

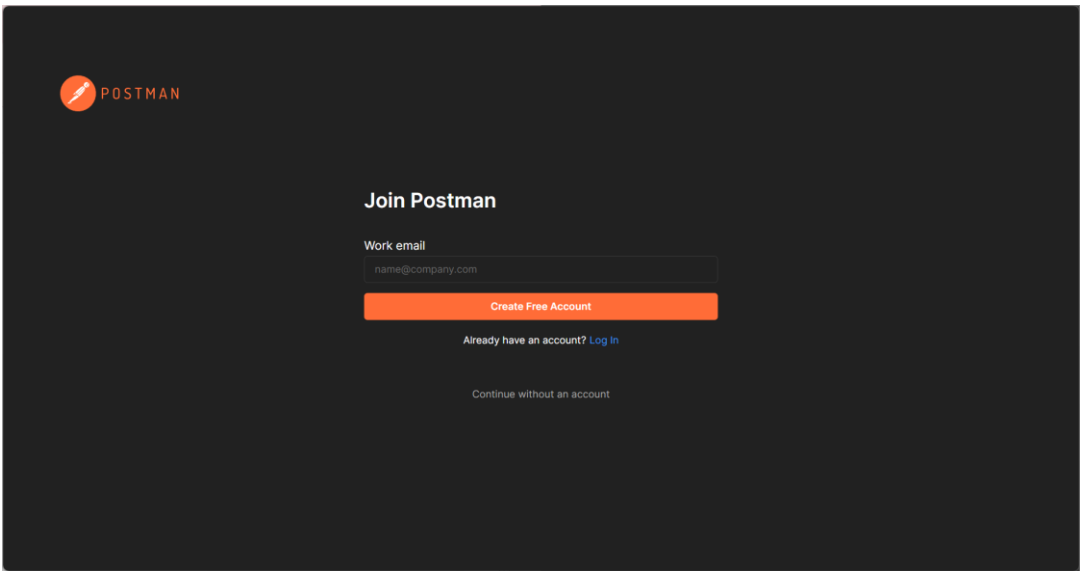


Mata Kuliah : Pemrograman Web Lanjut (PWL)  
Program Studi : D4 – Sistem Informasi Bisnis  
Semester : 5

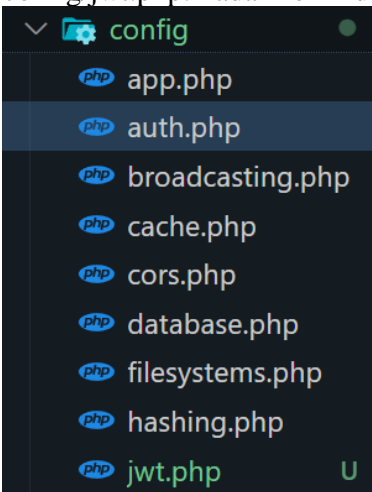
Kelas : SIB 3D  
NIM : 2241760030  
Nama : Fardiyani Afro'ul Yasinta  
Jobsheet Ke- : 10

## Laporan Jobsheet 10 | API Bagian 1

### Praktikum Ke-1 | Membuat RESTful API Register

Langkah	Jawaban/Deskripsi
1.	<p>Sebelum memulai membuat REST API, terlebih dahulu download aplikasi Postman di <a href="https://www.postman.com/downloads">https://www.postman.com/downloads</a>.</p> <p>Aplikasi ini akan digunakan untuk mengerjakan semua tahap praktikum pada Jobsheet ini.</p> 
2.	<p>Lakukan instalasi JWT dengan mengetikkan perintah berikut: <code>composer require tymon/jwt-auth:2.1.1</code></p> <p>Pastikan Anda terkoneksi dengan internet.</p>



	<pre>PS C:\laragon\www\PWL_POS&gt; 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) Package operations: 4 installs, 0 updates, 0 removals - Downloading stella-maris/clock (0.1.7) - Downloading lcobucci/clock (2.3.0) - Downloading lcobucci/jwt (4.0.4) - Downloading tymon/jwt-auth (2.1.1) - Installing stella-maris/clock (0.1.7): Extracting archive - Installing lcobucci/clock (2.3.0): Extracting archive - Installing lcobucci/jwt (4.0.4): Extracting archive - Installing tymon/jwt-auth (2.1.1): Extracting archive Generating optimized autoload files</pre>
3.	<p>Setelah berhasil menginstall JWT, lanjutkan dengan publish konfigurasi file dengan perintah berikut:</p> <pre>php artisan vendor:publish --provider="Tymon\JWTAuth\Providers\LaravelServiceProvider"</pre> <pre>PS C:\laragon\www\PWL_POS&gt; php artisan vendor:publish --provider="Tymon\JWTAuth\Providers\LaravelServicePro vider"  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> 



5.	<p>Setelah itu jalankan perintah berikut untuk membuat secret key JWT.</p> <pre>php artisan jwt:secret</pre> <p>Jika berhasil, maka pada file .env akan ditambahkan sebuah baris berisi nilai key JWT_SECRET.</p> <pre>PS C:\laragon\www\PWL_POS&gt; php artisan jwt:secret jwt-auth secret [kViVYj97JziXeG8Sks8HOP6YDZdpgUYmCxKeQHXYX3SN2i7ZF1r051Db6QD0d5tHX] set successfully.</pre>
6.	<p>Selanjutnya lakukan konfigurasi guard API. Buka config/auth.php. Ubah bagian 'guards' menjadi seperti berikut.</p> <pre>'guards' =&gt; [     'web' =&gt; [         'driver' =&gt; 'session',         'provider' =&gt; 'users',     ],     'api' =&gt; [         'driver' =&gt; 'jwt',         'provider' =&gt; 'users',     ], ],</pre>
7.	<p>Kita akan menambahkan kode di model UserModel, ubah kode seperti berikut:</p>



8.	<p>Berikutnya kita akan membuat controller untuk register dengan menjalankan perintah berikut.</p> <pre>php artisan make:controller Api/RegisterController</pre> <p>Jika berhasil maka akan ada tambahan controller pada folder Api dengan nama RegisterController.</p>
9.	<p>Buka file tersebut, dan ubah kode menjadi seperti berikut.</p>

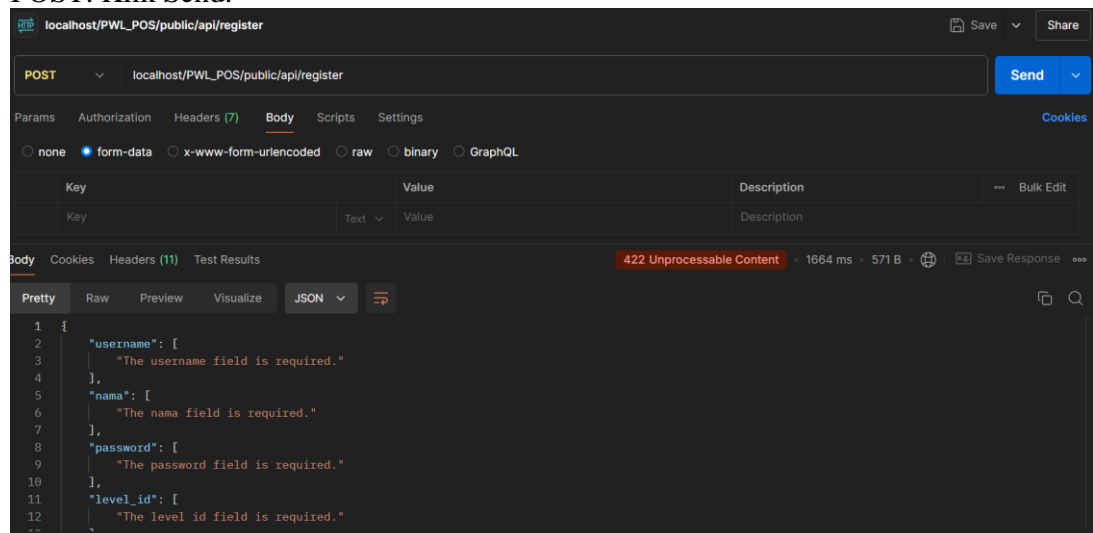


	<pre>1  &lt;?php 2 3  namespace App\Http\Controllers\Api; 4 5  use App\Http\Controllers\Controller; 6  use Illuminate\Http\Request; 7  use App\Models\UserModel; 8  use Illuminate\Support\Facades\Validator; 9 10 class RegisterController extends Controller 11 { 12     public function __invoke(Request \$request) 13     { 14         // set validation 15         \$validator = Validator::make(\$request-&gt;all(), [ 16             'username' =&gt; 'required', 17             'nama' =&gt; 'required', 18             'password' =&gt; 'required min:5 confirmed', 19             'level_id' =&gt; 'required' 20         ]); 21 22         // if validation fails 23         if(\$validator-&gt;fails()) { 24             return response()-&gt;json(\$validator-&gt;errors(), 422); 25         } 26 27         //create user 28         \$user = UserModel::create([ 29             'username' =&gt; \$request-&gt;username, 30             'nama' =&gt; \$request-&gt;nama, 31             'password' =&gt; bcrypt(\$request-&gt;password), 32             'level_id' =&gt; \$request-&gt;level_id 33         ]); 34 35         // return response JSON user is created 36         if (\$user) { 37             return response()-&gt;json([ 38                 'success' =&gt; true, 39                 'user' =&gt; \$user, 40             ], 201); 41         } 42 43         // return JSON process insert failed 44         return response()-&gt;json([ 45             'success' =&gt; false, 46         ], 409); 47     } 48 }</pre>
10.	Selanjutnya buka routes/api.php, ubah semua kode menjadi seperti berikut.



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.

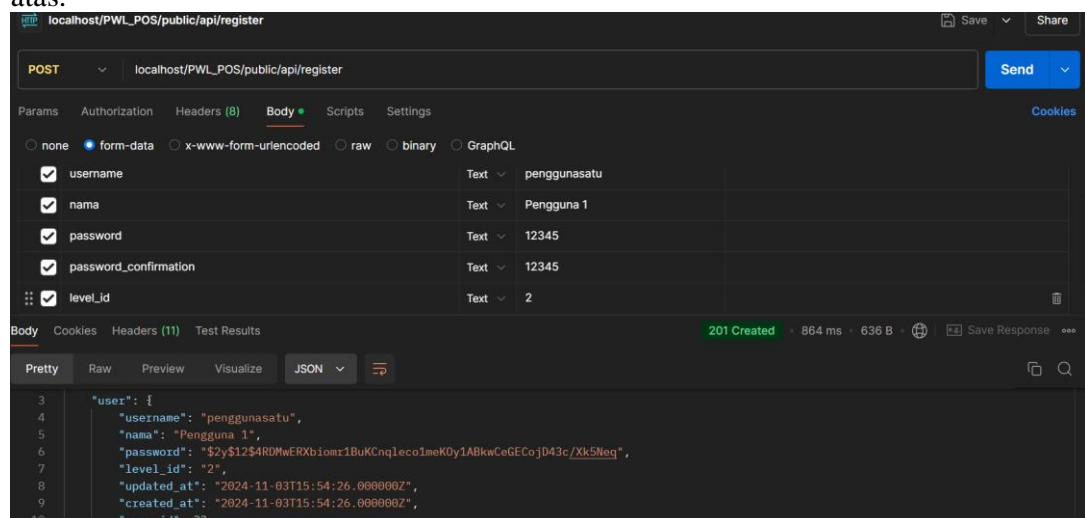


Jika berhasil akan muncul error validasi seperti gambar di atas.

**Lakukan percobaan yang sama dan berikan screenshot 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 screenshot hasil percobaan Anda.**

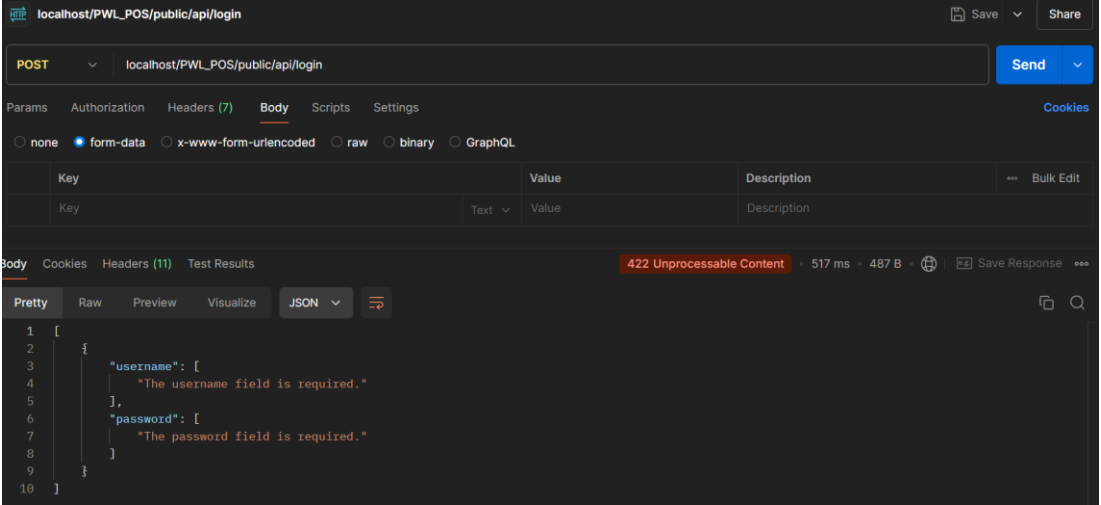


13.	Lakukan commit perubahan file pada Github.
-----	--

## Praktikum Ke-2 | Membuat RESTful API Login

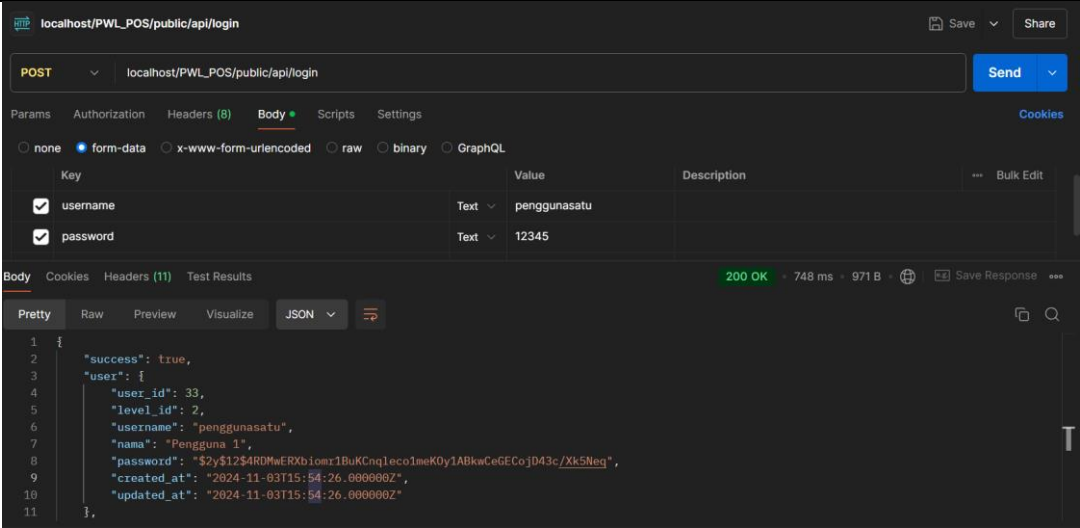
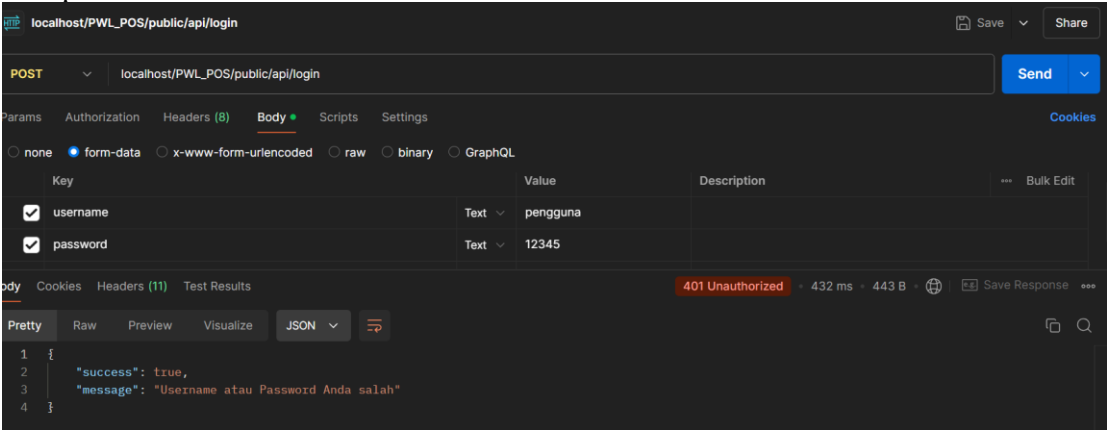
Langkah	Jawaban/Deskripsi
1.	<p>Kita buat file controller dengan nama LoginController.</p> <pre>php artisan make:controller Api/LoginController</pre> <p>Jika berhasil maka akan ada tambahan controller pada folder Api dengan nama LoginController.</p> <pre>PS C:\laragon\www\PWL_POS&gt; php artisan make:controller Api/LoginController</pre> <pre>INFO Controller [C:\laragon\www\PWL_POS\app\Http\Controllers\Api&gt;LoginController.php] created successfully.</pre>
2.	<p>Buka file tersebut, dan ubah kode menjadi seperti berikut.</p> <pre>1 &lt;?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-&gt;all(), [ 15             'username' =&gt; 'required', 16             'password' =&gt; 'required', 17         ]); 18 19         //if validation fails 20         if (\$validator-&gt;fails()) { 21             return response()-&gt;json([\$validator-&gt;errors()], 422); 22         } 23 24         //get credentials from request 25         \$credentials = \$request-&gt;only('username', 'password'); 26 27         //if auth failed 28         if (!\$token = auth()-&gt;guard('api')-&gt;attempt(\$credentials)) { 29             return response()-&gt;json([ 30                 'success' =&gt; true, 31                 'message' =&gt; 'Username atau Password Anda salah', 32             ], 401); 33         } 34 35         // if auth success 36         return response()-&gt;json([ 37             'success' =&gt; true, 38             'user' =&gt; auth()-&gt;guard('api')-&gt;user(), 39             'token' =&gt; \$token, 40         ], 200); 41     } 42 }</pre>
3.	Berikutnya tambahkan route baru pada file api.php yaitu /login dan /user.



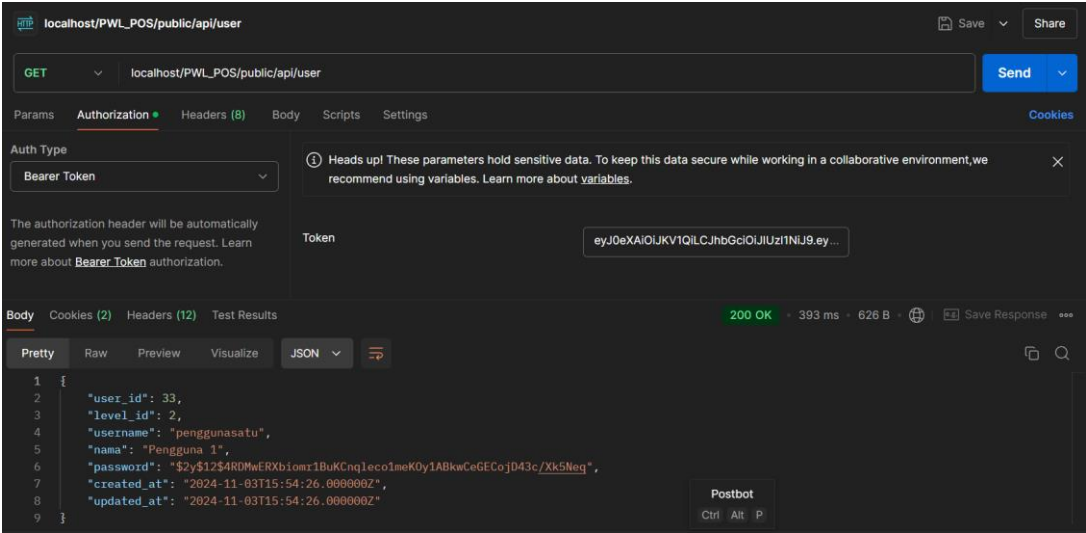
	<pre>routes &gt; api.php &gt; ... 1  &lt;?php 2 3  use Illuminate\Http\Request; 4  use Illuminate\Support\Facades\Route; 5  use App\Http\Controllers\Api\RegisterController; 6 7  /* 8   ----- 9    API Routes 10  ----- 11   12   Here is where you can register API routes for your application. These 13   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(uri: '/register', action: App\Http\Controllers\Api\RegisterController::class)-&gt;name(name: 'register'); 19 Route::post(uri: '/login', action: App\Http\Controllers\Api\LoginController::class)-&gt;name(name: 'login'); 20 Route::middleware(middleware: 'auth:api')-&gt;get(uri: '/user', action: function (Request \$request): mixed { 21     return \$request-&gt;user(); 22 }); </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>  <p>Jika berhasil akan muncul error validasi seperti gambar di atas. <b>Lakukan percobaan yang sama dan berikan screenshot hasil percobaan Anda.</b></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>





	
	Lakukan percobaan yang sama dan berikan screenshoot hasil percobaan Anda.
6.	<p>Lakukan percobaan yang untuk data yang salah dan berikan screenshoot hasil percobaan Anda.</p> <p>Mencoba menggunakan username pengguna, hasilnya Terdapat pesan bahwa “username atau password anda salah”</p> 
7.	<p>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. <b>Jelaskan hasil dari percobaan tersebut.</b></p>



	
8.	Lakukan commit perubahan file pada Github.

### Praktikum Ke-3 | Membuat RESTful API Logout

Langkah	Jawaban/Deskripsi
1.	<p>Tambahkan kode berikut pada file .env</p> <pre>JWT_SHOW_BLACKLIST_EXCEPTION=true</pre> 
2.	<p>Buat Controller baru dengan nama LogoutController.</p> <pre>php artisan make:controller Api/LogoutController</pre>



	<pre>PS C:\laragon\www\PWL_POS&gt; php artisan make:controller Api/LogoutController</pre> <p><b>INFO</b> Controller [C:\laragon\www\PWL_POS\app\Http\Controllers\Api\LogoutController.php] created successfully.</p>
3.	<p>Buka file tersebut dan ubah kode menjadi seperti berikut.</p> <pre>app &gt; Http &gt; Controllers &gt; Api &gt; LogoutController.php &gt; PHP Intelephense &gt; LogoutController</pre> <pre>1  &lt;?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\JWTException; 9  use Tymon\JWTAuth\Exceptions\TokenExpiredException; 10 use Tymon\JWTAuth\Exceptions\TokenInvalidException; 11 12 0 references   0 implementations 12 class LogoutController extends Controller 13 { 14 0 references   0 overrides 14 public function __invoke(Request \$request): JsonResponse mixed 15 { 16     //remove token 17     \$removeToken = JWTAuth::invalidate(JWTAuth::getToken()); 18 19     if(\$removeToken){ 20         //return response JSON 21         return response()-&gt;json(data: [ 22             'sukses' =&gt; true, 23             'message' =&gt; 'Logout Berhasil!', 24         ]); 25     } 26 } 27 }</pre>
4.	<p>Lalu kita tambahkan routes pada api.php</p> <pre>Route::post(uri: '/register', action: App\Http\Controllers\Api\RegisterController::class)-&gt;name(name: 'register'); Route::post(uri: '/login', action: App\Http\Controllers\Api&gt;LoginController::class)-&gt;name(name: 'login'); Route::middleware(middleware: 'auth:api')-&gt;get(uri: '/user', action: function (Request \$request): mixed {     return \$request-&gt;user(); }); Route::post(uri: '/logout', action: App\Http\Controllers\Api\LogoutController::class)-&gt;name(name: 'logout');</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> <p><b>Lakukan percobaan yang sama dan berikan screenshoot hasil percobaan Anda.</b></p>



	
7.	Lakukan commit perubahan file pada Github.

#### Praktikum Ke-4 | Implementasi CRUD dalam RESTful API

Langkah	Jawaban/Deskripsi
1.	<p>Pertama, buat controller untuk mengolah API pada data level. <code>php artisan make:controller Api/LevelController</code></p> <pre>PS C:\laragon\www\PWL_POS&gt; php artisan make:controller Api/LevelController INFO Controller [C:\laragon\www\PWL_POS\app\Http\Controllers\Api\LevelController.php] created successfully.</pre>
2.	Setelah berhasil, buka file tersebut dan tuliskan kode seperti berikut yang berisi fungsi CRUDnya.



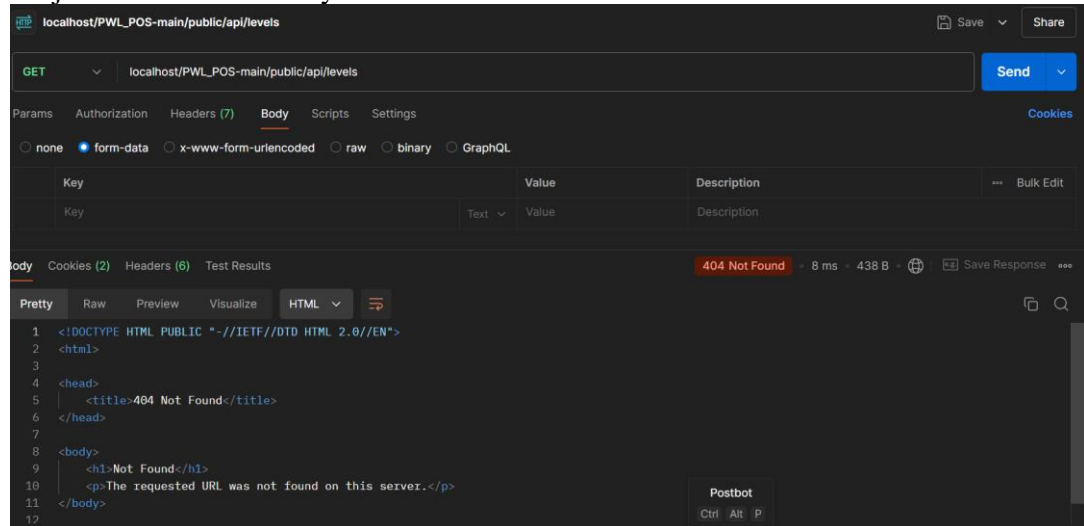
	<pre>1  &lt;?php 2 3  namespace App\Http\Controllers\Api; 4 5  use App\Http\Controllers\Controller; 6  use Illuminate\Http\Request; 7  use App\Models\LevelModel; 8 9  class LevelController extends Controller 10 { 11     public function index() 12     { 13         return LevelModel::all(); 14     } 15 16     public function store(Request \$request) 17     { 18         \$level = LevelModel::create(\$request-&gt;all()); 19         return response()-&gt;json(\$level, 201); 20     } 21 22     public function show(LevelModel \$level) 23     { 24         return LevelModel::find(\$level); 25     } 26 27     public function update(Request \$request, LevelModel \$level) 28     { 29         \$level-&gt;update(\$request-&gt;all()); 30         return LevelModel::find(\$level); 31     } 32 33     public function destroy(LevelModel \$user) 34     { 35         \$user-&gt;delete(); 36         return response()-&gt;json([ 37             'success' =&gt; true, 38             'message' =&gt; "Data terhapus", 39         ]); 40     } 41 }</pre>
3.	<p>Kemudian kita lengkapi routes pada api.php</p> <pre>19 Route::post(uri: '/register', action: App\Http\Controllers\Api\RegisterController::class)-&gt;name(name: 'register'); 20 Route::post(uri: '/login', action: App\Http\Controllers\Api&gt;LoginController::class)-&gt;name(name: 'login'); 21 Route::middleware(middleware: 'auth:api')-&gt;get(uri: '/user', action: function (Request \$request): mixed { 22     return \$request-&gt;user(); 23 }); 24 Route::post(uri: '/logout', action: App\Http\Controllers\Api\LogoutController::class)-&gt;name(name: 'logout'); 25 // Level 26 Route::get(uri: 'level', action: [LevelController::class, 'index']); 27 Route::post(uri: 'levels', action: [LevelController::class, 'store']); 28 Route::get(uri: 'levels/{level}', action: [LevelController::class, 'show']); 29 Route::put(uri: 'levels/{level}', action: [LevelController::class, 'update']); 30 Route::delete(uri: 'levels/{level}', action: [LevelController::class, 'destroy']);</pre>



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.**

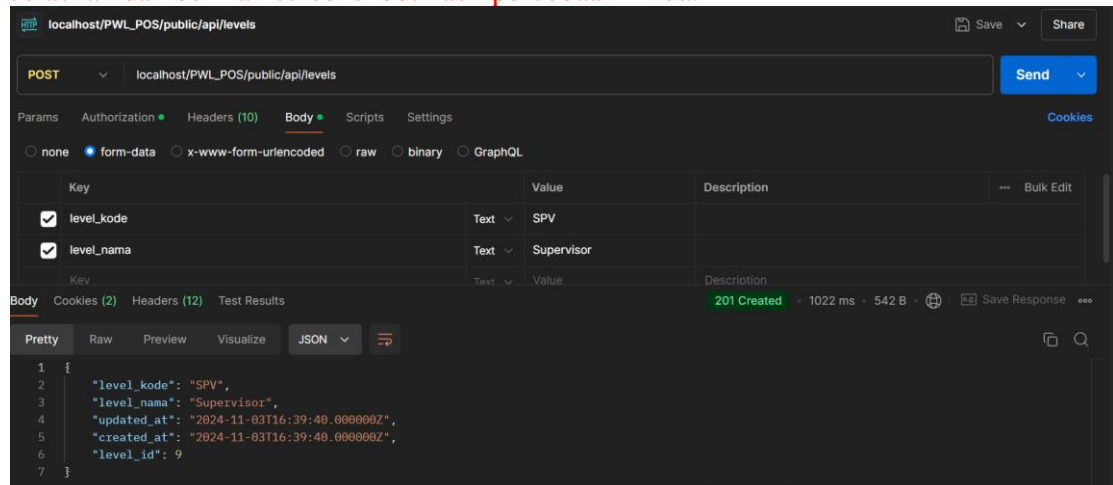
Terjadi error karena url nya tidak sesuai.



5.

Kemudian, lakukan percobaan penambahan data dengan URL : localhost/PWL\_POS-main/public/api/levels dan method POST seperti di bawah ini.

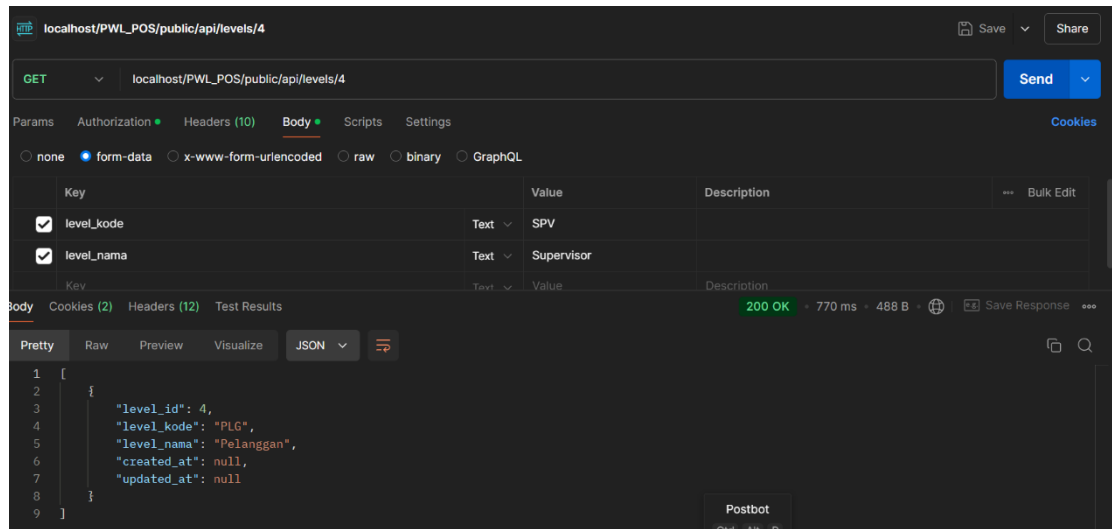
**Jelaskan dan berikan screenshot hasil percobaan Anda.**



6.

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

Percobaan melihat detail data level\_id 4

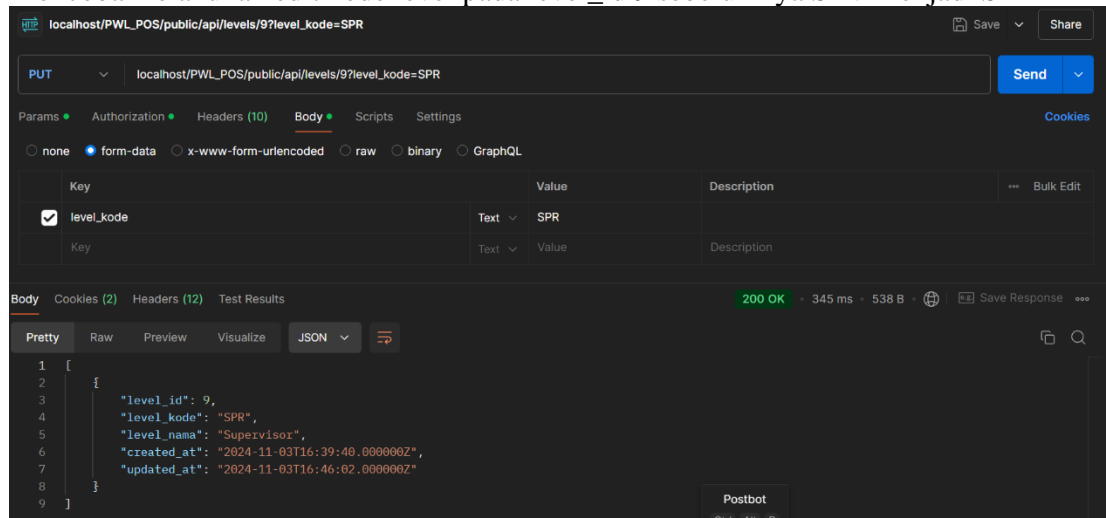


7.

Jika sudah, kita coba untuk melakukan edit data menggunakan localhost/PWL\_POS-main/public/api/levels/{id} dan method PUT. Isikan data yang ingin diubah pada tab Param.

**Jelaskan dan berikan screenshot hasil percobaan Anda.**

Mencoba melakukan edit kode level pada level\_id 9 sebelumnya SPV menjadi SPR



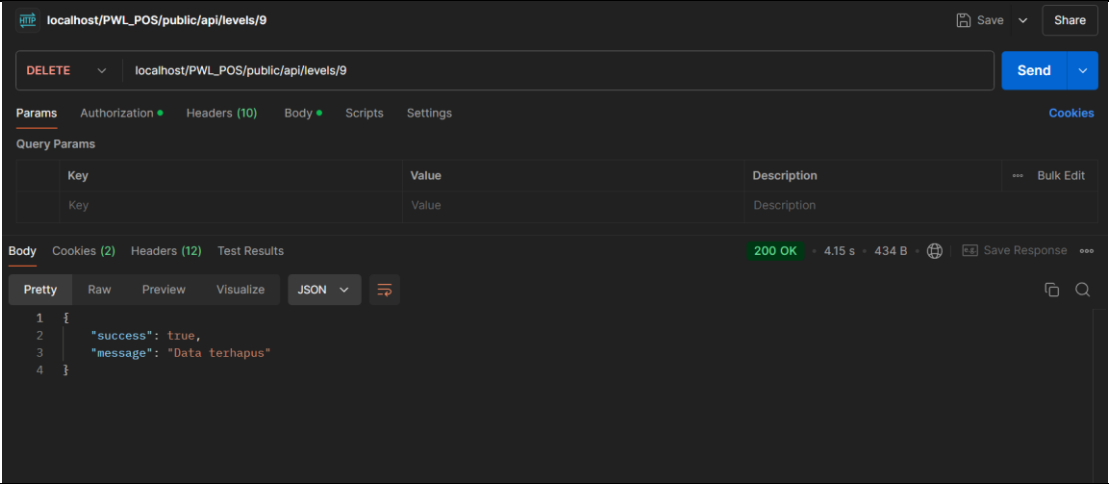
<div><div>←</div><div>→</div></div>				level_id	level_kode	level_nama	created_at	updated_at
<div><div><input type="checkbox"/></div><div><div><div></div></div></div><div>Edit</div><div><div><div></div></div></div><div>Copy</div><div><div><div></div></div></div><div>Delete</div></div>	1	ADM	Administrator	NULL	NULL			
<div><div><input type="checkbox"/></div><div><div><div></div></div></div><div>Edit</div><div><div><div></div></div></div><div>Copy</div><div><div><div></div></div></div><div>Delete</div></div>	2	MNG	Manager	NULL	NULL			
<div><div><input type="checkbox"/></div><div><div><div></div></div></div><div>Edit</div><div><div><div></div></div></div><div>Copy</div><div><div><div></div></div></div><div>Delete</div></div>	3	STF	Staff/kasir	NULL	NULL			
<div><div><input type="checkbox"/></div><div><div><div></div></div></div><div>Edit</div><div><div><div></div></div></div><div>Copy</div><div><div><div></div></div></div><div>Delete</div></div>	4	PLG	Pelanggan	NULL	NULL			
<div><div><input type="checkbox"/></div><div><div><div></div></div></div><div>Edit</div><div><div><div></div></div></div><div>Copy</div><div><div><div></div></div></div><div>Delete</div></div>	9	SPR	Supervisor	2024-11-03 16:39:40	2024-11-03 16:46:02			

8.

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

Mencoba menghapus level\_id 9 yaitu supervisor dengan level\_kode SPR



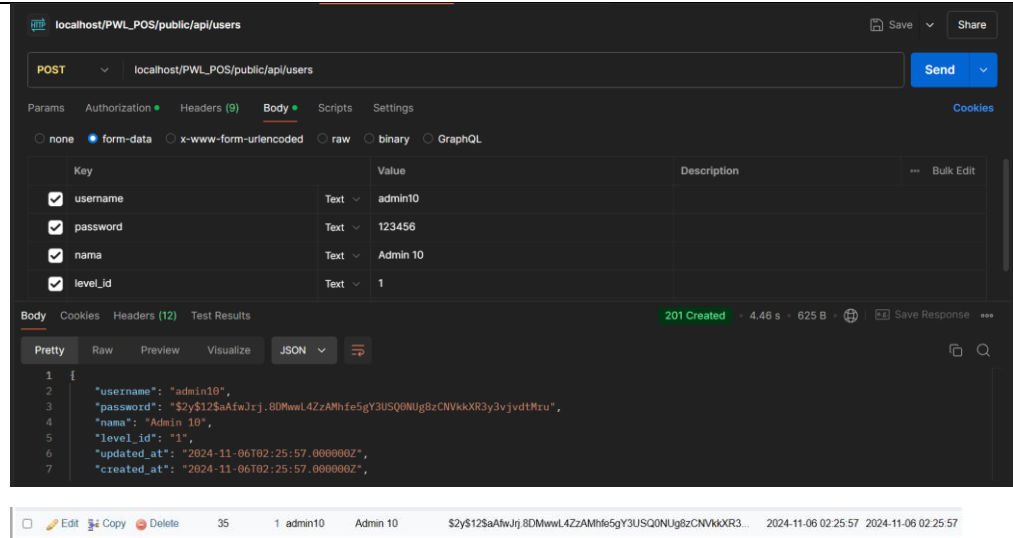
	
9.	Lakukan commit perubahan file pada Github.



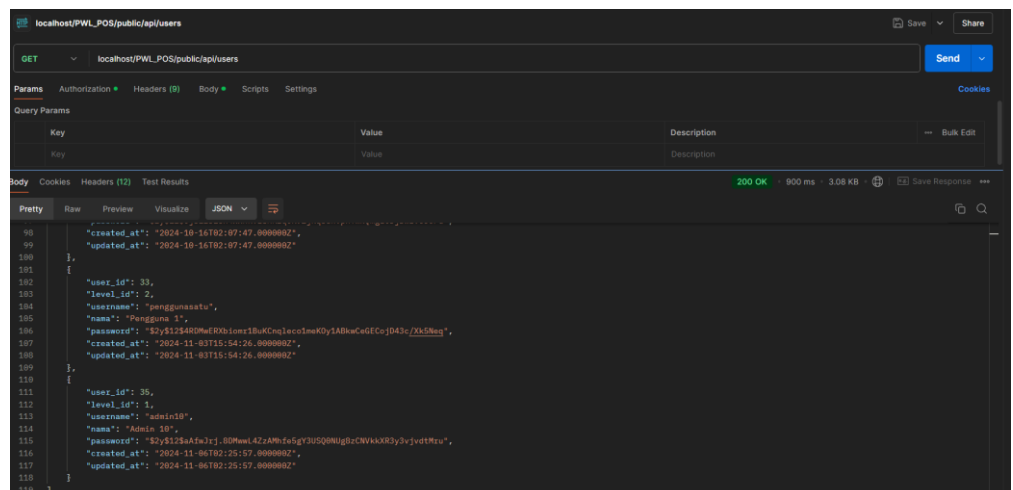


## Tugas | Implementasi CRUD pada tabel m\_user, m\_kategori, dan m\_barang

Langkah	Jawaban/Deskripsi
1.	<p><b>Tabel m_user</b></p> <p><b>Kode Program</b></p> <ul style="list-style-type: none"><li><b>UserController.php</b></li></ul> <pre>1 &lt;?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         \$id = UserModel::create(\$request-&gt;all()); 19         return response()-&gt;json(\$id, 201); 20     } 21 22     public function show(UserModel \$id) 23     { 24         return UserModel::find(\$id); 25     } 26 27     public function update(Request \$request, UserModel \$id) 28     { 29         \$id-&gt;update(\$request-&gt;all()); 30         return UserModel::find(\$id); 31     } 32 33     public function destroy(UserModel \$id) 34     { 35         \$id-&gt;delete(); 36         return response()-&gt;json([ 37             'success' =&gt; true, 38             'message' =&gt; "Data terhapus", 39         ]); 40     } 41 } 42 43</pre> <ul style="list-style-type: none"><li><b>Api.php</b></li></ul> <pre>32 33 // User 34 Route::get(uri: 'users', action: [UserController::class, 'index']); 35 Route::post(uri: 'users', action: [UserController::class, 'store']); 36 Route::get(uri: 'users/{id}', action: [UserController::class, 'show']); 37 Route::put(uri: 'users/{id}', action: [UserController::class, 'update']); 38 Route::delete(uri: 'users/{id}', action: [UserController::class, 'destroy']); </pre> <p><b>Postman</b></p> <ul style="list-style-type: none"><li><b>POST</b></li></ul>



- **GET**  
Menampilkan semua data user



- **DETAIL**  
Menampilkan detail user\_id 35



```
10 {
11   {
12     "user_id": 35,
13     "level_id": 1,
14     "username": "admin10",
15     "nama": "Admin 10",
16     "password": "$2y$12$AfwJrj.8DMwwL4ZzAMhfe5gY3USQ0Njg8zCNVkkXR3y3vjvdtMru",
17     "created_at": "2024-11-06T02:25:57.000000Z",
18     "updated_at": "2024-11-06T02:25:57.000000Z"
19   }
20 }
```

- **PUT**

Melakukan edit nama pada detail user\_id 35 yang sebelumnya “Admin 10” menjadi “Admin Sepuluh”

```
10 {
11   {
12     "user_id": 35,
13     "level_id": 1,
14     "username": "admin10",
15     "nama": "Admin Sepuluh",
16     "password": "$2y$12$AfwJrj.8DMwwL4ZzAMhfe5gY3USQ0Njg8zCNVkkXR3y3vjvdtMru",
17     "created_at": "2024-11-06T02:25:57.000000Z",
18     "updated_at": "2024-11-06T02:41:51.000000Z"
19   }
20 }
```

- **DELETE**

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



			24	2	manager12	Manager11	\$2y\$12\$A6y8Ze0KbJuGjL4m8zxqeiA6dDgEGi0DFs007EM.lq...	2024-09-19 13:57:59	2024-09-19 13:57:59
			32	1	pinkeu	Fafa	\$2y\$12\$Ojb12D1SH4xohkvtSMKtqOK9Zkqsun.pnTmNqmgLCD...	2024-10-16 02:07:47	2024-10-16 02:07:47
			33	2	penggunasatu	Pengguna 1	\$2y\$12\$4RDMwERXbionr1BuKCnqleco1meKOy1ABkwCeGECqjD...	2024-11-03 15:54:26	2024-11-03 15:54:26

Check all   With selected: Edit   Copy   Delete   Export

2.

## Tabel m\_kategori

### Kode Program

- **KategoriController.php**

```
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);
25     }
26
27     public function update(Request $request, KategoriModel $kategori)
28     {
29         $kategori->update($request->all());
30         return KategoriModel::find($kategori);
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.php**

```
// Kategori
Route::get(uri: 'kategoris', action: [KategoriController::class, 'index']);
Route::post(uri: 'kategoris', action: [KategoriController::class, 'store']);
Route::get(uri: 'kategoris/{kategori}', action: [KategoriController::class, 'show']);
Route::put(uri: 'kategoris/{kategori}', action: [KategoriController::class, 'update']);
Route::delete(uri: 'kategoris/{kategori}', action: [KategoriController::class, 'destroy']);
```

### Postman

- **POST**



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_kode	GFT	
kategori_nama	Gudang Furniture	

Body Cookies Headers (12) Test Results

201 Created · 1300 ms · 558 B · Save Response

Pretty Raw Preview Visualize JSON

```
1 {
2   "kategori_kode": "GFT",
3   "kategori_nama": "Gudang Furniture",
4   "updated_at": "2024-11-06T03:34:13.000000Z",
5   "created_at": "2024-11-06T03:34:13.000000Z",
6   "kategori_id": 12
7 }
```

	12	GFT	Gudang Furniture	2024-11-06 03:34:13	2024-11-06 03:34:13
<input type="checkbox"/>	<input type="text" value="Edit"/>	<input type="text" value="Copy"/>	<input type="text" value="Delete"/>		

- **GET**

localhost/PWL\_POS/public/api/kategori

GET 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_kode	GFT	
kategori_nama	Gudang Furniture	

Body Cookies Headers (12) Test Results

200 OK · 648 ms · 1.06 KB · Save Response

Pretty Raw Preview Visualize JSON

```
24 [
25   {
26     "kategori_id": 4,
27     "kategori_kode": "M",
28     "kategori_nama": "Makanan dan Minuman",
29     "updated_at": null,
30     "created_at": null
31   },
32   {
33     "kategori_id": 5,
34     "kategori_kode": "PK",
35     "kategori_nama": "Pakaian",
36     "updated_at": null,
37     "created_at": null
38   },
39   {
40     "kategori_id": 12,
41     "kategori_kode": "GFT",
42     "kategori_nama": "Gudang Furniture",
43     "updated_at": "2024-11-06T03:34:13.000000Z",
44     "created_at": "2024-11-06T03:34:13.000000Z"
45   }
46 ]
```

- **DETAIL**

Menampilkan detail kategori\_id 12

localhost/PWL\_POS/public/api/kategori/12

GET localhost/PWL\_POS/public/api/kategori/12

Params Authorization Headers (9) Body Scripts Settings Cookies

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

Key	Value	Description
kategori_kode	GFT	
kategori_nama	Gudang Furniture	

Body Cookies Headers (12) Test Results

200 OK · 780 ms · 555 B · Save Response

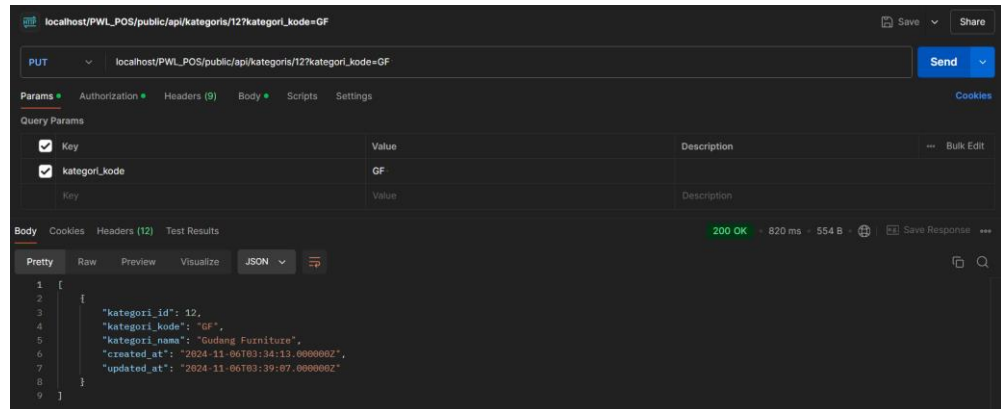
Pretty Raw Preview Visualize JSON

```
1 {
2   "kategori_id": 12,
3   "kategori_kode": "GFT",
4   "kategori_nama": "Gudang Furniture",
5   "updated_at": "2024-11-06T03:34:13.000000Z",
6   "created_at": "2024-11-06T03:34:13.000000Z"
7 }
8
9 }
```

- **PUT**

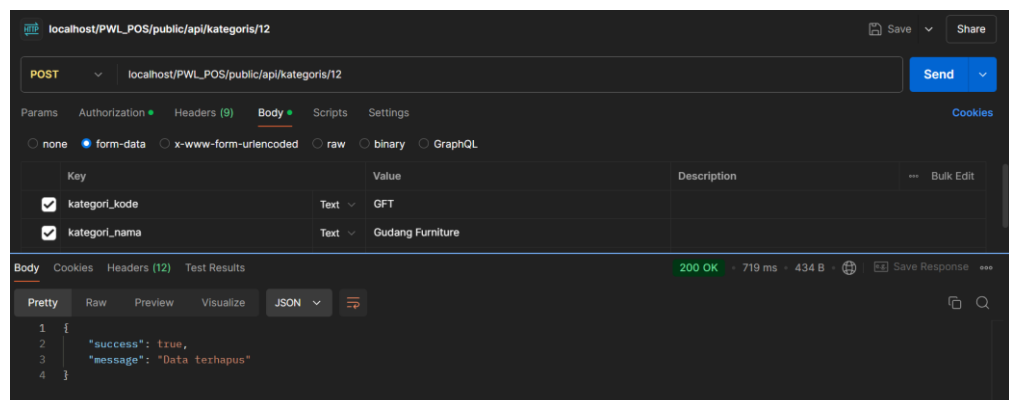


Melakukan edit kategori\_kode pada detail kategori\_id 12 yang sebelumnya “GFT” menjadi “GF”



- **DELETE**

Menghapus data kategori\_id 12



				kategori_id	kategori_kode	kategori_nama	created_at	updated_at
<input type="checkbox"/>	Edit	Copy	Delete	1	EK	Elektronik	NULL	NULL
<input type="checkbox"/>	Edit	Copy	Delete	2	KC	Kecantikan	NULL	NULL
<input type="checkbox"/>	Edit	Copy	Delete	3	OB	Obat-obatan	NULL	NULL
<input type="checkbox"/>	Edit	Copy	Delete	4	MM	Makanan dan Minuman	NULL	NULL
<input type="checkbox"/>	Edit	Copy	Delete	5	PK	Pakaian	NULL	NULL

3. **Tabel m\_barang**

**Kode Program**

- **BarangController.php**



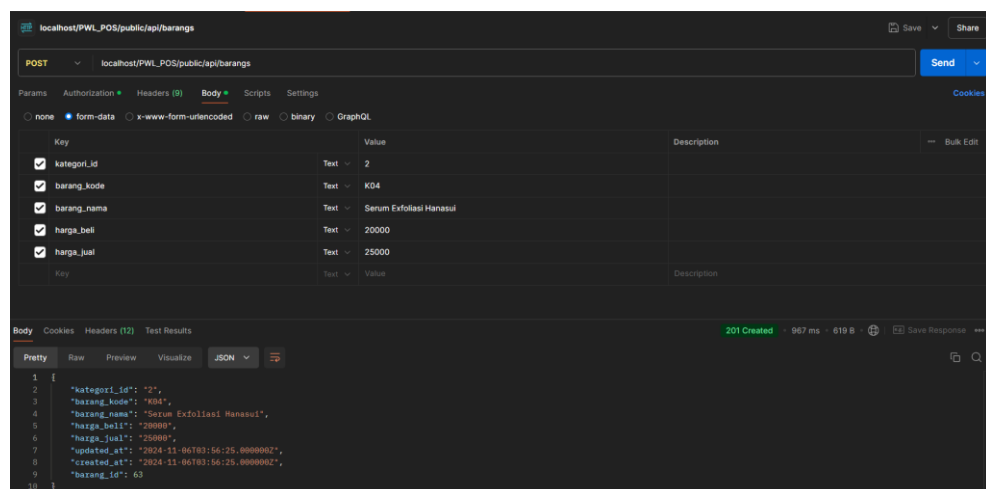
```
1 <?php
2
3 namespace App\Http\Controllers\Api;
4
5 use App\Http\Controllers\Controller;
6 use Illuminate\Http\Request;
7 use App\Models\BarangModel;
8
9 class BarangController extends Controller
10 {
11     public function index()
12     {
13         return BarangModel::all();
14     }
15
16     public function store(Request $request)
17     {
18         $barang = BarangModel::create($request->all());
19         return response()->json($barang, 201);
20     }
21
22     public function show($id)
23     {
24         $barang = BarangModel::findOrFail($id); // findOrFail akan mencari data dengan ID dan mengembalikan 404 jika tidak ditemukan
25         return response()->json($barang, 200);
26     }
27
28     // Update data berdasarkan ID yang diberikan
29     public function update(Request $request, $id)
30     {
31         $barang = BarangModel::findOrFail($id);
32         $barang->update($request->all());
33         return response()->json($barang, 200); // Mengembalikan data setelah diupdate
34     }
35
36     public function destroy(BarangModel $barang)
37     {
38         $barang->delete();
39         return response()->json([
40             'success' => true,
41             'message' => "Data terhapus",
42         ]);
43     }
44 }
```

- **Api.php**

```
// Barang
Route::get(uri: 'barangs', action: [BarangController::class, 'index']);
Route::post(uri: 'barangs', action: [BarangController::class, 'store']);
Route::get(uri: 'barangs/{barang}', action: [BarangController::class, 'show']);
Route::put(uri: 'barangs/{barang}', action: [BarangController::class, 'update']);
Route::delete(uri: 'barangs/{barang}', action: [BarangController::class, 'destroy']);
```

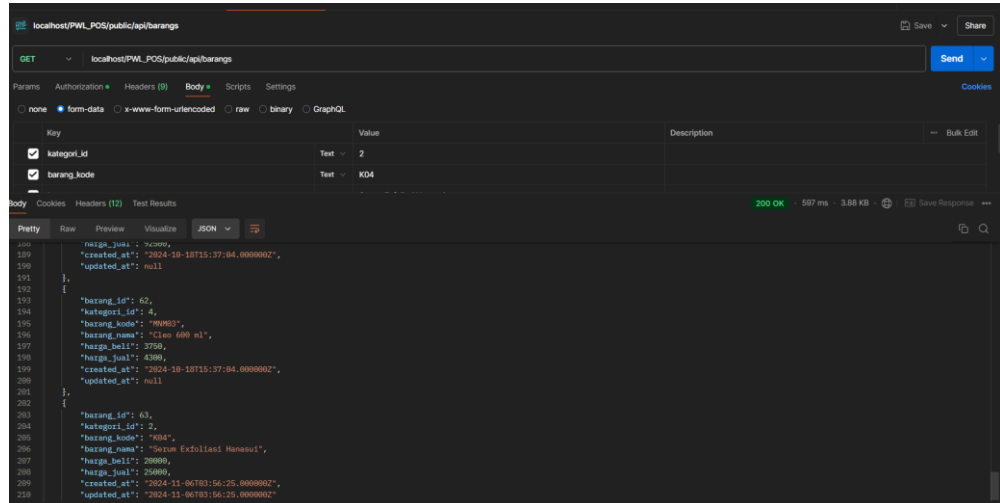
## Postman

- **POST**



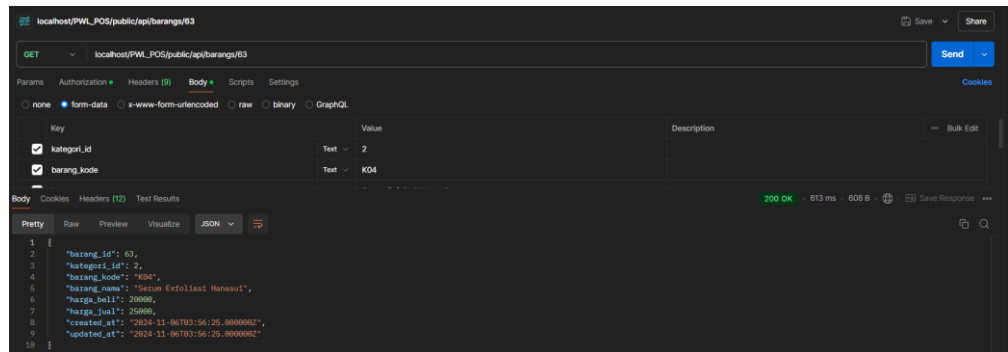


- **GET**



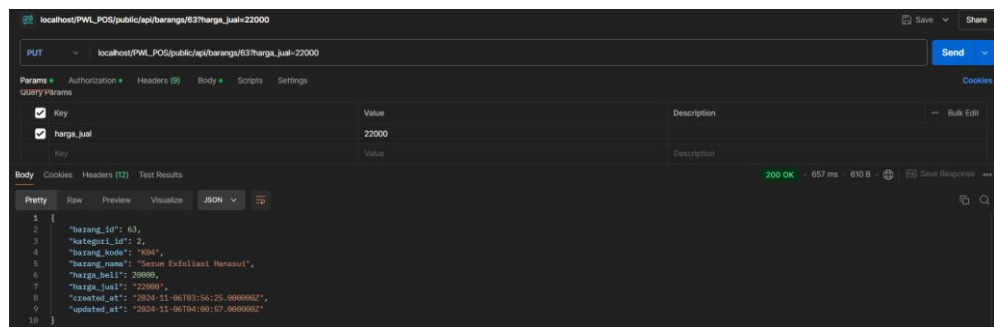
- **DETAIL**

Menampilkan detail barang\_id 63



- **PUT**

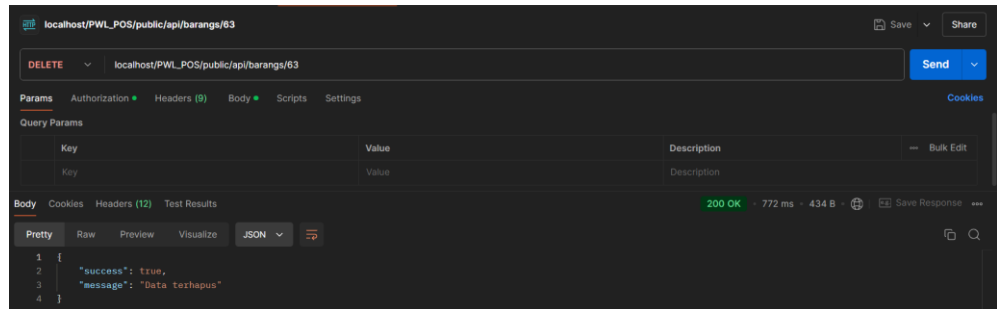
Melakukan edit harga\_jual pada detail barang\_id 63 yang sebelumnya “25000” menjadi “22000”







- **DELETE**



<input type="checkbox"/>				58	4	SBK03	Telur Omega (10 butir)	22000	25000	2024-10-18 15:37:04	NULL
<input type="checkbox"/>				59	4	SNK03	Sari roti	11500	12500	2024-10-18 15:37:04	NULL
<input type="checkbox"/>				60	2	MND03	Shampo pantene	17500	18500	2024-10-18 15:37:04	NULL
<input type="checkbox"/>				61	5	BAY03	Baju bayi 2 th	89000	92500	2024-10-18 15:37:04	NULL
<input type="checkbox"/>				62	4	MNM03	Cleo 600 ml	3750	4300	2024-10-18 15:37:04	NULL

☐ Check all    With selected: Edit   Copy   Delete   Export