



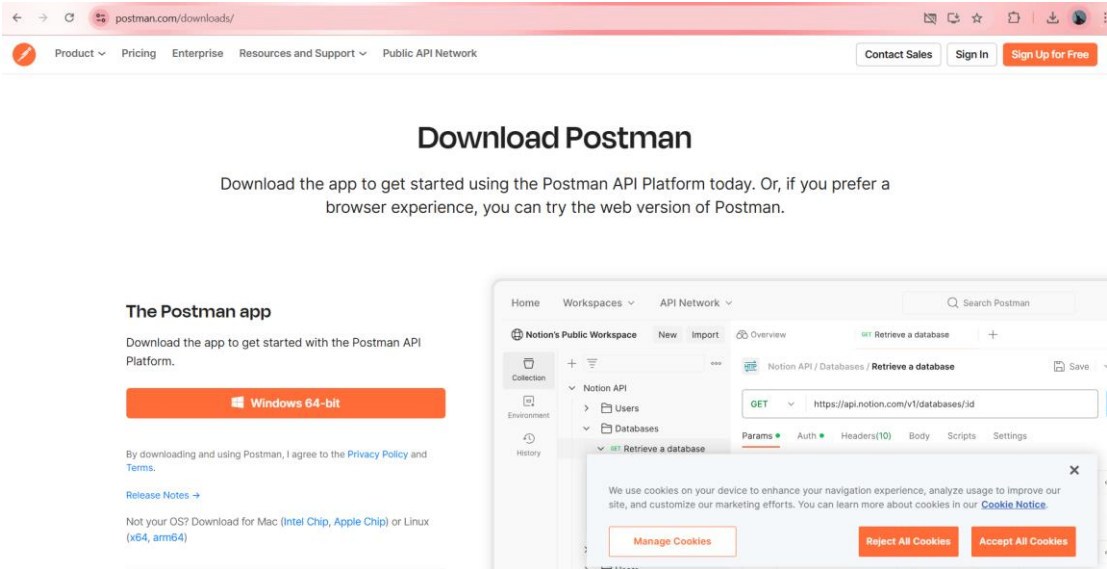
Mata Kuliah : Pemrograman Web Lanjut (PWL)  
Program Studi : D4 – Sistem Informasi Bisnis  
Semester : 5

Kelas : SIB  
NIM : 2241760007  
Nama : Anaradi Octa Lavechia  
Jobsheet Ke- : 10 (sepuluh)

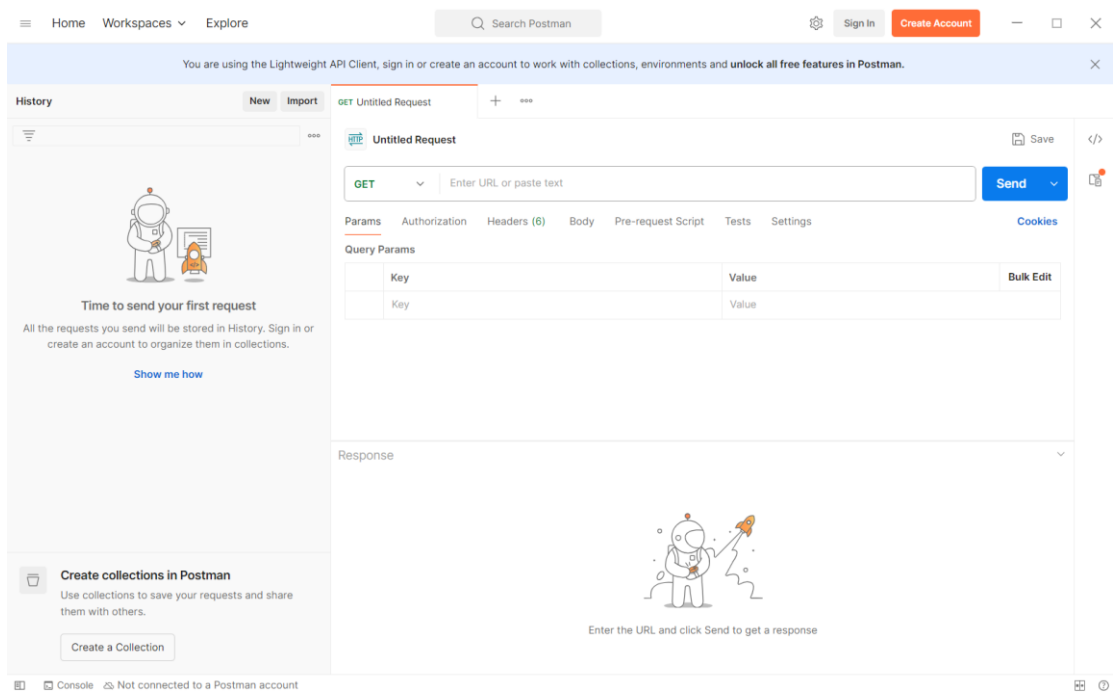
## Laporan Jobsheet – 10

### RESTFUL API

#### Praktikum 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>



	 <p>The screenshot shows the Postman web interface. On the left, there's a sidebar with 'History' and 'Collections'. The main area displays a 'GET' request for an 'Untitled Request'. The 'Query Params' section is empty. The 'Response' section shows a cartoon character and the text 'Enter the URL and click Send to get a response'.</p>
2	<p>Lakukan instalasi JWT dengan mengetikkan perintah berikut:</p> <p>composer require tymon/jwt-auth:2.1.1</p> <pre>C:\laragon\www\POS&gt;composer require tymon/jwt-auth:2.1.1 Warning: Module "gd" is already loaded in Unknown on line 0 ./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 &gt; illuminate\Foundation\ComposerScripts::postAutoloadDump &gt; @php artisan package:discover --ansi  Warning: Module "gd" is already loaded in Unknown on line 0  [INFO] Discovering packages. barryvdh/laravel-dompdf ..... DONE laravel/sail ..... DONE laravel/sanctum ..... DONE laravel/tinker ..... DONE nesbot/carbon ..... DONE nunomaduro/collision ..... DONE nunomaduro/termwind ..... DONE spatie/laravel-ignition ..... DONE tymon/jwt-auth ..... DONE yajra/laravel-datatables-buttons ..... DONE yajra/laravel-datatables-editor ..... DONE yajra/laravel-datatables-fractal ..... DONE yajra/laravel-datatables-html ..... DONE yajra/laravel-datatables-oracle ..... DONE  93 packages you are using are looking for funding. Use the "composer fund" command to find out more! &gt; @php artisan vendor:publish --tag=laravel-assets --ansi --force  Warning: Module "gd" is already loaded in Unknown on line 0  [INFO] No publishable resources for tag [laravel-assets].  No security vulnerability advisories found. C:\laragon\www\POS&gt;</pre> <p>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>C:\laragon\www\POS&gt;php artisan vendor:publish --provider="Tymon\JWTAuth\Providers\LaravelServiceProvider"  Warning: Module "gd" is already loaded in Unknown on line 0  [INFO] Publishing assets.  Copying file [C:\laragon\www\POS\vendor\tymon\jwt-auth\config\config.php] to [C:\laragon\www\POS\config\jwt.php] ..... DONE  C:\laragon\www\POS&gt;</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> <p>php artisan jwt:secret</p> <p>Jika berhasil, maka pada file .env akan ditambahkan sebuah baris berisi nilai key JWT_SECRET.</p> <pre>C:\laragon\www\POS&gt;php artisan jwt:secret  Warning: Module "gd" is already loaded in Unknown on line 0  jwt-auth secret [qkMopeTsTN0FR4TDIzjB6NUR8ACihpbD1BX26zicbB0Tjmx1nachnBBJe2oHu37U] set successfully.  C:\laragon\www\POS&gt;</pre>



6	<p>Selanjutnya lakukan konfigurasi guard API. Buka config/auth.php. Ubah bagian 'guards' menjadi seperti berikut.</p> <pre>config &gt; auth.php 3  return [ 37 38      'guards' =&gt; [ 39          'web' =&gt; [ 40              'driver' =&gt; 'session', 41              'provider' =&gt; 'users', 42          ], 43          'api' =&gt; [ 44              'driver' =&gt; 'jwt', 45              'provider' =&gt; 'users', 46          ], 47      ], </pre>
7	<p>Kita akan menambahkan kode di model UserModel</p> <pre>app &gt; Models &gt; UserModel.php &gt; PHP Intelephense &gt; UserModel &gt; getJWTIdentifier 1  &lt;?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 Authentication; //implementasi class Authenticatable 9  use Illuminate\Foundation\Auth\User as Authenticatable; 10 use Tymon\JWTAuth\Contracts\JWTSubject; 11 12 40 references   0 implementations 13 class UserModel extends Authenticatable implements JWTSubject 14 { 15     0 references   0 overrides 16     public function getJWTIdentifier(): mixed 17     { 18         return \$this-&gt;getKey(); 19     } 20     0 references   0 overrides 21     public function getJWTCustomClaims(): array 22     { 23         return []; 24     } 25     use HasFactory; 26     0 references 27     protected \$table = 'm_user'; 28     0 references 29     protected \$primaryKey = 'user_id'; </pre>
8	<p>Berikutnya kita akan membuat controller untuk register dengan menjalankan perintah berikut.</p>



php artisan make:controller Api/RegisterController

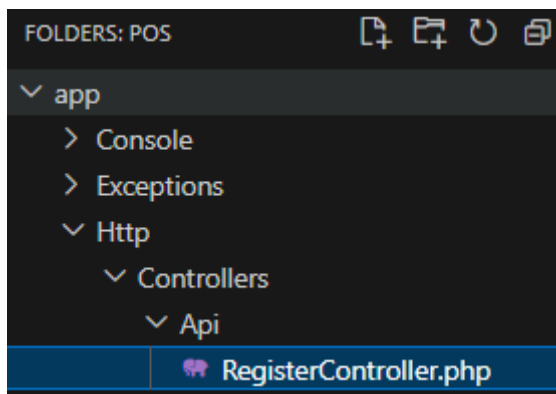
```
C:\laragon\www\POS>php artisan make:controller Api/RegisterController

Warning: Module "gd" is already loaded in Unknown on line 0

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

C:\laragon\www\POS>
```

Jika berhasil maka akan ada tambahan controller pada folder Api dengan nama RegisterController.



9

Buka file tersebut, dan ubah kode menjadi seperti berikut.

```
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  2 references | 0 implementations
8  class RegisterController extends Controller
9  {
10     0 references | 0 overrides
11     public function __invoke(Request $request): JsonResponse|mixed
12     {
13         $validator = Validator::make(data: $request->all(), rules: [
14             'username' => 'required',
15             'nama' => 'required',
16             'password' => 'required|min:5|confirmed',
17             'level_id' => 'required'
18         ]);
19         if ($validator->fails()) {
20             return response()->json(data: $validator->errors(), status: 422);
21         }
22         $user = UserModel::create(attributes: [
23             'username' => $request->username,
24             'nama' => $request->nama,
25             'password' => bcrypt(value: $request->password),
26             'level_id' => $request->level_id,
27         ]);
28         if ($user) {
29             return response()->json(data: [
30                 'success' => true,
31                 'user' => $user,
32             ], status: 201);
33         }
34         return response()->json(data: [
35             'success' => false,
36             ], status: 409);
37     }
38 }
```



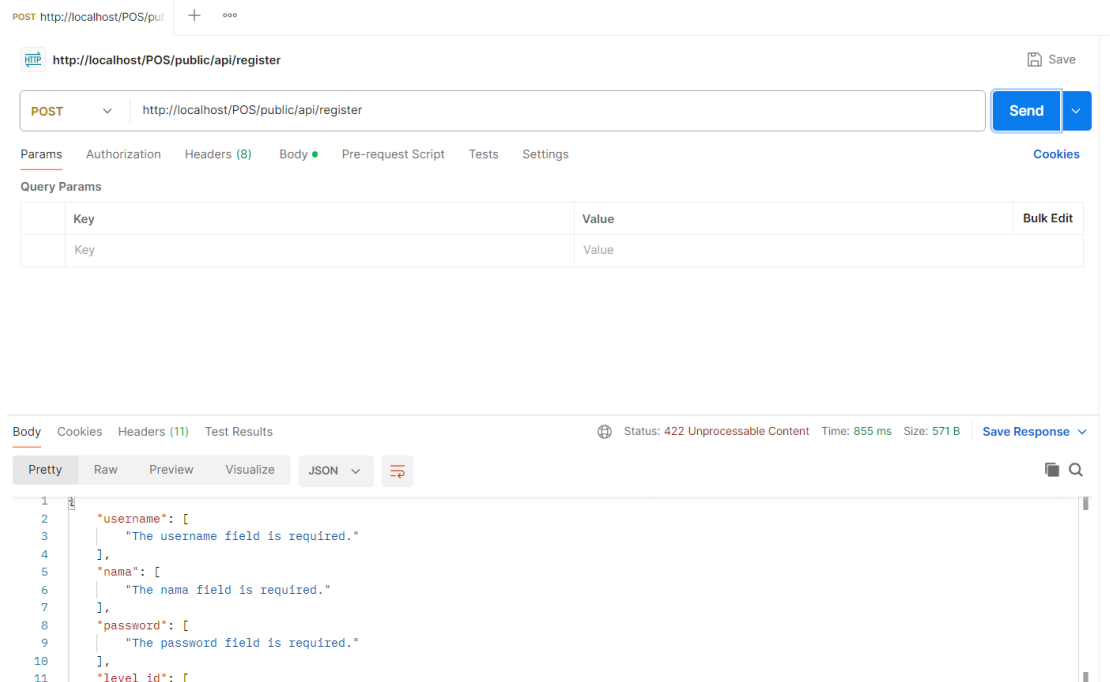
10

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(uri: '/register', action: App\Http\Controllers\Api\RegisterController::class)->name(name: 'register');
19
20 // Route::middleware('auth:sanctum')->get('/user', function (Request $request) {
21 //     return $request->user();
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.



Jika berhasil akan muncul error validasi seperti gambar di atas. Lakukan percobaan yang sama dan berikan screenshoot hasil percobaan Anda.



12

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

POST http://localhost/POS/public/api/register

Send

Params Authorization Headers (8) Body Pre-request Script Tests Settings Cookies

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

Key	Value
username	penggunasatu
nama	Pengguna 1
password	12345
password_confirmation	12345
level_id	2

Body Cookies Headers (11) Test Results Status: 422 Unprocessable Content Time: 855 ms Size: 571 B Save Response

Pretty Raw Preview Visualize JSON

```
1 {
2   "username": [
3     "The username field is required."
4   ],
5   "nama": [
6     "The nama field is required."
7   ],
8   "password": [
9     "The password field is required."
10  ],
11  "level_id": [
```

POST http://localhost/POS/public/api/register

Send

Params Authorization Headers (8) Body Pre-request Script Tests Settings Cookies

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

Key	Value
username	penggunasatu
nama	Pengguna 1
password	12345
password_confirmation	12345
level_id	2

Body Cookies Headers (11) Test Results Status: 201 Created Time: 1154 ms Size: 561 B Save Response

Pretty Raw Preview Visualize JSON

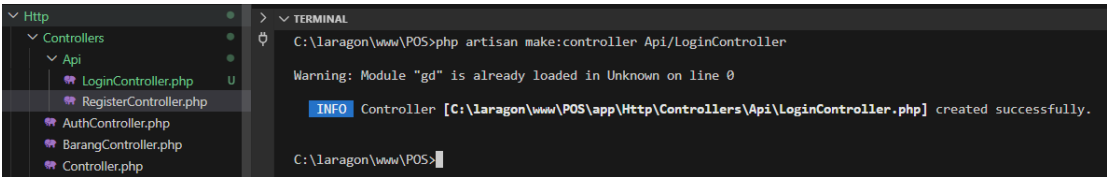
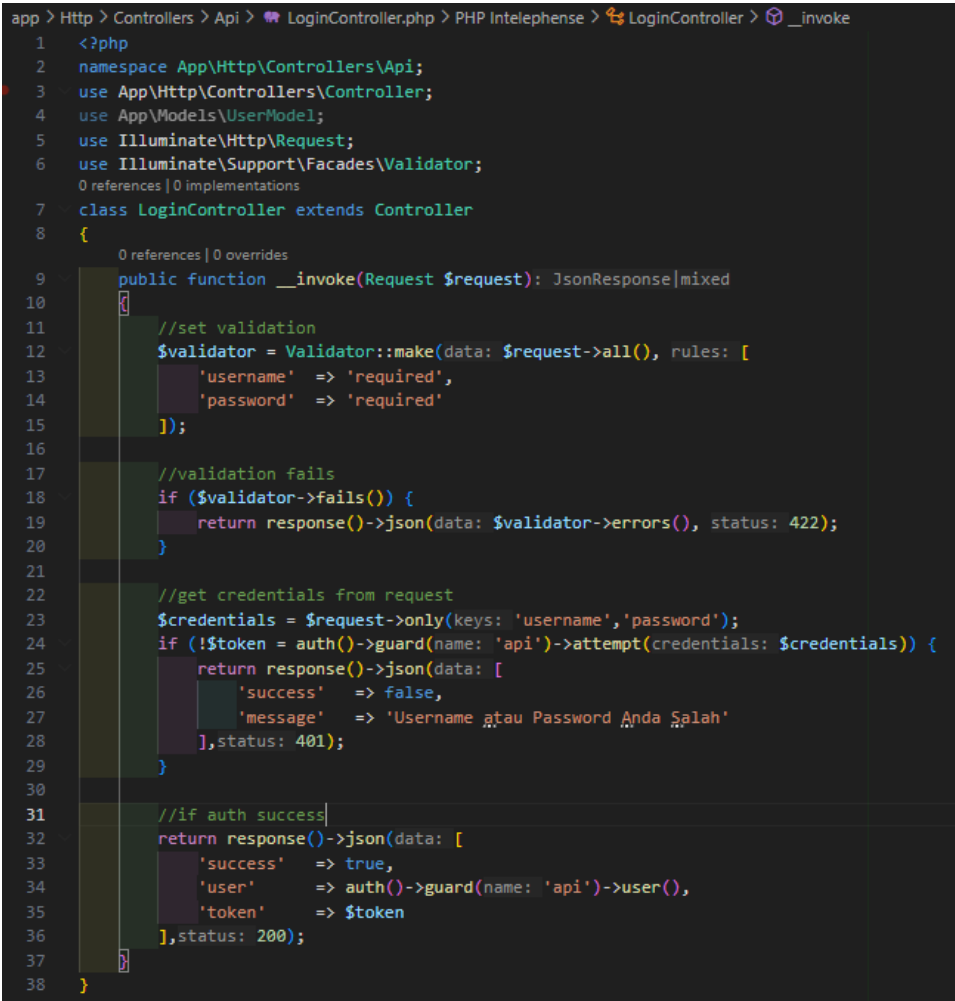
```
1 {
2   "success": true,
3   "user": {
4     "username": "penggunasatu",
5     "nama": "Pengguna 1",
6     "level_id": "2",
7     "updated_at": "2024-11-05T13:46:44.000000Z",
8     "created_at": "2024-11-05T13:46:44.000000Z",
9     "user_id": 70
10  }
11 }
```

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



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

## Praktikum 2 – Membuat RESTful API Login

Langkah	Jawaban/Deskripsi
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> 
2	<p>Buka file tersebut, dan ubah kode menjadi seperti berikut.</p> 





3

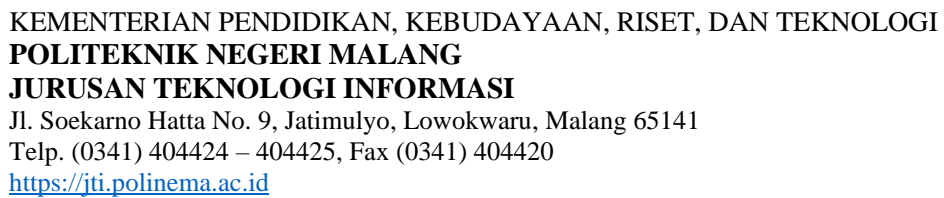
Berikutnya tambahkan route baru pada file api.php yaitu /login dan /user.

```
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(uri: '/register', action: App\Http\Controllers\Api\RegisterController::class)->name(name: 'register');
19
20 Route::post(uri: '/login', action: App\Http\Controllers\Api>LoginController::class)->name(name: 'login');
21 Route::middleware(middleware: 'auth:api')->get(uri: '/user', action: function (Request $request): mixed{
22     return $request->user();
23 });
24
25 // Route::middleware('auth:sanctum')->get('/user', function (Request $request) {
26 //     return $request->user();
27 // });
```

4

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

Jika berhasil akan muncul error validasi seperti gambar di atas. Lakukan percobaan yang sama dan berikan screenshoot hasil percobaan Anda.



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

POST http://localhost/POS/public/api/login

POST

http://localhost/POS/public/api/login

Send

Params

Authorization

Headers (8)

Body

Pre-request Script

Tests

Settings

none

form-data

x-www-form-urlencoded

raw

binary

Key	Value	...	Bulk Edit
<input checked="" type="checkbox"/> username	penggunadua		
<input checked="" type="checkbox"/> password	12345		
Key	Value		

Body

Cookies

Headers (11)

Test Results

Status: 401 Unauthorized Time: 334 ms Size: 444 B Save Response

Pretty

Raw

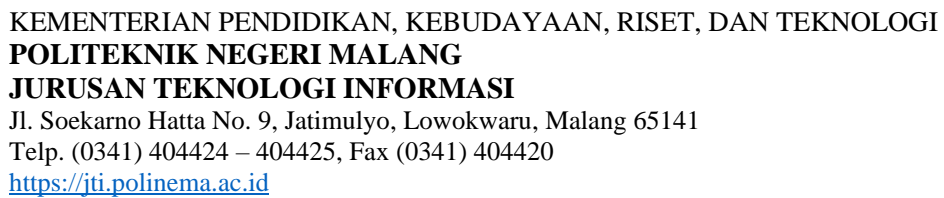
Preview

Visualize

JSON

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

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



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.

Untuk Menampilkan hasil data user ternyata diperlukan authorization dengan memasukan token yang di dapatkan saat di method POST sebelumnya, maka penginputannya di menu Authorization seperti berikut.

Hal. 11 / 31

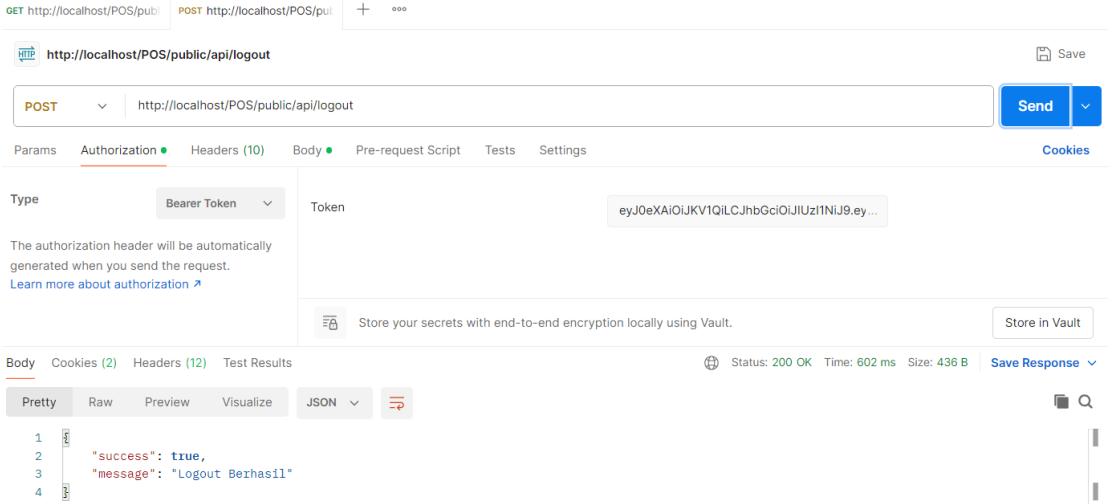


8	Lakukan commit perubahan file pada Github.
---	--

### Praktikum 3 – Membuat RESTful API Logout

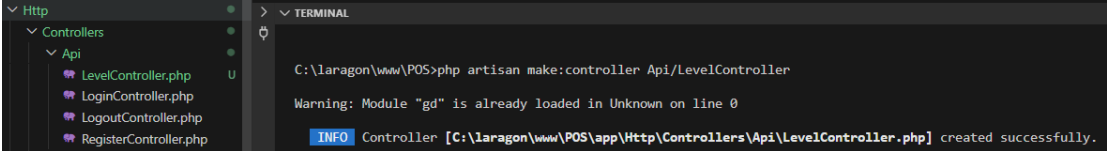
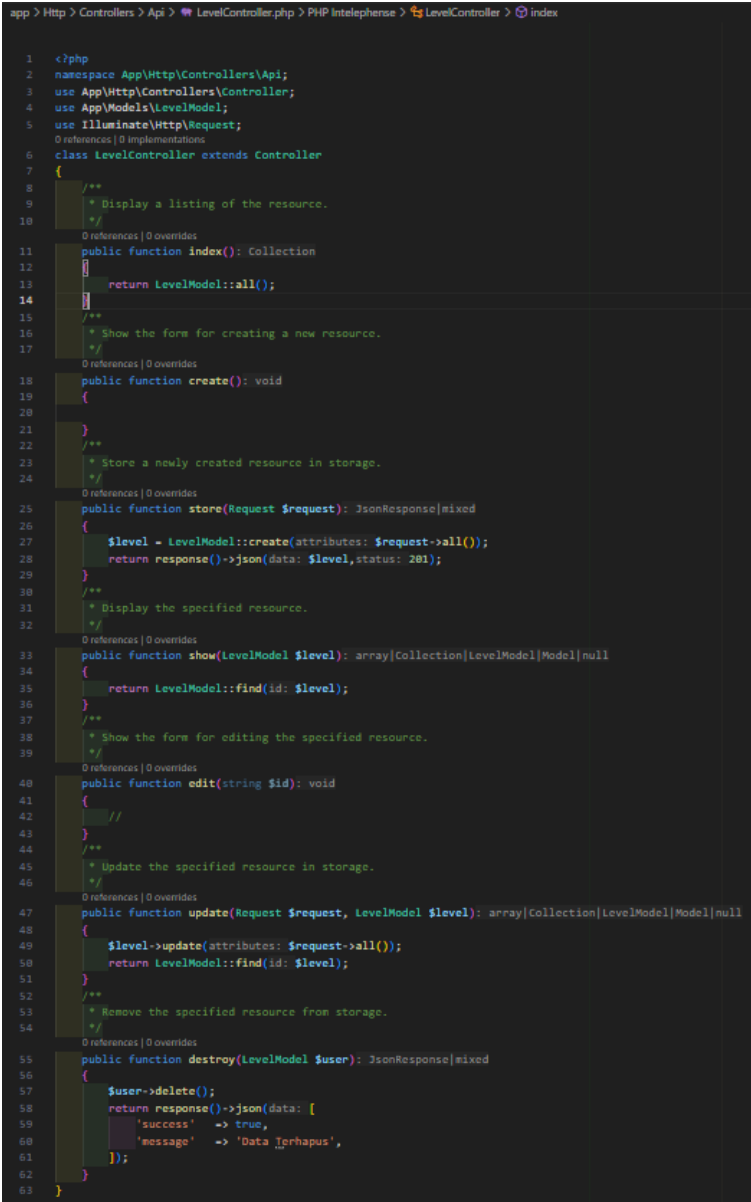
Langkah	Jawaban/Deskripsi
1	Tambahkan kode berikut pada file .env  JWT_SHOW_BLACKLIST_EXCEPTION=true
2	Buat Controller baru dengan nama LogoutController.  php artisan make:controller Api/LogoutController  
3	Buka file tersebut dan ubah kode menjadi seperti berikut.  



4	<p>Lalu kita tambahkan routes pada api.php</p> <pre>routes &gt; api.php &gt; ...  1  &lt;?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(uri: '/register', action: App\Http\Controllers\Api\RegisterController::class)-&gt;name(name: 'register'); 19 20 Route::post(uri: '/login', action: App\Http\Controllers\Api&gt;LoginController::class)-&gt;name(name: 'login'); 21 Route::middleware('auth:api')-&gt;get(uri: '/user', action: function (Request \$request): mixed { 22     return \$request-&gt;user(); 23 }); 24 25 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> 
7	<p>Lakukan commit perubahan file pada Github.</p>

## Praktikum 4 – Implementasi CRUD dalam RESTful API



Langkah	Jawaban/Deskripsi
1	<p>Pertama, buat controller untuk mengolah API pada data level.</p> <p>php artisan make:controller Api/LevelController</p> 
2	<p>Setelah berhasil, buka file tersebut dan tuliskan kode seperti berikut yang berisi fungsi CRUDnya.</p> 



3 Kemudian kita lengkapi routes pada api.php.

```
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  use App\Http\Controllers\Api\LevelController;
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(uri: '/register', action: App\Http\Controllers\Api\RegisterController::class)->name(name: 'register');
20
21 Route::post(uri: '/login', action: App\Http\Controllers\Api>LoginController::class)->name(name: 'login');
22 Route::middleware(middleware: 'auth:api')->get(uri: '/user', action: function (Request $request): mixed{
23     return $request->user();
24 });
25
26 Route::post(uri: '/logout', action: App\Http\Controllers\Api\LogoutController::class)->name(name: 'logout');
27
28 Route::get(uri: 'levels', action: [LevelController::class, 'index']);
29 Route::post(uri: 'levels', action: [LevelController::class, 'store']);
30 Route::get(uri: 'levels/{level}', action: [LevelController::class, 'show']);
31 Route::put(uri: 'levels/{level}', action: [LevelController::class, 'update']);
32 Route::delete(uri: 'levels/{level}', action: [LevelController::class, 'destroy']);
```

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

localhost/POS/public/api/user

GET localhost/POS/public/api/levels

Params Authorization Headers (6) Body Pre-request Script Tests Settings Cookies

Query Params

Key	Value	Bulk Edit
Key	Value	

Body Cookies Headers (11) Test Results

Status: 200 OK Time: 237 ms Size: 119 KB Save Response

Pretty Raw Preview Visualize JSON

```
[
  {
    "level_id": 1,
    "level_kode": "ADM",
    "level_nama": "Administrator",
    "created_at": null,
    "updated_at": null
  },
  {
    "level_id": 2,
    "level_kode": "MNG",
    "level_nama": "Manager",
    "created_at": null,
    "updated_at": null
  },
  {
    "level_id": 3,
    "level_kode": "STF",
    "level_nama": "Staf"
  }
]
```

Dari output di atas Menampilkan daftar data level yang ada dalam tabel level



5

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

POST localhost/POS/public/api/levels

localhost/POS/public/api/levels

POST localhost/POS/public/api/levels

Params Authorization Headers (9) Body Pre-request Script Tests Settings

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

Key	Value
level_kode	SPV
level_nama	Supervisor

Body Cookies (2) Headers (11) Test Results

Status: 201 Created Time: 249 ms Size: 521 B Save Response

```
1 {
2   "level_kode": "SPV",
3   "level_nama": "Supervisor",
4   "updated_at": "2024-11-05T14:48:42.000000Z",
5   "created_at": "2024-11-05T14:48:42.000000Z",
6   "level_id": 9
7 }
```

6

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

GET localhost/POS/public/api/levels

localhost/POS/public/api/levels

GET localhost/POS/public/api/levels

Params Authorization Headers (9) Body Pre-request Script Tests Settings

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

Key	Value
level_kode	SPV
level_nama	Supervisor

Body Cookies (2) Headers (11) Test Results

Status: 200 OK Time: 289 ms Size: 1.19 KB Save Response

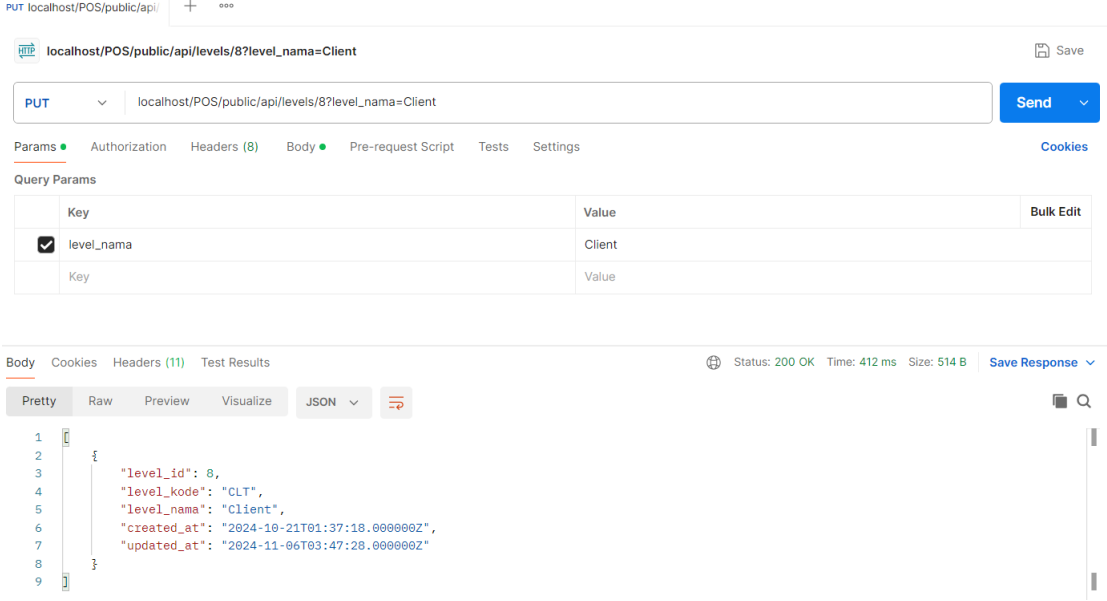
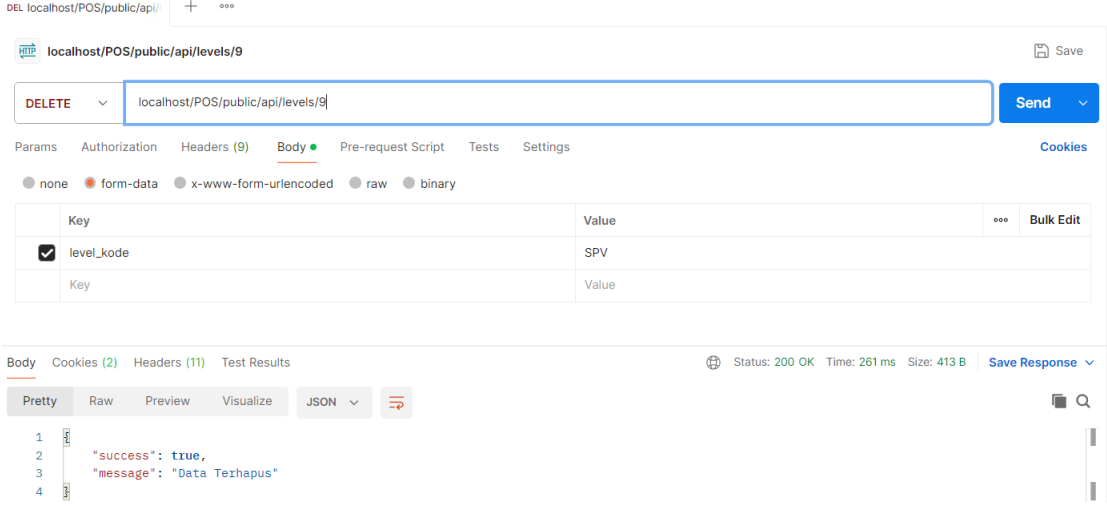
```
38 [
39   {
40     "level_id": 8,
41     "level_kode": "CLT",
42     "level_nama": "Clientt",
43     "created_at": "2024-10-21T01:37:18.000000Z",
44     "updated_at": "2024-10-23T07:08:15.000000Z"
45   },
46   {
47     "level_id": 9,
48     "level_kode": "SPV",
49     "level_nama": "Supervisor",
50     "created_at": "2024-11-05T14:48:42.000000Z",
51     "updated_at": "2024-11-05T14:48:42.000000Z"
52   }
53 ]
```

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





	<p>Param.</p> 
8	<p>Terakhir lakukan percobaan hapus data. Jelaskan dan berikan screenshoot hasil percobaan Anda.</p>  <p>Berdasarkan pemahaman saya dari hasil praktikum, jika data yang berdasarkan data level_kode ditemukan dan dapat terhapus maka akan menghasilkan output true Data terhapus namun jika tidak maka output akan false.</p>
9	Lakukan commit perubahan file pada Github.

## Tugas!



Implementasikan CRUD API pada tabel lainnya yaitu tabel m\_user, m\_kategori, dan m\_barang

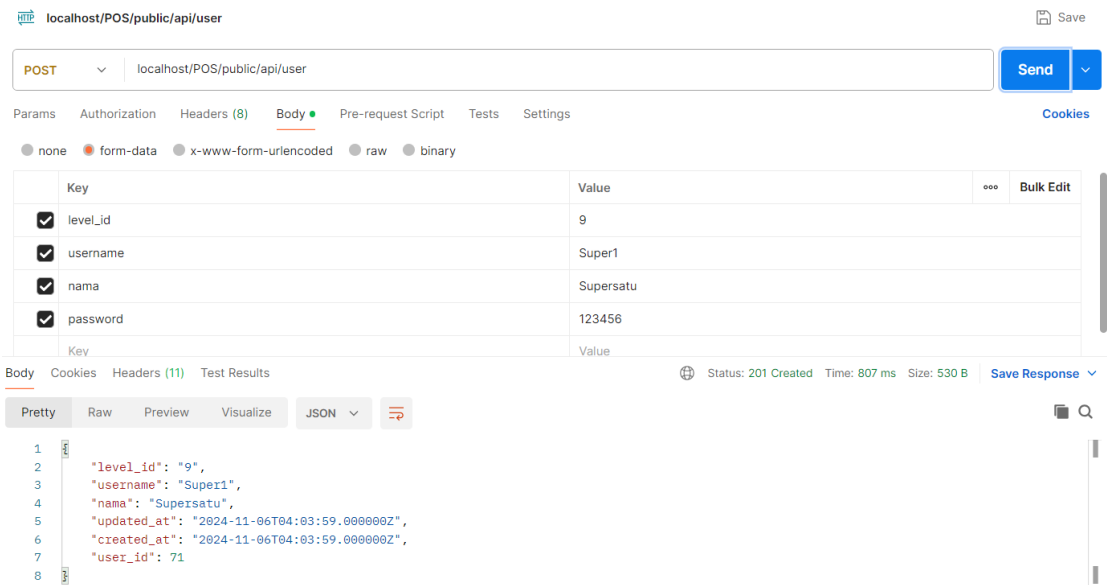
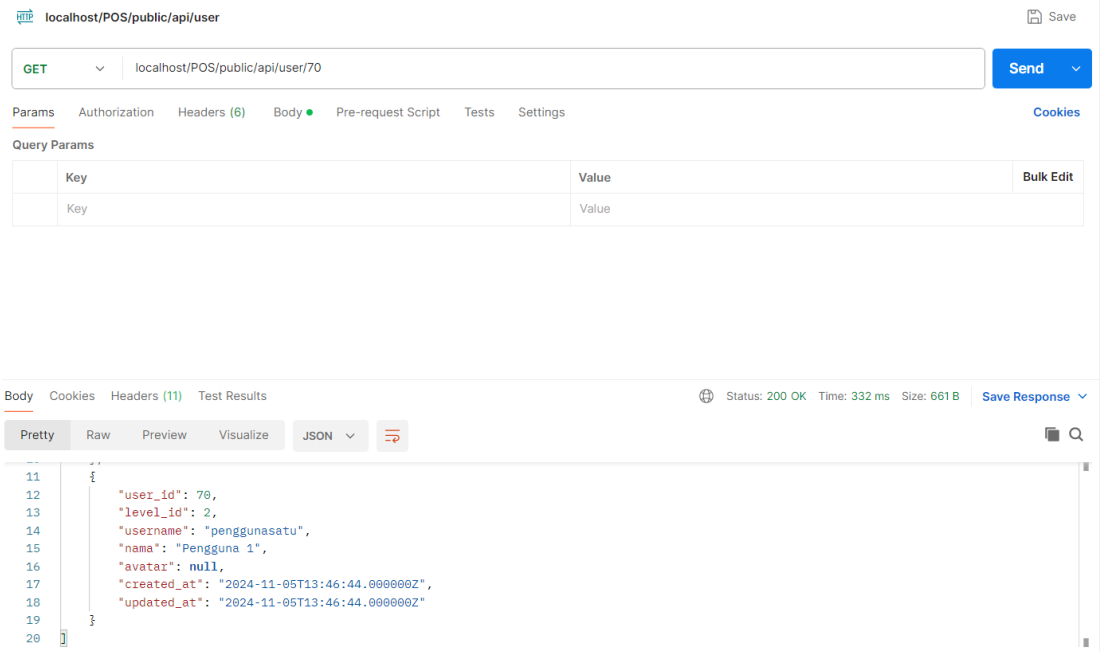
### Soal Praktikum – m\_user

Langkah	Jawaban/Deskripsi
1	<p>Pertama, buat controller untuk mengolah API pada data level.</p> <p>php artisan make:controller Api/UserController</p> <pre>C:\laragon\www\POS&gt;php artisan make:controller Api/UserController  Warning: Module "gd" is already loaded in Unknown on line 0  INFO Controller [C:\laragon\www\POS\app\Http\Controllers\Api\UserController.php] created successfully.  C:\laragon\www\POS&gt;</pre>
2	<p>Setelah berhasil, buka file tersebut dan tuliskan kode seperti berikut yang berisi fungsi CRUDnya.</p> <pre>app &gt; Http &gt; Controllers &gt; Api &gt; UserUserController.php &gt; ... 1 &lt;?php 2 3 namespace App\Http\Controllers\Api; 4 5 use App\Http\Controllers\Controller; 6 use App\Models\UserModel; 7 use Illuminate\Http\Request; 8 9 class UserController extends Controller 10 { 11     /** 12      * Display a listing of the resource. 13      */ 14     public function index(): Collection 15     { 16         return UserModel::all(); 17     } 18     /** 19      * Show the form for creating a new resource. 20      */ 21     public function create(): void 22     { 23         // 24     } 25     /** 26      * Store a newly created resource in storage. 27      */ 28     public function store(Request \$request): JsonResponse mixed 29     { 30         \$user = UserModel::create(attributes: \$request-&gt;all()); 31         return response()-&gt;json(data: \$user, status: 201); 32     } 33     /** 34      * Display the specified resource. 35      */ 36     public function show(UserModel \$user): array(Collection Model UserModel null) 37     { 38         return UserModel::find(id: \$user); 39     } 40     /** 41      * Show the form for editing the specified resource. 42      */ 43     public function edit(string \$id): void 44     { 45         // 46     } 47     /** 48      * Update the specified resource in storage. 49      */ 50     public function update(Request \$request, UserModel \$user): array(Collection Model UserModel null) 51     { 52         \$user-&gt;update(attributes: \$request-&gt;all()); 53         return UserModel::find(id: \$user); 54     } 55     /** 56      * Remove the specified resource from storage. 57      */ 58     public function destroy(UserModel \$user): JsonResponse mixed 59     { 60         \$user-&gt;delete(); 61         return response()-&gt;json(data: [ 62             'success' =&gt; true, 63             'message' =&gt; 'Data Terhapus', 64         ]); 65     } 66 }</pre>

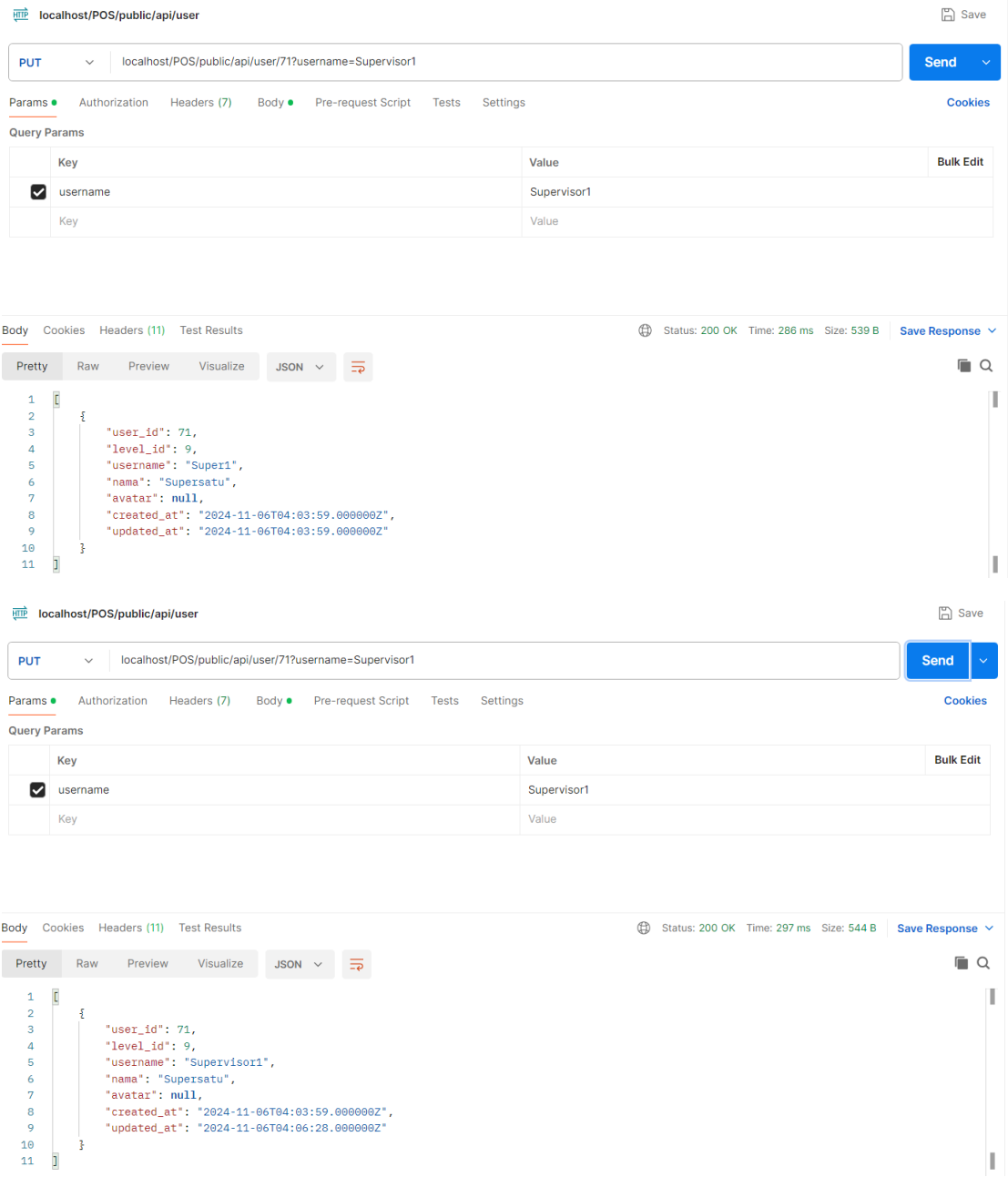


	<p>Kemudian kita lengkapi routes pada api.php.</p> <pre>routes &gt; api.php &gt; ... 35 //m_user 36 Route::get(uri: 'user', action: [UserController::class, 'index']); 37 Route::post(uri: 'user', action: [UserController::class, 'store']); 38 Route::get(uri: 'user/{user}', action: [UserController::class, 'show']); 39 Route::put(uri: 'user/{user}', action: [UserController::class, 'update']); 40 Route::delete(uri: 'user/{user}', action: [UserController::class, 'destroy']); 41</pre>
3	<p>Jika sudah. Lakukan uji coba API mulai dari fungsi untuk menampilkan data. Gunakan URL: localhost/PWL_POS-main/public/api/user dan method GET. Jelaskan dan berikan screenshoot hasil percobaan Anda.</p>  <p>Dari output di atas Menampilkan seluruh daftar data user yang ada dalam tabel user</p>
4	<p>Kemudian, lakukan percobaan penambahan data dengan URL : localhost/PWL_POS main/public/api/user dan method POST seperti di bawah ini.</p>

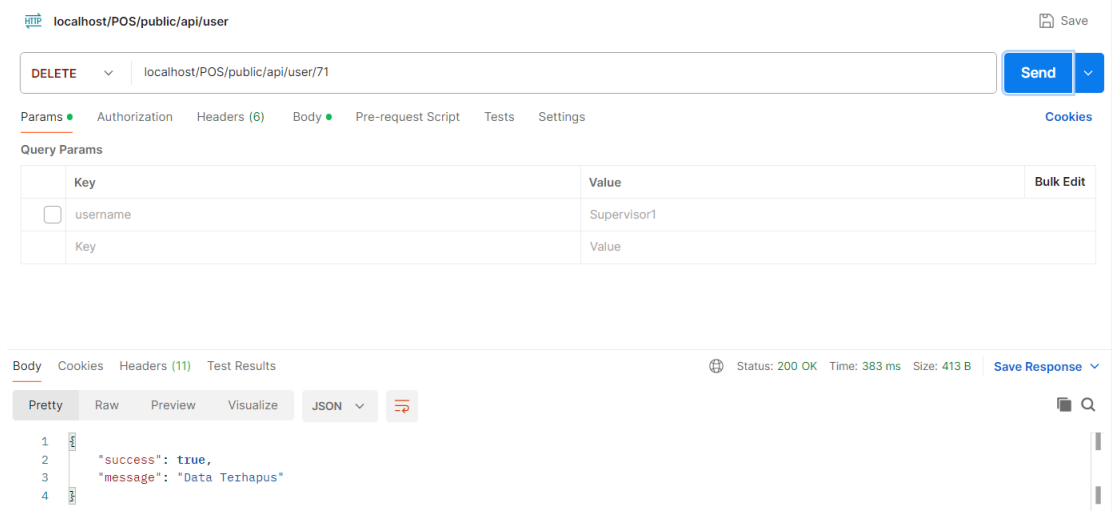


	
5	<p>Berikutnya lakukan percobaan menampilkan detail data. Jelaskan dan berikan screenshoot hasil percobaan Anda.</p> 
6	<p>Jika sudah, kita coba untuk melakukan edit data menggunakan localhost/PWL_POS main/public/api/user/{id} dan method PUT. Isikan data yang ingin diubah pada tab Param.</p>



	 <p>The screenshot displays a REST client interface with the following details:</p> <ul style="list-style-type: none"><li><b>URL:</b> localhost/POS/public/api/user</li><li><b>Method:</b> PUT</li><li><b>Query Params:</b> localhost/POS/public/api/user/71?username=Supervisor1</li><li><b>Body:</b> A JSON object containing user information: <pre>{  "user_id": 71,  "level_id": 9,  "username": "Super1",  "nama": "Supersatu",  "avatar": null,  "created_at": "2024-11-06T04:03:59.000000Z",  "updated_at": "2024-11-06T04:03:59.000000Z"}</pre></li><li><b>Status:</b> 200 OK, Time: 286 ms, Size: 539 B</li></ul>
7	Terakhir lakukan percobaan hapus data. Jelaskan dan berikan screenshoot hasil percobaan Anda.



	
8	Lakukan commit perubahan file pada Github.

### Soal Praktikum – m\_kategori

Langkah	Jawaban/Deskripsi
1	<p>Pertama, buat controller untuk mengolah API pada data level.</p> <p>php artisan make:controller Api/KategoriController</p> <pre>C:\laragon\www\POS&gt;php artisan make:controller Api/KategoriController Warning: Module "gd" is already loaded in Unknown on line 0 INFO Controller [C:\laragon\www\POS\app\Http\Controllers\Api\KategoriController.php] created successfully. C:\laragon\www\POS&gt;</pre>
2	<p>Setelah berhasil, buka file tersebut dan tuliskan kode seperti berikut yang berisi fungsi CRUDnya.</p>



	<pre>app &gt; Http &gt; Controllers &gt; Api &gt; KategoriController.php &gt; ... 1 &lt;?php 2 3 namespace App\Http\Controllers\Api; 4 5 use App\Http\Controllers\Controller; 6 use App\Models\KategoriModel; 7 use Illuminate\Http\Request; 8 9 /** 10  * @references   O implementations 11  */ 12 class KategoriController extends Controller 13 { 14     /** 15      * Display a listing of the resource. 16      */ 17     /** @references   O overrides */ 18     public function index(): Collection 19     { 20         return KategoriModel::all(); 21     } 22     /** 23      * Show the form for creating a new resource. 24      */ 25     /** @references   O overrides */ 26     public function create(): void 27     { 28         // 29     } 30     /** 31      * Store a newly created resource in storage. 32      */ 33     /** @references   O overrides */ 34     public function store(Request \$request): JsonResponse mixed 35     { 36         \$kategori = KategoriModel::create(attributes: \$request-&gt;all()); 37         return response()-&gt;json(data: \$kategori, status: 201); 38     } 39     /** 40      * Display the specified resource. 41      */ 42     /** @references   O overrides */ 43     public function show(KategoriModel \$kategori): array Collection KategoriModel Model null 44     { 45         return KategoriModel::find(id: \$kategori); 46     } 47     /** 48      * Show the form for editing the specified resource. 49      */ 50     /** @references   O overrides */ 51     public function edit(string \$id): void 52     { 53         // 54     } 55     /** 56      * Update the specified resource in storage. 57      */ 58     /** @references   O overrides */ 59     public function update(Request \$request, KategoriModel \$kategori): array Collection KategoriModel Model null 60     { 61         \$kategori-&gt;update(attributes: \$request-&gt;all()); 62         return KategoriModel::find(id: \$kategori); 63     } 64     /** 65      * Remove the specified resource from storage. 66      */ 67     /** @references   O overrides */ 68     public function destroy(KategoriModel \$kategori): JsonResponse mixed 69     { 70         \$kategori-&gt;delete(); 71         return response()-&gt;json(data: [ 72             'success' =&gt; true, 73             'message' =&gt; 'Data Terhapus', 74         ]); 75     } 76 }</pre>
3	<p>Kemudian kita lengkapi routes pada api.php.</p> <pre>routes &gt; api.php &gt; ... 43 // m_kategori 44 Route::get(uri: 'kategori', action: [KategoriController::class, 'index']); 45 Route::post(uri: 'kategori', action: [KategoriController::class, 'store']); 46 Route::get(uri: 'kategori/{kategori}', action: [KategoriController::class, 'show']); 47 Route::put(uri: 'kategori/{kategori}', action: [KategoriController::class, 'update']); 48 Route::delete(uri: 'kategori/{kategori}', action: [KategoriController::class, 'destroy']); 49</pre>
4	<p>Jika sudah. Lakukan uji coba API mulai dari fungsi untuk menampilkan data. Gunakan</p>



URL: localhost/ POS /public/api/kategori dan method GET. Jelaskan dan berikan screenshot hasil percobaan Anda.

localhost/POS/public/api/user

GET localhost/POS/public/api/kategori

Send

Params Authorization Headers (6) Body Pre-request Script Tests Settings Cookies

Query Params

Key	Value	Bulk Edit
<input type="checkbox"/> username	Supervisor1	
Key	Value	

Body Cookies Headers (11) Test Results

Status: 200 OK Time: 282 ms Size: 1.66 KB Save Response

Pretty Raw Preview Visualize JSON

```
1 {
2   {
3     "kategori_id": 1,
4     "kategori_kode": "fmb",
5     "kategori_nama": "Food and Beverage",
6     "created_at": null,
7     "updated_at": null
8   },
9   {
10    "kategori_id": 2,
11    "kategori_kode": "bnh",
12    "kategori_nama": "Beauty and Health",
13    "created_at": null,
14    "updated_at": null
15  },
16 }
```

Dari output di atas Menampilkan seluruh daftar data kategori yang ada dalam tabel kategori

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

localhost/POS/public/api/user

POST localhost/POS/public/api/kategori

Send

Params Authorization Headers (8) Body Pre-request Script Tests Settings Cookies

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

Key	Value	Bulk Edit
<input checked="" type="checkbox"/> kategori_kode	SPRT	
<input checked="" type="checkbox"/> kategori_nama	SPORT	
Key	Value	

Body Cookies Headers (11) Test Results

Status: 201 Created Time: 327 ms Size: 527 B Save Response

Pretty Raw Preview Visualize JSON

```
1 {
2   "kategori_kode": "SPRT",
3   "kategori_nama": "SPORT",
4   "updated_at": "2024-11-06T04:19:37.000000Z",
5   "created_at": "2024-11-06T04:19:37.000000Z",
6   "kategori_id": 16
7 }
```

6 Berikutnya lakukan percobaan menampilkan detail data. Jelaskan dan berikan





screenshoot hasil percobaan Anda.

localhost/POS/public/api/kategori?kategori\_nama=Sembako

GET localhost/POS/public/api/kategori?kategori\_nama=Sembako

Params Authorization Headers (8) Body Pre-request Script Tests Settings Cookies

Query Params

Key	Value	Bulk Edit
<input checked="" type="checkbox"/> kategori_nama	Sembako	
Key	Value	

Body Cookies Headers (11) Test Results Status: 200 OK Time: 212 ms Size: 1.96 KB Save Response

Pretty Raw Preview Visualize JSON

```
37 {
38   "kategori_id": 9,
39   "kategori_kode": "SBK",
40   "kategori_nama": "Sembako",
41   "created_at": "2024-10-18T04:37:37.000000Z",
42   "updated_at": "2024-10-18T04:37:37.000000Z"
43 },
```

7

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

localhost/POS/public/api/kategori?kategori\_nama=Sembako

PUT localhost/POS/public/api/kategori/9

Params Authorization Headers (8) Body Pre-request Script Tests Settings Cookies

Query Params

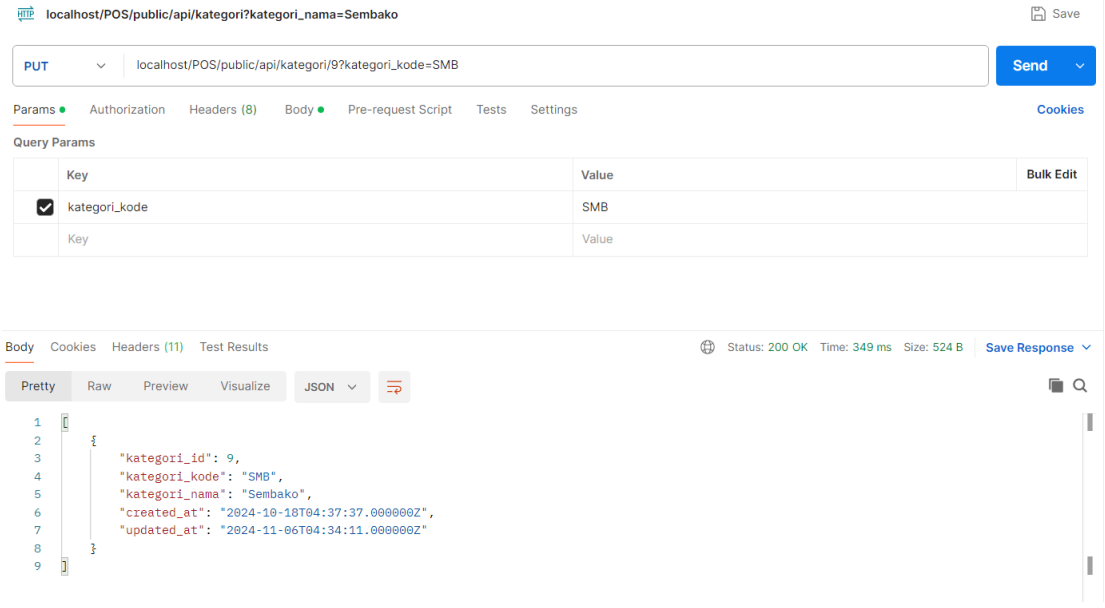
Key	Value	Bulk Edit
Key	Value	

Body Cookies Headers (11) Test Results Status: 200 OK Time: 328 ms Size: 524 B Save Response

Pretty Raw Preview Visualize JSON

```
1 {
2   "kategori_id": 9,
3   "kategori_kode": "SBK",
4   "kategori_nama": "Sembako",
5   "created_at": "2024-10-18T04:37:37.000000Z",
6   "updated_at": "2024-10-18T04:37:37.000000Z"
7 }
8
9
```



	
8	<p>Terakhir lakukan percobaan hapus data. Jelaskan dan berikan screenshoot hasil percobaan Anda.</p> 
9	Lakukan commit perubahan file pada Github.

### Soal Praktikum – m\_barang

Langkah	Jawaban/Deskripsi
1	Pertama, buat controller untuk mengolah API pada data level.



	<p>php artisan make:controller Api/BarangController</p> <pre>C:\laragon\www\POS&gt;php artisan make:controller Api/BarangController  Warning: Module "gd" is already loaded in Unknown on line 0  INFO Controller [C:\laragon\www\POS\app\Http\Controllers\Api\BarangController.php] created successfully.  C:\laragon\www\POS&gt;</pre>
2	<p>Setelah berhasil, buka file tersebut dan tuliskan kode seperti berikut yang berisi fungsi CRUDnya.</p> <pre>app &gt; http &gt; Controllers &gt; Api &gt; BarangController.php &gt; PHP Intellisense &gt; % BarangController &gt; @ create 1 &lt;?php 2 3 namespace App\Http\Controllers\Api; 4 5 use App\Http\Controllers\Controller; 6 use App\Models\BarangModel; 7 use Illuminate\Http\Request; 8 9 /** 10  * @return \Illuminate\Contracts\Foundation\Application, \Illuminate\Http\Response 11  */ 12 /** 13  * Display a listing of the resource. 14  */ 15 /** 16  * @return \Illuminate\Http\Response 17  */ 18 public function index(): Collection 19 { 20     return BarangModel::all(); 21 } 22 23 /** 24  * Show the form for creating a new resource. 25  */ 26 /** 27  * @return \Illuminate\Http\Response 28  */ 29 public function create(): void 30 { 31     // 32 } 33 34 /** 35  * Store a newly created resource in storage. 36  */ 37 /** 38  * @return \Illuminate\Http\Response 39  */ 40 public function store(Request \$request): JsonResponse void 41 { 42     \$barang = BarangModel::create(attributes: \$request-&gt;all()); 43     return response()-&gt;json(data: \$barang, status: 201); 44 } 45 46 /** 47  * Display the specified resource. 48  */ 49 /** 50  * @return \Illuminate\Http\Response 51  */ 52 public function show(BarangModel \$barang): array BarangModel Collection Model null 53 { 54     return BarangModel::find(\$barang); 55 } 56 57 /** 58  * Show the form for editing the specified resource. 59  */ 60 /** 61  * @return \Illuminate\Http\Response 62  */ 63 public function edit(string \$id): void 64 { 65     // 66 } 67 68 /** 69  * Update the specified resource in storage. 70  */ 71 /** 72  * @return \Illuminate\Http\Response 73  */ 74 public function update(Request \$request, BarangModel \$barang): array BarangModel Collection Model null 75 { 76     \$barang-&gt;update(attributes: \$request-&gt;all()); 77     return BarangModel::find(\$barang); 78 } 79 80 /** 81  * Remove the specified resource from storage. 82  */ 83 /** 84  * @return \Illuminate\Http\Response 85  */ 86 public function destroy(BarangModel \$barang): JsonResponse void 87 { 88     \$barang-&gt;delete(); 89     return response()-&gt;json(data: [ 90         'success' =&gt; true, 91         'message' =&gt; 'Data Terhapus', 92     ]); 93 } 94 }</pre>
3	<p>Kemudian kita lengkapi routes pada api.php.</p> <pre>routes &gt; api.php &gt; ... 51 // m_barang 52 Route::get(uri: 'barang', action: [BarangController::class, 'index']); 53 Route::post(uri: 'barang', action: [BarangController::class, 'store']); 54 Route::get(uri: 'barang/{barang}', action: [BarangController::class, 'show']); 55 Route::put(uri: 'barang/{barang}', action: [BarangController::class, 'update']); 56 Route::delete(uri: 'barang/{barang}', action: [BarangController::class, 'destroy']);</pre>



4

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

localhost/POS/public/api/kategori?kategori\_nama=Sembako

GET localhost/POS/public/api/barang

Params Authorization Headers (8) Body Pre-request Script Tests Settings Cookies

Query Params

Key	Value	Bulk Edit
Key	Value	

Body Cookies Headers (11) Test Results

Status: 200 OK Time: 367 ms Size: 3.66 KB Save Response

Pretty Raw Preview Visualize JSON

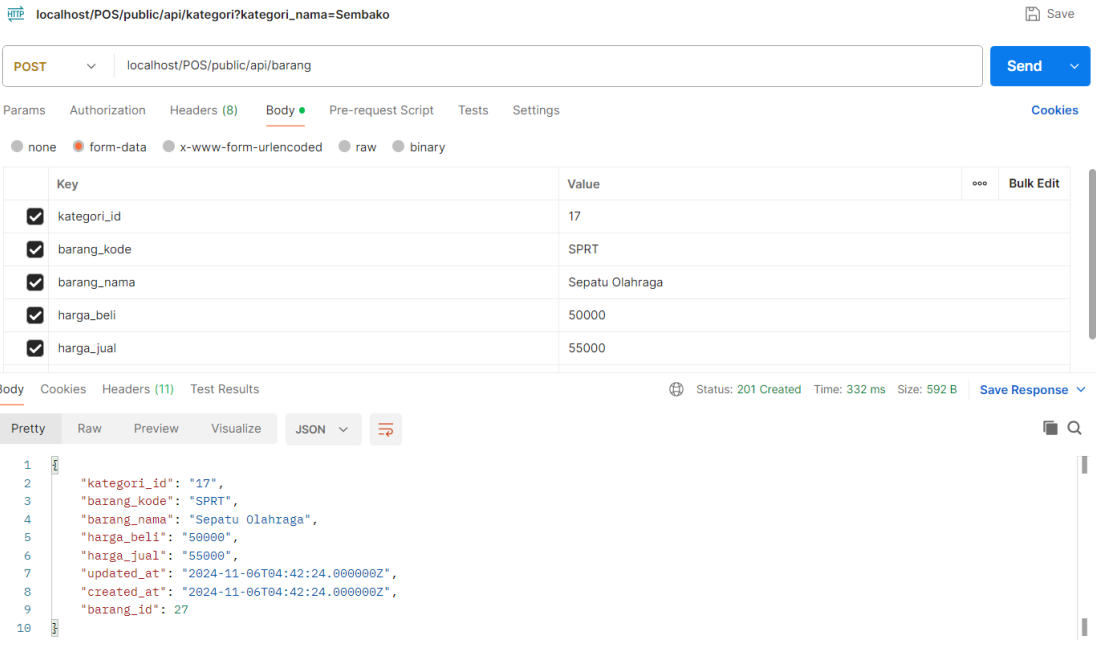
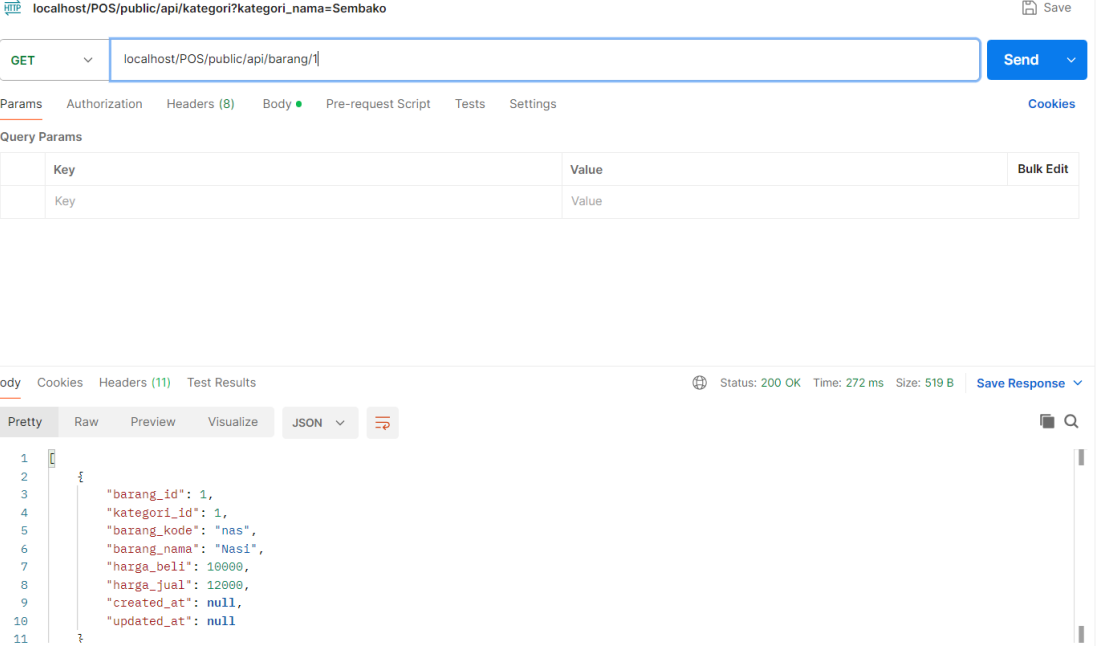
```
1 {
2   {
3     "barang_id": 1,
4     "kategori_id": 1,
5     "barang_kode": "nas",
6     "barang_nama": "Nasi",
7     "harga_beli": 10000,
8     "harga_jual": 12000,
9     "created_at": null,
10    "updated_at": null
11  },
12  {
13    "barang_id": 2,
14    "kategori_id": 1,
15    "barang_kode": "aym",
16    "barang_nama": "Ayam",
17    "harga_beli": 8000,
18    "harga_jual": 10000,
19    "created_at": null
20  }
21 }
```

Dari output di atas menampilkan seluruh daftar data barang yang ada dalam tabel barang

5

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



	
6	<p>Berikutnya lakukan percobaan menampilkan detail data. Jelaskan dan berikan screenshot hasil percobaan Anda.</p> 
7	<p>Jika sudah, kita coba untuk melakukan edit data menggunakan localhost/POS/main/public/api/barang/{id} dan method PUT. Isikan data yang ingin diubah pada tab Param.</p>



localhost/POS/public/api/kategori?kategori\_nama=Sembako

PUT

localhost/POS/public/api/barang/27?harga\_jual=100000

Send

Params Authorization Headers (8) Body Pre-request Script Tests Settings Cookies

Query Params

Key	Value	Bulk Edit
<input checked="" type="checkbox"/> harga_jual	100000	
Key	Value	

Body Cookies Headers (11) Test Results

Status: 200 OK Time: 302 ms Size: 583 B Save Response

Pretty Raw Preview Visualize JSON

```
1 { 2   "barang_id": 27, 3   "kategori_id": 17, 4   "barang_kode": "SPRT", 5   "barang_nama": "Sepatu Olahraga", 6   "harga_beli": 50000, 7   "harga_jual": 50000, 8   "created_at": "2024-11-06T04:42:24.000000Z", 9   "updated_at": "2024-11-06T04:42:24.000000Z" 10 } 11
```

localhost/POS/public/api/kategori?kategori\_nama=Sembako

PUT

localhost/POS/public/api/barang/27?harga\_jual=100000

Send

Params Authorization Headers (8) Body Pre-request Script Tests Settings Cookies

Query Params

Key	Value	Bulk Edit
<input checked="" type="checkbox"/> harga_jual	100000	
Key	Value	

Body Cookies Headers (11) Test Results

Status: 200 OK Time: 322 ms Size: 584 B Save Response

Pretty Raw Preview Visualize JSON

```
1 { 2   "barang_id": 27, 3   "kategori_id": 17, 4   "barang_kode": "SPRT", 5   "barang_nama": "Sepatu Olahraga", 6   "harga_beli": 50000, 7   "harga_jual": 100000, 8   "created_at": "2024-11-06T04:42:24.000000Z", 9   "updated_at": "2024-11-06T04:47:45.000000Z" 10 } 11
```

8

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



	
9	Lakukan commit perubahan file pada Github.