

PEMROGRAMAN WEB LANJUT

“RESTFUL API”



Kelas : TI-2H

Disusun Oleh :

Fanesabhirawaning Sulistyio

PROGRAM STUDI D-IV TEKNIK INFORMATIKA

JURUSAN TEKNOLOGI INFORMASI

POLITEKNIK NEGERI MALANG

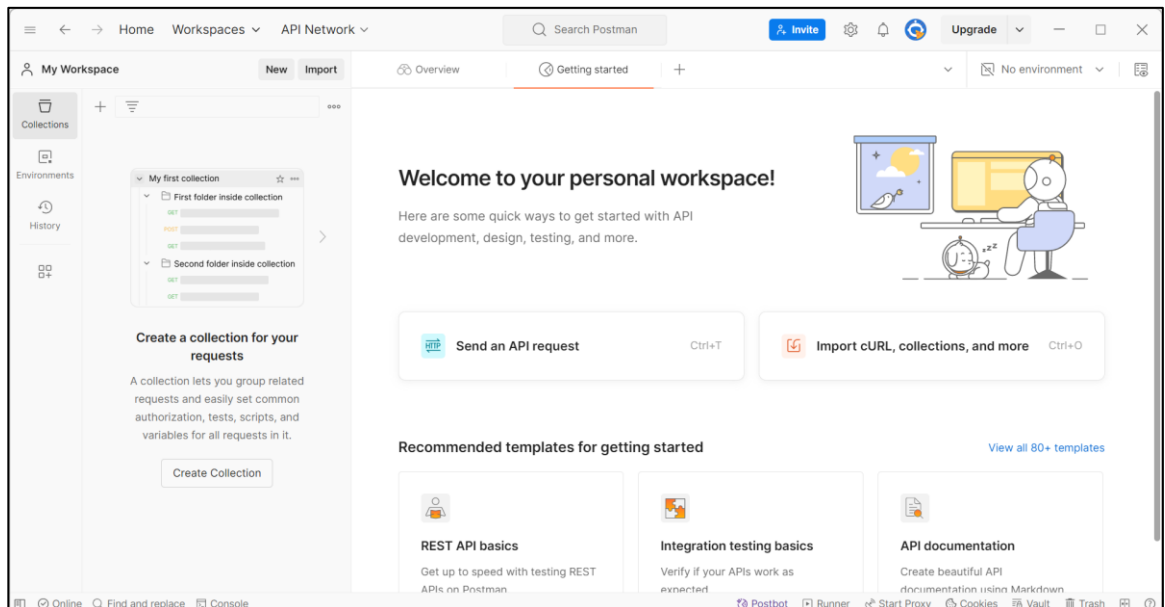
Jl. Soekarno Hatta No.9, Jatimulyo, Kec. Lowokwaru, Kota Malang, Jawa Timur 65141

\

Praktikum 1 – Membuat RESTful API Register

1. Sebelum memulai membuat REST API, terlebih dahulu download aplikasi Postman di <https://www.postman.com/downloads>.

Aplikasi ini akan digunakan untuk mengerjakan semua tahap praktikum pada Jobsheet ini.



2. Lakukan instalasi JWT dengan mengetikkan perintah berikut:

```
composer require tymon/jwt-auth:2.1.1
```

Pastikan Anda terkoneksi dengan internet.

```
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)
- 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
Class App\Http\Controllers\levelController located in C:\laragon\www\PWL_POS\app\Http\Controllers\levelControllerold.php not found in C:\laragon\www\PWL_POS\app\Http\Controllers\levelControllerold.php does not comply with psr-4 autoloading standard. Skipping.
> Illuminate\Foundation\ComposerScripts::postAutoloadDump
> @php artisan package:discover --ansi

[INFO] Discovering packages.

barryvdh/laravel-dompdf DONE
jeroennoten/laravel-adminlte DONE
laravel/sail .. DONE
laravel/sanctum DONE
laravel/tinker DONE
laravel/ui .... DONE
nesbot/carbon . DONE

!
> @php artisan vendor:publish --tag=laravel-assets --ansi --force

[INFO] No publishable resources for tag [laravel-assets].
```

3. Setelah berhasil menginstall JWT, lanjutkan dengan publish konfigurasi file dengan perintah berikut:

```
php artisan vendor:publish --  
provider="Tymon\JWTAuth\Providers\LaravelServiceProvider"
```

```
PS C:\laragon\www\PWL_POS> php artisan vendor:publish --provider="Tymon\JWTAuth\Providers\LaravelServiceProvider"  
INFO Publishing assets.  
Copying file [C:\laragon\www\PWL_POS\vendor\tymon\jwt-auth\config\config.php] to [C:\laragon\www\PWL_POS\config\jwt.php] ..... DONE
```

4. Jika perintah di atas berhasil, maka kita akan mendapatkan 1 file baru yaitu config/jwt.php. Pada file ini dapat dilakukan konfigurasi jika memang diperlukan.

```
jwt.php U
```

5. Setelah itu jalankan perintah berikut untuk membuat secret key JWT.

```
php artisan jwt:secret
```

```
PS C:\laragon\www\PWL_POS> php artisan jwt:secret  
jwt-auth secret [FEMJxDZsZm30cvu1riC7bWMPoqi3eLLIdbpgpkZOL60D7Rq0BdUVb2i14F9RppId] set successfully.
```

Jika berhasil, maka pada file .env akan ditambahkan sebuah baris berisi nilai key JWT_SECRET.

```
.env  
61 JWT_SECRET=FEMJxDZsZm30cvu1riC7bWMPoqi3eLLIdbpgpkZOL60D7Rq0BdUVb2i14F9RppId
```

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

```
38     'guards' => [  
39         'web' => [  
40             'driver' => 'session',  
41             'provider' => 'users',  
42         ],  
43         'api' => [  
44             'driver' => 'jwt',  
45             'provider' => 'users',  
46         ],  
47     ],
```

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

```
app > Models > UserModel.php > ...
1  <?php
2
3  namespace App\Models;
4
5  use Illuminate\Database\Eloquent\Factories\HasFactory;
6  use Illuminate\Database\Eloquent\Model;
7  use Illuminate\Database\Eloquent\Relations\BelongsTo;
8  use Illuminate\Database\Eloquent\Relations\HasOne;
9  use Illuminate\Foundation\Auth\User as Authenticatable;
10 use Tymon\JWTAuth\Contracts\JWTSubject;
11
12 use App\Models\LevelModel; // Add this import statement
13
14 class UserModel extends Authenticatable implements JWTSubject
15 {
16     use HasFactory;
17     public function getJWTIdentifier()
18     {
19         return $this->getKey();
20     }
21     public function getJWTCustomClaims()
22     {
23         return [];
24     }
25 }
```

8. Berikutnya kita akan membuat controller untuk register dengan menjalankan perintah berikut.

`php artisan make:controller Api/RegisterController`

```
PS C:\laragon\www\PWL_POS> php artisan make:controller Api/RegisterController
INFO Controller [C:\laragon\www\PWL_POS\app\Http\Controllers\Api\RegisterController.php] created successfully.
```

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

`RegisterController.php`

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

```
app > Http > Controllers > Api > RegisterController.php > RegisterController
1  <?php
2
3  namespace App\Http\Controllers\Api;
4
5  use App\Models\UserModel;
6  use App\Http\Controllers\Controller;
7  use Illuminate\Http\Request;
8  use Illuminate\Support\Facades\Validator;
9
10 class RegisterController extends Controller
11 {
12 }
```

```

12 public function __invoke(Request $request)
13 {
14     //set validator
15     $validator = Validator::make($request->all(), [
16         'username' => 'required',
17         'nama' => 'required',
18         'password' => 'required|min:5|confirmed',
19         'level_id' => 'required'
20     ]);
21
22     //if validator fails
23     if ($validator->fails()) {
24         return response()->json($validator->errors(), 422);
25     }
26
27     //create user
28     $user = UserModel::create([
29         'username' => $request->username,
30         'nama' => $request->nama,
31         'password' => bcrypt($request->password),
32         'level_id' => $request->level_id,
33     ]);
34
35     //return response JSON user is created
36     if ($user) {
37         return response()->json([
38             'success' => true,
39             'user' => $user
40         ], 201);
41     }
42
43     //return JSON process insert failed
44     return response()->json([
45         'success' => false,
46     ], 409);
47 }
48 ]

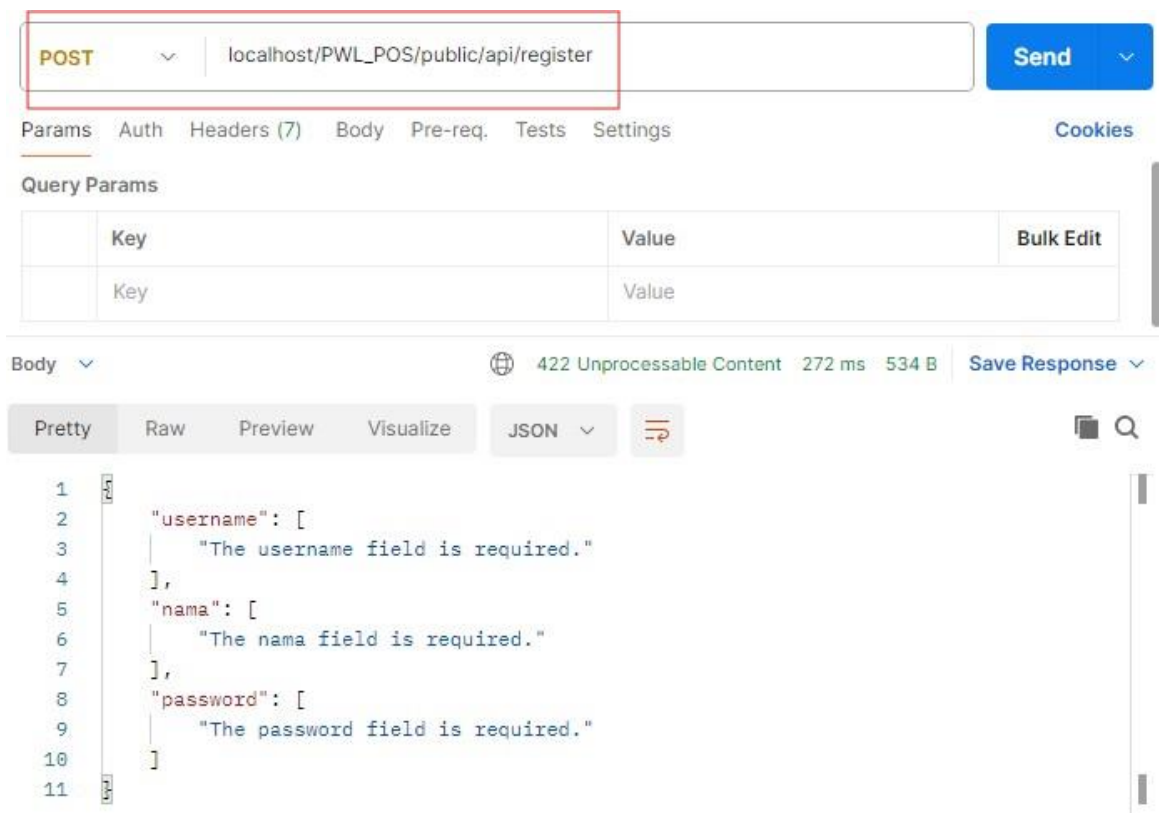
```

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::middleware('auth:sanctum')->get('/user', function (Request $request) {
19     return $request->user();
20 });
21
22 Route::post('/register', App\Http\Controllers\Api\RegisterController::class)->name('register');
23
```

11. Jika sudah, kita akan melakukan uji coba REST API melalui aplikasi Postman.

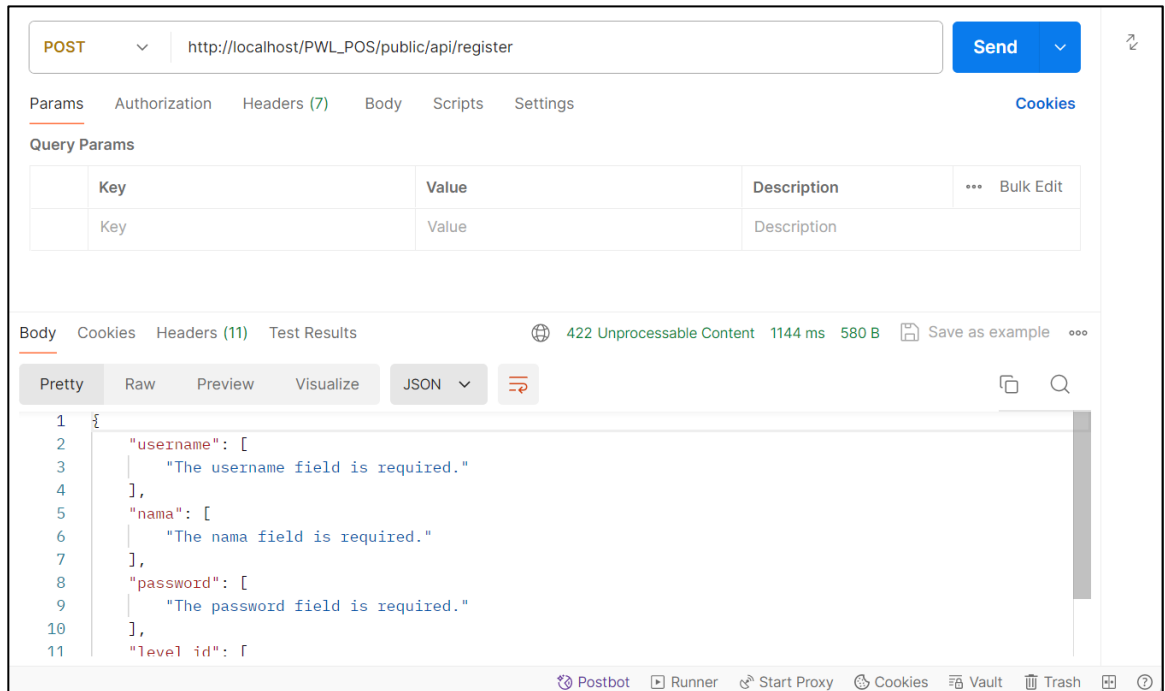
Buka aplikasi Postman, isi URL localhost/PWL_POS/public/api/register serta method POST. Klik Send.



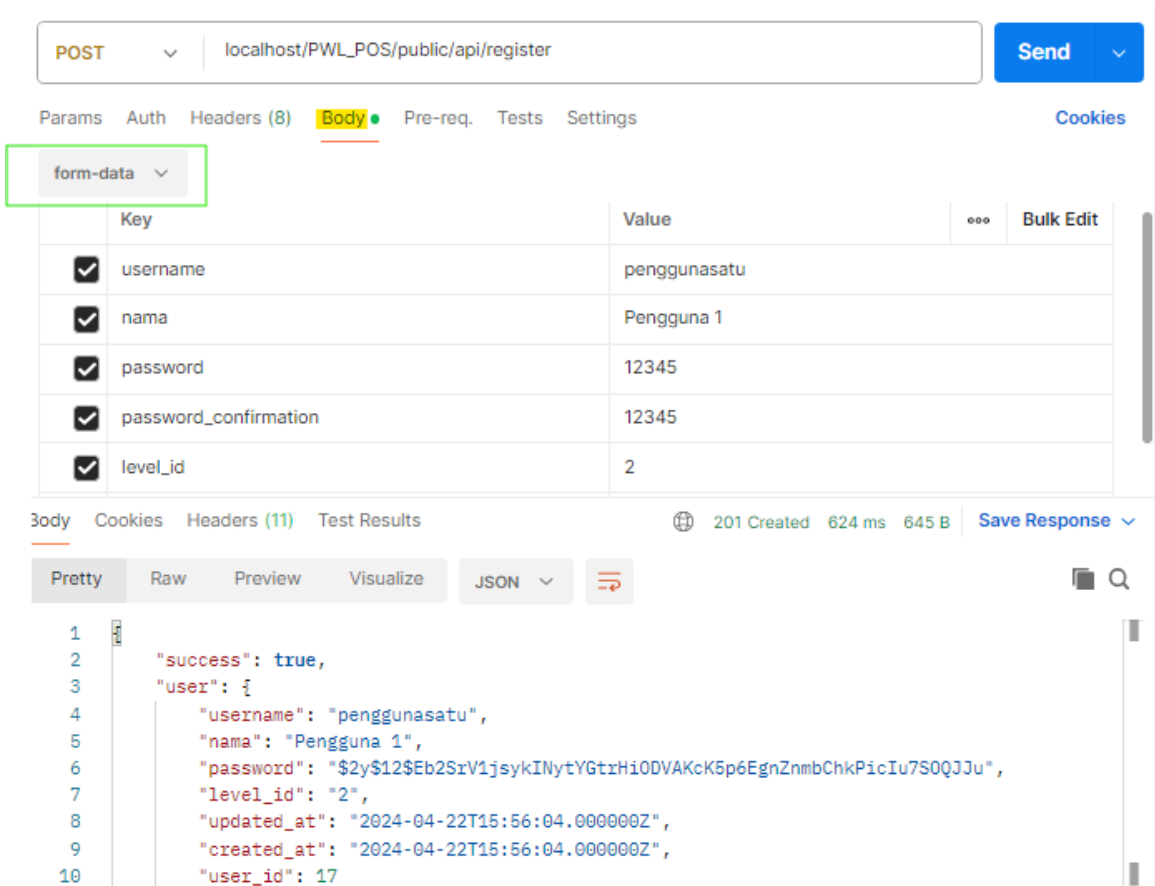
Jika berhasil akan muncul error validasi seperti gambar di atas.

Lakukan percobaan yang sama dan berikan screenshot hasil percobaan Anda.

Jawab:



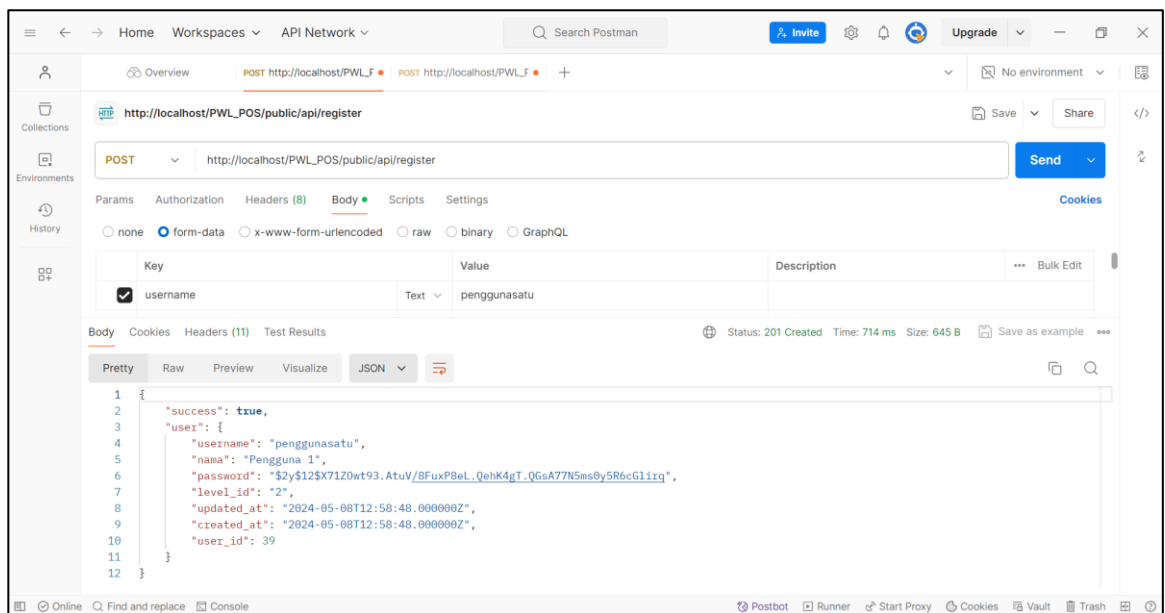
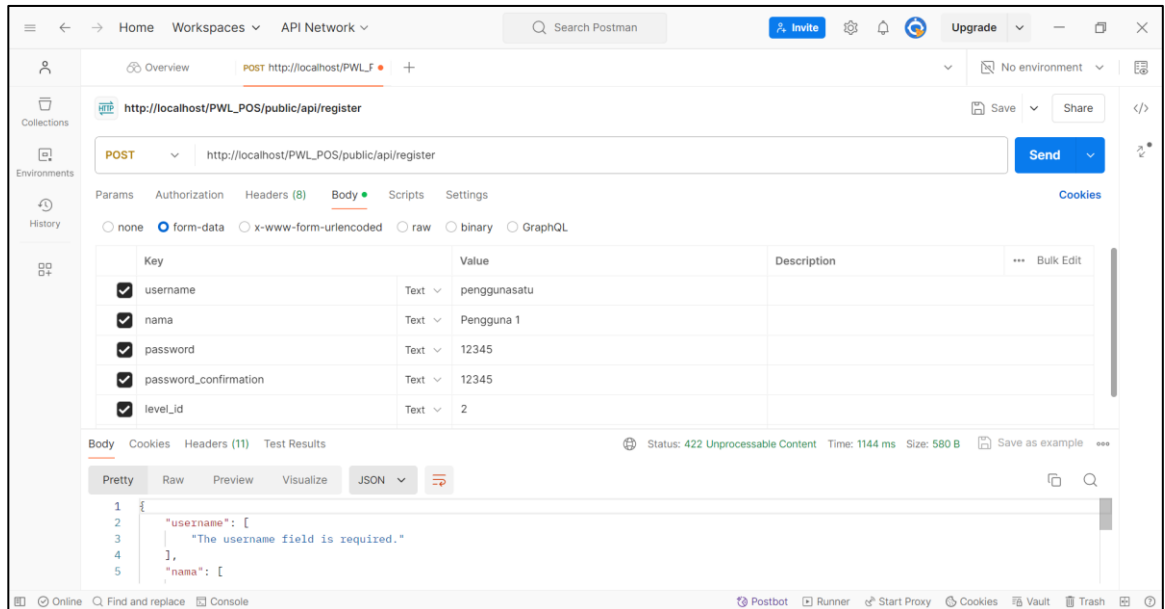
12. Sekarang kita coba masukkan data. Klik tab Body dan pilih form-data. Isikan key sesuai dengan kolom data, serta isikan data registrasi menggunakan nilai yang Anda inginkan.



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

Lakukan percobaan yang sama dan berikan screenshoot hasil percobaan Anda.

Jawab:



13. Lakukan commit perubahan file pada Github.

Praktikum 2 – Membuat RESTful API Login


1. Kita buat file controller dengan nama LoginController.

`php artisan make:controller Api/LoginController`

```
PS C:\laragon\www\PWL_POS> php artisan make:controller Api/LoginController
```

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

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

 LoginController.php

U

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

```
app > Http > Controllers > Api > LoginController.php > LoginController

1  <?php
2
3  namespace App\Http\Controllers\Api;
4
5  use App\Http\Controllers\Controller;
6  use Illuminate\Http\Request;
7  use Illuminate\Support\Facades\Validator;
8
9  class LoginController extends Controller
10 {
11     public function __invoke(Request $request)
12     {
13         //set validation
14         $validator = Validator::make($request->all(), [
15             'username' => 'required',
16             'password' => 'required'
17         ]);
18
19         //if validation fails
20         if ($validator->fails()) {
21             return response()->json($validator->errors(), 422);
22         }
23
24         //get credentials from request
25         $credentials = $request->only('username', 'password');
26
27         //if auth failed
28         if (!$token = auth()->guard('api')->attempt($credentials)) {
29             return response()->json([
30                 'success' => false,
31                 'message' => 'Username atau Password Anda salah'
32             ], 401);
33         }
34
35         //if auth success
36         return response()->json([
37             'success' => true,
38             // 'user' => auth()->user(),
39             'user' => auth()->guard('api')->user(),
40             'token' => $token
41         ], 200);
42     }
43 }
```

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

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.

POST localhost/PWL_POS/public/api/login Send

Params Auth Headers (7) Body Pre-req. Tests Settings Cookies

Query Params

Key	Value	Bulk Edit
Key	Value	

Body Cookies Headers (11) Test Results 422 Unprocessable Content 563 ms 495 B Save Response

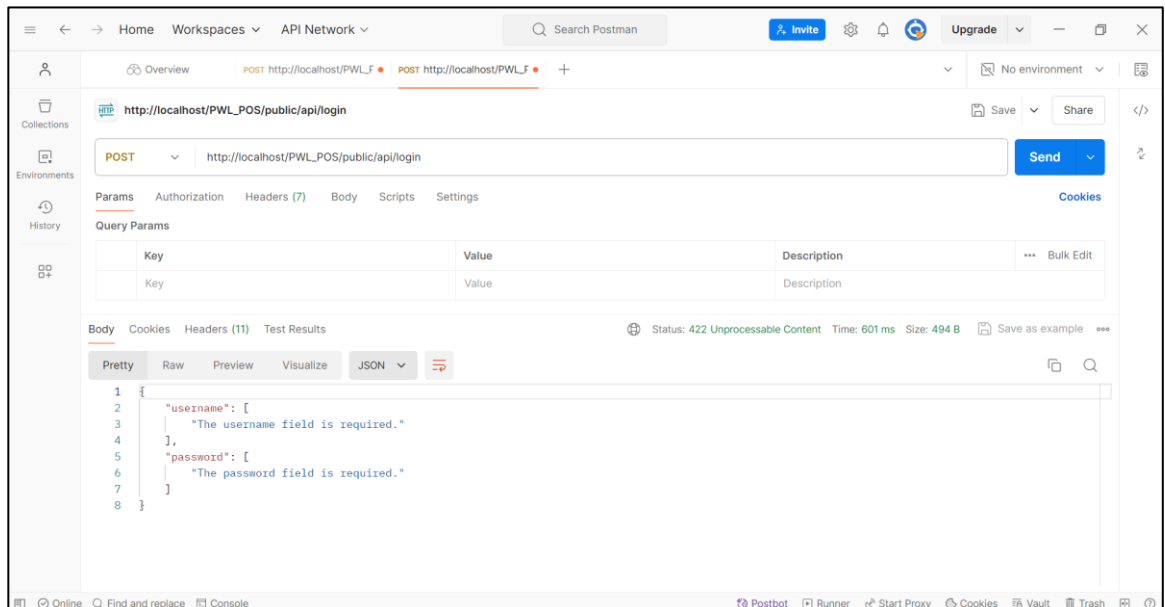
Pretty Raw Preview Visualize JSON

```
1 {
2   "username": [
3     "The username field is required."
4   ],
5   "password": [
6     "The password field is required."
7   ]
8 }
```

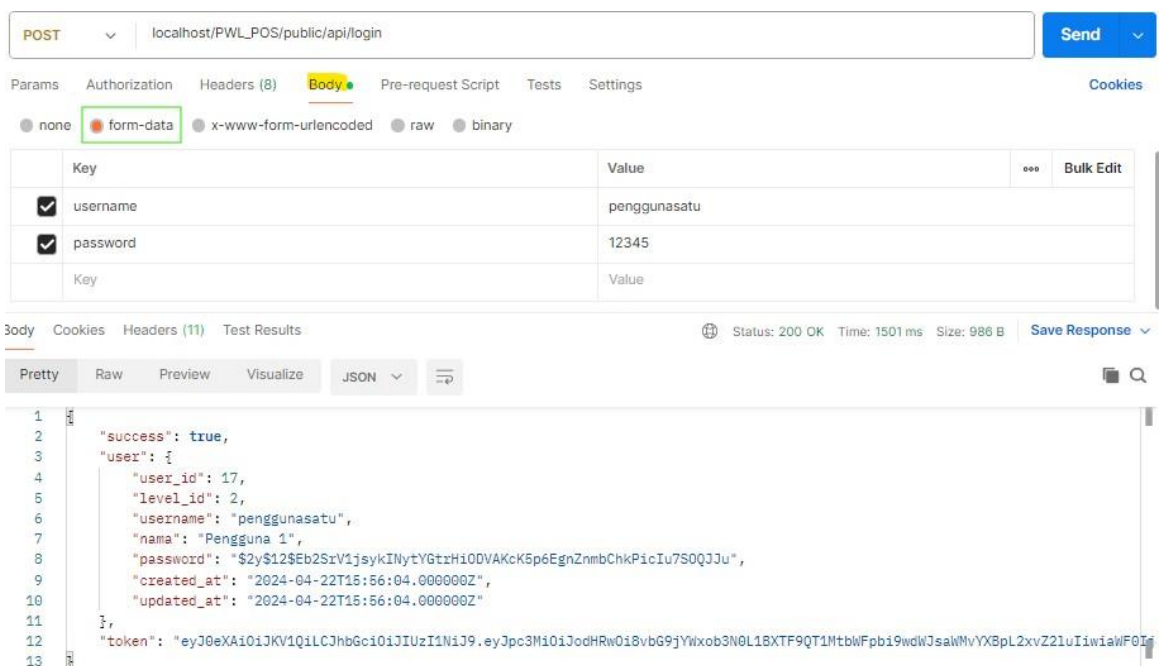
Jika berhasil akan muncul error validasi seperti gambar di atas.

Lakukan percobaan yang sama dan berikan screenshoot hasil percobaan Anda.

Jawab:

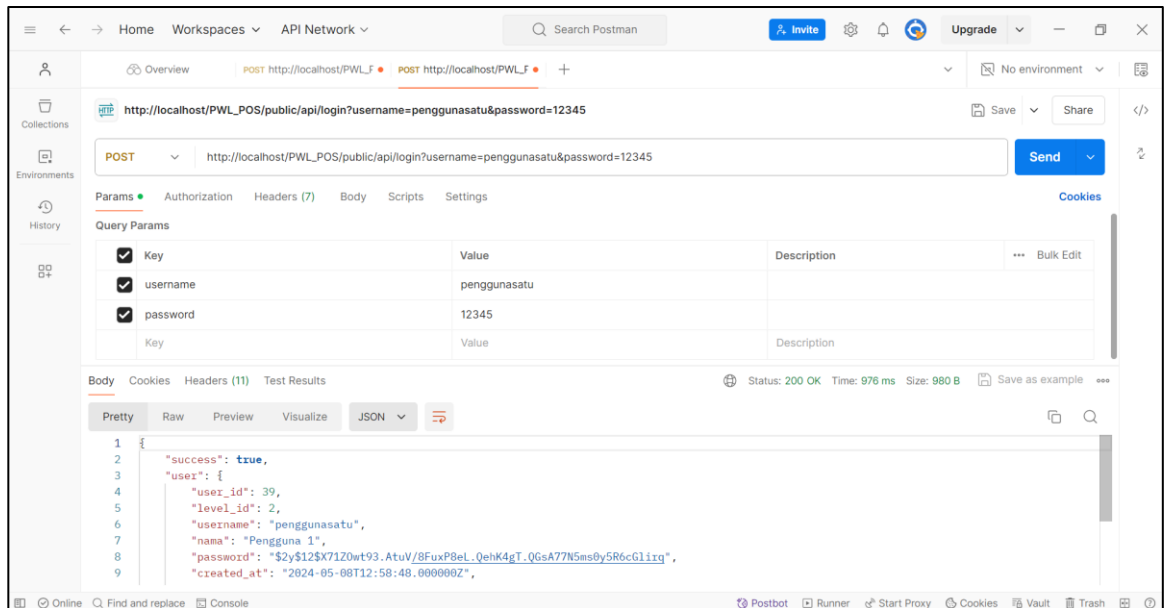


5. Selanjutnya, isikan username dan password sesuai dengan data user yang ada pada database. Klik tab Body dan pilih form-data. Isikan key sesuai dengan kolom data, serta isikan data user. Klik tombol Send, jika berhasil maka akan keluar tampilan seperti berikut. Copy nilai token yang diperoleh pada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



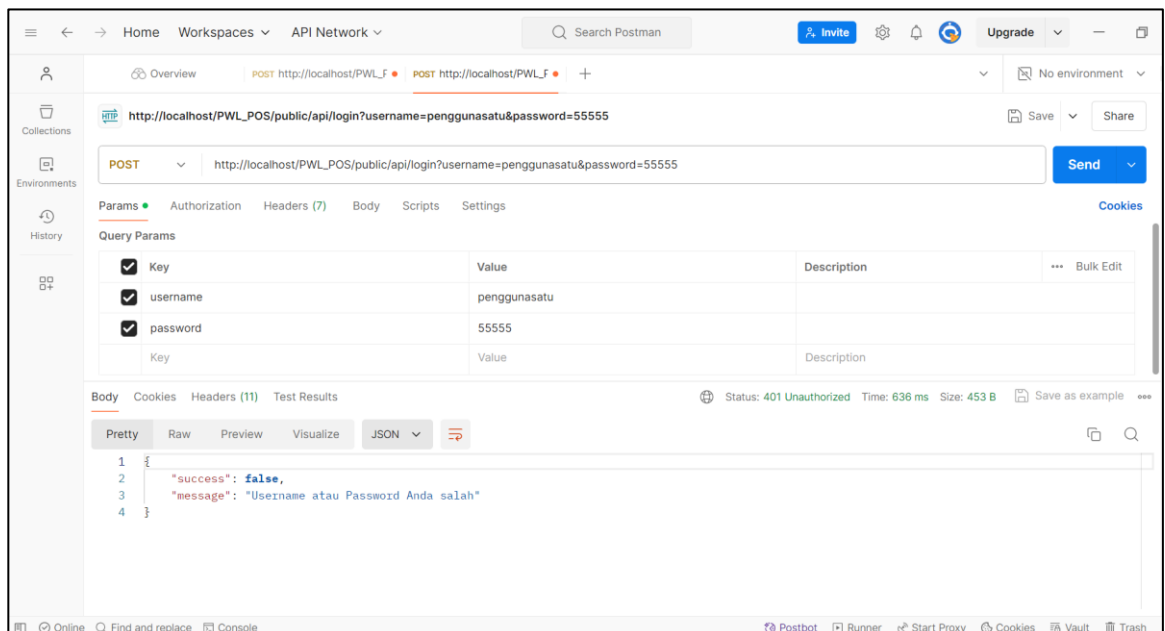
Lakukan percobaan yang sama dan berikan screenshoot hasil percobaan Anda.

Jawab:



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

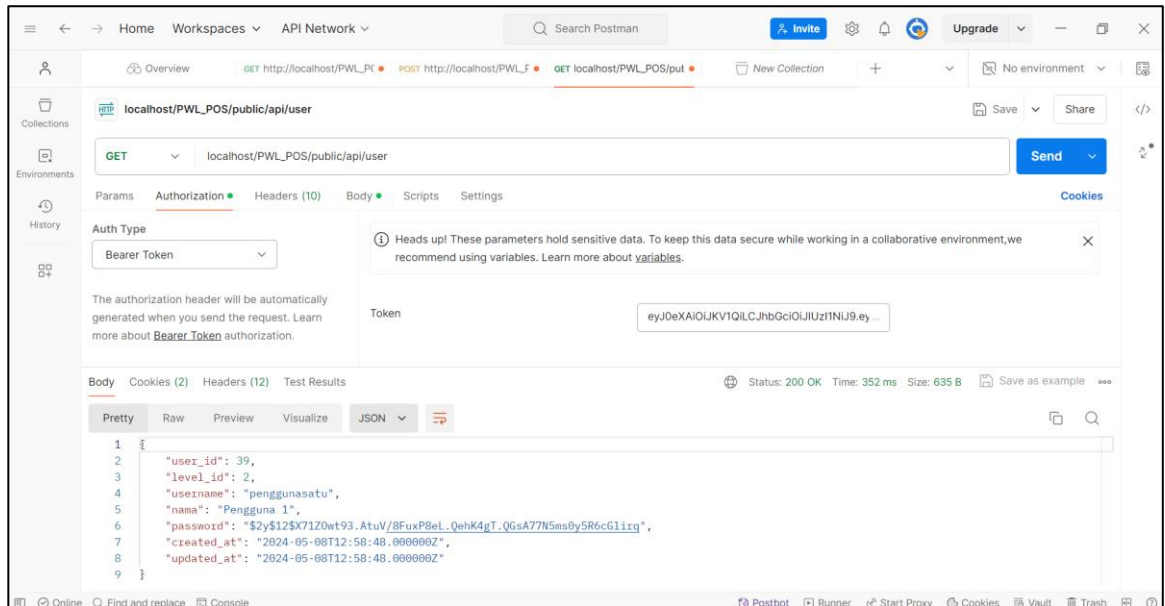
Jawab:



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.

Jawab:



hasil dari percobaan tersebut adalah mendapatkan respons yang berisi informasi data user.

8. Lakukan commit perubahan file pada Github.

Praktikum 3 – Membuat RESTful API Logout

1. Tambahkan kode berikut pada file .env
`JWT_SHOW_BLACKLIST_EXCEPTION=true`
2. Buat Controller baru dengan nama LogoutController.
`php artisan make:controller Api/LogoutController`

```
PS C:\laragon\www\PWL_POS> php artisan make:controller Api/LogoutController
INFO Controller [C:\laragon\www\PWL_POS\app\Http\Controllers\Api\LogoutController.php] created successfully.
```

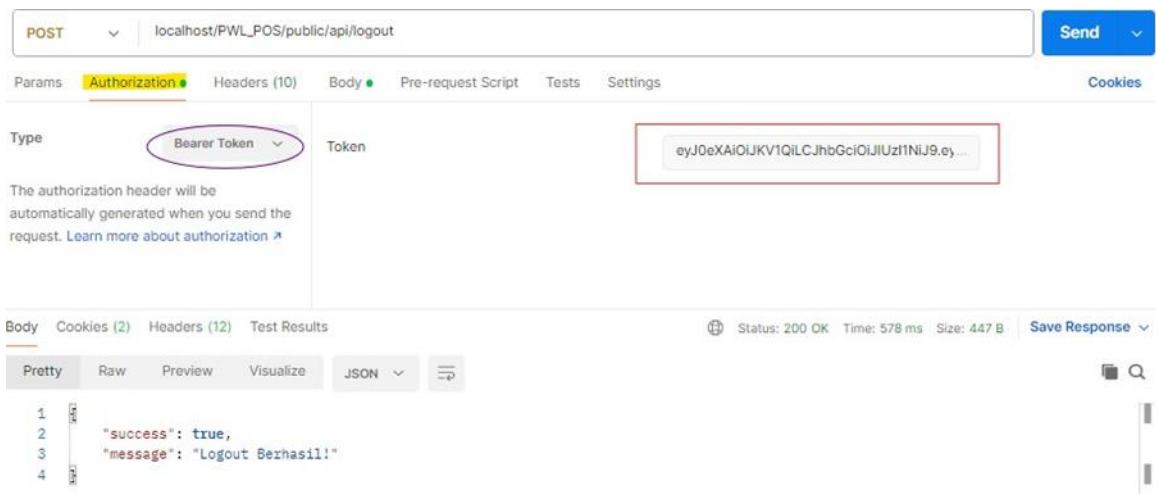
3. Buka file tersebut dan ubah kode menjadi seperti berikut.

```
app > Http > Controllers > Api > LogoutController.php > LogoutController
1  <?php
2
3  namespace App\Http\Controllers\Api;
4
5  use App\Http\Controllers\Controller;
6  use Illuminate\Http\Request;
7  use Tymon\JWTAuth\Facades\JWTAuth;
8  use Tymon\JWTAuth\Exceptions\JWTException;
9  use Tymon\JWTAuth\Exceptions\TokenExpiredException;
10 use Tymon\JWTAuth\Exceptions\TokenInvalidException;
11
12 class LogoutController extends Controller
13 {
14     public function __invoke(Request $request)
15     {
16         //remove token
17         $removeToken = JWTAuth::invalidate(JWTAuth::getToken());
18
19         if ($removeToken) {
20             //return response JSON
21             return response()->json([
22                 'success' => true,
23                 'message' => 'Logout berhasil!',
24             ]);
25         }
26     }
27 }
```

4. Lalu kita tambahkan routes pada api.php

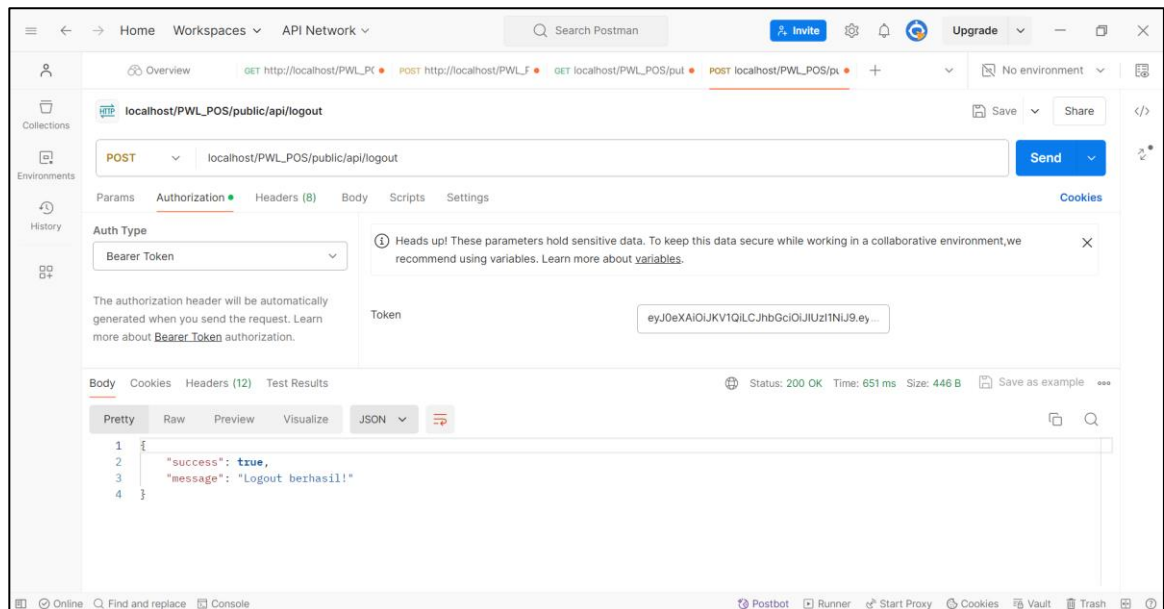
```
24 Route::post('/register', RegisterController::class)->name('register');
25 Route::post('/login', LoginController::class)->name('login');
26 Route::middleware('auth:api')->get('/user', function (Request $request) {
27     return $request->user();
28 });
29 Route::post('/logout', LogoutController::class)->name('logout');
```

5. Jika sudah, kita akan melakukan uji coba REST API melalui aplikasi Postman. Buka aplikasi Postman, isi URL localhost/PWL_POS/public/api/logout serta method POST.
6. Isi token pada tab Authorization, pilih Type yaitu Bearer Token. Isikan token yang didapat saat login. Jika sudah klik Send.



Lakukan percobaan yang sama dan berikan screenshoot hasil percobaan Anda.

Jawab:



7. Lakukan commit perubahan file pada Github.

Praktikum 4 – Implementasi CRUD dalam RESTful API

Pada praktikum ini kita akan menggunakan tabel m_level untuk dimodifikasi menggunakan RESTful API.

1. Pertama, buat controller untuk mengolah API pada data level.

`php artisan make:controller Api/LevelController`

PS C:\laragon\www\PWL_POS> `php artisan make:controller Api/LevelController`

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

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

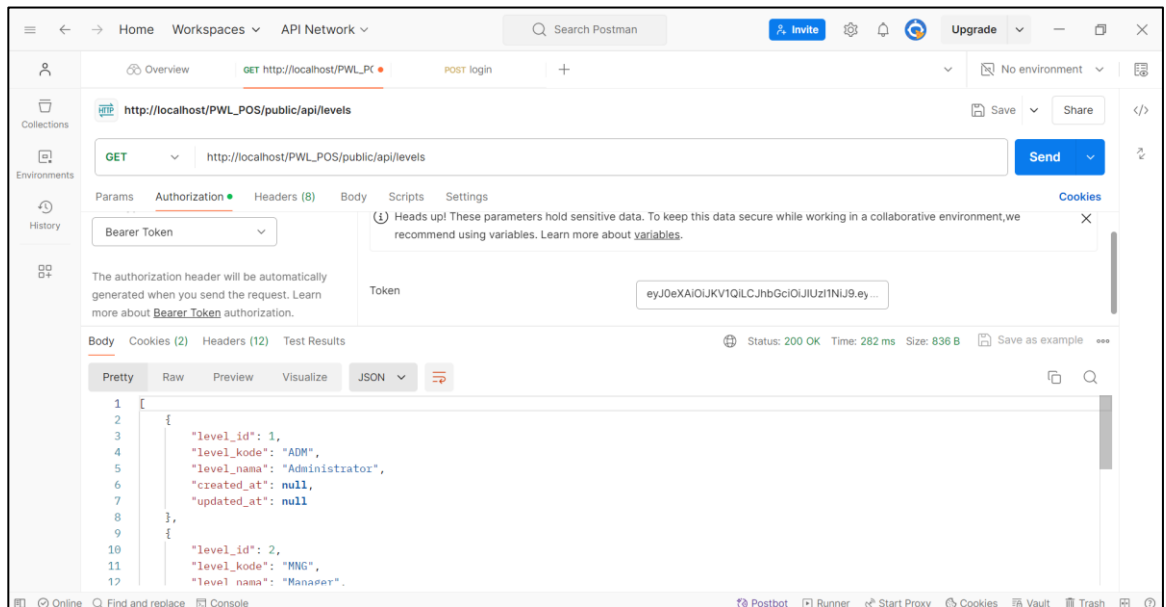
```
LevelController.php U X
app > Http > Controllers > Api > LevelController.php > LevelController
1  <?php
2
3  namespace App\Http\Controllers\Api;
4
5  use App\Http\Controllers\Controller;
6  use Illuminate\Http\Request;
7  use App\Models\LevelModel;
8
9  class LevelController extends Controller
10 {
11     public function index()
12     {
13         return LevelModel::all();
14     }
15
16     public function store(Request $request)
17     {
18         $level = LevelModel::create($request->all());
19         return response()->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         $level->update($request->all());
30         return LevelModel::find($level);
31     }
32
33     public function destroy(LevelModel $level)
34     {
35         $level->delete();
36         return response()->json([
37             "success" => true,
38             "message" => "Data terhapus",
39         ]);
40     }
41 }
```

3. Kemudian kita lengkapi routes pada api.php

```
31 use App\Http\Controllers\Api\LevelController;
32
33 Route::get('/levels', [LevelController::class, 'index']);
34 Route::post('/levels', [LevelController::class, 'store']);
35 Route::get('/levels/{level}', [LevelController::class, 'show']);
36 Route::put('/levels/{level}', [LevelController::class, 'update']);
37 Route::delete('/levels/{level}', [LevelController::class, 'destroy']);
```

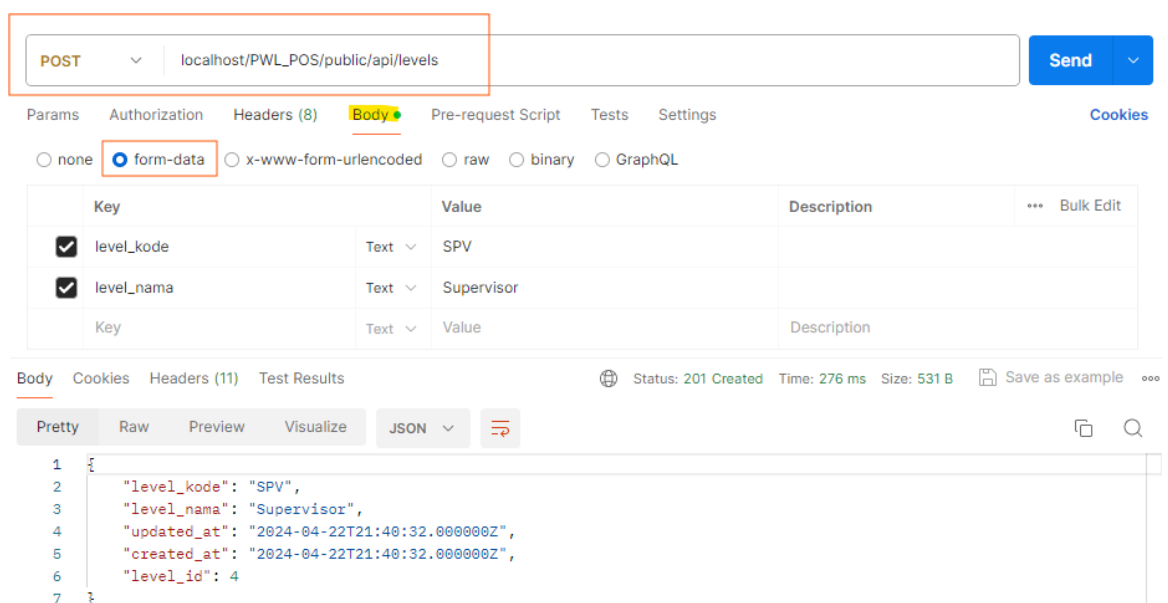

5. Jika sudah. Lakukan uji coba API mulai dari fungsi untuk menampilkan data. Gunakan URL: localhost/PWL_POS-main/public/api/levels dan method GET. **Jelaskan dan berikan screenshoot hasil percobaan Anda.**

Jawab:



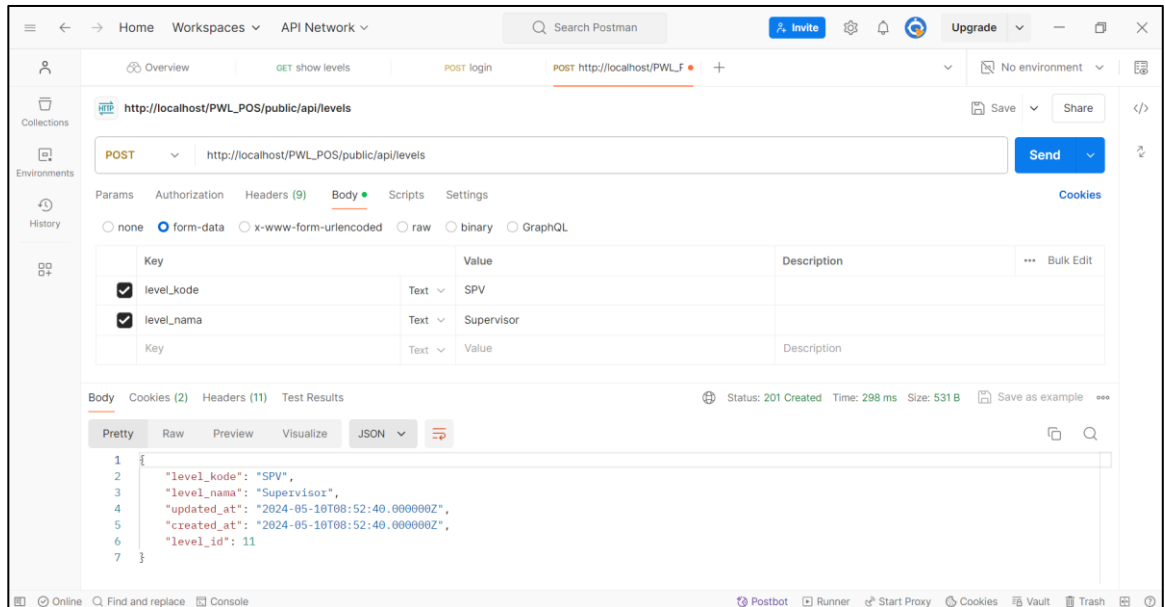
Maka akan menampilkan daftar level yang ada pada database dengan keterangan level_id, level_kode, level_nama, created_at, dan updated_at

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



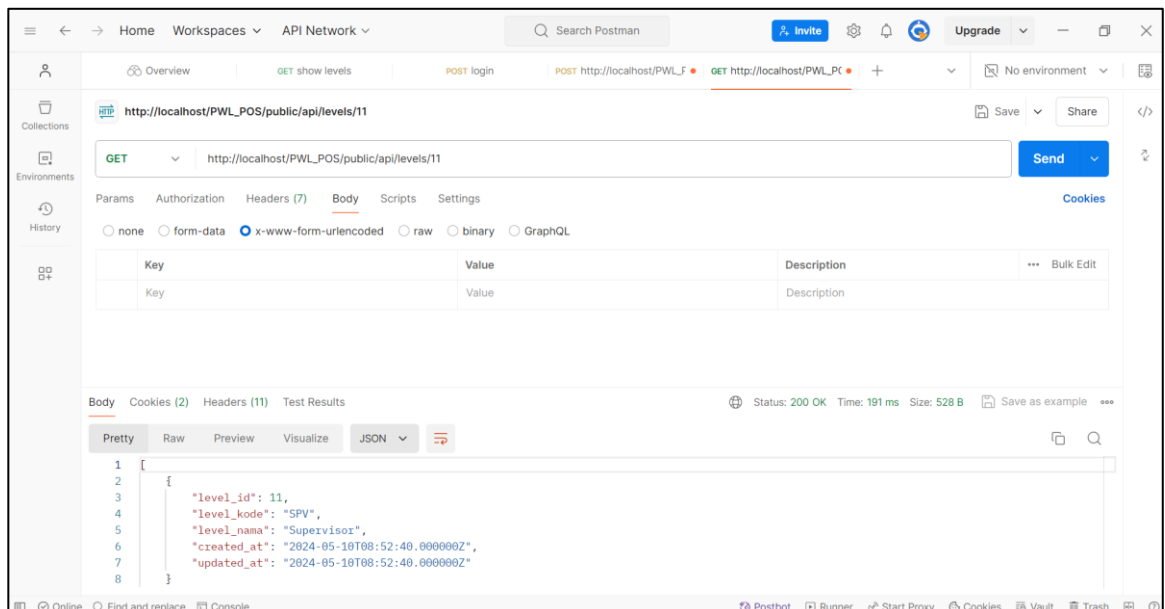
Jelaskan dan berikan screenshoot hasil percobaan Anda.

Jawab:



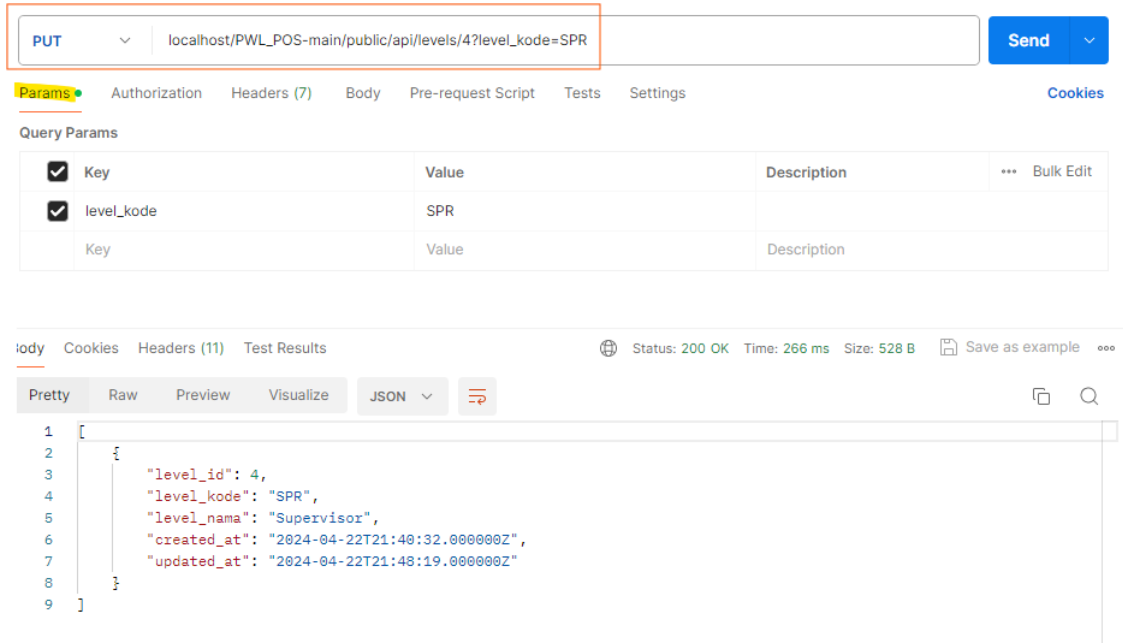
Membuat level baru dengan level_kode SPV dan level_nama Supervisor. Setelah itu jika berhasil akan menampilkan detail keterangan dibawahnya.

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

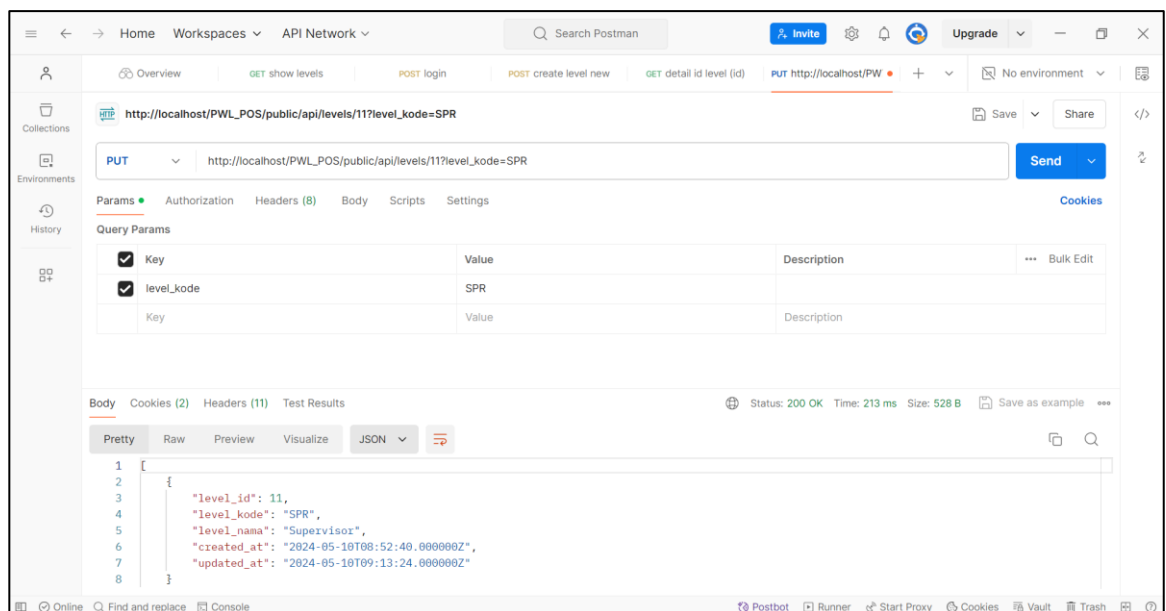


Menampilkan detail level dari id user 11

8. 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.



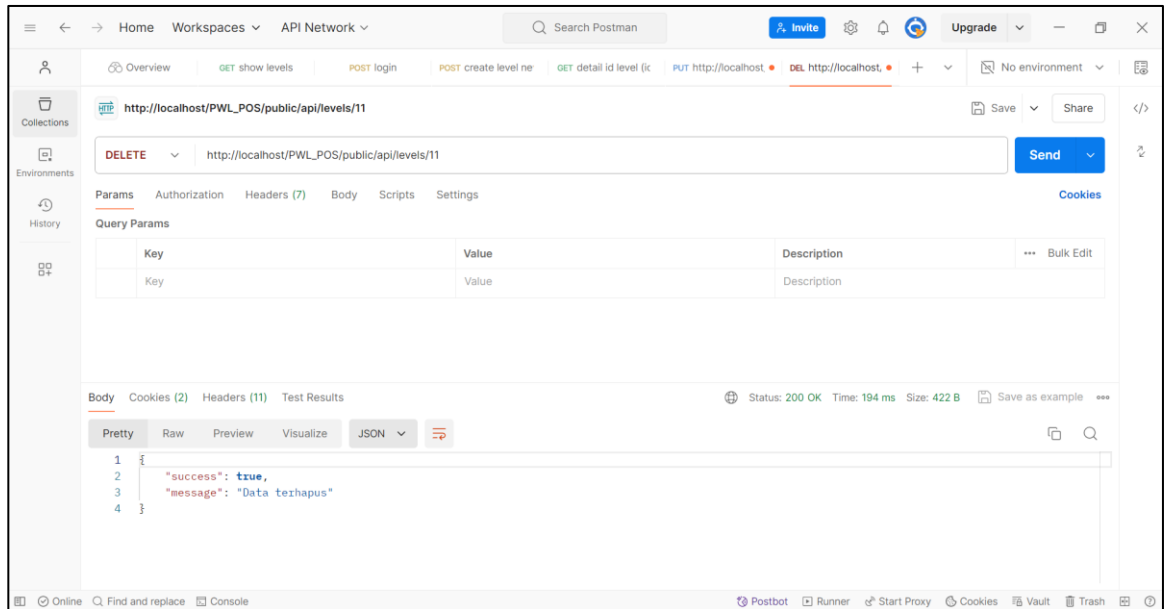
Jelaskan dan berikan screenshoot hasil percobaan Anda.



Hasil dari edit data diatas adalah perubahan level_kode yang awalnya SPV menjadi SPR

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

Jawab:



Data level dengan id_level 11 akan terhapus

10. Lakukan commit perubahan file pada Github.

TUGAS

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

```
PS C:\laragon\www\PWL_POS> php artisan make:controller Api/BarangController
INFO Controller [C:\laragon\www\PWL_POS\app\Http\Controllers\Api\BarangController.php] created successfully.
PS C:\laragon\www\PWL_POS> php artisan make:controller Api/KategoriController
INFO Controller [C:\laragon\www\PWL_POS\app\Http\Controllers\Api\KategoriController.php] created successfully.
PS C:\laragon\www\PWL_POS> php artisan make:controller Api/UserController
INFO Controller [C:\laragon\www\PWL_POS\app\Http\Controllers\Api\UserController.php] created successfully.
```

```

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);
25     }
26
27     public function update(Request $request, BarangModel $barang)
28     {
29         $barang->update($request->all());
30         return BarangModel::find($barang);
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 }

```

```

55 use App\Http\Controllers\Api\BarangController;
56
57 Route::get('/barangs', [BarangController::class, 'index']);
58 Route::post('/barangs', [BarangController::class, 'store']);
59 Route::get('/barangs/{barang}', [BarangController::class, 'show']);
60 Route::put('/barangs/{barang}', [BarangController::class, 'update']);
61 Route::delete('/barangs/{barang}', [BarangController::class, 'destroy']);

```

```

app > Http > Controllers > Api > KategoriController.php > KategoriController
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 }

```

```

47 use App\Http\Controllers\Api\KategoriController;
48
49 Route::get('/kategoris', [KategoriController::class, 'index']);
50 Route::post('/kategoris', [KategoriController::class, 'store']);
51 Route::get('/kategoris/{kategori}', [KategoriController::class, 'show']);
52 Route::put('/kategoris/{kategori}', [KategoriController::class, 'update']);
53 Route::delete('/kategoris/{kategori}', [KategoriController::class, 'destroy']);
54

```

```

app > Http > Controllers > Api > UserController.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\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);
25     }
26
27     public function update(Request $request, UserModel $user)
28     {
29         $user->update($request->all());
30         return UserModel::find($user);
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 }

```

```

39 use App\Http\Controllers\Api\UserController;
40
41 Route::get('/users', [UserController::class, 'index']);
42 Route::post('/users', [UserController::class, 'store']);
43 Route::get('/users/{user}', [UserController::class, 'show']);
44 Route::put('/users/{user}', [UserController::class, 'update']);
45 Route::delete('/users/{user}', [UserController::class, 'destroy']);

```

• Show daftar barangs

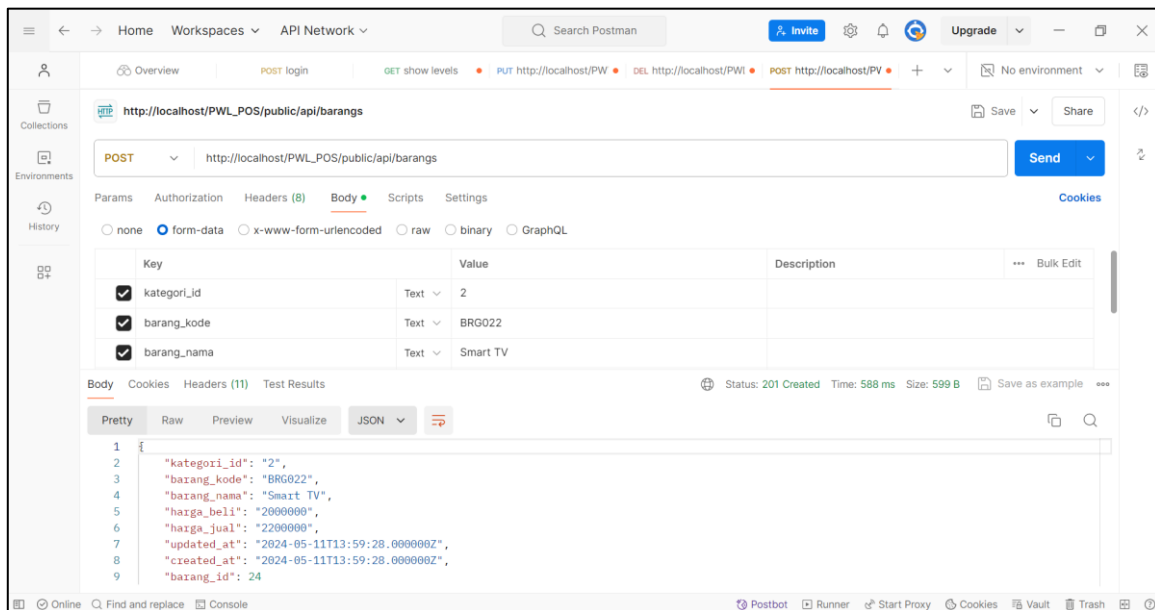
The screenshot shows the Postman interface with a GET request to `http://localhost/PWL_POS/public/api/barangs` successfully executed. The response status is 200 OK, and the body contains a JSON array of two items. The first item has a `barang_id` of 22 and a `barang_nama` of "Setelan Bayi". The second item has a `barang_id` of 23 and a `barang_nama` of "Kemeja Pria".

```

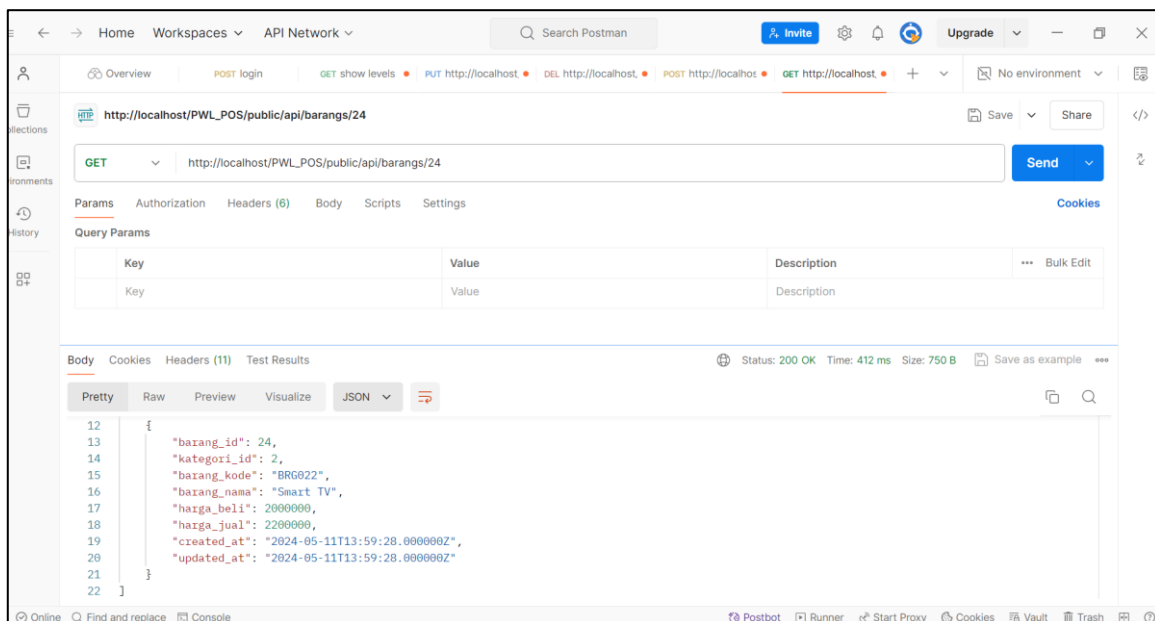
[
  {
    "barang_id": 22,
    "kategori_id": 2,
    "barang_kode": "PAK0187",
    "barang_nama": "Setelan Bayi",
    "harga_beli": 90000,
    "harga_jual": 109000,
    "created_at": "2024-04-14T12:12:12.000000Z",
    "updated_at": "2024-04-14T12:25:47.000000Z"
  },
  {
    "barang_id": 23,
    "kategori_id": 2,
    "barang_kode": "PK00002",
    "barang_nama": "Kemeja Pria",
    "harga_beli": 90000,
    "harga_jual": 109000,
    "created_at": "2024-04-17T08:12:02.000000Z",
    "updated_at": "2024-04-17T08:12:02.000000Z"
  }
]

```

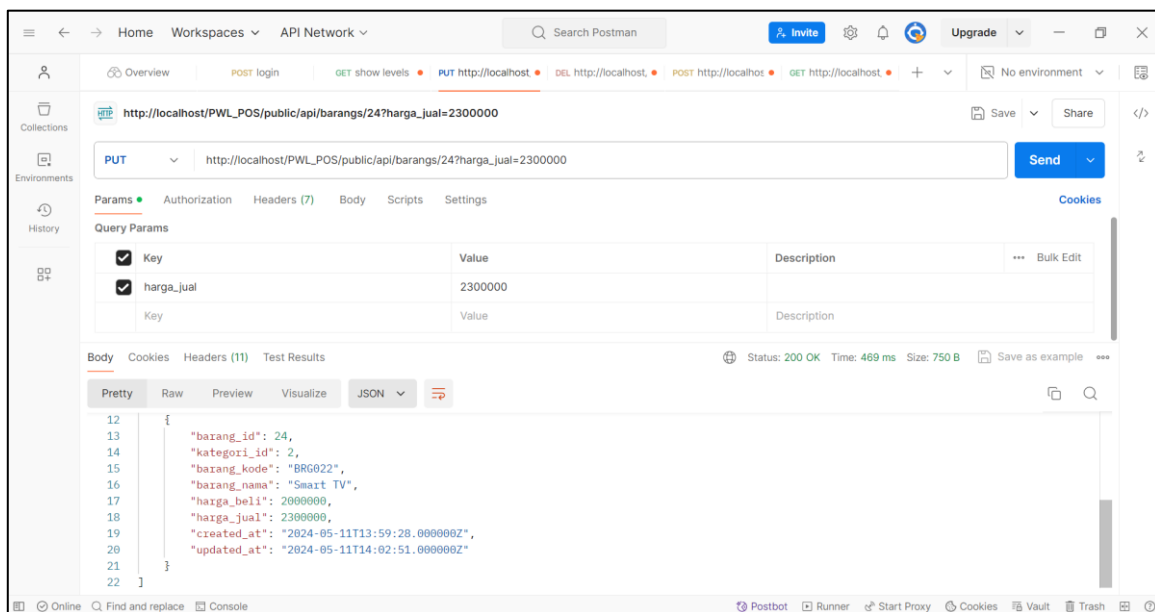
- Tambah barangs



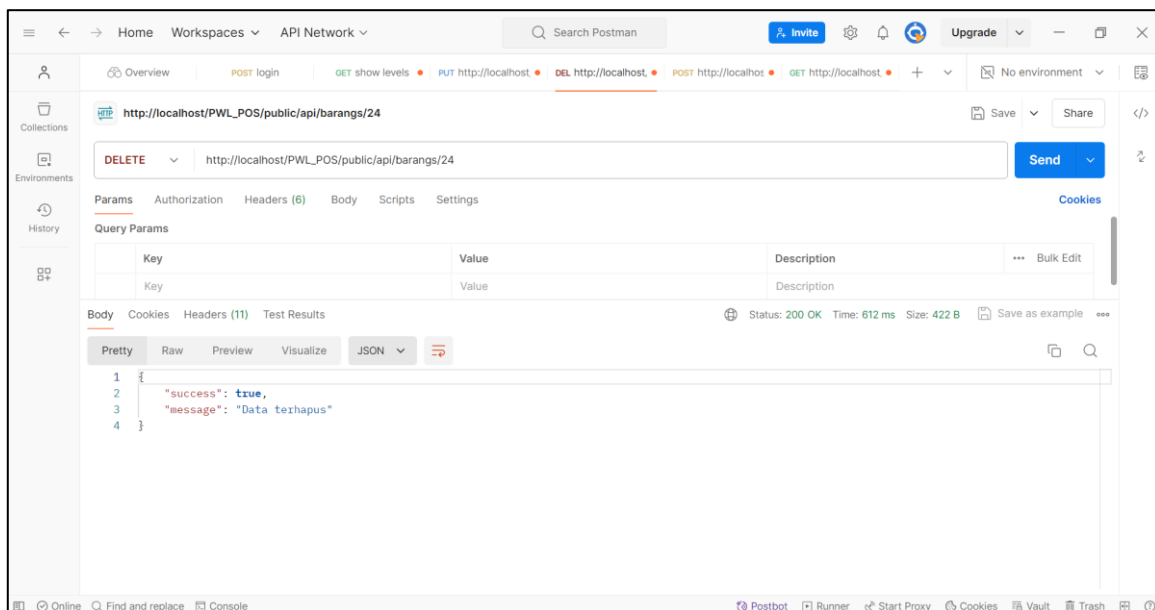
- Show detail barangs



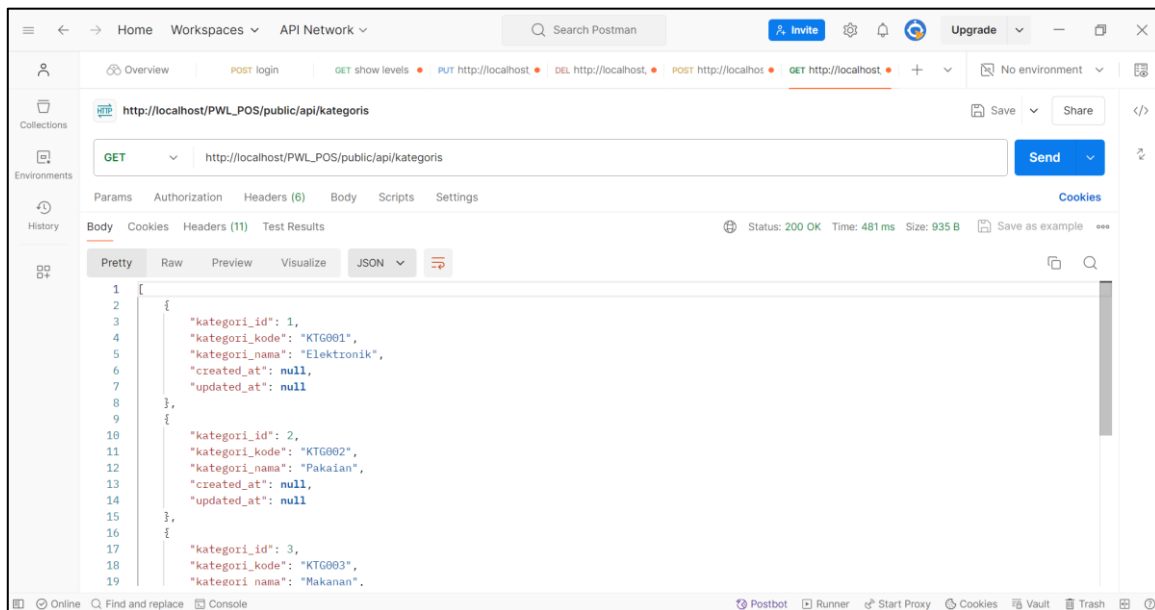
- Edit barangs



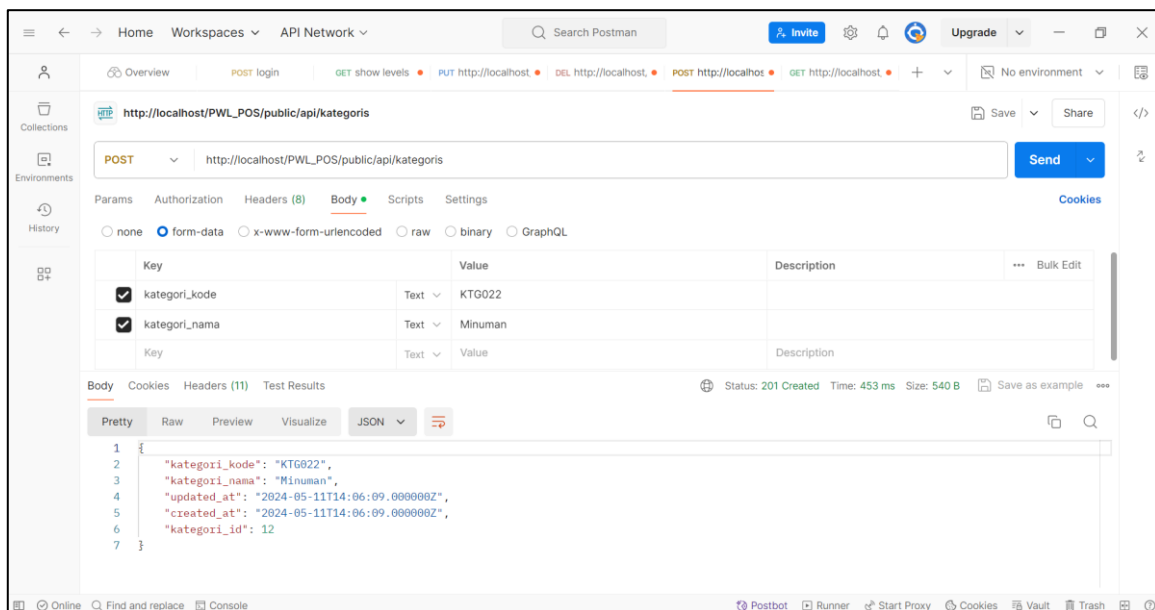
- Delete barangs



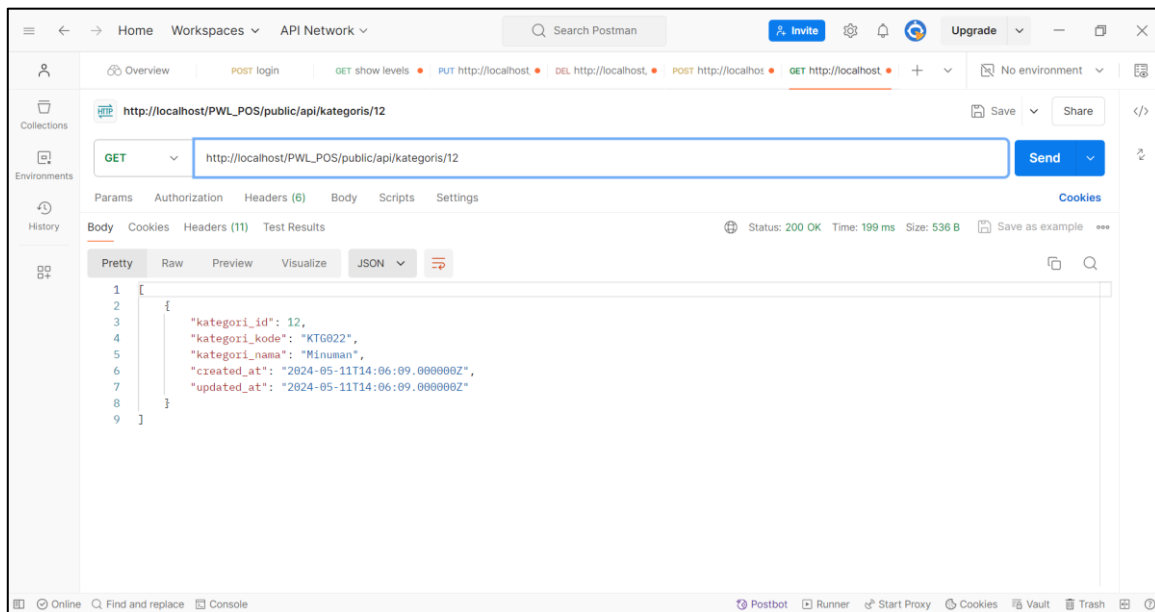
- Show daftar kategori



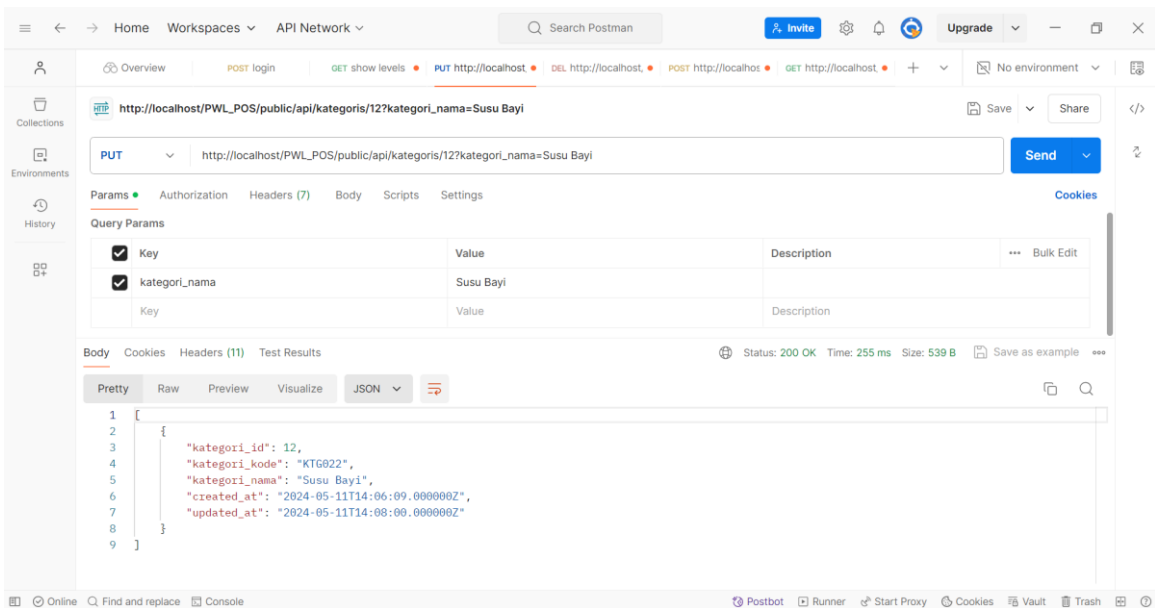
- Tambah categories



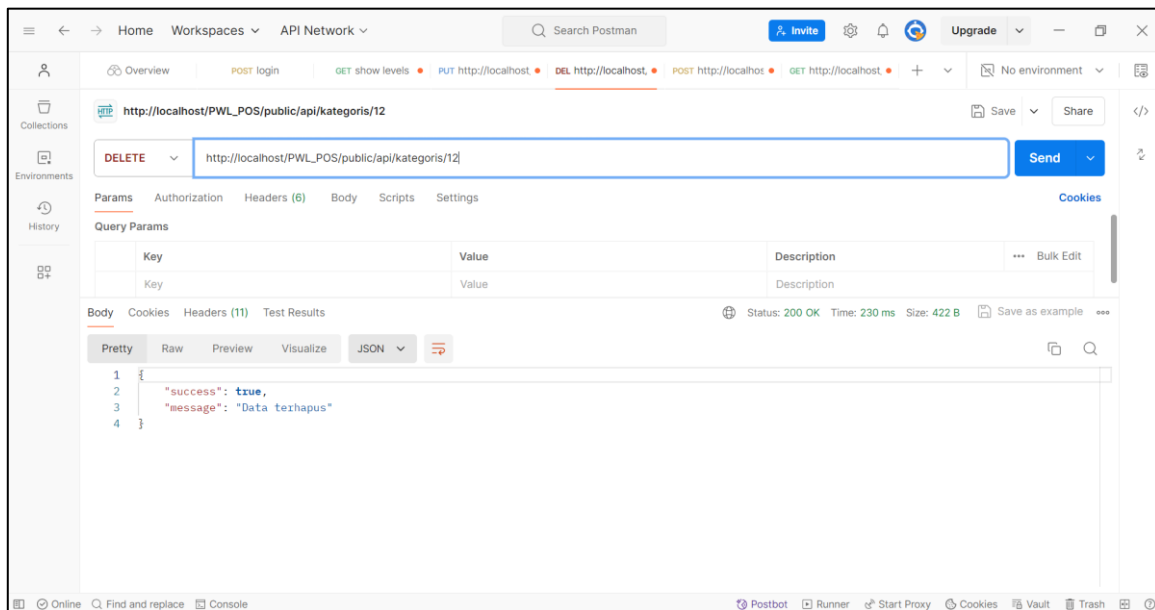
- Show detail kategoris



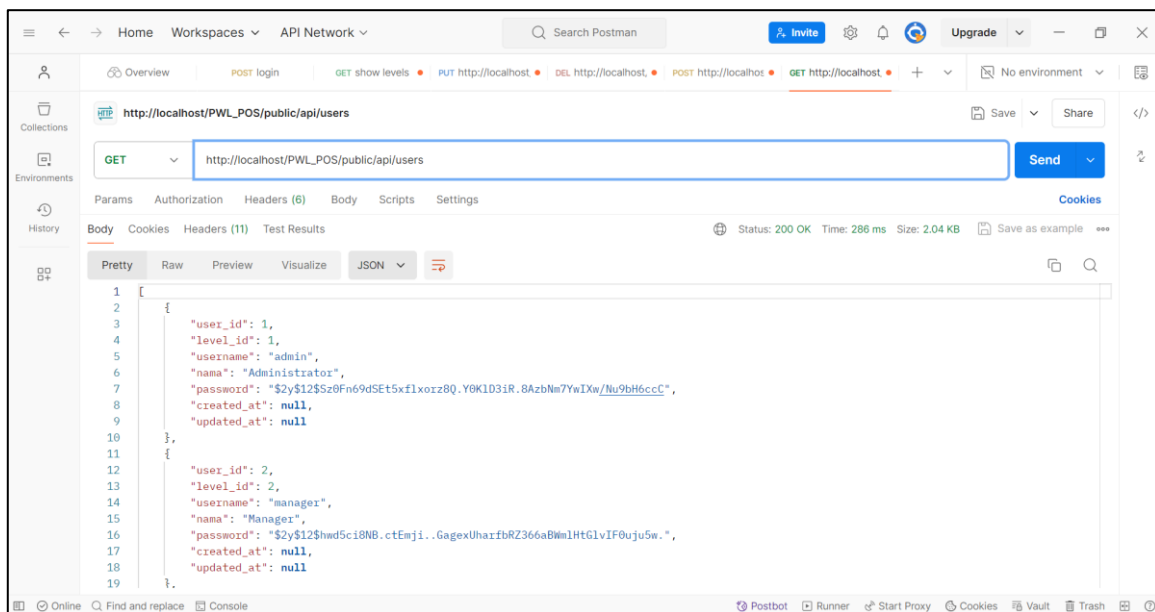
- Edit kategoris



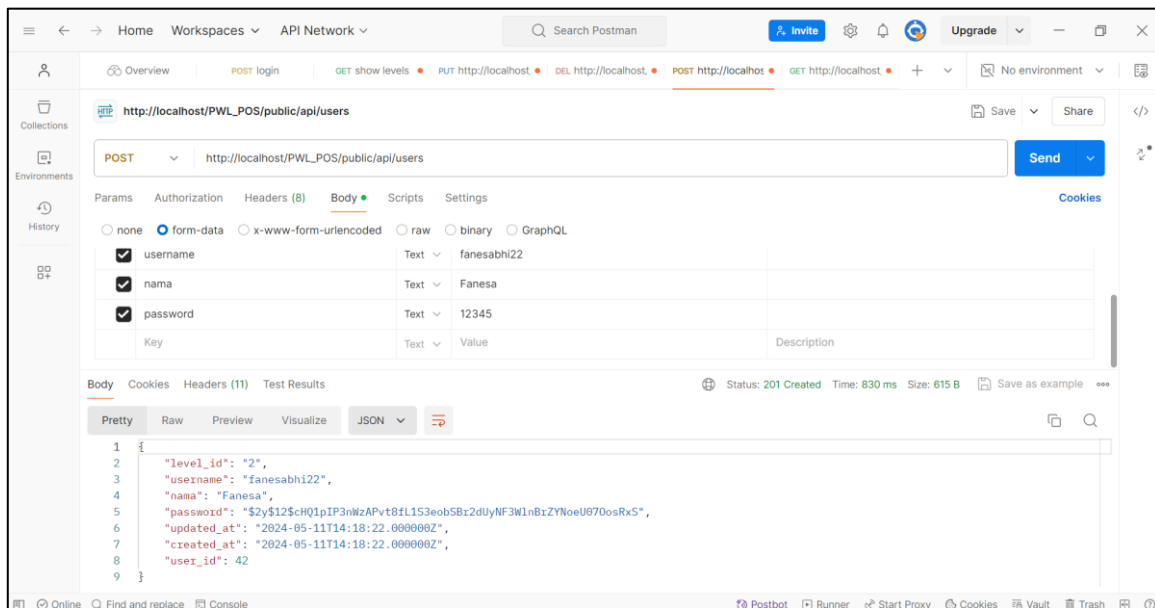
- Delete kategoris



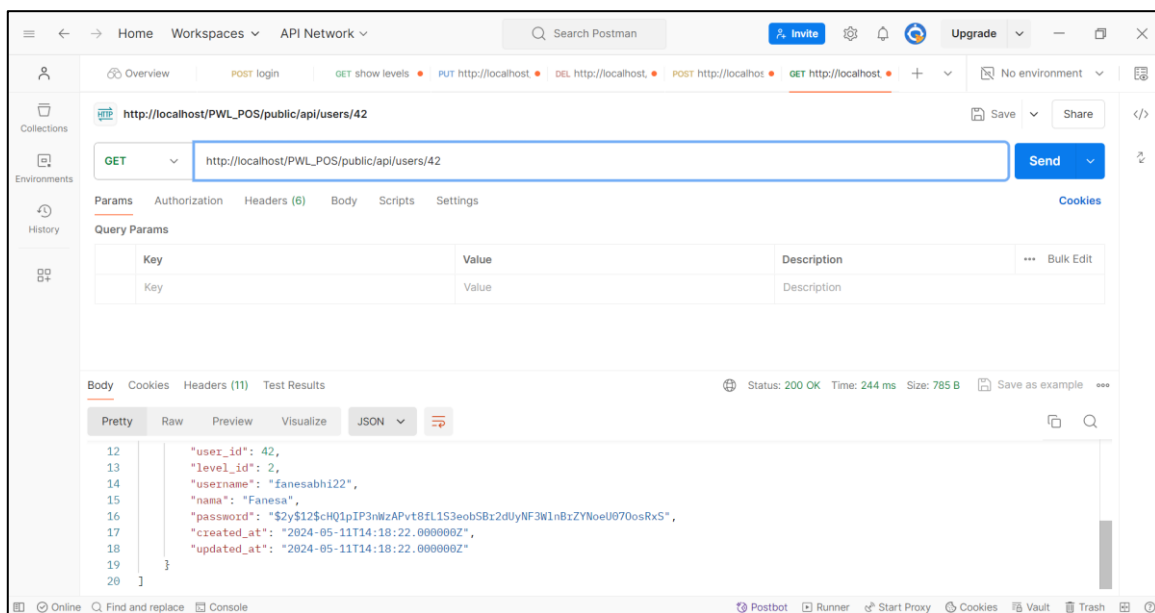
- Show daftar users



- Tambah users



- Show detail users



- Edit users

The screenshot shows the Postman interface with a PUT request to `http://localhost/PWL_POS/public/api/users/42?nama=Fanesabhirawaning`. The request is configured with the following details:

- Method:** PUT
- URL:** `http://localhost/PWL_POS/public/api/users/42?nama=Fanesabhirawaning`
- Params:**
 - Query Params:**

Key	Value	Description
nama	Fanesabhirawaning	
- Body:**

```

11 {
12   "user_id": 42,
13   "level_id": 2,
14   "username": "Fanesabhi22",
15   "nama": "Fanesabhirawaning",
16   "password": "$2y$12$cH01pTP3nMzAPvt8fL1S3eobSBz2dUyNF3WlnBzZYNoeU870osRx5",
17   "created_at": "2024-05-11T14:18:22.000000Z",
18   "updated_at": "2024-05-11T14:19:44.000000Z"
19 }
20

```
- Status:** 200 OK, Time: 279 ms, Size: 796 B

- Delete users

The screenshot shows the Postman interface with a DELETE request to `http://localhost/PWL_POS/public/api/users/42`. The request is configured with the following details:

- Method:** DELETE
- URL:** `http://localhost/PWL_POS/public/api/users/42`
- Params:**
 - Query Params:**

Key	Value	Description
Key	Value	Description
- Body:**

```

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

```
- Status:** 200 OK, Time: 358 ms, Size: 422 B