

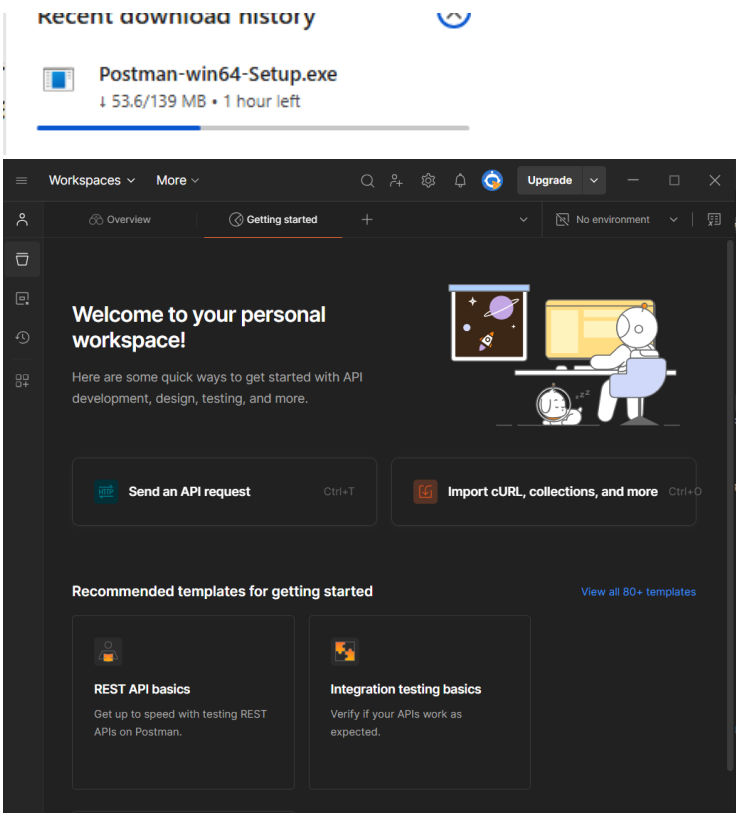


Mata Kuliah : Pemrograman Web Lanjut (PWL)
Program Studi : D4 – Sistem Informasi Bisnis
Semester : 5

Kelas : SIB
NIM : 2241760026
Nama : Siti Faiqoh
Jobsheet Ke- : 10

Laporan Jobsheet

Praktikum 1 – Membuat RESTful API Register

Langkah	Jawaban/Deskripsi
1	<p>Aplikasi ini akan digunakan untuk mengerjakan semua tahap praktikum pada Jobsheet ini.</p>  <p>The screenshot shows the Postman application interface. At the top, there's a 'recent download history' section showing 'Postman-win64-Setup.exe' with a progress bar indicating 53.6/139 MB downloaded and 1 hour left. Below this is the 'Workspaces' section with tabs for 'Overview' and 'Getting started'. The 'Getting started' tab is active, displaying a 'Welcome to your personal workspace!' message and several quick actions: 'Send an API request' (Ctrl+T), 'Import cURL, collections, and more' (Ctrl+Q), and 'Recommended templates for getting started' including 'REST API basics' and 'Integration testing basics'.</p>



2	<p>Lakukan instalasi JWT dengan mengetikkan perintah berikut:</p> <pre>PS C:\laragon\www\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)</pre>
3	<p>Setelah berhasil menginstall JWT, lanjutkan dengan publish konfigurasi file dengan perintah berikut:</p> <pre>PS C:\laragon\www\PWL_POS> php artisan vendor:publish -- >> provider="Tymon\JWTAuth\Providers\LaravelServiceProvider" Which provider or tag's files would you like to publish? All providers and tags 0 Provider: Barryvdh\DomPDF\ServiceProvider 1 Provider: Illuminate\Foundation\Providers\FoundationServiceProvider 2 Provider: Illuminate\Mail\MailServiceProvider 3 Provider: Illuminate\Notifications\NotificationServiceProvider 4 Provider: Illuminate\Pagination\PaginationServiceProvider 5 Provider: Laravel\Sail\SailServiceProvider 6 Provider: Laravel\Sanctum\SanctumServiceProvider 7 Provider: Laravel\Tinker\TinkerServiceProvider 8 Provider: Spatie\LaravelIgnition\IgnitionServiceProvider 9 Provider: Tymon\JWTAuth\Providers\LaravelServiceProvider 10 Tag: sanctum-config 30 Tag: sanctum-migrations 31 > 10 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</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> <pre>config ├── app.php ├── auth.php ├── broadcasting.php ├── cache.php ├── cors.php ├── database.php ├── filesystems.php ├── hashing.php ├── jwt.php └── logging.php</pre>



5	<p>Setelah itu jalankan perintah berikut untuk membuat secret key JWT</p> <pre>PS C:\laragon\www\PWL_POS> php artisan jwt:secret jwt-auth secret [TBmhw8Cg0yDS7cUxmKQBswYnpYKi6dMBW0Ri0MHA4Gvt6sBf1nnQA0F2dkQL4dD] set successfully. PS C:\laragon\www\PWL_POS></pre> <pre>.env 61 JWT_SECRET=TBmhw8Cg0yDS7cUxmKQBswYnpYKi6dMBW0Ri0MHA4Gvt6sBf1nnQA0F2dkQL4dD 62</pre>
6	<p>Selanjutnya lakukan konfigurasi guard API. Buka config/auth.php. Ubah bagian 'guards' menjadi seperti berikut.</p> <pre>config > auth.php 3 return [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>app > Models > UserModel.php > UserModel > level 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 Illuminate\Foundation\Auth\User as Authenticatable; 9 use Tymon\JWTAuth\Contracts\JWTSubject; 10 11 class UserModel extends Authenticatable implements JWTSubject 12 { 13 14 public function getJWTIdentifier(){ 15 return \$this->getKey(); 16 } 17 public function getJWTCustomClaims(){ 18 return []; 19 } 20</pre>



8	<p>Berikutnya kita akan membuat controller untuk register dengan menjalankan perintah berikut.</p> <p>php artisan make:controller Api/RegisterController</p> <pre>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.</pre> 
9	<p>Buka file tersebut, dan ubah kode menjadi seperti berikut</p> <pre>app > Http > Controllers > Api > RegisterController.php > ... 1 <?php 2 namespace App\Http\Controllers\Api; 3 use App\Http\Controllers\Controller; 4 use App\Models\UserModel; 5 use Illuminate\Http\Request; 6 use Illuminate\Support\Facades\Validator; 7 8 class RegisterController extends Controller 9 { 10 public function __invoke(Request \$request) 11 { 12 //validasi 13 \$validator = Validator::make(\$request->all(), [14 'username' => 'required', 15 'nama' => 'required', 16 'password' => 'required min:5 confirmed', 17 'level_id' => 'required' 18]); 19 20 // if validasi gagal 21 if (\$validator->fails()) { 22 return response()->json(\$validator->errors(), 422); 23 } 24 25 // create user 26 \$user = UserModel::create([27 'username' => \$request->username, 28 'nama' => \$request->nama, 29 'password' => bcrypt(\$request->password), 30 'level_id' => \$request->level_id, 31]); 32 33 // return response JSON user is created 34 if (\$user) { 35 return response()->json([36 'success' => true, 37 'user' => \$user, 38],201); 39 } 40 41 // return JSON process insert failed 42 return response()->json([43 'success' => false, 44],409); 45 } 46 }</pre>



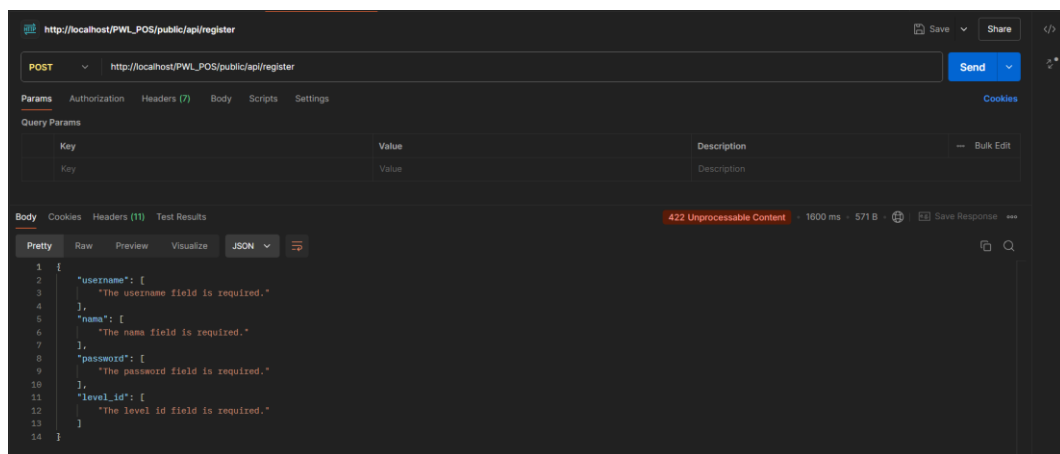
10

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

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

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

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

- URL:** `http://localhost/PWL_POS/public/api/register`
- Method:** `POST`
- Body Type:** `form-data`
- Body Data:**

Key	Value
username	penggunasatu
nama	Pengguna 1
password	12345
password_confirmation	12345
level_id	2
- Response:**

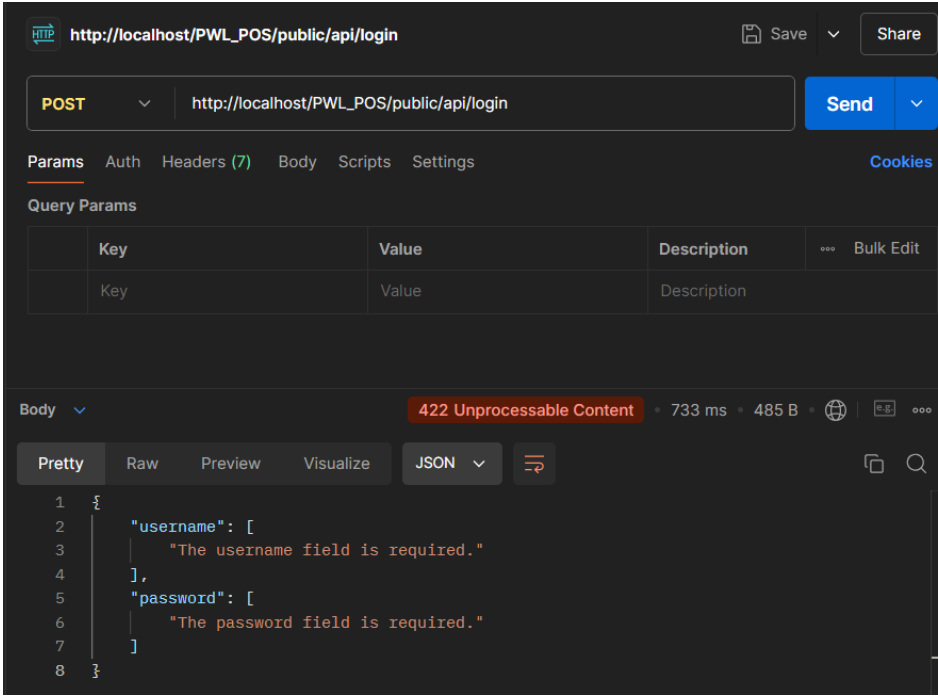
```
1 {
2   "success": true,
3   "user": {
4     "username": "penggunasatu",
5     "nama": "Pengguna 1",
6     "level_id": "2",
7     "updated_at": "2024-11-06T02:57:50.000000Z",
8     "created_at": "2024-11-06T02:57:50.000000Z",
9     "user_id": 36
10  }
11 }
```

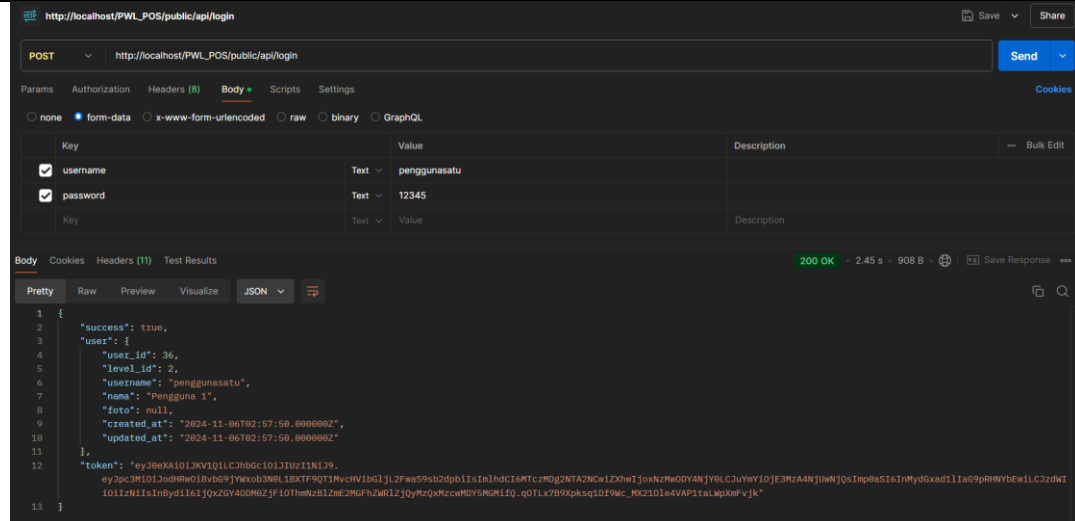


Praktikum Ke-2 Membuat RESTful API Login

Langkah	Jawaban/Deskripsi
1	<p>Kita buat file controller dengan nama LoginController.</p> <pre>PS C:\laragon\www\PWL_POS> php artisan make:controller Api/LoginController</pre> <p>INFO Controller [C:\laragon\www\PWL_POS\app\Http\Controllers\Api>LoginController.php] created successfully.</p>
2	<p>Buka file tersebut, dan ubah kode menjadi seperti berikut.</p> <pre>app > Http > Controllers > Api > LoginController.php > LoginController</pre> <pre>1 <?php 2 namespace App\Http\Controllers\Api; 3 use App\Http\Controllers\Controller; 4 use App\Models\UserModel; 5 use Illuminate\Http\Request; 6 use Illuminate\Support\Facades\Validator; 7 class LoginController extends Controller 8 { 9 public function __invoke(Request \$request) 10 { 11 \$validator = Validator::make(\$request->all(), [12 'username' => 'required', 13 'password' => 'required' 14]); 15 if (\$validator->fails()) { 16 return response()->json(\$validator->errors(), 422); 17 } 18 \$credentials = \$request->only('username','password'); 19 if (!\$token = auth()->guard('api')->attempt(\$credentials)) { 20 return response()->json([21 'success' => false, 22 'message' => 'Username atau Password Anda Salah' 23],401); 24 } 25 return response()->json([26 'success' => true, 27 'user' => auth()->guard('api')->user(), 28 'token' => \$token 29],200); 30 } 31 }</pre>
3	<p>Berikutnya tambahkan route baru pada file api.php yaitu /login dan /user.</p>

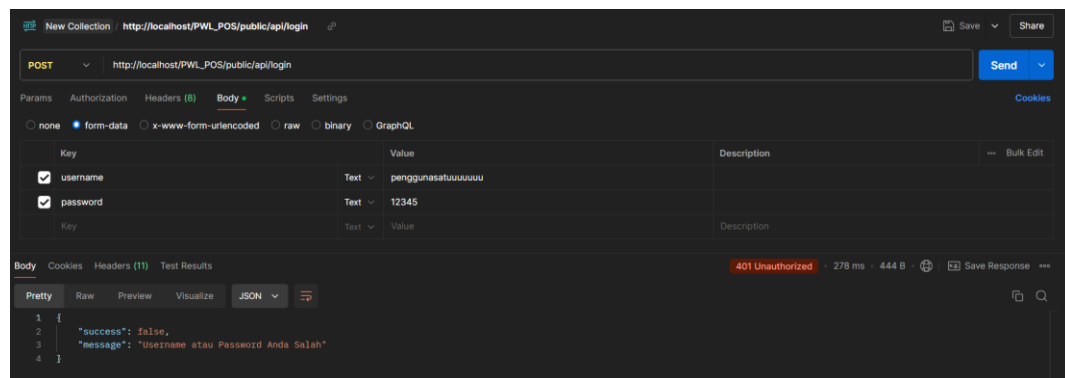


	<pre>routes > api.php > ... 1 <?php 2 3 use App\Http\Controllers\Api\RegisterController; 4 use App\Http\Controllers\Api>LoginController; 5 use Illuminate\Http\Request; 6 use Illuminate\Support\Facades\Route; 7 8 /* 9 ----- 10 API Routes 11 ----- 12 13 Here is where you can register API routes for your 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</pre>
4	<p>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</p> 
5	<p>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.</p>



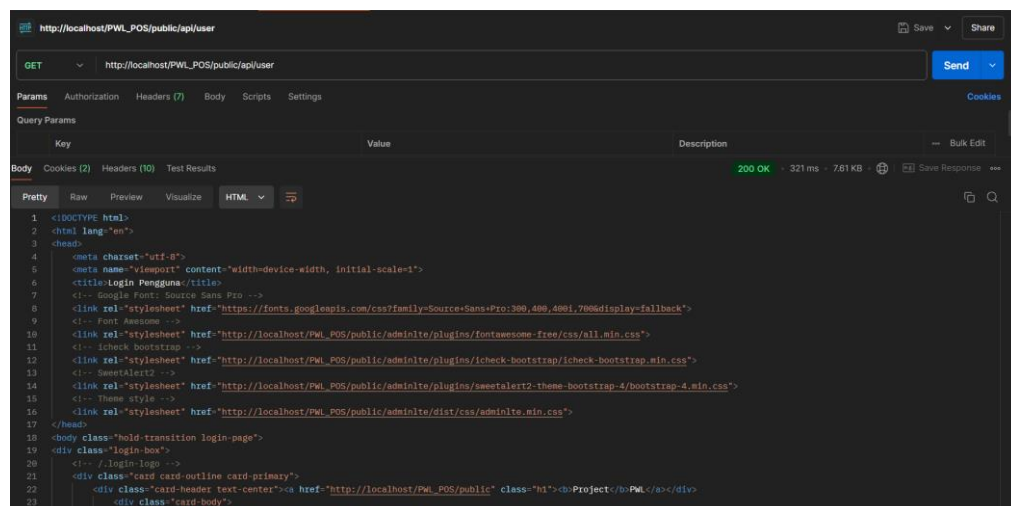
6

Lakukan percobaan yang untuk data yang salah dan berikan screenshot 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



Karena belum membuat usercontroller maka tampilan seperti diatas



Praktikum Ke-3 Membuat RESTful API Logou

Langkah	Jawaban/Deskripsi
1	<p>Tambahkan kode berikut pada file .env</p> <p>JWT_SHOW_BLACKLIST_EXCEPTION=true</p> <pre>.env 61 JWT_SECRET=TBmhw8Cg0yDS7cUxmKQBskYnpYKi6dMBW0Ri0MHA4Gvt6sBf1nnQA0F2dkQL4dD 62 JWT_SHOW_BLACKLIST_EXCEPTION=true</pre>
2	<p>Buat Controller baru dengan nama LogoutController.</p> <pre>PS C:\laragon\www\PWL_POS> php artisan make:controller Api/LogoutController</pre> <p>INFO Controller [C:\laragon\www\PWL_POS\app\Http\Controllers\Api\LogoutController.php] created successfully.</p> <pre>app > Http > Controllers > Api > LogoutController.php > LogoutController 1 <?php 2 namespace App\Http\Controllers\Api; 3 use App\Http\Controllers\Controller; 4 use Illuminate\Http\Request; 5 use Illuminate\Support\Facades\Validator; 6 use Tymon\JWTAuth\Facades\JWTAuth; 7 class LogoutController extends Controller 8 { 9 public function __invoke(Request \$request) 10 { 11 \$removeToken = JWTAuth::invalidate(JWTAuth::getToken()); 12 if (\$removeToken) { 13 return response()->json([14 'success' => true, 15 'message' => 'Logout Berhasil', 16]); 17 } 18 } 19 }</pre>
3	<p>Lalu kita tambahkan routes pada api.php</p> <pre>routes > api.php > ... 26 Route::post('/logout', App\Http\Controllers\Api\LogoutController::class)->name('logout'); 27</pre>
4	<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>



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
<https://jti.polinema.ac.id>

HTTP http://localhost/PWL_POS/public/api/logout Save Share

POST http://localhost/PWL_POS/public/api/logout Send

Params Auth Headers (9) Body Scripts Settings Cookies

Auth Type
Bearer Token

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

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
eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI...

Body 200 OK 312 ms 436 B

Pretty Raw Preview Visualize JSON

```
1 {  
2   "success": true,  
3   "message": "Logout Berhasil"  
4 }
```



Praktikum ke-4 – Implementasi CRUD dalam RESTful API

Langkah	Jawaban/Deskripsi
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>
2	<p>Setelah berhasil, buka file tersebut dan tuliskan kode seperti berikut yang berisi fungsi CRUDnya.</p> <pre>app > Http > Controllers > Api > LevelController.php > LevelController > update 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 return LevelModel::all(); 13 } 14 public function store(Request \$request) 15 { 16 \$level = LevelModel::create(\$request->all()); 17 return response()->json(\$level, 201); 18 } 19 20 public function show(LevelModel \$level) 21 { 22 return LevelModel::find(\$level); 23 } 24 25 public function update(Request \$request, LevelModel \$level) 26 { 27 \$level->update(\$request->all()); 28 return LevelModel::find(\$level); 29 } 30 31 public function destroy(LevelModel \$level) 32 { 33 \$level->delete(); 34 return response()->json([35 'success' => true, 36 'message' => 'Data terhapus' 37]); 38 } 39 }</pre>
3	<p>Kemudian kita lengkapi routes pada api.php.</p>



```
routes > api.php > ...
```

```
1  <?php
2
3  use App\Http\Controllers\Api\RegisterController;
4  use App\Http\Controllers\Api>LoginController;
5  use App\Http\Controllers\Api\LogoutController;
6  use App\Http\Controllers\Api\LevelController;
7  use Illuminate\Http\Request;
8  use Illuminate\Support\Facades\Route;
```

```
routes > api.php > ...
```

```
28 Route::group(['prefix'=>'levels'], function(){
29     Route::get('/', [LevelController::class, 'index']);
30     Route::post('/', [LevelController::class, 'store']);
31     Route::get('/{level}', [LevelController::class, 'show']);
32     Route::put('/{level}', [LevelController::class, 'update']);
33     Route::delete('/{level}', [LevelController::class, 'destroy']);
34 });
```

4

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

localhost/PWL_POS/public/api/levels

GET localhost/PWL_POS/public/api/levels

Send

Params Auth Headers (7) Body Scripts Settings Cookies

form-data

Key	Value	Description
-----	-------	-------------

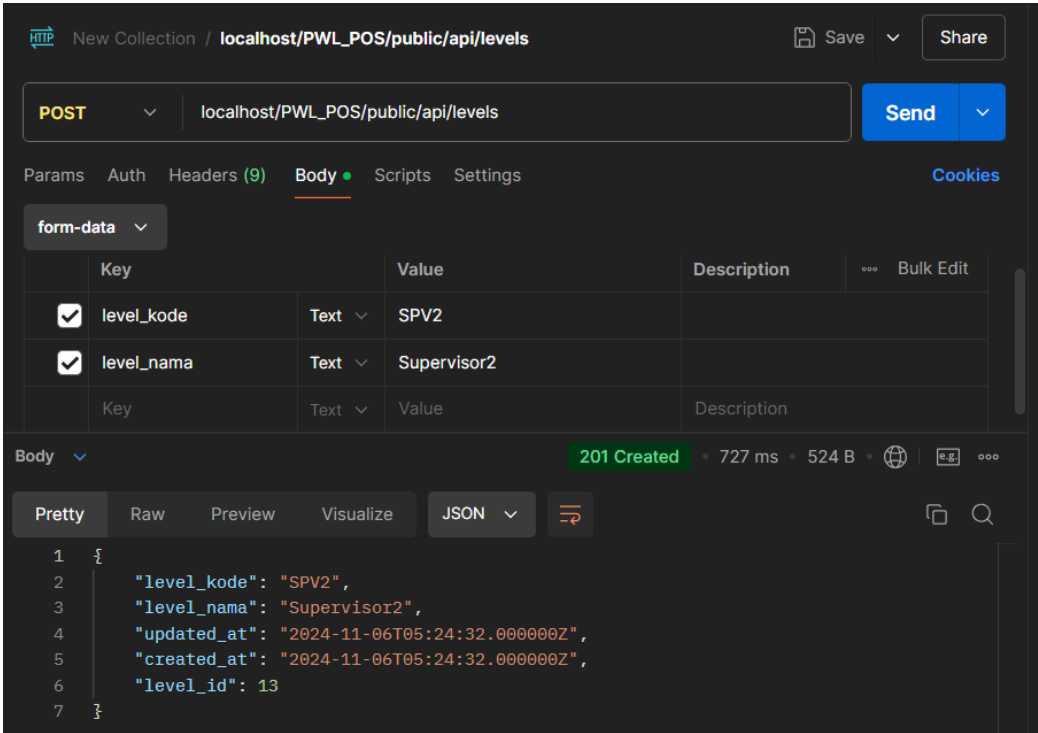
Body

200 OK • 11.80 s • 995 B

Pretty Raw Preview Visualize JSON

```
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,
```



	<p>Ketika pengujian dilakukan menggunakan metode indeks untuk mengambil semua data yang ada di tabel level, hal ini terjadi karena penggunaan sintaks `all()` dalam proses pengambilan data.</p>
5	<p>Kemudian, lakukan percobaan penambahan data dengan URL : localhost/PWL_POSmain/public/api/levels dan method POST seperti di bawah ini.</p>  <p>Pada proses ini, metode yang dijalankan adalah POST untuk menambahkan data. Metode ini akan mengirimkan data baru ke server untuk disimpan dalam tabel. Hasilnya adalah server akan menerima data tersebut dan menambahkannya ke dalam tabel level.</p>
6	<p>Berikutnya lakukan percobaan menampilkan detail data. Jelaskan dan berikan screenshoot hasil percobaan Anda.</p>



New Collection / localhost/PWL_POS/public/api/levels

GET localhost/PWL_POS/public/api/levels/13

Send

Params Auth Headers (9) Body • Scripts Settings Cookies

Query Params

Key	Value	Description	Bulk Edit
Key	Value	Description	

Body

200 OK • 293 ms • 522 B

Pretty Raw Preview Visualize JSON

```
1 {
2   "level_id": 13,
3   "level_kode": "SPV22",
4   "level_nama": "Supervisor2",
5   "created_at": "2024-11-06T05:24:32.000000Z",
6   "updated_at": "2024-11-06T05:36:42.000000Z"
7 }
8
9
```

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

New Collection / localhost/PWL_POS/public/api/levels

PUT localhost/PWL_POS/public/api/levels/13?level_kode=SPV22

Send

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

Query Params

<input checked="" type="checkbox"/> Key	Value	Description	Bulk Edit
<input checked="" type="checkbox"/> level_kode	SPV22		
Key	Value	Description	

Body

200 OK • 294 ms • 522 B

Pretty Raw Preview Visualize JSON

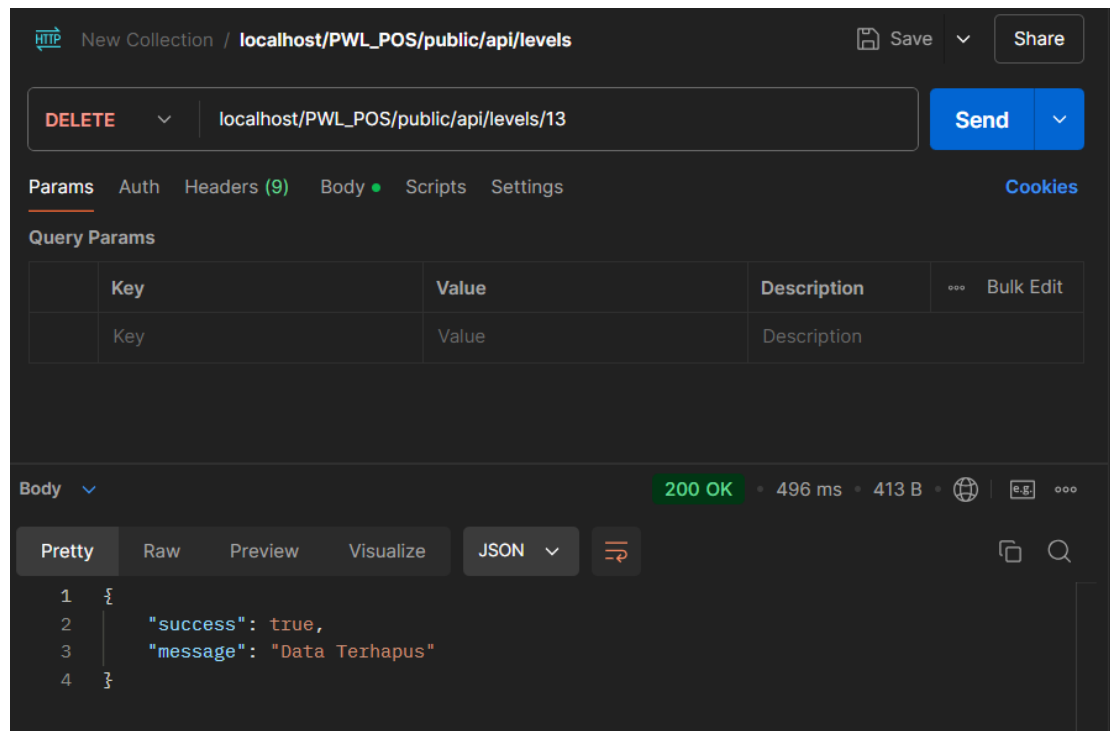
```
1 [
2   {
3     "level_id": 13,
4     "level_kode": "SPV22",
5     "level_nama": "Supervisor2",
6     "created_at": "2024-11-06T05:24:32.000000Z",
7     "updated_at": "2024-11-06T05:36:42.000000Z"
8   }
9 ]
```



Pada proses ini, metode yang dijalankan adalah PUT untuk mengedit data. Metode ini akan mengupdate data yang sesuai dengan ID yang diberikan. Pada kasus kali ini, ID yang digunakan adalah 13, dan data yang diubah mencakup level_kode yang diisi dengan "SPV22". Hasilnya adalah server akan menerima data baru tersebut dan memperbarui entri yang sesuai di dalam tabel level.

8

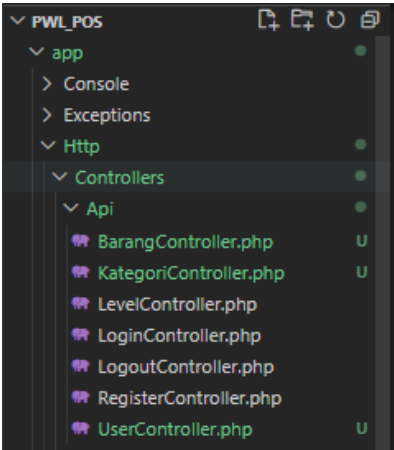
Terakhir lakukan percobaan hapus data. Jelaskan dan berikan screenshot hasil percobaan Anda



Pada proses di atas, metode yang dijalankan adalah `destroy`. Metode ini berfungsi untuk menghapus data berdasarkan ID yang diberikan, yaitu `level_id`. Proses penghapusan data akan dilakukan menggunakan perintah `delete()`.



Soal Praktikum

Langkah	Jawaban/Deskripsi
1	<p>Implementasikan CRUD API pada tabel lainnya yaitu tabel m_user, m_kategori, dan m_barang</p> <p>Create</p> <ul style="list-style-type: none">- BarangController (php artisan make:controller Api/BarangController)- KategoriController (php artisan make:controller Api/KategoriController)- UserController (php artisan make:controller Api/UserController) 
	<p>User</p> <ul style="list-style-type: none">- UserController

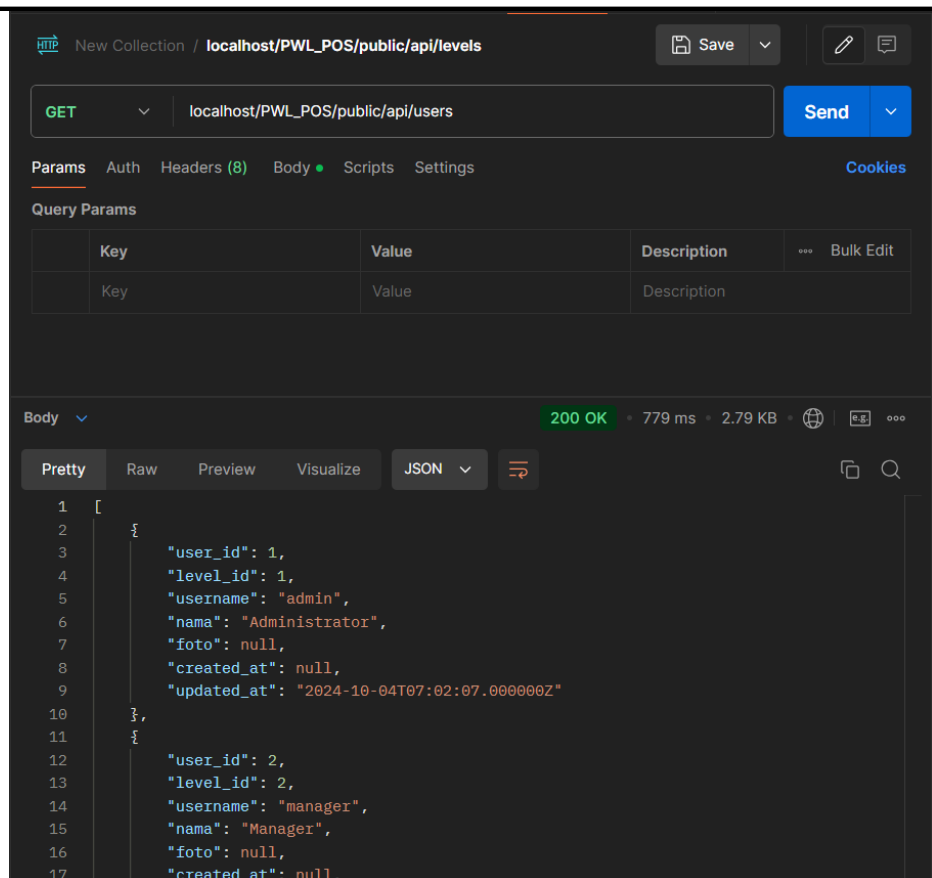


```
app > Http > Controllers > Api > UserController.php > UserController
1  <?php
2
3  namespace App\Http\Controllers\Api;
4
5  use App\Http\Controllers\Controller;
6  use Illuminate\Http\Request;
7  use App\Models\UserModel;
8
9  class UserController extends Controller
10 {
11     public function index()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

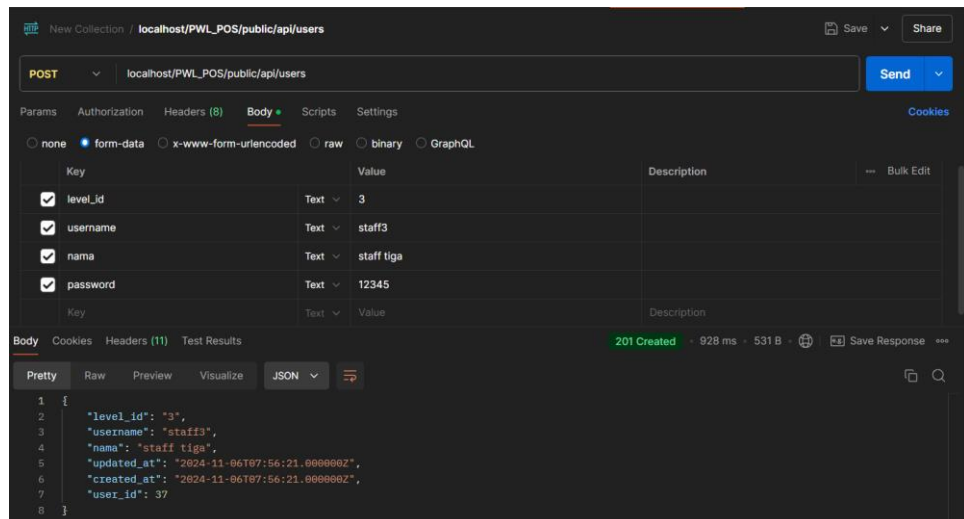
- API

```
routes > api.php > ...
39 Route::group(['prefix' => 'users'], function() {
40     Route::get('/', [UserController::class, 'index']);
41     Route::post('/', [UserController::class, 'store']);
42     Route::get('/{user}', [UserController::class, 'show']);
43     Route::put('/{user}', [UserController::class, 'update']);
44     Route::delete('/{user}', [UserController::class, 'destroy']);
45 });
```

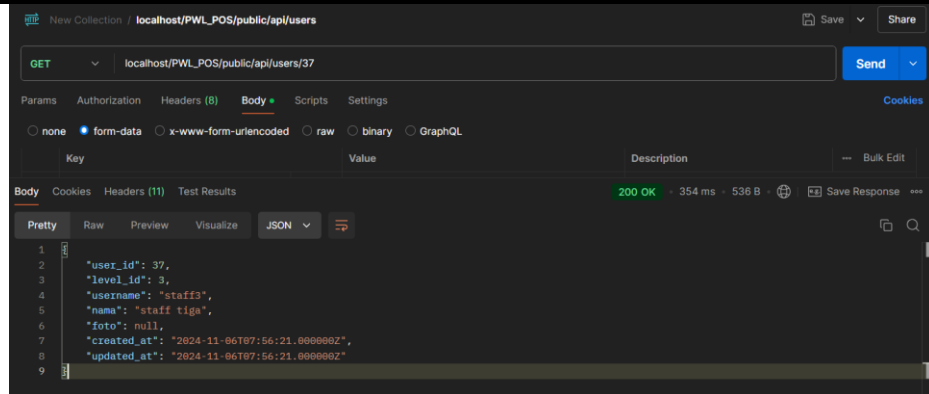
- Get data / Tampil list data



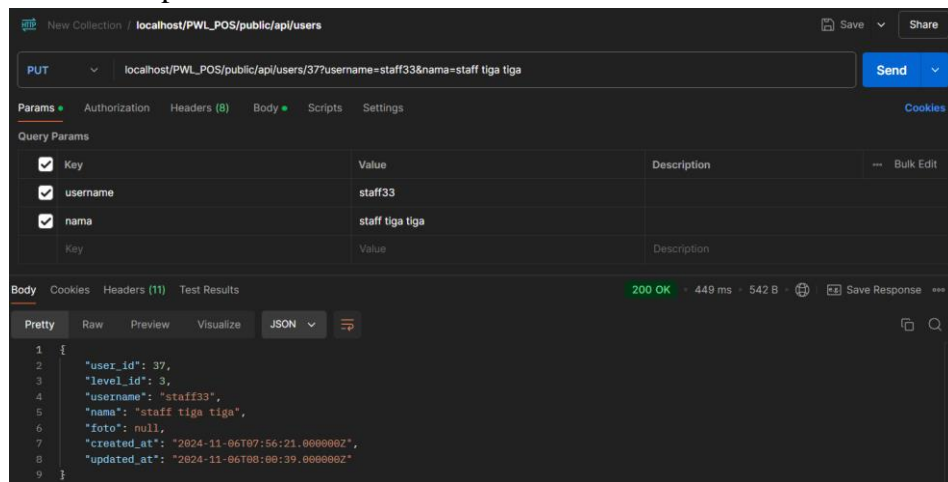
- Post data / Menambahkan data



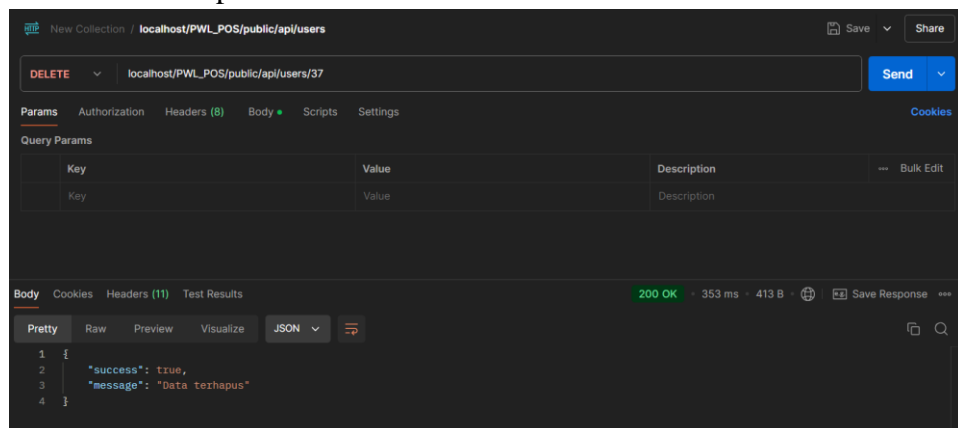
- Get data / Tampil data tertentu



- Put data / Update edit data



- Delete data / Hapus data



Kategori

- KategoriController



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

- API

```
routes > api.php > ...
47 Route::group(['prefix' => 'kategoris'], function() {
48     Route::get('/', [KategoriController::class, 'index']);
49     Route::post('/', [KategoriController::class, 'store']);
50     Route::get('/{kategori}', [KategoriController::class, 'show']);
51     Route::put('/{kategori}', [KategoriController::class, 'update']);
52     Route::delete('/{kategori}', [KategoriController::class, 'destroy']);
53 });
54
```

- Get data / Tampil list data

localhost/PWL_POS/public/api/kategori

GET localhost/PWL_POS/public/api/kategori

Params Authorization Headers (6) Body Scripts Settings Cookies

Query Params

Key	Value	Description
Key	Value	Description

Body Cookies Headers (11) Test Results

200 OK - 355 ms - 1.03 KB

Pretty Raw Preview Visualize JSON

```
1 [
2   {
3     "kategori_id": 1,
4     "kategori_kode": "ELEC",
5     "kategori_nama": "Elektronik",
6     "created_at": null,
7     "updated_at": null
8   },
9   {
10    "kategori_id": 2,
11    "kategori_kode": "FURN",
12    "kategori_nama": "Furniture",
13    "created_at": null,
14    "updated_at": null
15  },
16  {
17    "kategori_id": 3
```



- Post data / Menambahkan data

localhost/PWL_POS/public/api/kategori

POST localhost/PWL_POS/public/api/kategori

Params Authorization Headers (9) Body Scripts Settings Cookies

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

Key	Value	Description
kategori_nama	Peralatan Rumah Tangga	
kategori_kode	PRT	

Body Cookies Headers (12) Test Results

201 Created • 690 ms • 564 B

Pretty Raw Preview Visualize JSON

```
1 {
2   "kategori_nama": "Peralatan Rumah Tangga",
3   "kategori_kode": "PRT",
4   "updated_at": "2024-11-06T08:16:09.000000Z",
5   "created_at": "2024-11-06T08:16:09.000000Z",
6   "kategori_id": 15
7 }
```

- Get data / Tampil data tertentu

New Collection / localhost/PWL_POS/public/api/kategori

GET localhost/PWL_POS/public/api/kategori/15

Params Authorization Headers (9) Body Scripts Settings Cookies

Query Params

Key	Value	Description
Key	Value	Description

Body Cookies Headers (12) Test Results

200 OK • 335 ms • 559 B

Pretty Raw Preview Visualize JSON

```
1 {
2   "kategori_id": 15,
3   "kategori_kode": "PRT",
4   "kategori_nama": "Peralatan Rumah Tangga",
5   "created_at": "2024-11-06T08:16:09.000000Z",
6   "updated_at": "2024-11-06T08:16:09.000000Z"
7 }
```

- Put data / Update edit data

New Collection / localhost/PWL_POS/public/api/kategori

PUT localhost/PWL_POS/public/api/kategori/15?kategori_kode=PRT1

Params Authorization Headers (9) Body Scripts Settings Cookies

Query Params

Key	Value	Description
kategori_kode	PRT1	
Key	Value	Description

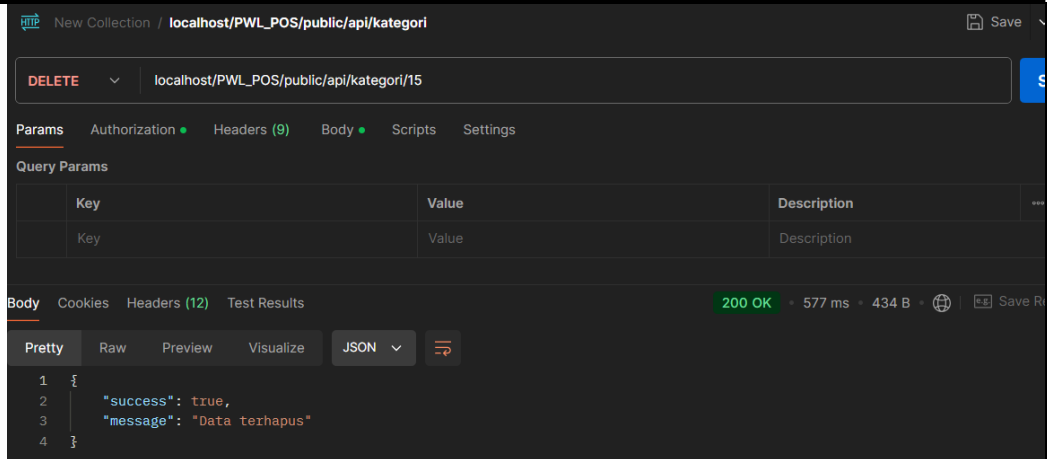
Body Cookies Headers (12) Test Results

200 OK • 391 ms • 560 B

Pretty Raw Preview Visualize JSON

```
1 {
2   "kategori_id": 15,
3   "kategori_kode": "PRT1",
4   "kategori_nama": "Peralatan Rumah Tangga",
5   "created_at": "2024-11-06T08:16:09.000000Z",
6   "updated_at": "2024-11-06T08:18:01.000000Z"
7 }
```

- Delete data / Hapus data



Barang

- BarangController

```
app > Http > Controllers > Api > BarangController.php > BarangController
1  <?php
2
3  namespace App\Http\Controllers\Api;
4
5  use App\Http\Controllers\Controller;
6  use Illuminate\Http\Request;
7  use App\Models\BarangModel;
8
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 terhapus',
39         ]);
40     }
41 }
```

- API



```
routes > api.php > ...  
55 Route::group(['prefix' => 'barangs'], function() {  
56     Route::get('/', [BarangController::class, 'index']);  
57     Route::post('/', [BarangController::class, 'store']);  
58     Route::get('/{barang}', [BarangController::class, 'show']);  
59     Route::put('/{barang}', [BarangController::class, 'update']);  
60     Route::delete('/{barang}', [BarangController::class, 'destroy']);  
61 });
```

- Get data / Tampil list data

localhost/PWL_POS/public/api/barang

GET localhost/PWL_POS/public/api/barang

Send

Params Authorization Headers (6) Body Scripts Settings Cookies

Query Params

Key	Value	Description
Key	Value	Description

Body Cookies Headers (11) Test Results 200 OK 452 ms 3.41 KB Save Response

Pretty Raw Preview Visualize JSON

```
1 [  
2   {  
3     "barang_id": 1,  
4     "kategori_id": 2,  
5     "barang_kode": "BRG1",  
6     "barang_nama": "Barang 1",  
7     "harga_beli": 40951,  
8     "harga_jual": 90560,  
9     "created_at": null,  
10    "updated_at": null  
11  },  
12  {  
13    "barang_id": 2,  
14    "kategori_id": 2,  
15    "barang_kode": "BRG2",
```

- Post data / Menambahkan data

localhost/PWL_POS/public/api/barang

POST localhost/PWL_POS/public/api/barang

Send

Params Authorization Headers (8) Body Scripts Settings Cookies

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

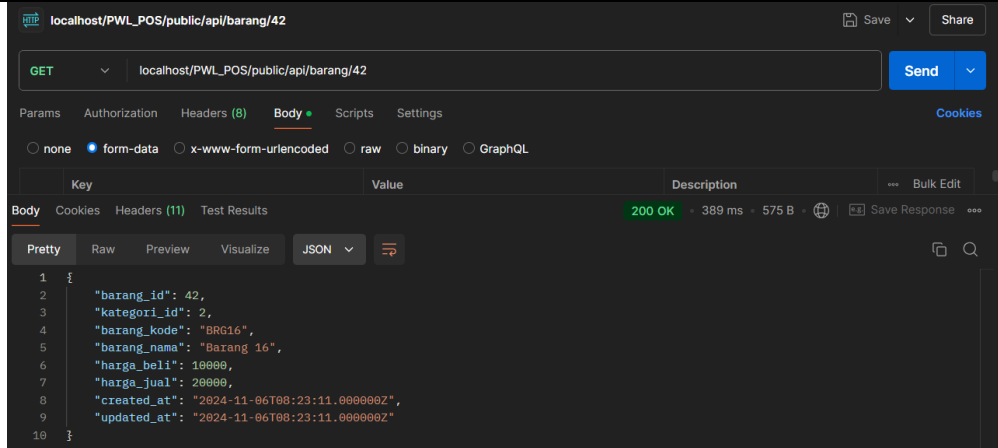
Key	Value	Description
barang_nama	Text Barang 16	
barang_kode	Text BRG16	
kategori_id	Text 2	
harga_beli	Text 10000	
harga_jual	Text 20000	

Body Cookies Headers (11) Test Results 201 Created 351 ms 586 B Save Response

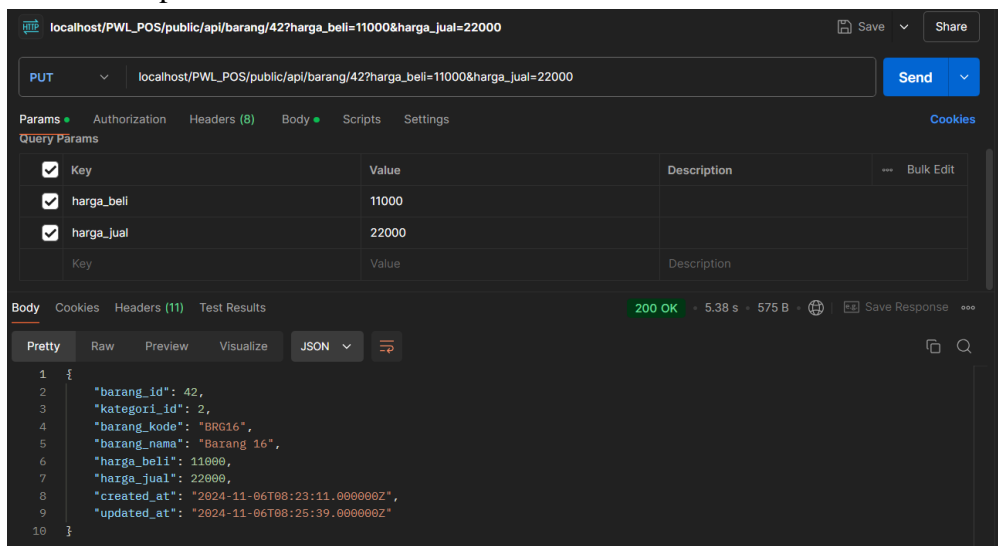
Pretty Raw Preview Visualize JSON

```
1 {  
2   "barang_nama": "Barang 16",  
3   "barang_kode": "BRG16",  
4   "kategori_id": "2",  
5   "harga_beli": "10000",  
6   "harga_jual": "20000",  
7   "updated_at": "2024-11-06T08:23:11.000000Z",  
8   "created_at": "2024-11-06T08:23:11.000000Z",  
9   "barang_id": 42  
10 }
```

- Get data / Tampil data tertentu



- Put data / Update edit data



- Delete data / Hapus data

