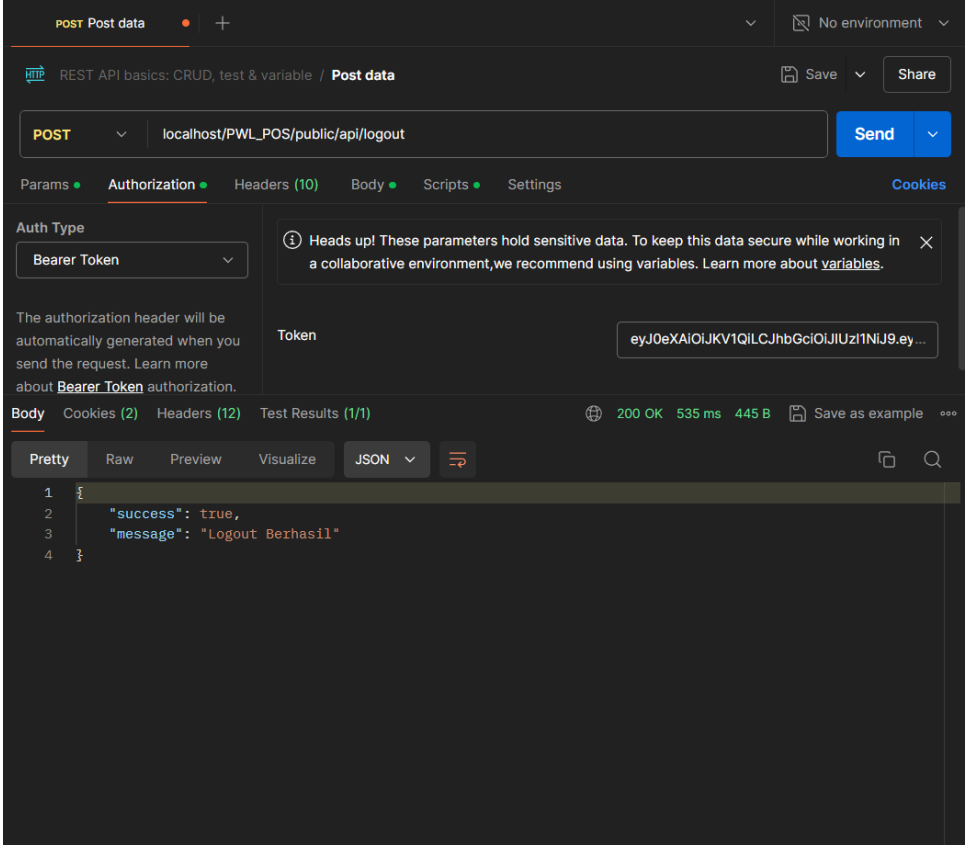


- Pada percobaan di atas, saya mencoba untuk mengakses user berdasarkan token yang didapat saat proses login dilakukan. Setelah dijalankan akan muncul detail informasi dari user berdasarkan token yang ada. Hal tersebut bisa terjadi karena saya memberi kode program

	route middleware yang outputnya akan mengembalikan nilai berupa data user, sehingga output yang diberikan berupa detail dari data user.
8.	Lakukan commit perubahan file pada Github.

3. Praktikum 3 - Membuat RESTful API Logout

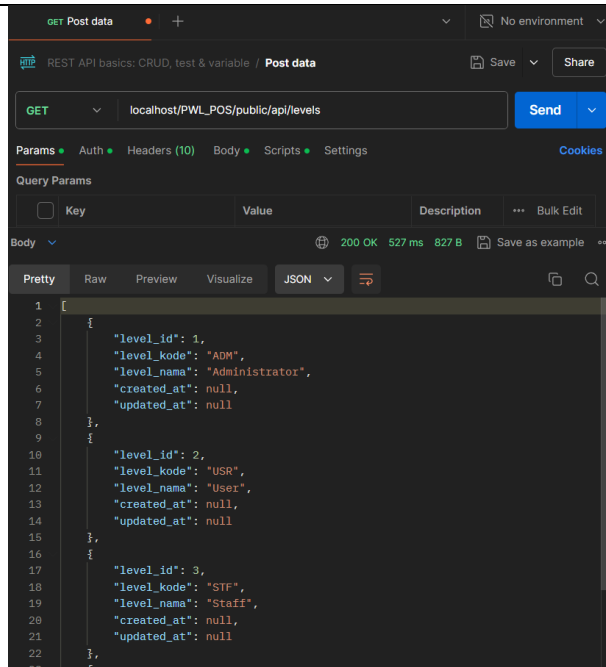
No.	Langkah Praktikum
1.	<p>Tambahkan kode berikut pada file .env</p> <p>JWT_SHOW_BLACKLIST_EXCEPTION=true</p>  <pre> 40 AWS_ACCESS_KEY_ID= 41 AWS_SECRET_ACCESS_KEY= 42 AWS_DEFAULT_REGION=us-east-1 43 AWS_BUCKET= 44 AWS_USE_PATH_STYLE_ENDPOINT=false 45 46 PUSHER_APP_ID= 47 PUSHER_APP_KEY= 48 PUSHER_APP_SECRET= 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=0QSghur63TBoXFY89z4JjVvCp40x2WmJzy7JTRfTpBAj8sP99hAr1UzFgghupDU 62 JWT_SHOW_BLACKLIST_EXCEPTION=true 63 </pre>
2.	<p>Buat Controller baru dengan nama LogoutController. php artisan make:controller Api/LogoutController</p>  <pre> INFO Controller [C:\laragon\www\PWL_POS\app\Http\Controllers\Api\LogoutController.php] created successfully. PS C:\laragon\www\PWL_POS> </pre>
3.	Buka file tersebut dan ubah kode menjadi seperti berikut.

	<pre> 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 use Tymon\JWTAuth\Facades\JWTAuth; 8 use Tymon\JWTAuth\Exceptions\TokenExpiredException; 9 use Tymon\JWTAuth\Exceptions\TokenInvalidException; 10 11 12 class LogoutController extends Controller 13 { 14 public function __invoke(Request \$request) 15 { 16 \$removeToken = JWTAuth::invalidate(JWTAuth::getToken()); 17 18 if (\$removeToken){ 19 return response()->json([20 'success' => true, 21 'message' => 'Logout Berhasil' 22]); 23 } 24 } 25 } 26 </pre>
4.	<p>Lalu kita tambahkan routes pada api.php</p> <pre> 21 }); 22 Route::post('/logout', App\Http\Controllers\Api\LogoutController::class->name('logout')); 23 </pre>
5.	<p>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.</p>
6.	<p>Isi token pada tab Authorization, pilih Type yaitu Bearer Token. Isikan token yang didapat saat login. Jika sudah klik Send.</p> 

7.	Lakukan commit perubahan file pada Github.
----	--

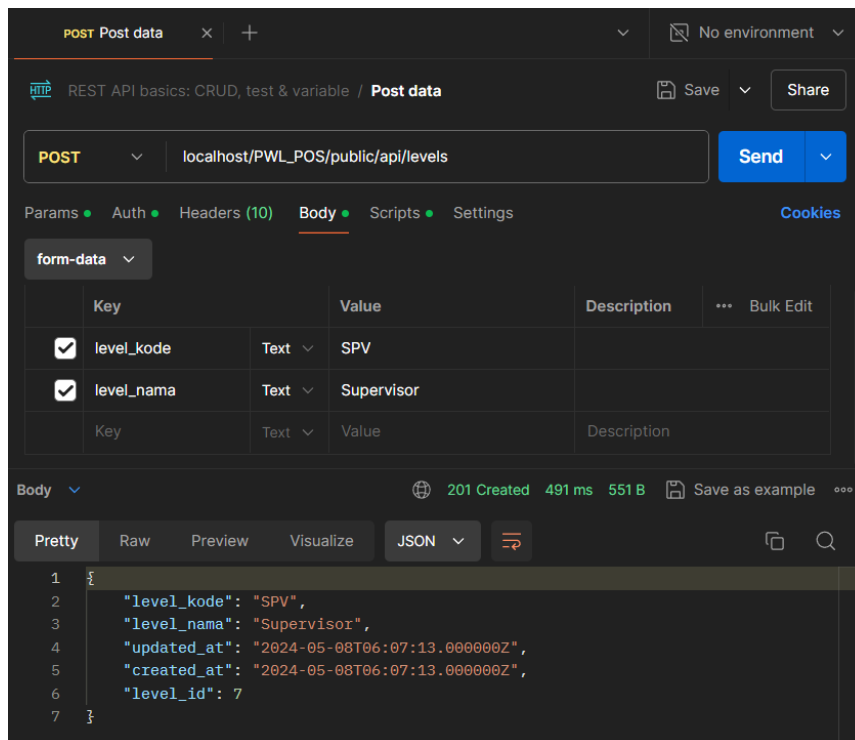
4. Praktikum 4 - Implementasi CRUD dalam RESTful API

No.	Langkah Praktikum
1.	<p>Pertama, buat controller untuk mengolah API pada data level. php artisan make:controller Api/LevelController</p> <pre>PS C:\laragon\www\PWL_POS> php artisan make:controller Api/LevelController</pre> <p>INFO Controller [C:\laragon\www\PWL_POS\app\Http\Controllers\Api\LevelController.php] created successfully.</p> <pre>PS C:\laragon\www\PWL_POS></pre>
2.	<p>Setelah berhasil, buka file tersebut dan tuliskan kode seperti berikut yang berisi fungsi CRUDnya.</p> <pre>app > Http > Controllers > Api > LevelController.php > PHP Intelephense > LevelController > update</pre> <pre> 4 5 use App\Http\Controllers\Controller; 6 use Illuminate\Http\Request; 7 use App\Models\LevelModel; 8 use Laravel\Ui\Presets\React; 9 10 class LevelController extends Controller 11 { 12 public function index() 13 { 14 return LevelModel::all(); 15 } 16 17 public function store(Request \$request) 18 { 19 \$level = LevelModel::create(\$request->all()); 20 return response()->json(\$level, 201); 21 } 22 23 public function show(LevelModel \$level) 24 { 25 return LevelModel::find(\$level); 26 } 27 28 public function update(Request \$request, LevelModel \$level) 29 { 30 \$level->update(\$request->all()); 31 return LevelModel::find(\$level); 32 } 33 34 public function destroy(LevelModel \$user) 35 { 36 \$user->delete(); 37 return response()->json([38 'success' => true, 39 'message' => 'Data Terhapus' 40]); 41 } 42 }</pre>
3.	<p>Kemudian kita lengkapi routes pada api.php.</p> <pre> 25 Route::get('levels', [LevelController::class, 'index']); 26 Route::post('levels', [LevelController::class, 'store']); 27 Route::get('levels', [LevelController::class, 'show']); 28 Route::put('levels', [LevelController::class, 'update']); 29 Route::delete('levels', [LevelController::class, 'destroy']); </pre>
4.	<p>Jika sudah. Lakukan uji coba API mulai dari fungsi untuk menampilkan data. Gunakan URL: localhost/PWL_POS-main/public/api/levels dan method GET. Jelaskan dan berikan screenshoot hasil percobaan Anda.</p>



- Pada percobaan ini dilakukan untuk menampilkan seluruh data level yang ada dan tersimpan dalam database.

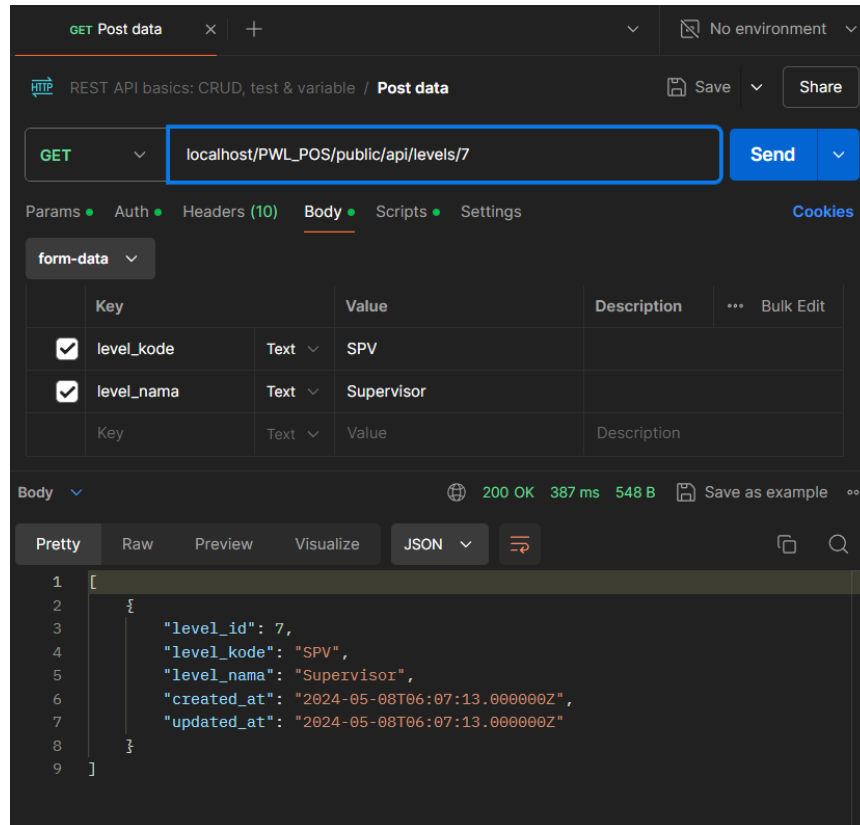
5. Kemudian, lakukan percobaan penambahan data dengan URL : localhost/PWL_POSmain/public/api/levels dan method POST seperti di bawah ini.



- Pada percobaan ini, akan ditambahkan data pada tabel m_level yang ada pada database, disini ditambahkan berupa:
 - level_kode bernilai “SPV”
 - level_nama bernilai “Supervisor”

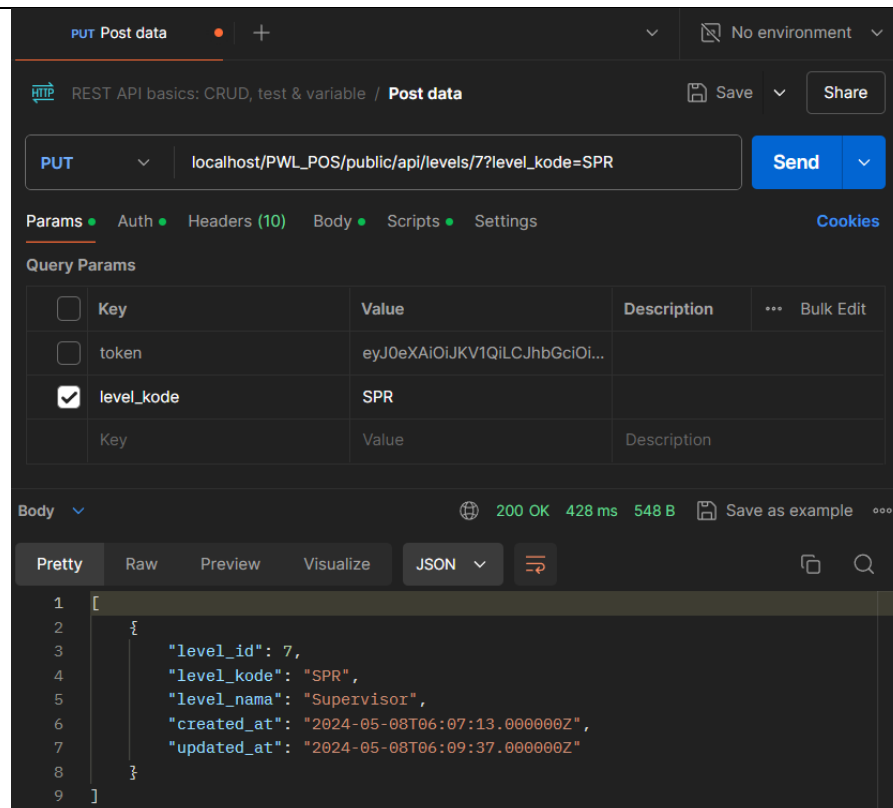
untuk nilai dari level_id akan auto increment serta nilai dari kolom updated_at dan created_at akan terisi secara otomatis berdasarkan waktu dibuatnya atau diupdatenya data tersebut.

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



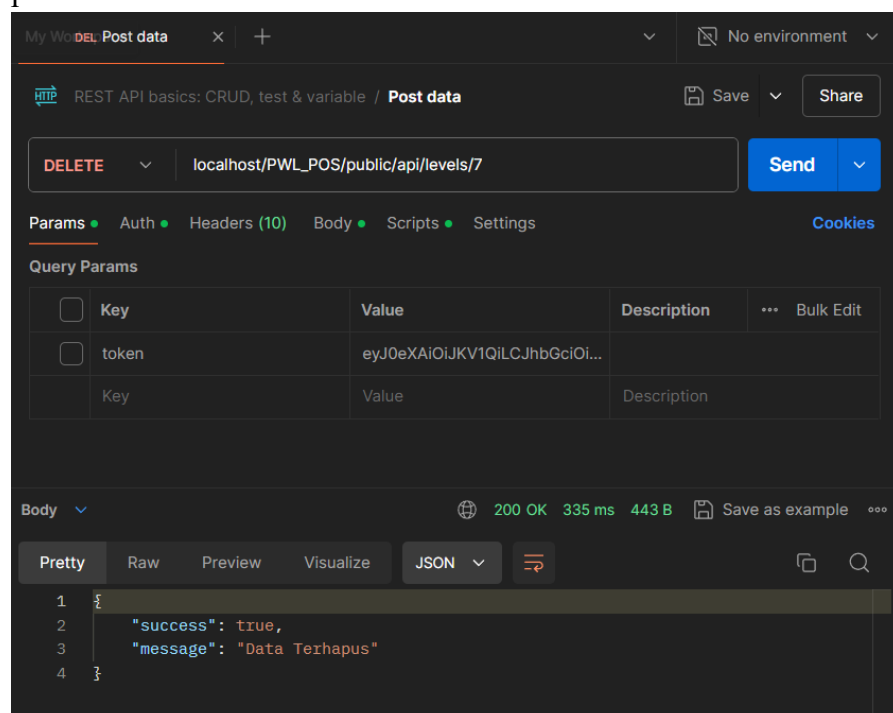
- Percobaan ini dilakukan untuk menampilkan data level berdasarkan level_id nya. Pada percobaan ini saya mencoba untuk menampilkan data yang memiliki level_id bernilai 7.

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.



- Pada percobaan ini digunakan untuk mengubah nilai dari suatu kolom tertentu. Pada percobaan ini saya akan mengedit suatu data yang memiliki level_id = 7, dimana nilai level_kode pada level_id = 7 yang awalnya bernilai “SPV” saya ubah menjadi “SPR”.

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



	<ul style="list-style-type: none"> Pada percobaan ini digunakan untuk menghapus suatu data pada tabel level berdasarkan level_id yang ada. Pada percobaan ini saya mencoba akan menghapus data yang memiliki level_id bernilai 5. Dan jika berhasil terhapus maka akan muncul pesan seperti di atas.
9.	Lakukan commit perubahan file pada Github.

5. Tugas

a. User

No.	Langkah Pengerjaan
1.	<p>Membuat UserController pada folder Api</p> <pre>PS C:\laragon\www\PWL_POS> php artisan make:controller Api/UserController</pre> <p>INFO Controller [C:\laragon\www\PWL_POS\app\Http\Controllers\Api\UserController.php] created successfully.</p>
2.	<p>Mengisi UserController</p> <pre>app > Http > Controllers > Api > UserController.php > PHP Intelephense > UserController > destroy</pre> <pre> 1 use App\Models\UserModel; 2 3 class UserController extends Controller 4 { 5 public function index() 6 { 7 return UserModel::all(); 8 } 9 10 public function store(Request \$request) 11 { 12 \$user = UserModel::create(\$request->all()); 13 return response()->json(\$user, 201); 14 } 15 16 public function show(UserModel \$user) 17 { 18 return UserModel::find(\$user); 19 } 20 21 public function update(Request \$request, UserModel \$user) 22 { 23 \$user->update(\$request->all()); 24 return UserModel::find(\$user->user_id); 25 } 26 27 public function destroy([UserModel \$user]) 28 { 29 \$user->delete(); 30 return response()->json([31 'success' => true, 32 'message' => 'Data Berhasil Dihapus' 33]); 34 } 35 }</pre>
3.	<p>Menentukan Route</p> <pre> 31 32 Route::get('users', [UserController::class, 'index']); 33 Route::get('users/{user}', [UserController::class, 'show']); 34 Route::post('users', [UserController::class, 'store']); 35 Route::put('users/{user}', [UserController::class, 'update']); 36 Route::delete('users/{user}', [UserController::class, 'destroy']); </pre>
4.	<ul style="list-style-type: none"> Menambahkan data ke tabel m_user

POST Post data x + No environment

REST API basics: CRUD, test & variable / Post data Save Share

POST localhost/PWL_POS/public/api/users Send

Params Auth Headers (9) Body Scripts Settings Cookies

form-data

	Key		Value	Description	...	Bulk Edit
<input checked="" type="checkbox"/>	username	Text	indra			
<input checked="" type="checkbox"/>	name	Text	Bagus Indrawan			
<input checked="" type="checkbox"/>	password	Text	12345			
<input checked="" type="checkbox"/>	level_id	Text	3			
	Key	Text	Value	Description		

Body 201 Created 432 ms 638 B Save as example

Pretty Raw Preview Visualize JSON

```
1 {
2   "username": "indra",
3   "name": "Bagus Indrawan",
4   "password": "$2y$12$vFzZ67IeU1Vcyu4XryFTRupQ48IFHsTUr25f63h3y7R4.nJT.jYJ.",
5   "level_id": "3",
6   "updated_at": "2024-05-08T07:06:48.000000Z",
7   "created_at": "2024-05-08T07:06:48.000000Z",
8   "user_id": 14
9 }
```

- Menampilkan seluruh data pada tabel m_user

GET Post data + No environment

REST API basics: CRUD, test & variable / Post data Save Share

GET localhost/PWL_POS/public/api/users Send

Params Auth Headers (9) Body Scripts Settings Cookies

Body 200 OK 224 ms 1.85 KB Save as example

Pretty Raw Preview Visualize JSON

```
1 [
2   {
3     "user_id": 1,
4     "level_id": 1,
5     "username": "adm",
6     "name": "Administrator",
7     "password": "$2y$12$xE.yn1NZC4DX/AxwUdm940evJGmExXiTeJj3EI33Tqln/3.54L2hK",
8     "created_at": null,
9     "updated_at": "2024-05-08T06:24:33.000000Z"
10  },
11  {
12    "user_id": 2,
13    "level_id": 2,
14    "username": "manager",
15    "name": "Manager",
16    "password": "$2y$12$PzkkEyoBqKwfpVEDhTQDsuRHHaIEFsFwAtu6g56i6vxCGTkdx941e",
17    "created_at": null,
18    "updated_at": null
19  },
20  {
21    "user_id": 3,
22    "level_id": 3,
23    "username": "staff",
```


- Menampilkan data user berdasarkan user_id yang dimiliki

GET Post data

REST API basics: CRUD, test & variable / Post data

GET localhost/PWL_POS/public/api/users/1

Send

Params Auth Headers (10) Body Scripts Settings Cookies

Query Params

<input type="checkbox"/>	Key	Value	Description	...	Bulk Edit
<input type="checkbox"/>	token	eyJ0eXAiOiJKV1QiLCJhbGciOi...			
	Key	Value	Description		

Body

200 OK 352 ms 608 B Save as example

Pretty Raw Preview Visualize JSON

```
1 [
2   {
3     "user_id": 1,
4     "level_id": 1,
5     "username": "admin",
6     "name": "Administrator",
7     "password": "$2y$12$xE.yn1NZC4DX/AxwUdm940evJGmExXiTeJj3EI33Tqln/3.
8       54L2hK",
9     "created_at": null,
10    "updated_at": "2024-04-17T13:12:06.000000Z"
11  }
```

- Memperbarui nilai suatu kolom berdasarkan user_id

PUT Post data

REST API basics: CRUD, test & variable / Post data

PUT localhost/PWL_POS/public/api/users/1?username=adm

Send

Params Auth Headers (10) Body Scripts Settings Cookies

<input type="checkbox"/>	Key	Value	Description	...	Bulk Edit
<input type="checkbox"/>	token	eyJ0eXAiOiJKV1QiLCJhbGciOi...			
<input checked="" type="checkbox"/>	username	adm			
	Key	Value	Description		

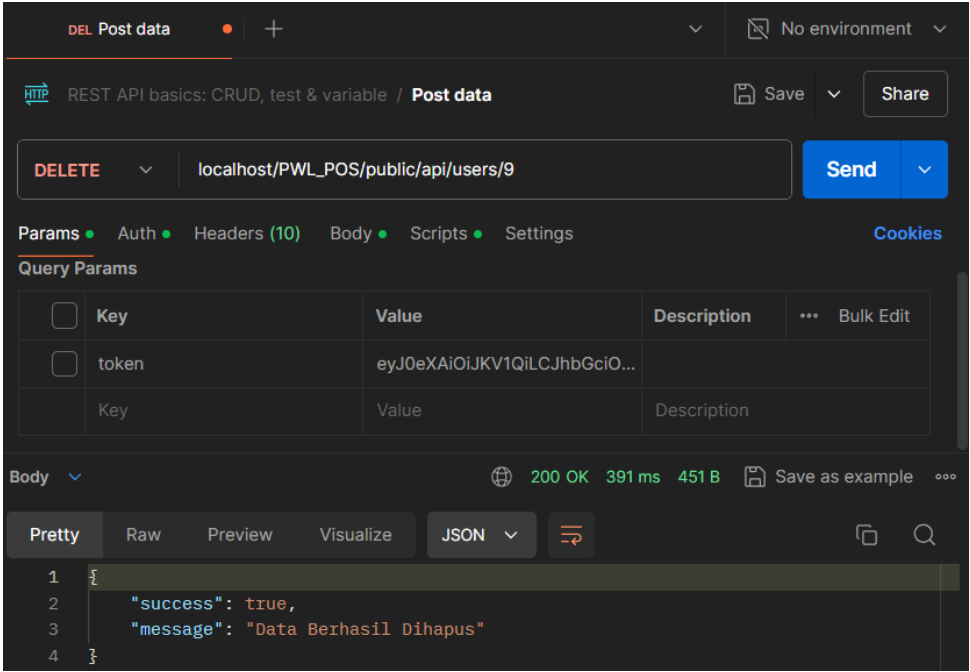
Body

200 OK 398 ms 606 B Save as example

Pretty Raw Preview Visualize JSON

```
1 [
2   {
3     "user_id": 1,
4     "level_id": 1,
5     "username": "adm",
6     "name": "Administrator",
7     "password": "$2y$12$xE.yn1NZC4DX/AxwUdm940evJGmExXiTeJj3EI33Tqln/3.
8       54L2hK",
9     "created_at": null,
10    "updated_at": "2024-05-08T06:24:33.000000Z"
11  }
```

- Menghapus data user berdasarkan user_id nya



The screenshot shows the Postman interface for a DELETE request. The URL is `localhost/PWL_POS/public/api/users/9`. The response status is 200 OK, and the body is a JSON object: `{ "success": true, "message": "Data Berhasil Dihapus" }`.

b. Kategori

No.	Langkah Pengerjaan
1.	<p>Membuat KategoriController pada folder Api</p> <pre>PS C:\laragon\www\PWL_POS> php artisan make:controller Api/KategoriController</pre> <p>INFO Controller [C:\laragon\www\PWL_POS\app\Http\Controllers\Api\KategoriController.php] created successfully.</p> <pre>PS C:\laragon\www\PWL_POS></pre>
2.	<p>Mengisi KategoriController</p> <pre>app > Http > Controllers > Api > KategoriController.php > PHP Intelephense > KategoriController > update</pre> <pre>use App\Models\KategoriModel; class KategoriController extends Controller { public function index() { return KategoriModel::all(); } public function store(Request \$request) { \$kategori = KategoriModel::create(\$request->all()); return response()->json(\$kategori, 201); } public function show(KategoriModel \$kategori) { return KategoriModel::find(\$kategori); } public function update(Request \$request, KategoriModel \$kategori) { \$kategori->update(\$request->all()); return KategoriModel::find(\$kategori->kategori_id); } public function destroy(KategoriModel \$kategori) { \$kategori->delete(); return response()->json(['success' => true, 'message' => 'Data Berhasil Dihapus']); } }</pre>

3. Menentukan Route

```
38  
39 Route::get('kategoris', [KategoriController::class, 'index']);  
40 Route::get('kategoris/{kategori}', [KategoriController::class, 'show']);  
41 Route::post('kategoris', [KategoriController::class, 'store']);  
42 Route::put('kategoris/{kategori}', [KategoriController::class, 'update']);  
43 Route::delete('kategoris/{kategori}', [KategoriController::class, 'destroy']);
```

4.

- Menambahkan data pada tabel kategori

POST Post data

REST API basics: CRUD, test & variable / Post data

POST localhost/PWL_POS/public/api/kategoris

form-data

Key	Value	Description
kategori_kode	OLR	
kategori_nama	Olahraga	
updated_at	2024-05-08T07:02:05.000000Z	
created_at	2024-05-08T07:02:05.000000Z	
kategori_id	11	

Body

201 Created 220 ms 559 B

Save as example

Pretty Raw Preview Visualize JSON

```
1 {  
2   "kategori_kode": "OLR",  
3   "kategori_nama": "Olahraga",  
4   "updated_at": "2024-05-08T07:02:05.000000Z",  
5   "created_at": "2024-05-08T07:02:05.000000Z",  
6   "kategori_id": 11  
7 }
```

- Menampilkan seluruh data pada tabel kategori

GET Post data

REST API basics: CRUD, test & variable / Post data

GET localhost/PWL_POS/public/api/kategoris

Body

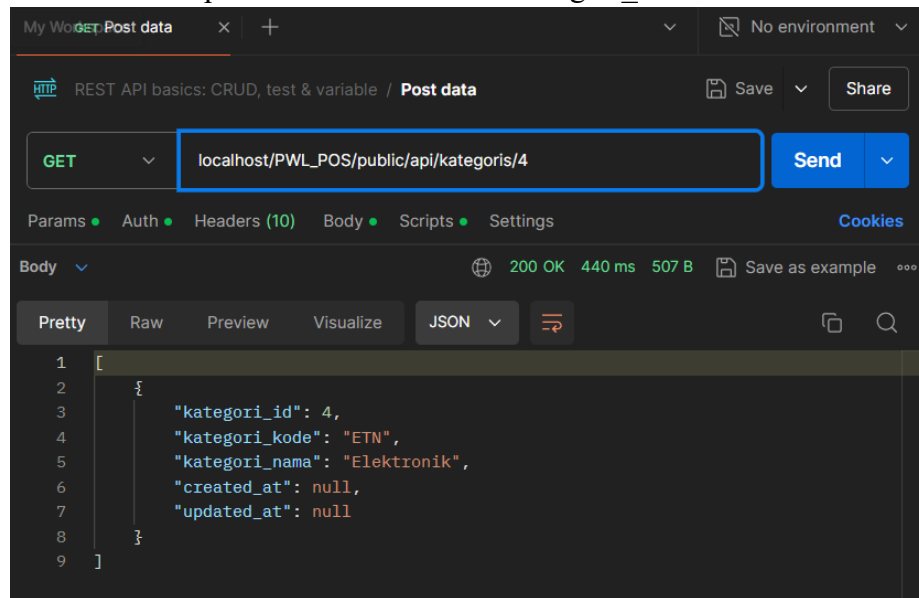
200 OK 387 ms 1.36 KB

Save as example

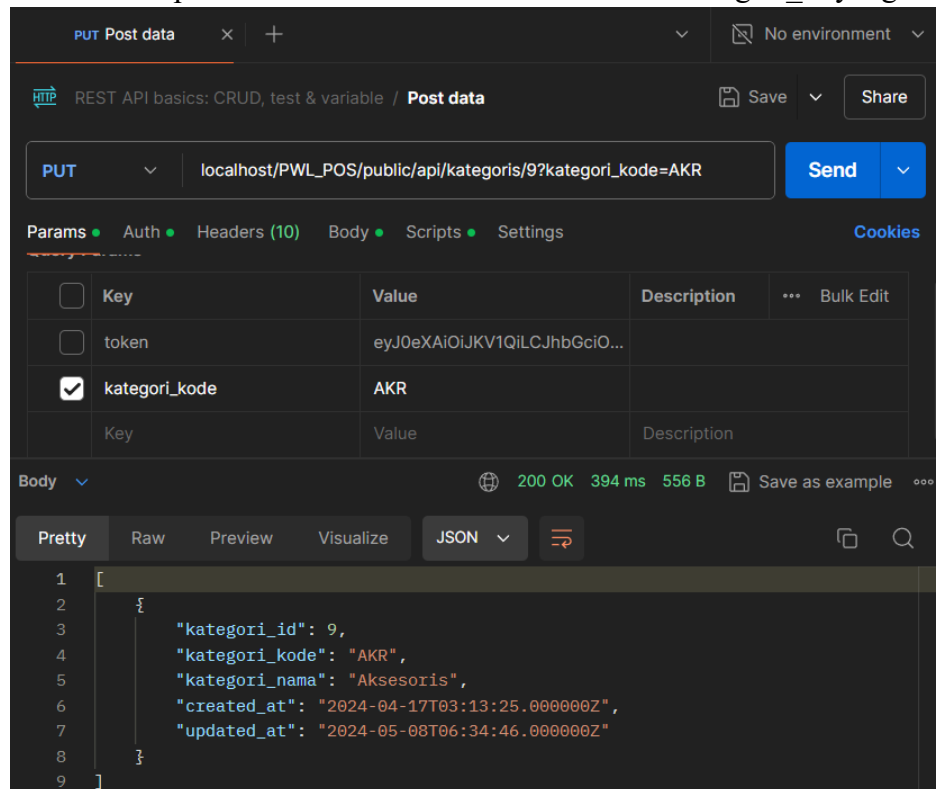
Pretty Raw Preview Visualize JSON

```
1 [  
2   {  
3     "kategori_id": 1,  
4     "kategori_kode": "MKN",  
5     "kategori_nama": "Makanan",  
6     "created_at": null,  
7     "updated_at": null  
8   },  
9   {  
10    "kategori_id": 2,  
11    "kategori_kode": "MNM",  
12    "kategori_nama": "Minuman",  
13    "created_at": null,  
14    "updated_at": null  
15  },  
16  {  
17    "kategori_id": 3,  
18    "kategori_kode": "PKN",  
19    "kategori_nama": "Pakaian",  
20    "created_at": null,  
21    "updated_at": null  
22  },  
23  {  
24    "kategori_id": 4,  
25    "kategori_kode": "ETN",  
26    "kategori_nama": "Etnisitas"  
27  }  
28 ]
```

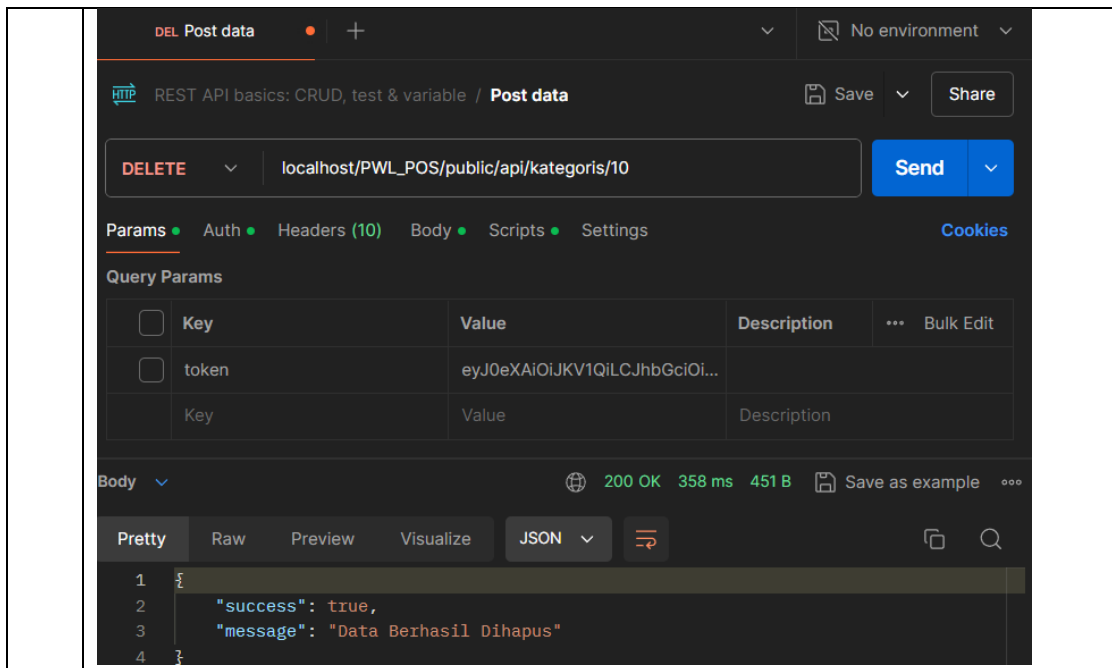
- Menampilkan data berdasarkan kategori_id



- Memperbarui nilai suatu kolom berdasarkan kategori_id yang dimiliki



- Menghapus suatu data kategori berdasarkan kategori_id nya



c. Barang

No.	Langkah Pengerjaan
1.	<p>Membuat BarangController</p> <pre>PS C:\laragon\www\PWL_POS> php artisan make:controller Api/BarangController</pre> <p>INFO Controller [C:\laragon\www\PWL_POS\app\Http\Controllers\Api\BarangController.php] created successfully.</p> <pre>PS C:\laragon\www\PWL_POS></pre>
2.	<p>Mengisi BarangController</p> <pre>app > Http > Controllers > Api > BarangController.php > PHP Intelephense > BarangController > update</pre> <pre> 1 use App\Models\BarangModel; 2 3 4 5 6 7 8 9 class BarangController extends Controller 10 { 11 public function index() 12 { 13 return BarangModel::all(); 14 } 15 16 public function store(Request \$request) 17 { 18 \$barang = BarangModel::create(\$request->all()); 19 return response()->json(\$barang, 201); 20 } 21 22 public function show(BarangModel \$barang) 23 { 24 return BarangModel::find(\$barang->barang_id); 25 } 26 27 public function update(Request \$request, BarangModel \$barang) 28 { 29 \$barang->update(\$request->all()); 30 return BarangModel::find(\$barang->barang_id); 31 } 32 33 public function destroy(BarangModel \$barang) 34 { 35 \$barang->delete(); 36 return response()->json([37 'success' => true, 38 'message' => 'Data Berhasil Dihapus' 39]); 40 } 41 }</pre>

3. Menentukan Route

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

4. Menambahkan data pada tabel m_barang

POST Post data

localhost/PWL_POS/public/api/barangs

Send

Params Auth Headers (9) Body Scripts Settings Cookies

form-data

	Key	Value	Description	...	Bulk Edit
<input type="checkbox"/>	barang_id	Text			
<input checked="" type="checkbox"/>	barang_kode	Text	PHP		
<input checked="" type="checkbox"/>	kategori_id	Text	6		
<input checked="" type="checkbox"/>	barang_nama	Text	Penghapus		
<input checked="" type="checkbox"/>	harga_beli	Text	1000		
<input checked="" type="checkbox"/>	harga_jual	Text	2000		

Body

201 Created 189 ms 612 B Save as example

Pretty Raw Preview Visualize JSON

```
1 {
2   "barang_kode": "PHP",
3   "kategori_id": "6",
4   "barang_nama": "Penghapus",
5   "harga_beli": "1000",
6   "harga_jual": "2000",
7   "updated_at": "2024-05-08T06:58:49.000000Z",
8   "created_at": "2024-05-08T06:58:49.000000Z",
9   "barang_id": 16
10 }
```

Menampilkan seluruh data yang ada pada tabel m_barang

GET Post data

localhost/PWL_POS/public/api/barangs

Send

Params Auth Headers (10) Body Scripts Settings Cookies

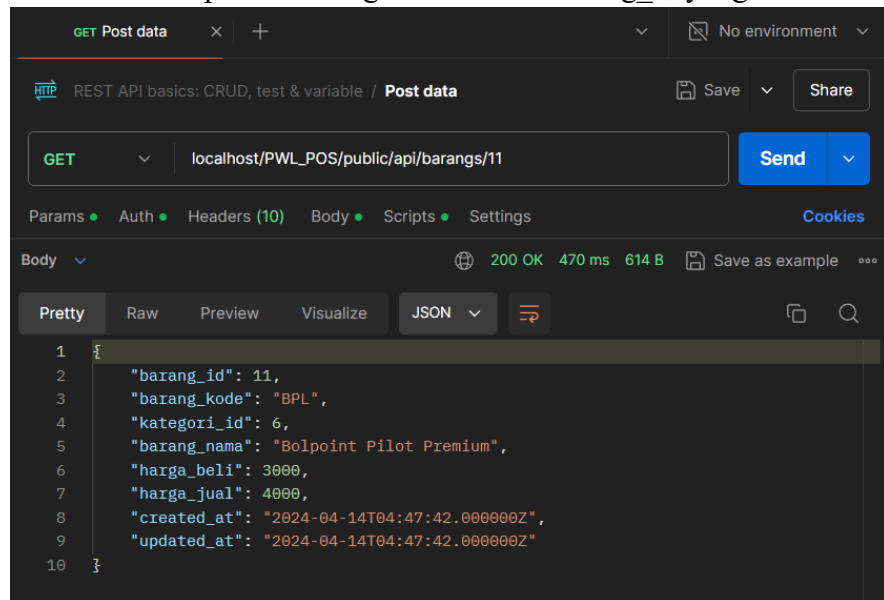
Body

200 OK 529 ms 2.12 KB Save as example

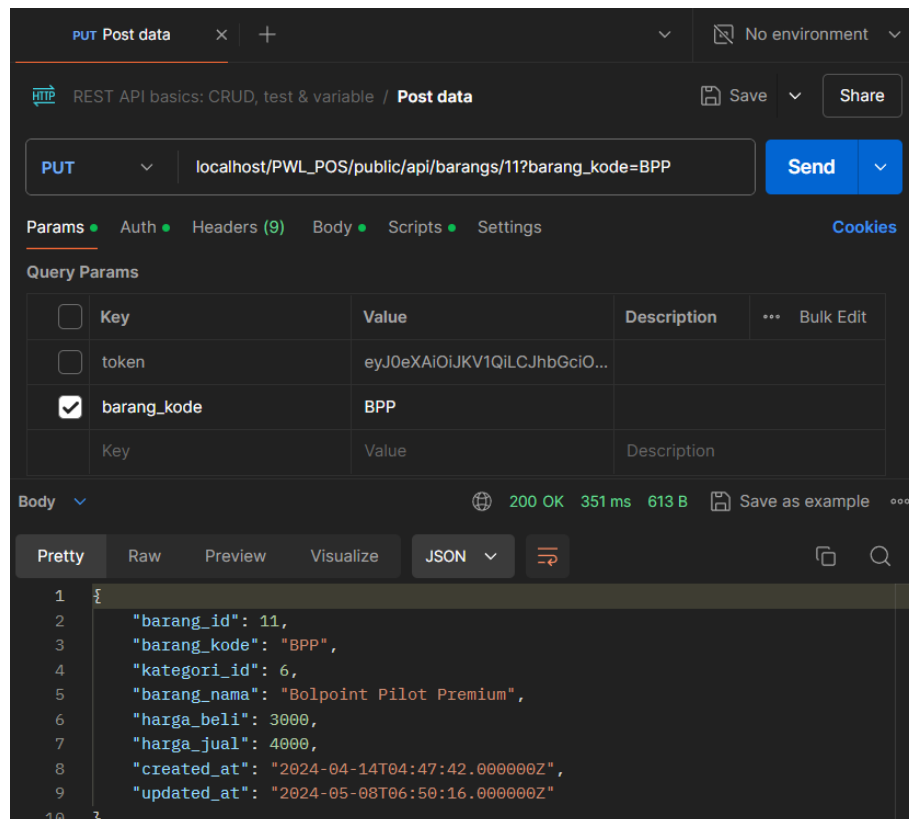
Pretty Raw Preview Visualize JSON

```
1 [
2   {
3     "barang_id": 1,
4     "barang_kode": "SBH",
5     "kategori_id": 1,
6     "barang_nama": "Salad Buah",
7     "harga_beli": 15000,
8     "harga_jual": 20000,
9     "created_at": null,
10    "updated_at": null
11  },
12  {
13    "barang_id": 2,
14    "barang_kode": "CHC",
15    "kategori_id": 1,
16    "barang_nama": "Cheese Cake",
17    "harga_beli": 20000,
18    "harga_jual": 30000,
19    "created_at": null,
20    "updated_at": null
21  },
22  {
23    "barang_id": 3,
24    "barang_kode": "JAP",
```

- Menampilkan barang berdasarkan barang_id yang dimiliki



- Memperbarui nilai dari suatu kolom berdasarkan barang_id yang dimiliki



- Menghapus data barang berdasarkan barang_id

