Nama	**: Dini Elminingtyas**
Kelas/No.absen	**: SIB 2B / 07**
NIM	**: 2341760180**

PEMROGRAMAN WEB LANJUT

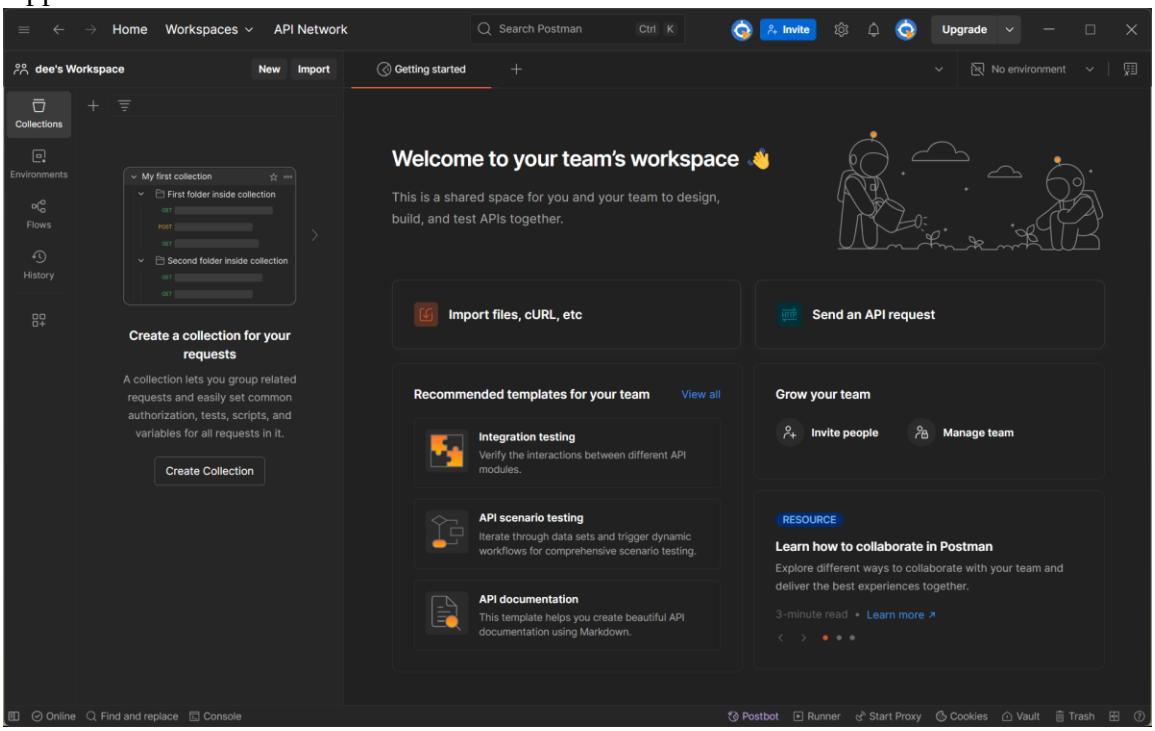
JOBSHEET 10

RESTFULL API

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.

   ⇨ Apps



2. Lakukan instalasi JWT dengan mengetikkan perintah berikut:
   ```
   composer require tymon/jwt-auth:2.1.1
   ```
   Pastikan Anda terkoneksi dengan internet.

   ⇨ Source code

```
PS D:\laragon\www\Pemrograman_Web_Lanjut_2025\Minggu 10 (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
```

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

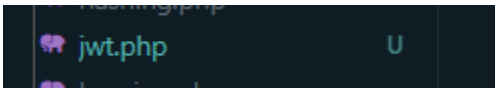
```
php artisan vendor:publish --
provider="Tymon\JWTAuth\Providers\LaravelServiceProvider"
```

⇨ Source code

```
PS D:\laragon\www\Pemrograman_Web_Lanjut_2025\Minggu 10 (PWL_POS)> php artisan vendor:publish --provider="Tymon\JWTAuth\Providers\LaravelServiceProvider"

  INFO  Publishing assets.

  Copying file [D:\laragon\www\Pemrograman_Web_Lanjut_2025\Minggu 10 (PWL_POS)\vendor\tymon\jwt-auth\config\config.php] to [D:\laragon\www\Pemrograman_Web_Lanjut_2025\Minggu 10 (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.

⇨ File



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

```
php artisan jwt:secret
```

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

⇨ Source code

```
PS D:\laragon\www\Pemrograman_Web_Lanjut_2025\Minggu 10 (PWL_POS)> php artisan jwt:secret
jwt-auth secret [fDsIN6RxfbmpDEjgSAzWATCvxh34QoqCYgdm0ehYZ8m74vhvBrqZ61BsRkQRAJpU] set successfully.
```

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

```
'guards' => [
    'web' => [
        'driver' => 'session',
        'provider' => 'users',
    ],
    'api' => [
        'driver' => 'jwt',
        'provider' => 'users',
    ],
],
```

⇨ Source code

```
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:

```php
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Model;
use Tymon\JWTAuth\Contracts\JWTSubject;
use Illuminate\Foundation\Auth\User as Authenticatable;

class UserModel extends Authenticatable implements JWTSubject
{

    public function getJWTIdentifier(){
        return $this->getKey();
    }

    public function getJWTCustomClaims(){
        return [];
    }

    protected $table = 'm_user';
    protected $primaryKey = 'user_id';
```

⇨ Source code

app > Models > UserModel.php > UserModel

```php
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Model;
use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Relations\BelongsTo;
use Tymon\JWTAuth\Contracts\JWTSubject;
use Illuminate\Foundation\Auth\User as Authenticatable; // implementasi class Authenticatable

class UserModel extends Authenticatable implements JWTSubject
{
    use HasFactory;
    public function getJWTIdentifier(){
        return $this->getKey();
    }

    public function getJWTCustomClaims(){
        return [];
    }

    protected $table = 'm_user';
    protected $primaryKey = 'user_id';
    protected $fillable = ['username', 'password', 'nama', 'level_id', 'created_at', 'updated_at', 'user_profile_picture'];
    protected $hidden = ['password']; // jangan di tampilkan saat select
    protected $casts = ['password' => 'hashed']; // casting password agar otomatis di hash

```

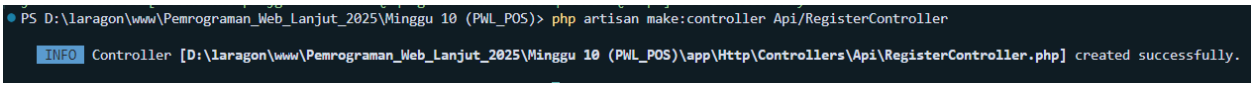
Hasil yang didapatkan adalah menampilkan halaman form login dikarenakan ketika pada tahapan sebelumnya tidak mengirimkan ke dalam server untuk login. Sehingga pada pengujian hanya menampilkan halaman form login.

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

   ```
   php artisan make:controller Api/RegisterController
   ```

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

   ⇨ Source code



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

```php
<?php

namespace App\Http\Controllers\Api;

use App\Models\UserModel;
use App\Http\Controllers\Controller;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Validator;

class RegisterController extends Controller
{
    public function __invoke(Request $request)
    {
```

   ⇨ Source code

app > Http > Controllers > Api > RegisterController.php > ...

```php
<?php

namespace App\Http\Controllers\Api;

use App\Models\UserModel;
use App\Http\Controllers\Controller;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Validator;

class RegisterController extends Controller
{
    public function __invoke(Request $request)
    {
        //set validation
        $validator = Validator::make($request->all(), [
            'username' => 'required',
            'nama' => 'required',
            'password' => 'required|min:5|confirmed',
            'level_id' => 'required'
        ]);

        //if validations fails
        if($validator->fails()){
            return response()->json($validator->errors(), 422);
        }

        //create user
        $user = UserModel::create([
            'username' => $request->username,
            'nama' => $request->nama,
            'password' => bcrypt($request->password),
            'level_id' => $request->level_id,
        ]);

        //return response JSON user is created
        if($user){
            return response()->json([
                'success' => true,
                'user' => $user,
            ], 201);
        }

        //return JSON process insert failed
        return response()->json([
            'success' => false,
        ], 409);
    }
}
```

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

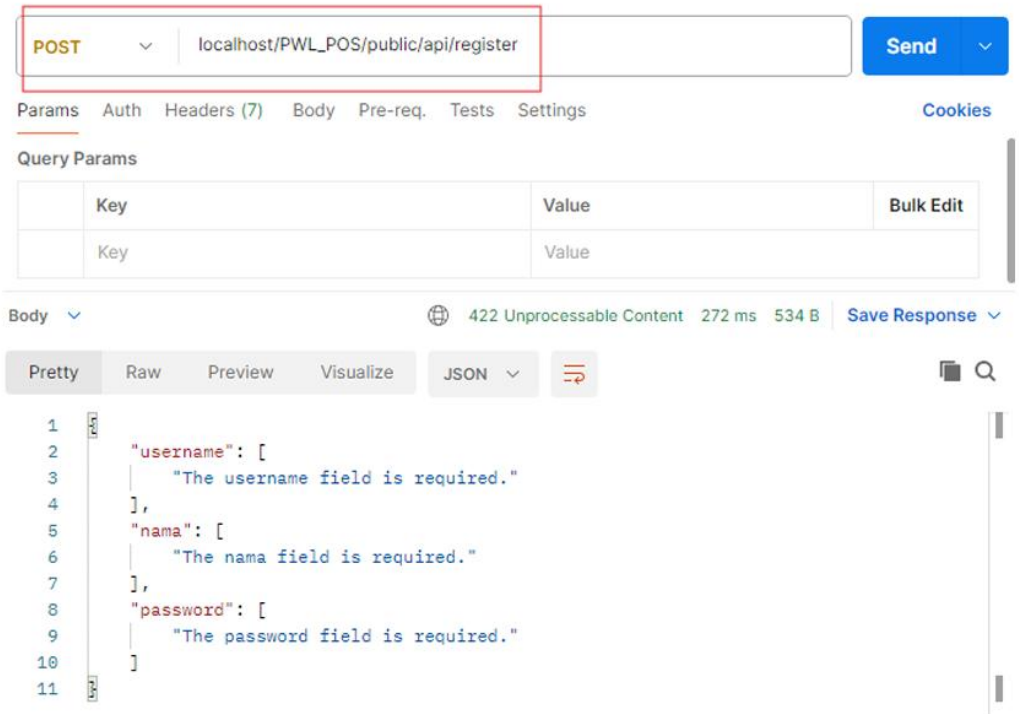
```php
<?php

use App\Http\Controllers\Api\RegisterController;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Route;

/*
|--------------------------------------------------------------------------
| API Routes
|--------------------------------------------------------------------------
|
| Here is where you can register API routes for your application. These
| routes are loaded by the RouteServiceProvider and all of them will
| be assigned to the "api" middleware group. Make something great!
|
*/

Route::post('/register', App\Http\Controllers\Api\RegisterController::class)->name('register');
```

⇨ Source code

routes > api.php

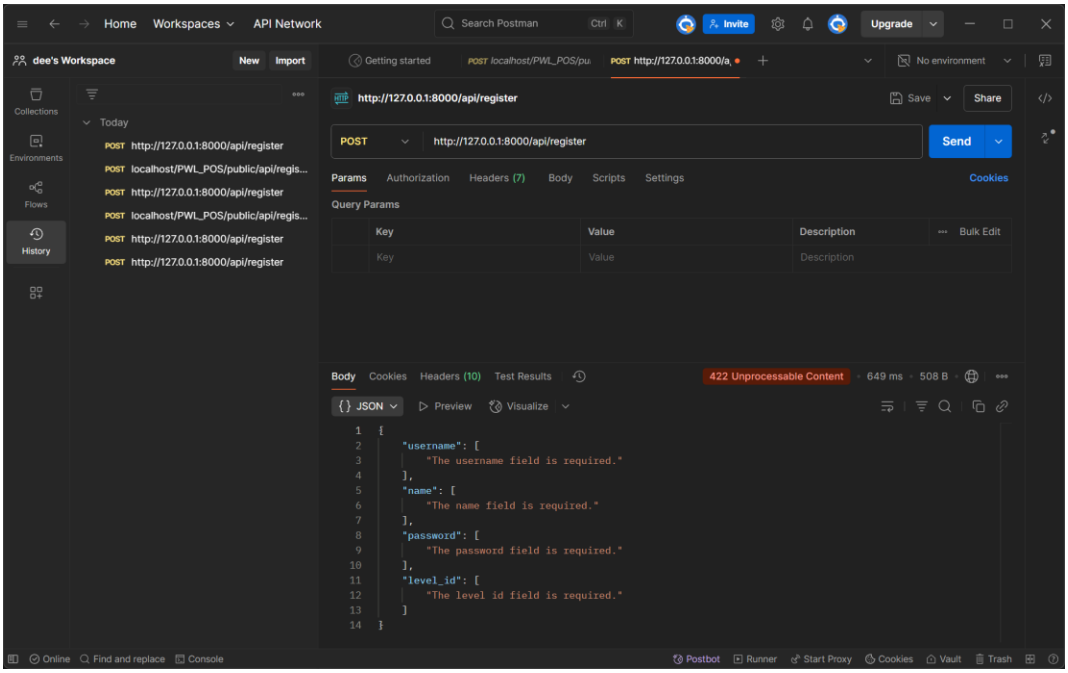
```php
<?php

use App\Http\Controllers\Api\RegisterController;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Route;

/*
|--------------------------------------------------------------------------
| API Routes
|--------------------------------------------------------------------------
|
| Here is where you can register API routes for your application. These
| routes are loaded by the RouteServiceProvider and all of them will
| be assigned to the "api" middleware group. Make something great!
|
*/

Route::post('/register', App\Http\Controllers\Api\RegisterController::class)->name('register');

```

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.

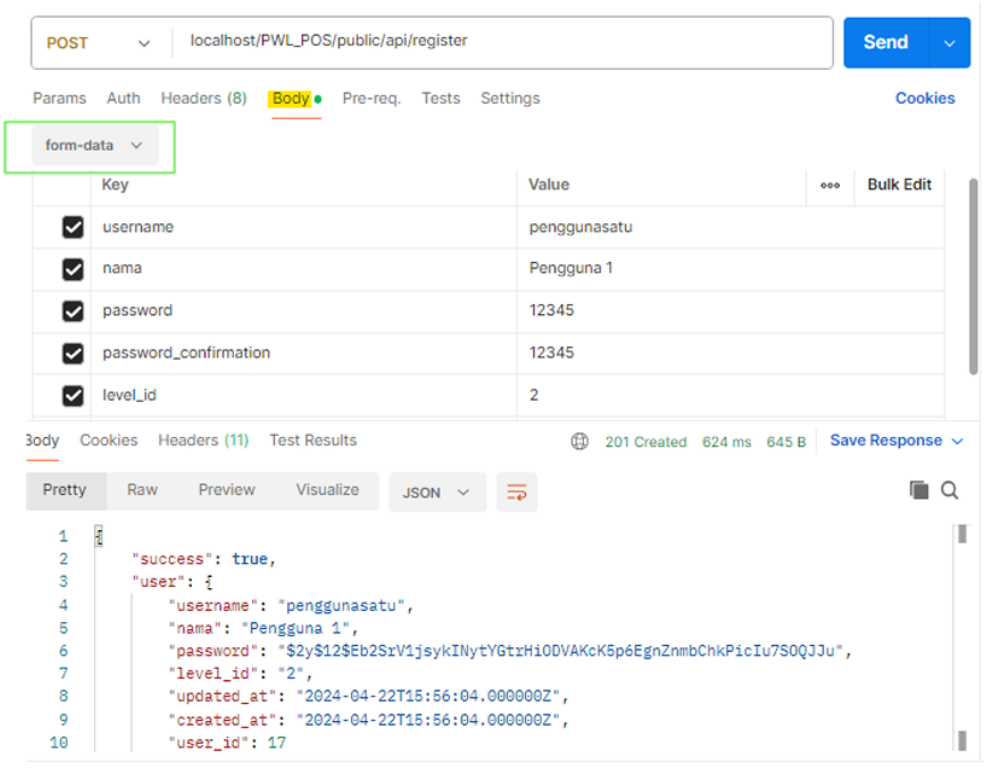


Jika berhasil akan muncul error validasi seperti gambar di atas.
Lakukan percobaan yang sama dan berikan screenshoot hasil percobaan Anda.

⇨ postman



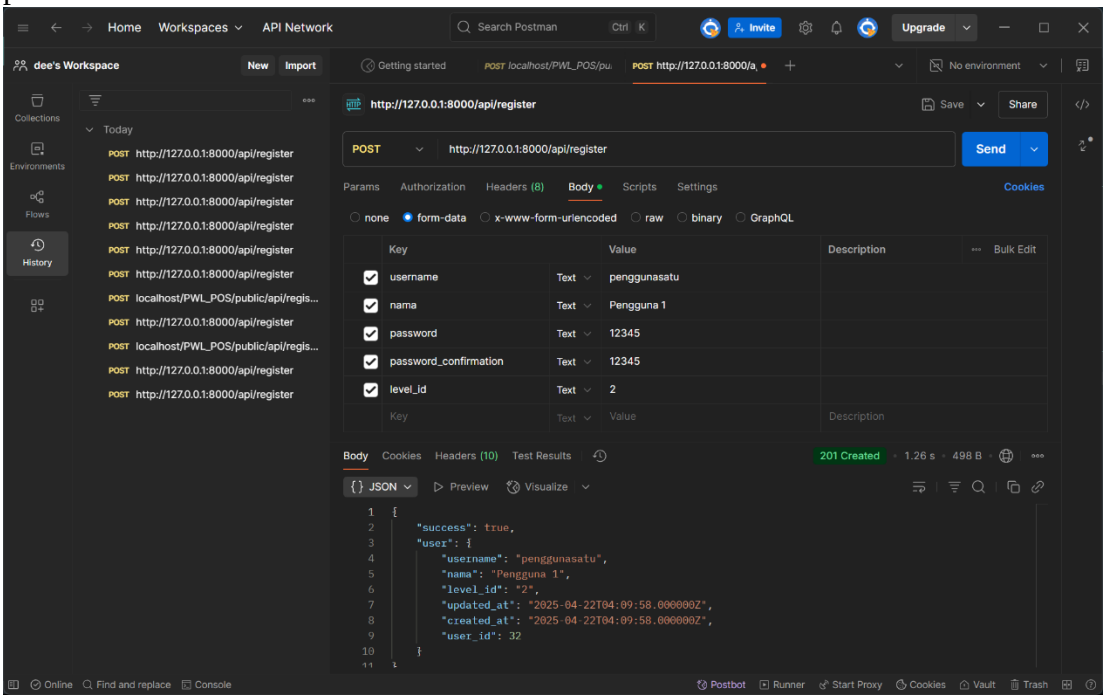
12. Sekarang kita coba masukkan data. Klik tab Body dan pilih form-data. Isikan key sesuai dengan kolom data, serta isikan data registrasi menggunakan nilai yang Anda inginkan.



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

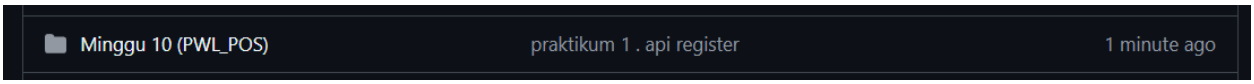
Lakukan percobaan yang sama dan berikan screenshoot hasil percobaan Anda.

⇨ postman



13. Lakukan commit perubahan file pada Github.

⇨ commit



Praktikum 2 – Membuat RESTful API Login

1. 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.

   ⇨ create controller

```
PS D:\laragon\www\Pemrograman_Web_Lanjut_2025\Minggu 10 (PWL_POS)> php artisan make:controller Api/LoginController

  INFO  Controller [D:\laragon\www\Pemrograman_Web_Lanjut_2025\Minggu 10 (PWL_POS)\app\Http\Controllers\Api\LoginController.php] created successfully.
```

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

```php
<?php

namespace App\Http\Controllers\Api;

use App\Http\Controllers\Controller;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Validator;

```

   ⇨ controller

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

```php
<?php
namespace App\Http\Controllers\Api;

use App\Http\Controllers\Controller;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Validator;

class LoginController extends Controller
{
    public function __invoke(Request $request)
    {
        //set validation
        $validator = Validator::make($request->all(), [
            'username' => 'required',
            'password' => 'required',
        ]);

        //if validation fails
        if ($validator->fails()) {
            return response()->json($validator->errors(), 422);
        }

        //get credentials from request
        $credentials = $request->only('username', 'password');

        //if auth failed
        if(!$token = auth()->guard('api')->attempt($credentials)) {
            return response()->json([
                'success' => false,
                'message' => 'Username atau Password Anda salah'
            ], 401);
        }

        //if auth success
        return response()->json([
            'success' => true,
            'user'    => auth()->guard('api')->user(),
            'token'   => $token
        ], 200);
    }
}
```

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

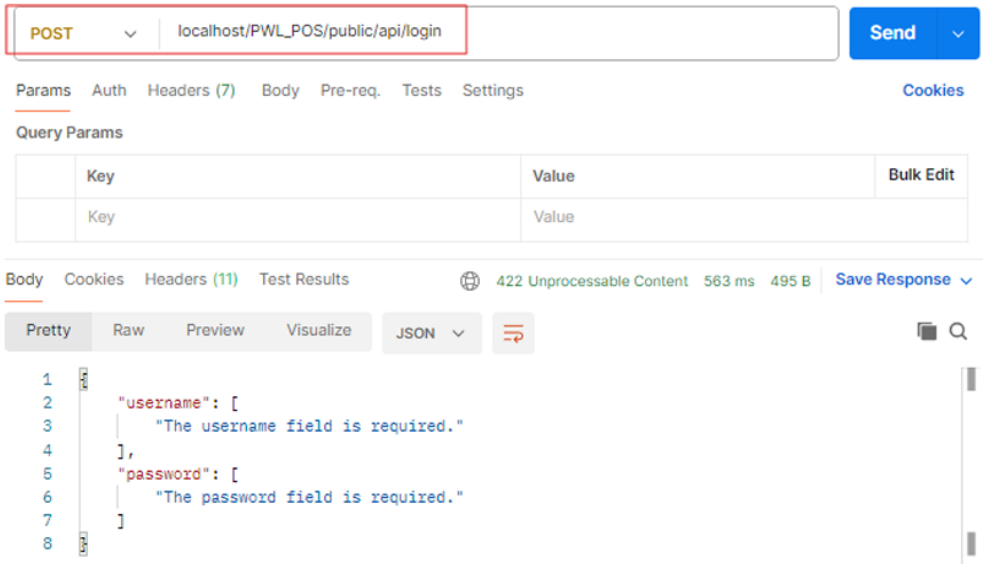
```php
use App\Http\Controllers\Api\LoginController;

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

⇨ route

routes > api.php > ...

```php
<?php

use App\Http\Controllers\Api\RegisterController;
use App\Http\Controllers\Api\LoginController;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Route;

/*
|--------------------------------------------------------------------------
| API Routes
|--------------------------------------------------------------------------
|
| Here is where you can register API routes for your application. These
| routes are loaded by the RouteServiceProvider and all of them will
| be assigned to the "api" middleware group. Make something great!
|
*/

Route::post('/register', App\Http\Controllers\Api\RegisterController::class)->name('register');
Route::post('/login', App\Http\Controllers\Api\LoginController::class)->name('login');
Route::middleware('auth:api')->get('/user', function (Request $request) {
    return $request->user();
});

```

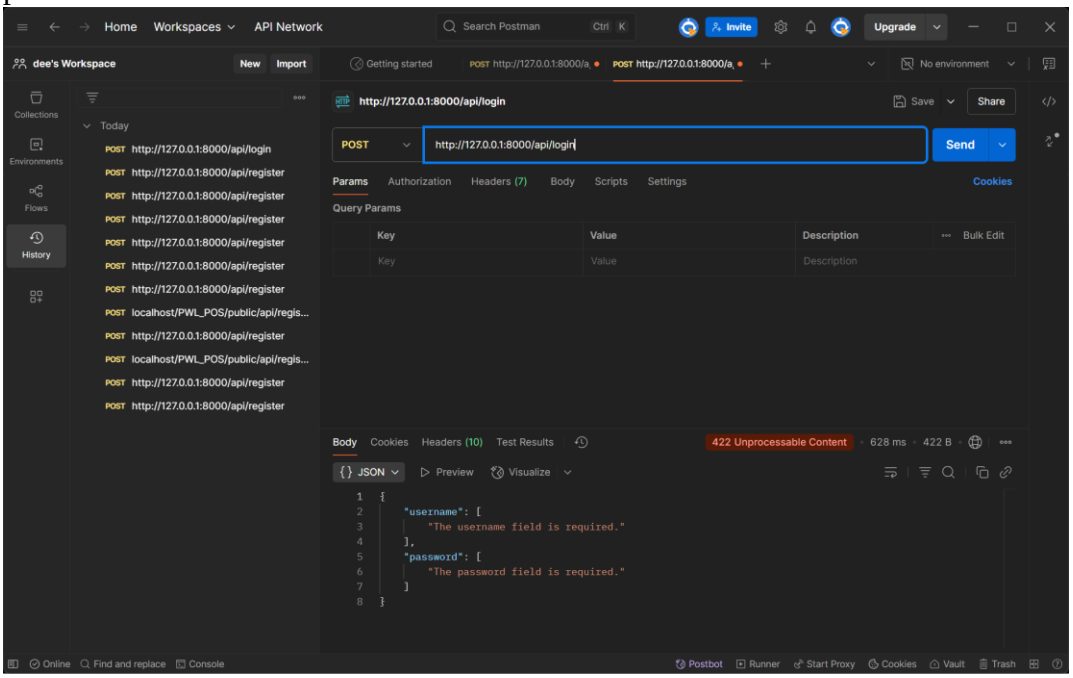
4. Jika sudah, kita akan melakukan uji coba REST API melalui aplikasi Postman. Buka aplikasi Postman, isi URL localhost/PWL_POS/public/api/login serta method POST. Klik Send.



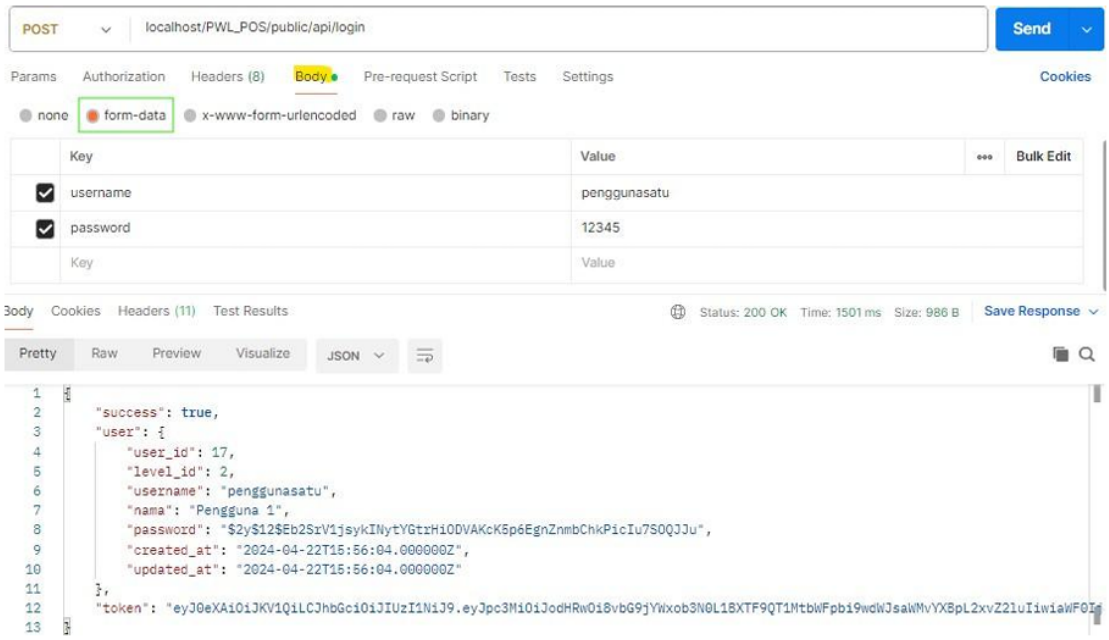
Jika berhasil akan muncul error validasi seperti gambar di atas.

Lakukan percobaan yang sama dan berikan screenshoot hasil percobaan Anda.

⇨ postman

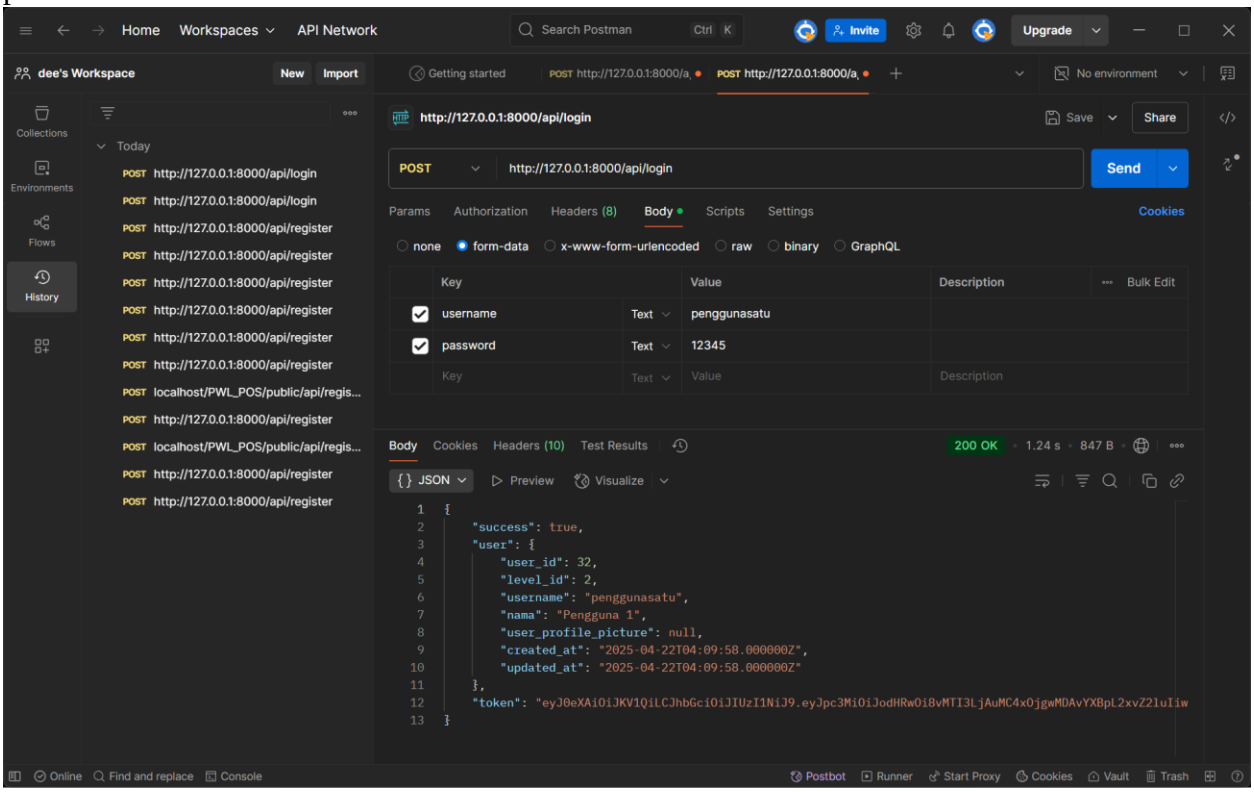


5. Selanjutnya, isikan username dan password sesuai dengan data user yang ada pada database. Klik tab Body dan pilih form-data. Isikan key sesuai dengan kolom data, serta isikan data user. Klik tombol Send, jika berhasil maka akan keluar tampilan seperti berikut. Copy nilai token yang diperoleh pada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



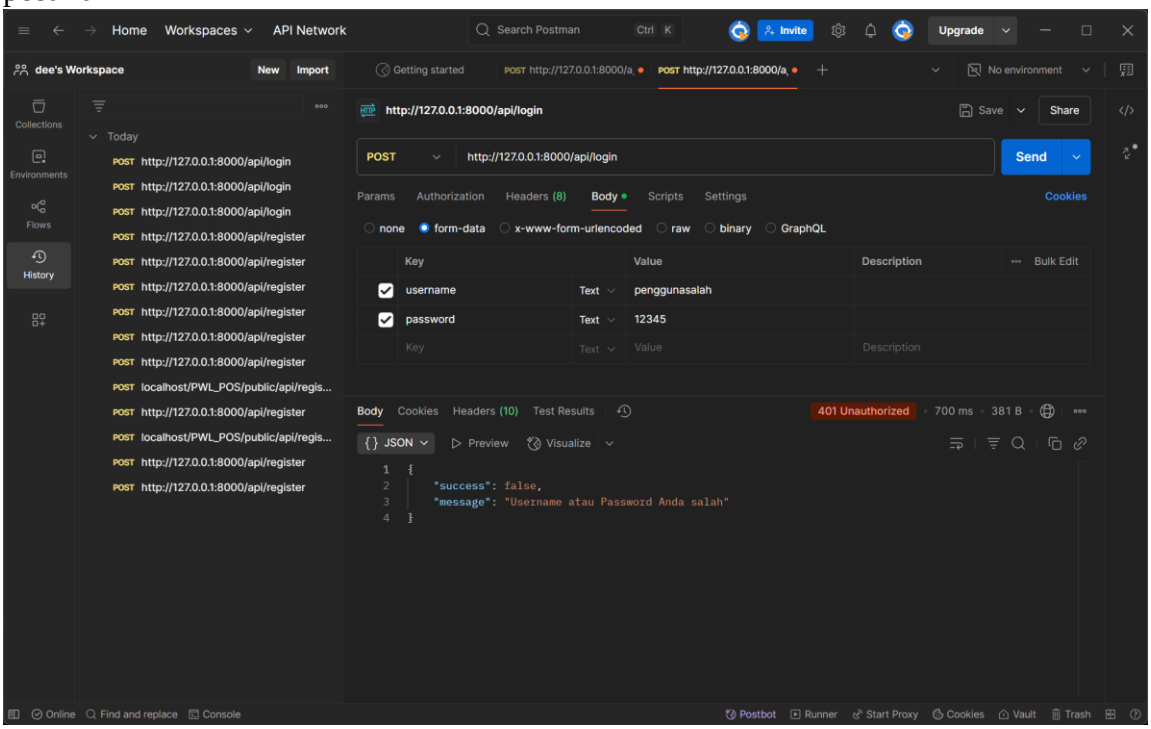
Lakukan percobaan yang sama dan berikan screenshoot hasil percobaan Anda.

⇨ postman



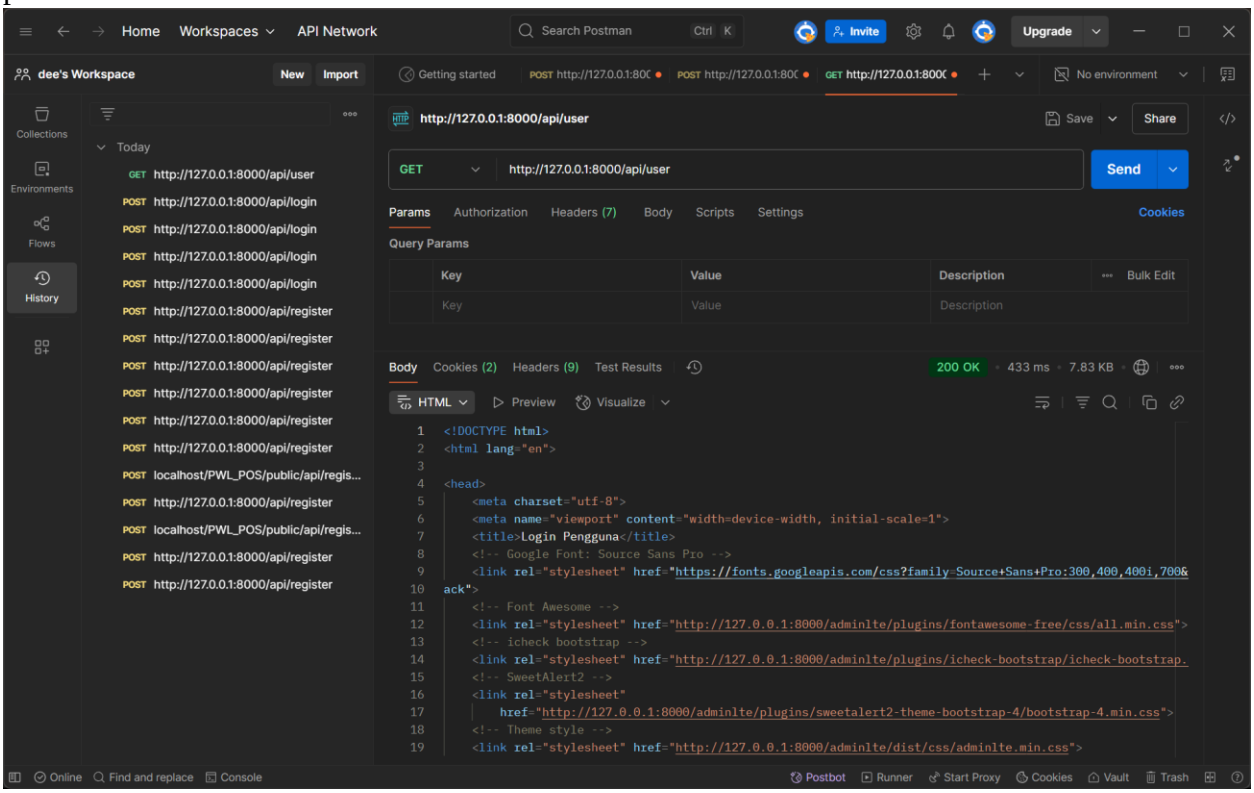
6. Lakukan percobaan yang untuk data yang salah dan berikan screenshoot hasil percobaan Anda.

⇨ postman



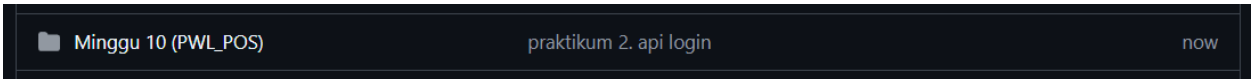
7. Coba kembali melakukan login dengan data yang benar. Sekarang mari kita coba menampilkan data user yang sedang login menggunakan URL localhost/PWL_POS/public/api/user dan method GET. Jelaskan hasil dari percobaan tersebut.

⇨ postman



8. Lakukan commit perubahan file pada Github.

⇨ commit



Praktikum 3 – Membuat RESTful API Logout

1. Tambahkan kode berikut pada file .env

```
JWT_SHOW_BLACKLIST_EXCEPTION=true
```

⇨ .env

```
JWT_SECRET=fDsIN6RxfbmpDEjgSAzWATCvxh34QoqCYgdm0ehYZ8m74vhvBrqZ61BsRkQRAJpU
JWT_SHOW_BLACKLIST_EXCEPTION=true
```

2. Buat Controller baru dengan nama LogoutController.

```
php artisan make:controller Api/LogoutController
```

⇨

```
PS D:\laragon\www\Pemrograman_Web_Lanjut_2025\Minggu 10 (PWL_POS)> php artisan make:controller Api/LogoutController

  INFO  Controller [D:\laragon\www\Pemrograman_Web_Lanjut_2025\Minggu 10 (PWL_POS)\app\Http\Controllers\Api\LogoutController.php] created successfully.
```

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

```php
<?php

namespace App\Http\Controllers\Api;
use Illuminate\Http\Request;
use App\Http\Controllers\Controller;
use Tymon\JWTAuth\Facades\JWTAuth;
use Tymon\JWTAuth\Exceptions\JWTException;
use Tymon\JWTAuth\Exceptions\TokenExpiredException;
use Tymon\JWTAuth\Exceptions\TokenInvalidException;

class LogoutController extends Controller
{
    public function __invoke(Request $request)
    {
        //remove token
        $removeToken = JWTAuth::invalidate(JWTAuth::getToken());

        if($removeToken) {
            //return response JSON
            return response()->json([
                'success' => true,
                'message' => 'Logout Berhasil!',
            ]);
        }
    }
}
```

⇨ source code

app > Http > Controllers > Api > LogoutController.php > LogoutController

```php
<?php

namespace App\Http\Controllers\Api;

use Illuminate\Http\Request;
use App\Http\Controllers\Controller;
use Tymon\JWTAuth\Facades\JWTAuth;
use Tymon\JWTAuth\Exceptions\JWTException;
use Tymon\JWTAuth\Exceptions\TokenExpiredException;
use Tymon\JWTAuth\Exceptions\TokenInvalidException;

class LogoutController extends Controller
{
    public function __invoke(Request $request)
    {
        //remove token
        $removeToken = JWTAuth::invalidate(JWTAuth::getToken());

        if($removeToken) {
            //return response JSON
            return response()->json([
                'success' => true,
                'message' => 'Logout Berhasil!',
            ]);
        }
    }
}
```

4. Lalu kita tambahkan routes pada api.php

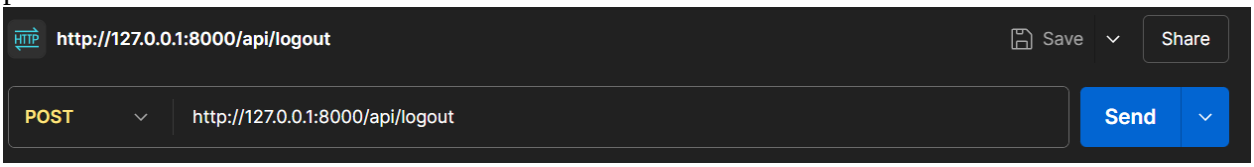
```php
Route::post('/logout', App\Http\Controllers\Api\LogoutController::class)->name('logout');
```

⇨ route

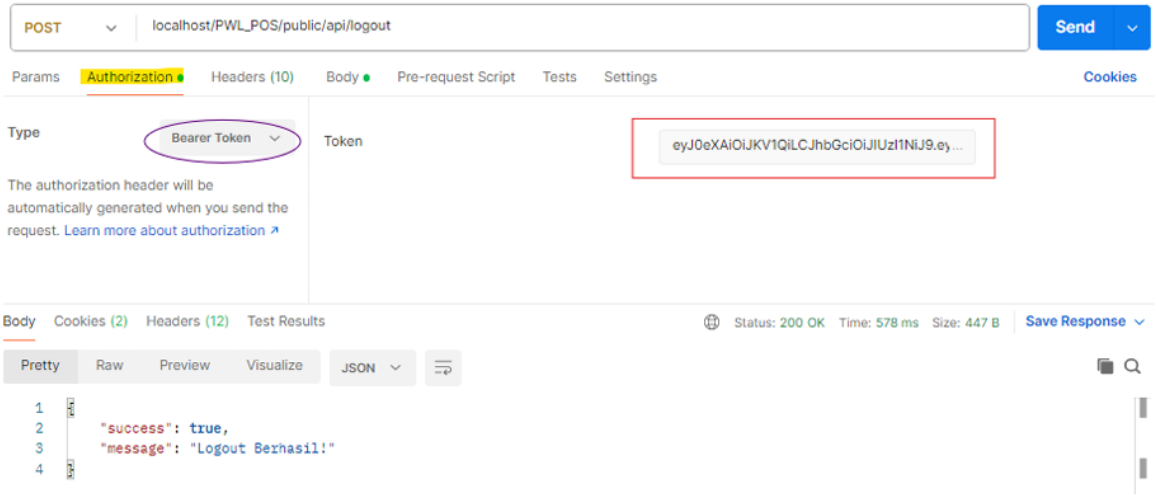
```php
Route::post('/logout', App\Http\Controllers\Api\LogoutController::class)->name('logout');
```

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.

⇨ postman

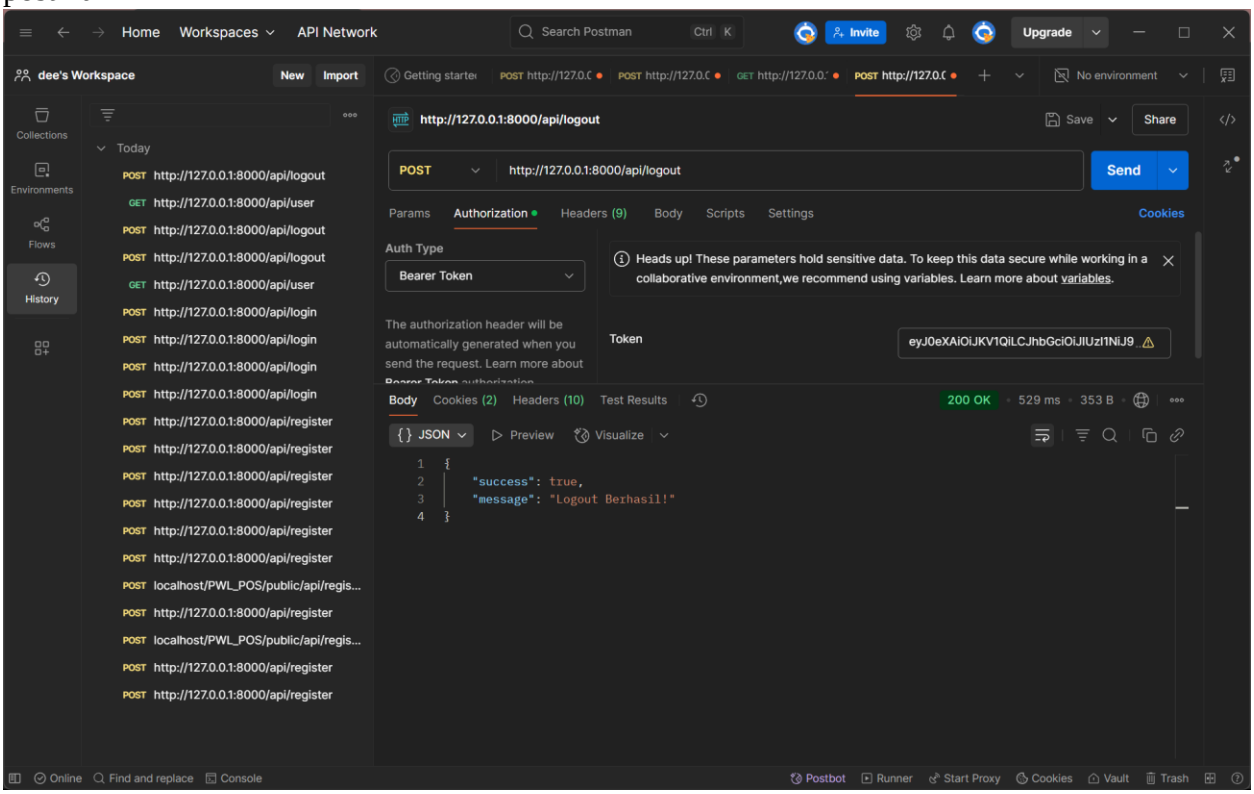


6. Isi token pada tab Authorization, pilih Type yaitu Bearer Token. Isikan token yang didapat saat login. Jika sudah klik Send.



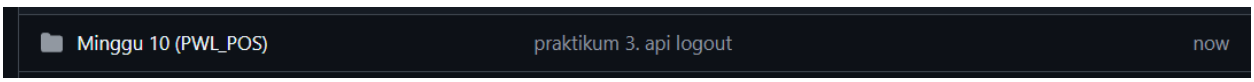
Lakukan percobaan yang sama dan berikan screenshoot hasil percobaan Anda.

⇨ postman



7. Lakukan commit perubahan file pada Github.

⇨ commit



Praktikum 4 – Implementasi CRUD dalam RESTful API

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

```
php artisan make:controller Api/LevelController
```

⇨ make controller

```
PS D:\laragon\www\Pemrograman_Web_Lanjut_2025\Minggu 10 (PWL_POS)> php artisan make:controller Api/LevelController

  INFO  Controller [D:\laragon\www\Pemrograman_Web_Lanjut_2025\Minggu 10 (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.

```php
namespace App\Http\Controllers\Api;
use App\Http\Controllers\Controller;
use Illuminate\Http\Request;
use App\Models\LevelModel;

class LevelController extends Controller
{
    public function index()
    {
        return LevelModel::all();
    }
```

⇨ controller

app > Http > Controllers > Api > LevelController.php > ...

```php
<?php

namespace App\Http\Controllers\Api;

use App\Http\Controllers\Controller;
use App\Models\LevelModel;
use Illuminate\Http\Request;

class LevelController extends Controller
{
    public function index(){
        return LevelModel::all();
    }
    public function store(Request $request)
    {
        $level = LevelModel::create($request->all());
        return response()->json($level, 201);
    }

    public function show(LevelModel $level)
    {
        return LevelModel::find($level);
    }

    public function update(Request $request, LevelModel $level)
    {
        $level->update($request->all());
        return LevelModel::find($level);
    }

}
```

3. Kemudian kita lengkapi routes pada api.php.

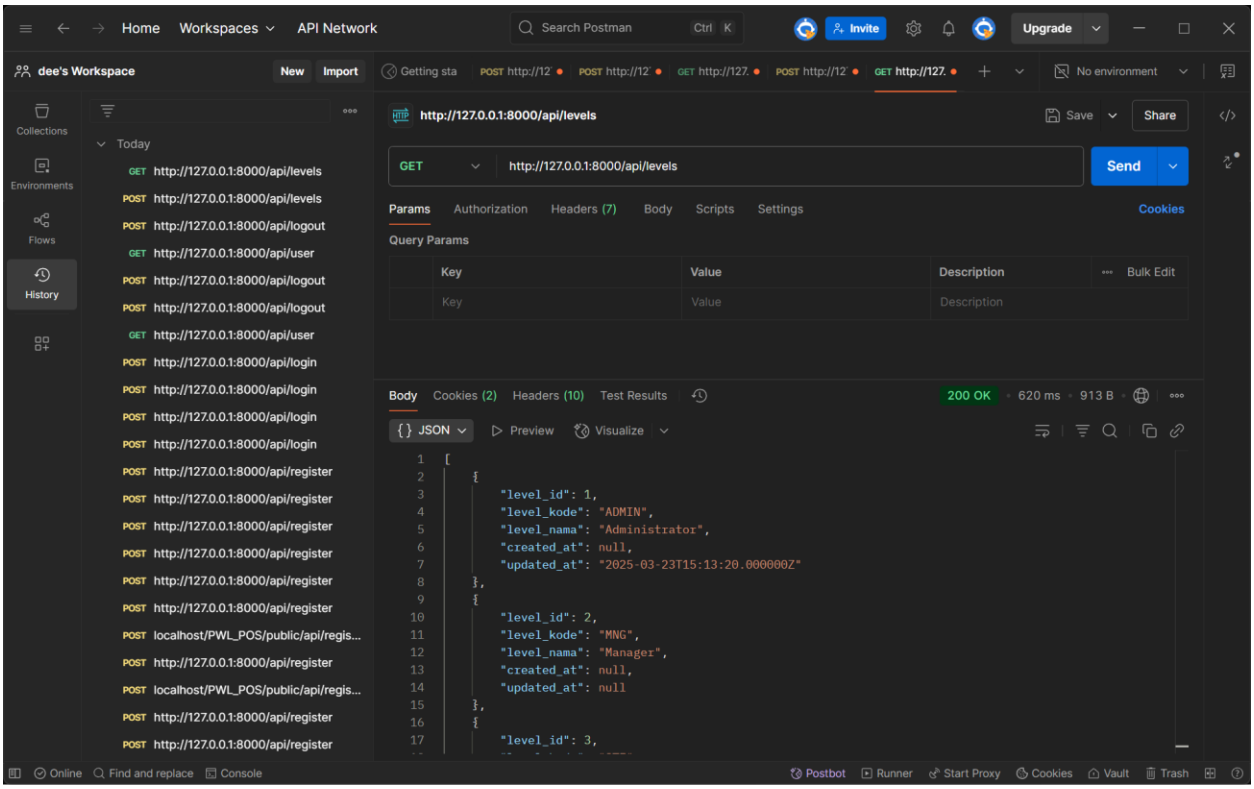
```php
use App\Http\Controllers\Api\LevelController;

Route::get('levels', [LevelController::class, 'index']);
Route::post('levels', [LevelController::class, 'store']);
Route::get('levels/{level}', [LevelController::class, 'show']);
Route::put('levels/{level}', [LevelController::class, 'update']);
Route::delete('levels/{level}', [LevelController::class, 'destroy']);
```

⇨ routes

```php
use App\Http\Controllers\Api\LoginController;
use App\Http\Controllers\Api\LevelController;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Route;

/*
|--------------------------------------------------------------------------
| API Routes
|--------------------------------------------------------------------------
|
| Here is where you can register API routes for your application. These
| routes are loaded by the RouteServiceProvider and all of them will
| be assigned to the "api" middleware group. Make something great!
|
*/

Route::post('/register', App\Http\Controllers\Api\RegisterController::class)->name('register');
Route::post('/login', App\Http\Controllers\Api\LoginController::class)->name('login');
Route::middleware('auth:api')->get('/user', function (Request $request) {
    return $request->user();
});
Route::post('/logout', App\Http\Controllers\Api\LogoutController::class)->name('logout');

Route::get('levels', [LevelController::class, 'index']);
Route::post('levels', [LevelController::class, 'store']);
Route::get('levels/{level}', [LevelController::class, 'show']);
Route::put('levels/{level}', [LevelController::class, 'update']);
Route::delete('levels/{level}', [LevelController::class, 'destroy']);
```

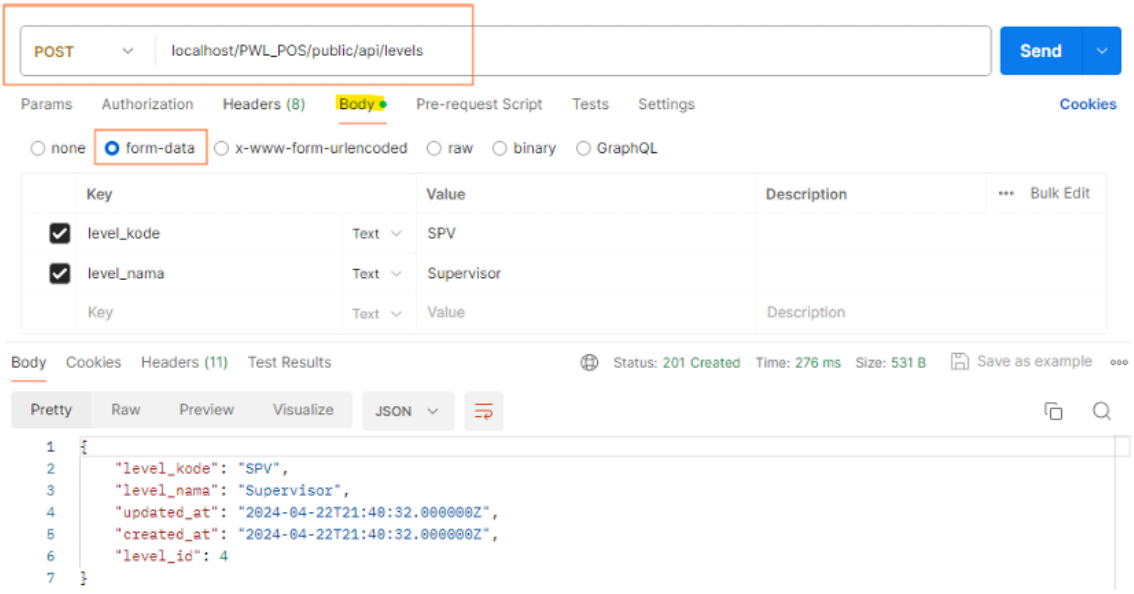
4. Jika sudah. Lakukan uji coba API mulai dari fungsi untuk menampilkan data. Gunakan URL: localhost/PWL_POS-main/public/api/levels dan method GET. Jelaskan dan berikan screenshoot hasil percobaan Anda.

⇨ postman



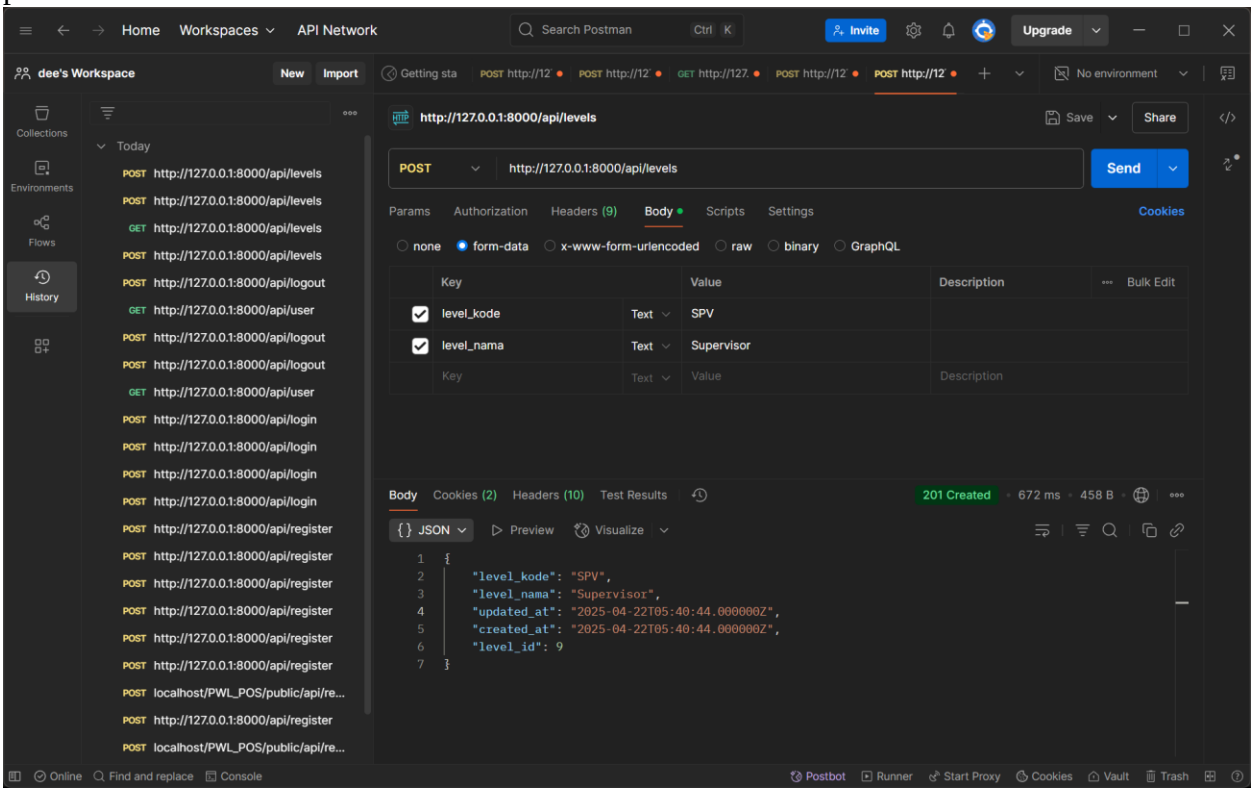
Pengujian yang dijalankan menggunakan method indeks dimana sistem mengambil semua data yang tersedia pada tabel level. Hal tersebut terjadi karena menggunakan sintaks all() untuk proses pengambilan datanya.

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



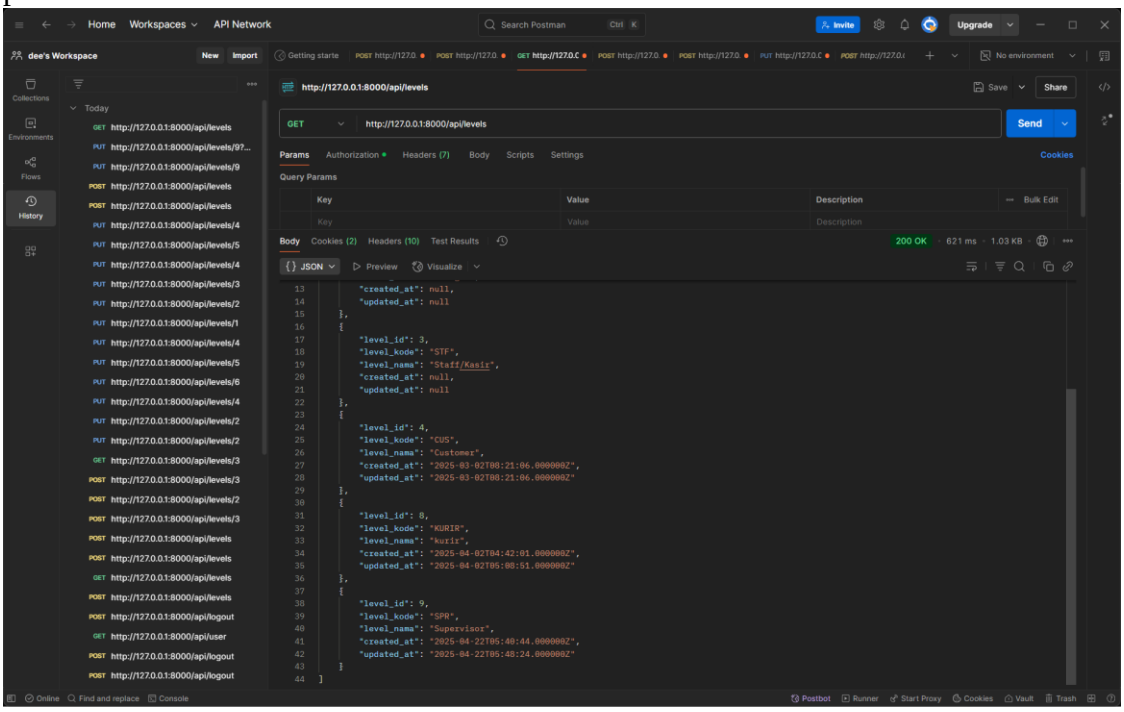
Jelaskan dan berikan screenshoot hasil percobaan Anda.

⇨ postman

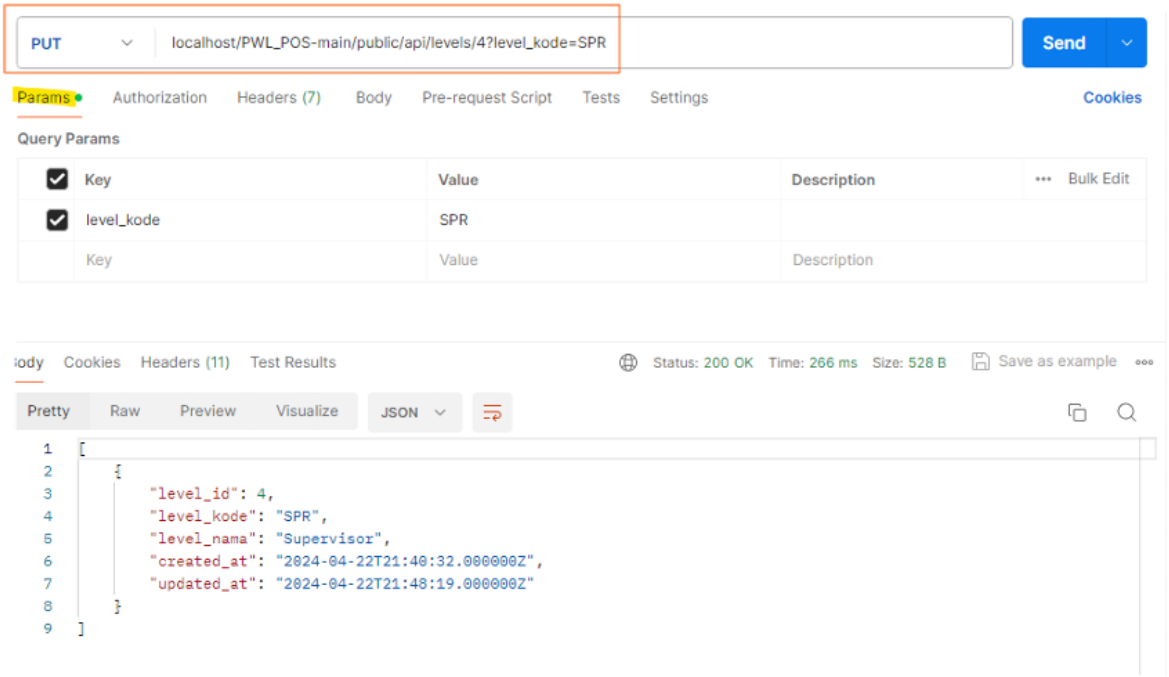


6. Berikutnya lakukan percobaan menampilkan detail data. Jelaskan dan berikan screenshoot hasil percobaan Anda.

⇨ postman

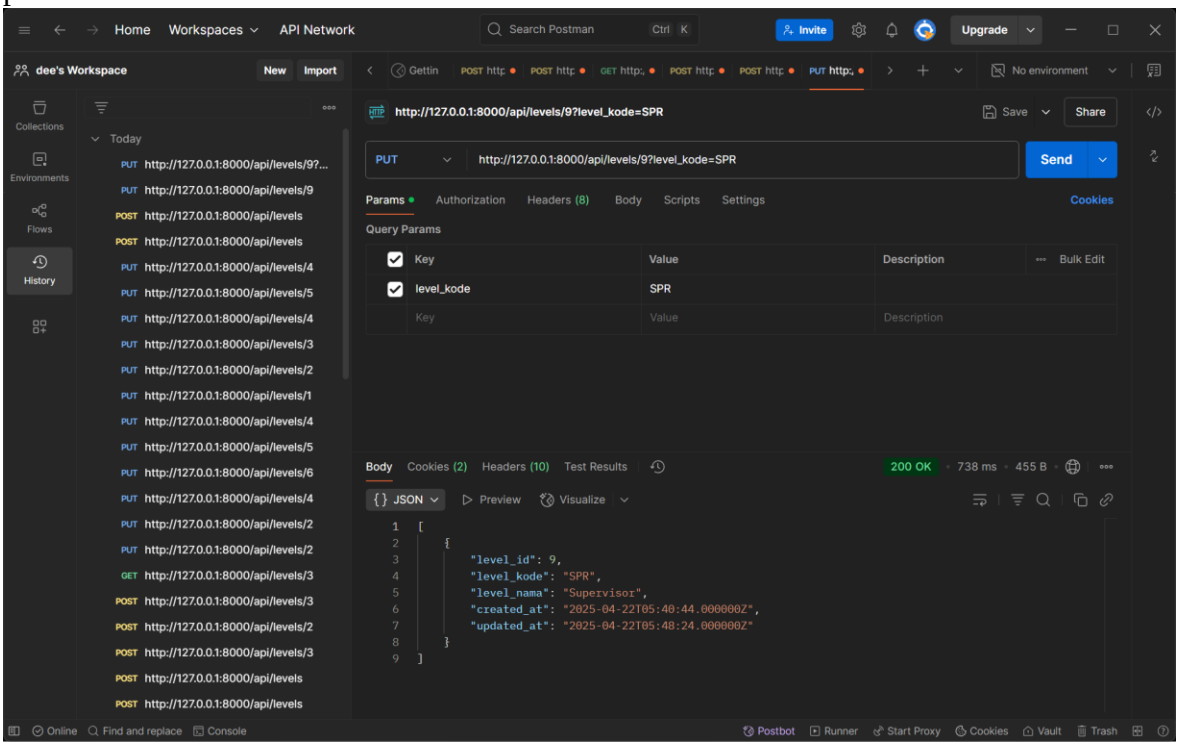


7. Jika sudah, kita coba untuk melakukan edit data menggunakan localhost/PWL_POS-main/public/api/levels/{id} dan method PUT. Isikan data yang ingin diubah pada tab Param.



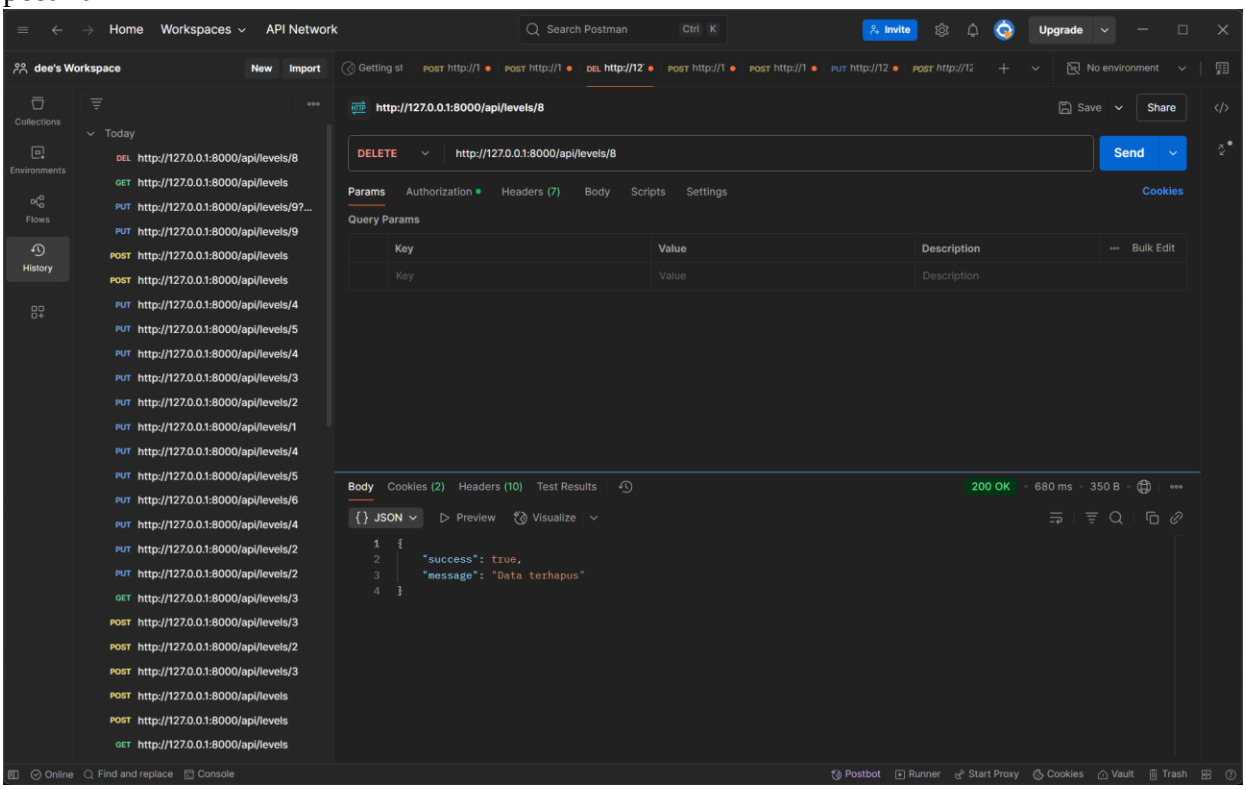
Jelaskan dan berikan screenshoot hasil percobaan Anda.

⇨ postman



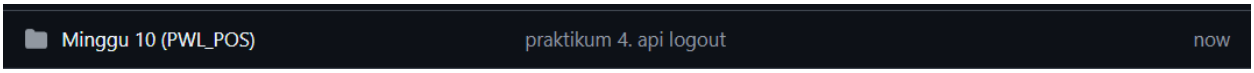
8. Terakhir lakukan percobaan hapus data. Jelaskan dan berikan screenshoot hasil percobaan Anda.

⇨ postman



9. Lakukan commit perubahan file pada Github.

⇨ commit



TUGAS

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

⇨ create controller

```
PS D:\laragon\www\Pemrograman_Web_Lanjut_2025\Minggu 10 (PWL_POS)> php artisan make:controller Api/UserController
  INFO  Controller [D:\laragon\www\Pemrograman_Web_Lanjut_2025\Minggu 10 (PWL_POS)\app\Http\Controllers\Api\UserController.php] created successfully.
PS D:\laragon\www\Pemrograman_Web_Lanjut_2025\Minggu 10 (PWL_POS)> php artisan make:controller Api/BarangController
  INFO  Controller [D:\laragon\www\Pemrograman_Web_Lanjut_2025\Minggu 10 (PWL_POS)\app\Http\Controllers\Api\BarangController.php] created successfully.
PS D:\laragon\www\Pemrograman_Web_Lanjut_2025\Minggu 10 (PWL_POS)> php artisan make:controller Api/KategoriController
  INFO  Controller [D:\laragon\www\Pemrograman_Web_Lanjut_2025\Minggu 10 (PWL_POS)\app\Http\Controllers\Api\KategoriController.php] created successfully.
```

⇨ controller

- user

app > Http > Controllers > Api > UserController.php > ...

```php
<?php

namespace App\Http\Controllers\Api;

use App\Http\Controllers\Controller;
use App\Models\UserModel;
use Illuminate\Http\Request;

class UserController extends Controller
{
    public function index()
    {
        return UserModel::all();
    }

    public function store(Request $request)
    {
        $user = UserModel::create($request->all());
        return response()->json($user, 201);
    }

    public function show(UserModel $user)
    {
        return $user;
    }

    public function update(Request $request, UserModel $user)
    {
        $user->update($request->all());
        return $user;
    }

    public function destroy(UserModel $user)
    {
        $user->delete();

        return response()->json([
            'success' => true,
            'message' => 'Data terhapus',
        ]);
    }
}
```

- barang

app > Http > Controllers > Api > BarangController.php > ...

```php
<?php

namespace App\Http\Controllers\Api;

use App\Http\Controllers\Controller;
use App\Models\BarangModel;
use Illuminate\Http\Request;

class BarangController extends Controller
{
    public function index()
    {
        return BarangModel::all();
    }

    public function store(Request $request)
    {
        $barang = BarangModel::create($request->all());
        return response()->json($barang, 201);
    }

    public function show(BarangModel $barang)
    {
        return $barang;
    }

    public function update(Request $request, BarangModel $barang)
    {
        $barang->update($request->all());
        return $barang;
    }

    public function destroy(BarangModel $barang)
    {
        $barang->delete();

        return response()->json([
            'success' => true,
            'message' => 'Data terhapus',
        ]);
    }
}
```

- kategori

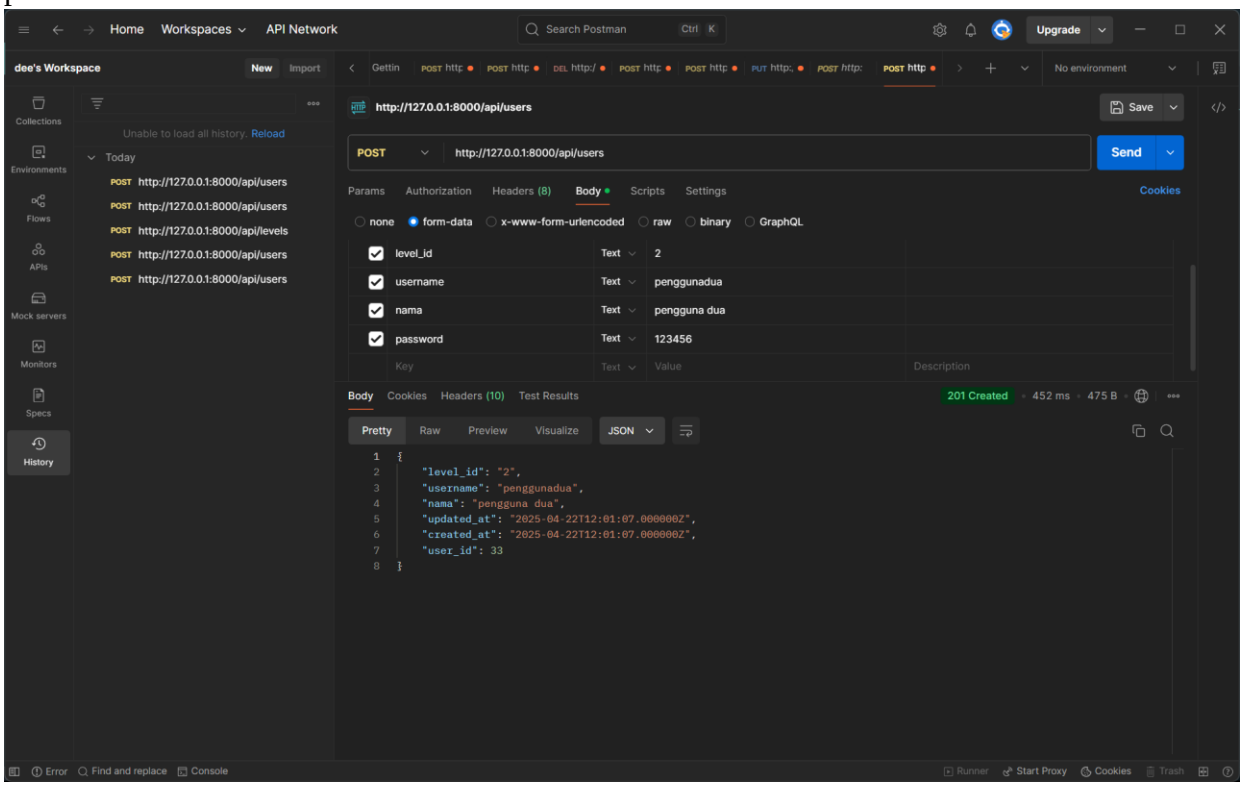
app > Http > Controllers > Api > KategoriController.php > ...

```php
<?php

namespace App\Http\Controllers\Api;

use App\Http\Controllers\Controller;
use App\Models\KategoriModel;
use Illuminate\Http\Request;

class KategoriController extends Controller
{
    public function index()
    {
        return KategoriModel::all();
    }

    public function store(Request $request)
    {
        $kategori = KategoriModel::create($request->all());
        return response()->json($kategori, 201);
    }

    public function show(KategoriModel $kategori)
    {
        return $kategori;
    }

    public function update(Request $request, KategoriModel $kategori)
    {
        $kategori->update($request->all());
        return $kategori;
    }

    public function destroy(KategoriModel $kategori)
    {
        $kategori->delete();

        return response()->json([
            'success' => true,
            'message' => 'Data terhapus',
        ]);
    }
}
```

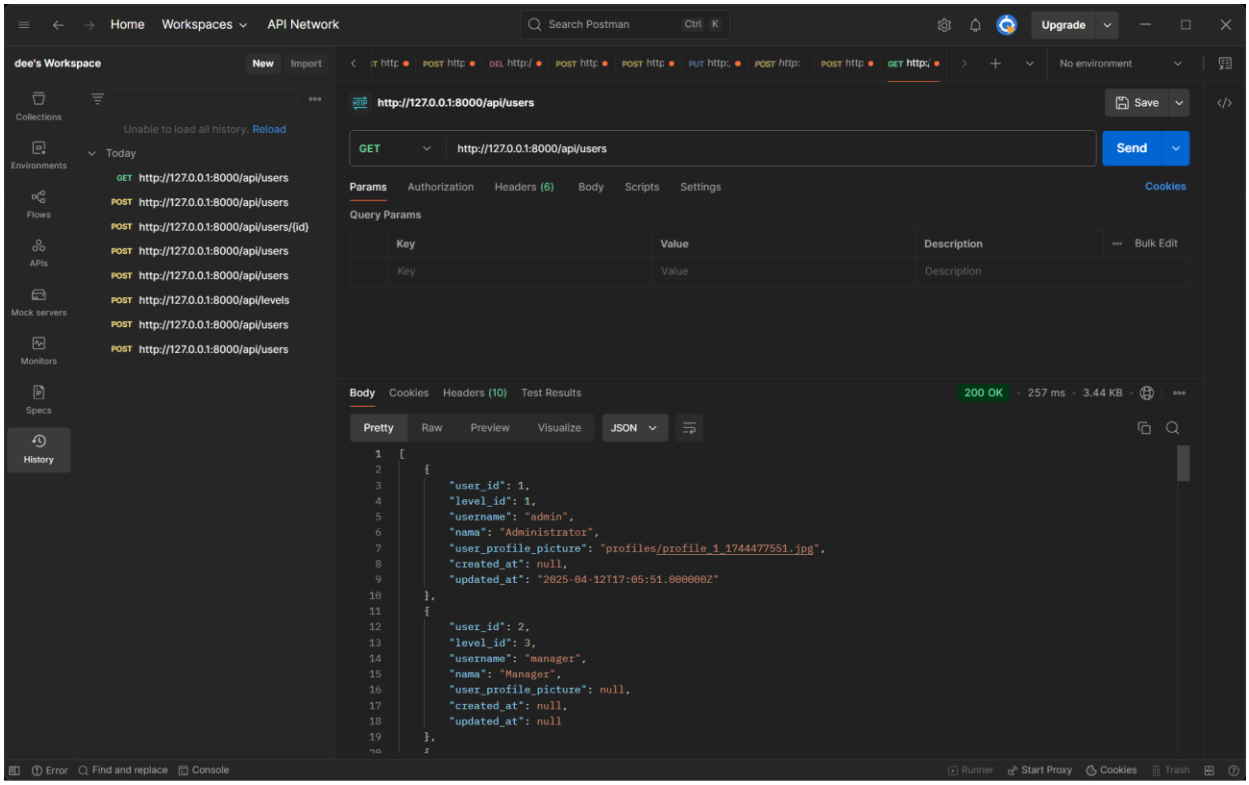
⇨ postman

- user

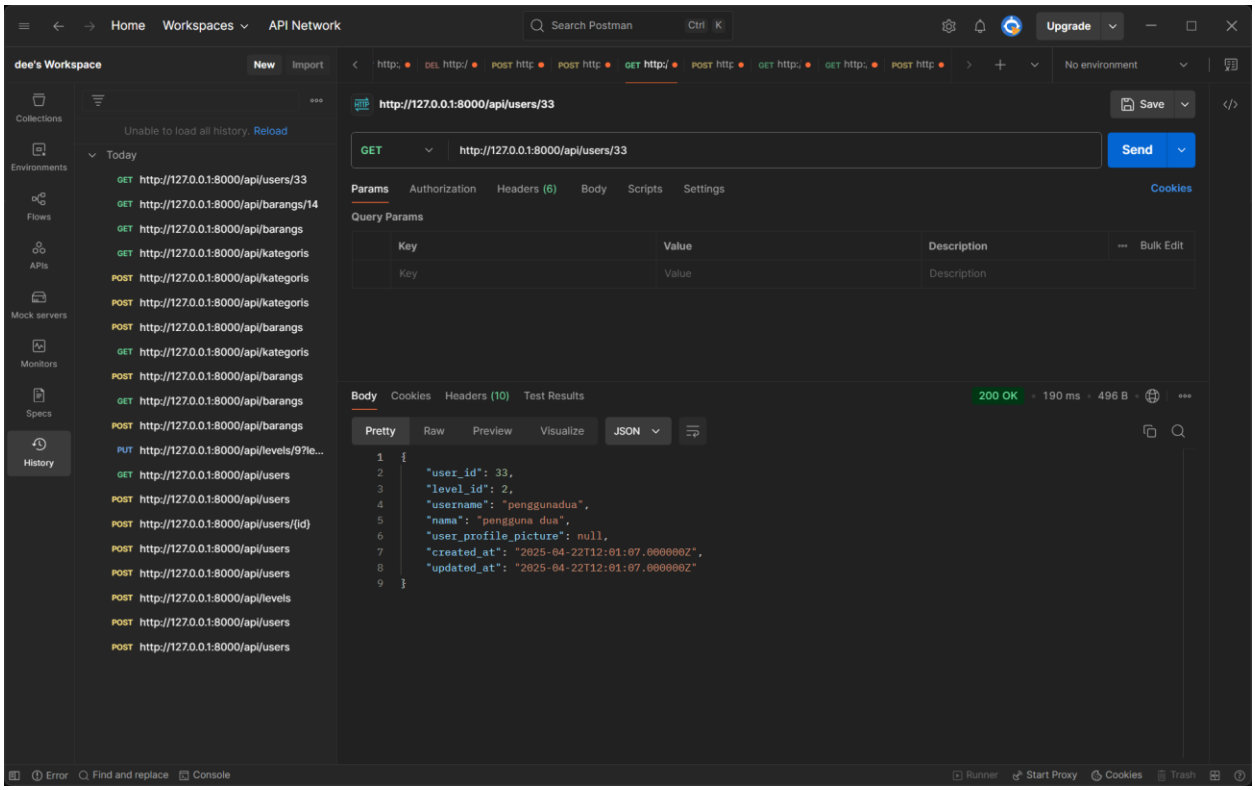
post



Get

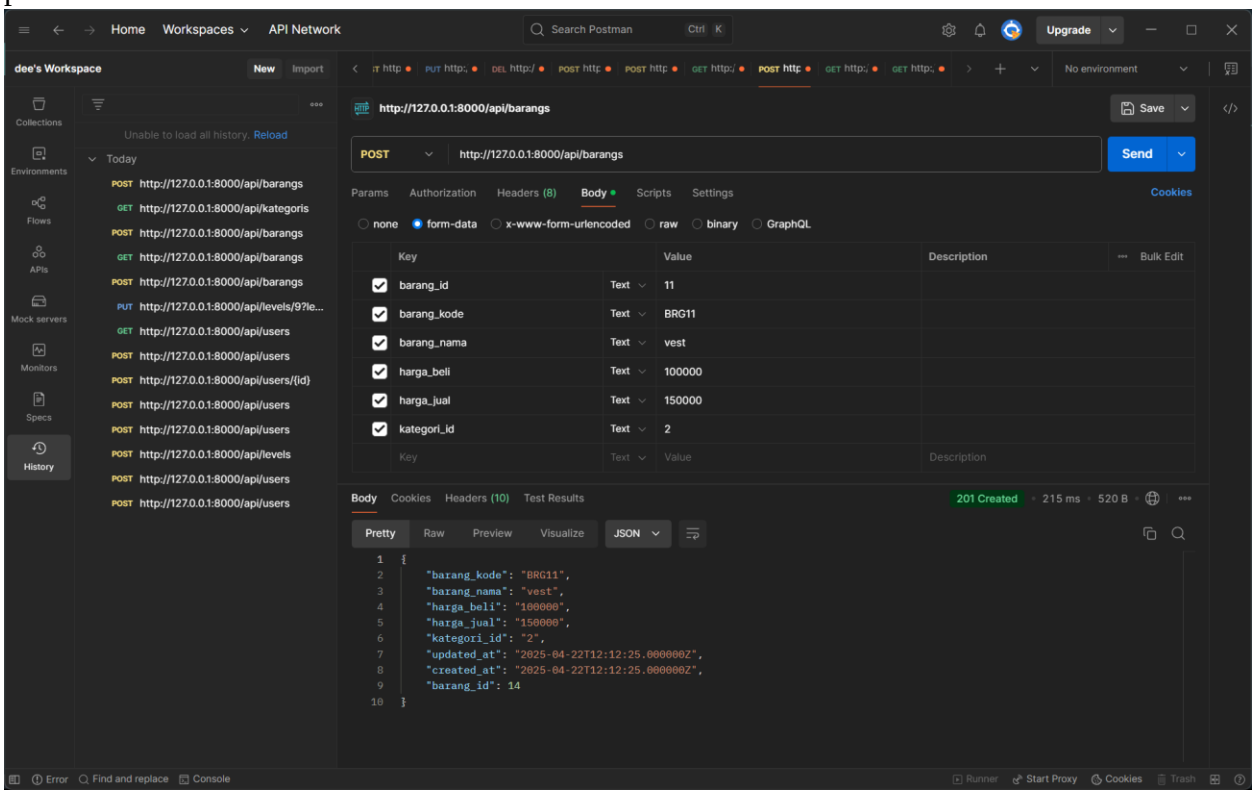

Home
Workspaces
API Network
Search Postman
Ctrl K
Upgrade
dee's Workspace
New
Import
No environment
http://127.0.0.1:8000/api/users
Save
GET
http://127.0.0.1:8000/api/users
Send
Params
Authorization
Headers (6)
Body
Scripts
Settings
Cookies
Query Params
Key
Value
Description
Bulk Edit
Body
Cookies
Headers (10)
Test Results
200 OK
257 ms
3.44 KB
Pretty
Raw
Preview
Visualize
JSON
Collections
Environments
Flows
APIs
Mock servers
Monitors
Specs
History
Unable to load all history. Reload
Today
GET http://127.0.0.1:8000/api/users
POST http://127.0.0.1:8000/api/users
POST http://127.0.0.1:8000/api/users/{id}
POST http://127.0.0.1:8000/api/users
POST http://127.0.0.1:8000/api/users
POST http://127.0.0.1:8000/api/levels
POST http://127.0.0.1:8000/api/users
POST http://127.0.0.1:8000/api/users
[
{
"user_id": 1,
"level_id": 1,
"username": "admin",
"nama": "Administrator",
"user_profile_picture": "profiles/profile_1_1744477551.jpg",
"created_at": null,
"updated_at": "2025-04-12T17:05:51.000000Z"
},
{
"user_id": 2,
"level_id": 3,
"username": "manager",
"nama": "Manager",
"user_profile_picture": null,
"created_at": null,
"updated_at": null
},
Error
Find and replace
Console
Runner
Start Proxy
Cookies
Trash

Get/id

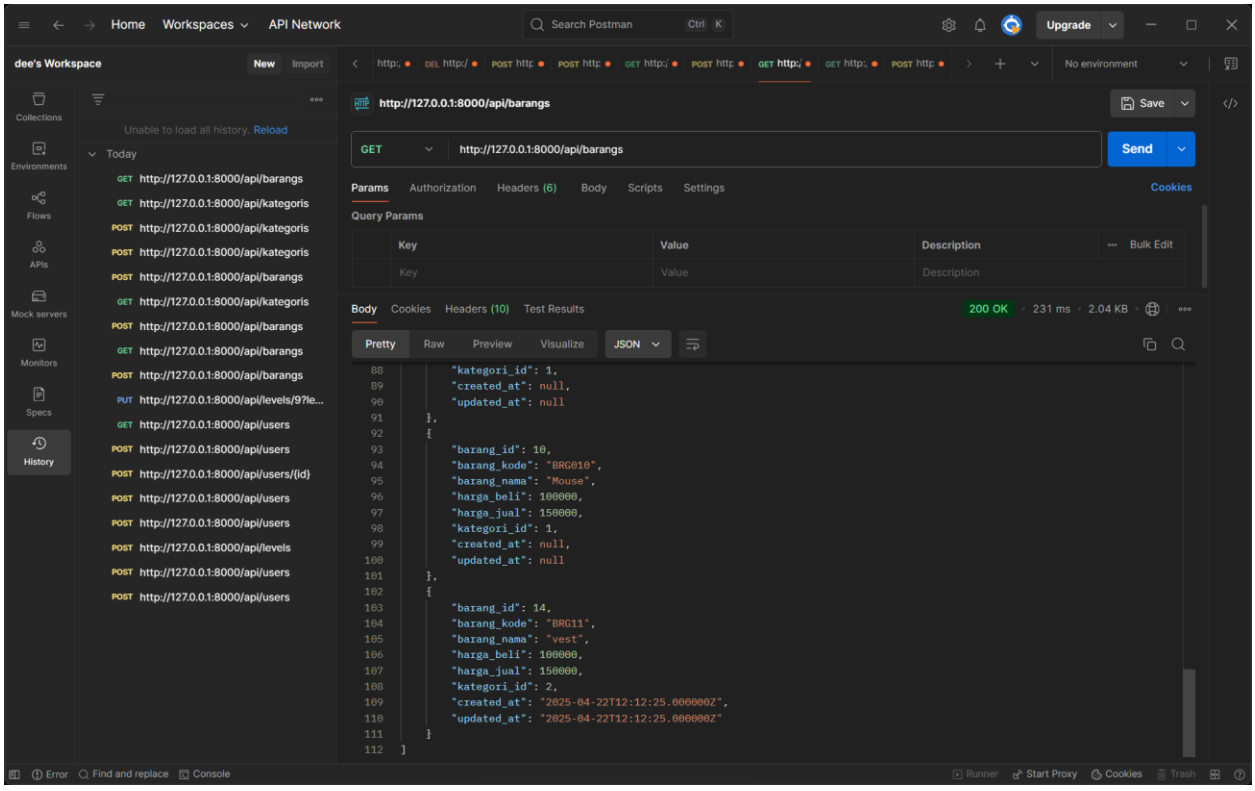

Home
Workspaces
API Network
Search Postman
Ctrl K
Upgrade
dee's Workspace
New
Import
No environment
http://127.0.0.1:8000/api/users/33
Save
GET
http://127.0.0.1:8000/api/users/33
Send
Params
Authorization
Headers (6)
Body
Scripts
Settings
Cookies
Query Params
Key
Value
Description
Bulk Edit
Body
Cookies
Headers (10)
Test Results
200 OK
190 ms
496 B
Pretty
Raw
Preview
Visualize
JSON
Collections
Environments
Flows
APIs
Mock servers
Monitors
Specs
History
Unable to load all history. Reload
Today
GET http://127.0.0.1:8000/api/users/33
GET http://127.0.0.1:8000/api/barangs/14
GET http://127.0.0.1:8000/api/barangs
GET http://127.0.0.1:8000/api/kategoris
POST http://127.0.0.1:8000/api/kategoris
POST http://127.0.0.1:8000/api/kategoris
POST http://127.0.0.1:8000/api/barangs
GET http://127.0.0.1:8000/api/kategoris
POST http://127.0.0.1:8000/api/barangs
GET http://127.0.0.1:8000/api/barangs
POST http://127.0.0.1:8000/api/barangs
PUT http://127.0.0.1:8000/api/levels/9?le...
GET http://127.0.0.1:8000/api/users
POST http://127.0.0.1:8000/api/users
POST http://127.0.0.1:8000/api/users/{id}
POST http://127.0.0.1:8000/api/users
POST http://127.0.0.1:8000/api/users
POST http://127.0.0.1:8000/api/levels
POST http://127.0.0.1:8000/api/users
POST http://127.0.0.1:8000/api/users
{
"user_id": 33,
"level_id": 2,
"username": "penggunadua",
"nama": "pengguna dua",
"user_profile_picture": null,
"created_at": "2025-04-22T12:01:07.000000Z",
"updated_at": "2025-04-22T12:01:07.000000Z"
}
Error
Find and replace
Console
Runner
Start Proxy
Cookies
Trash

- barang

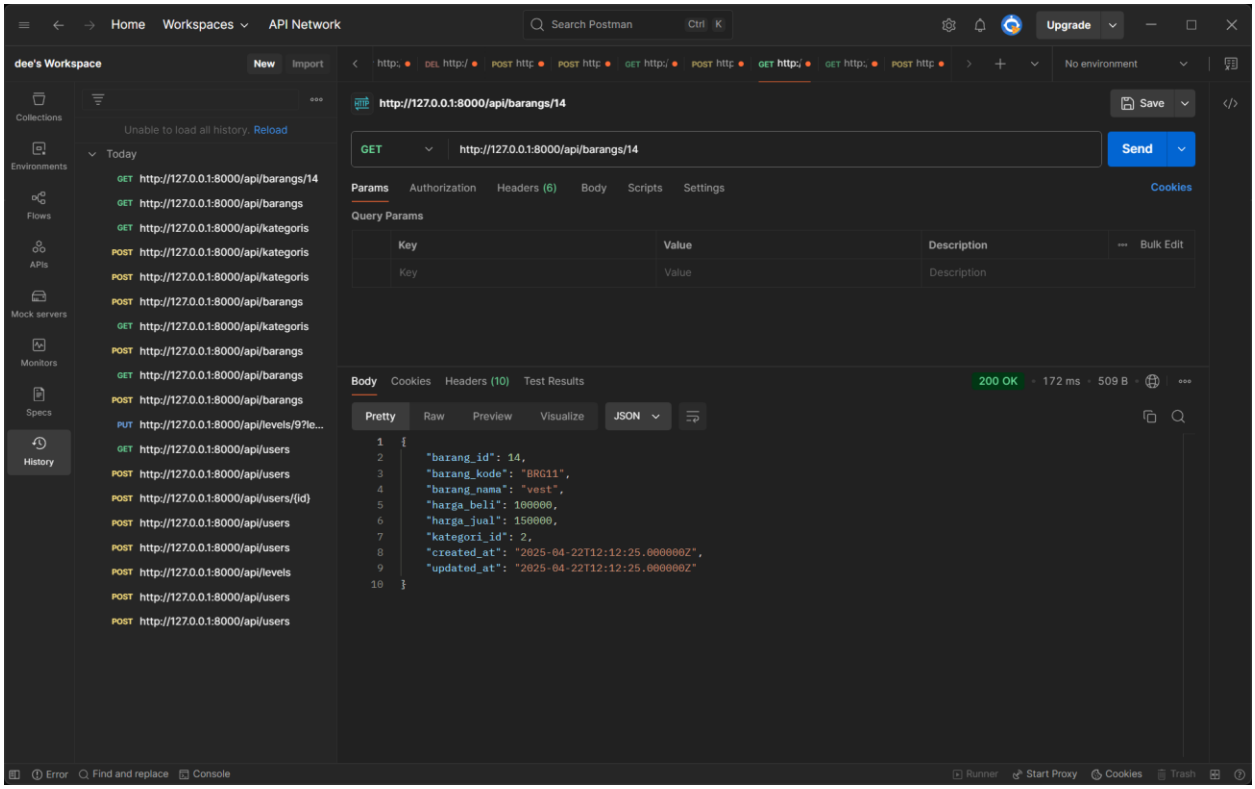
post



Get

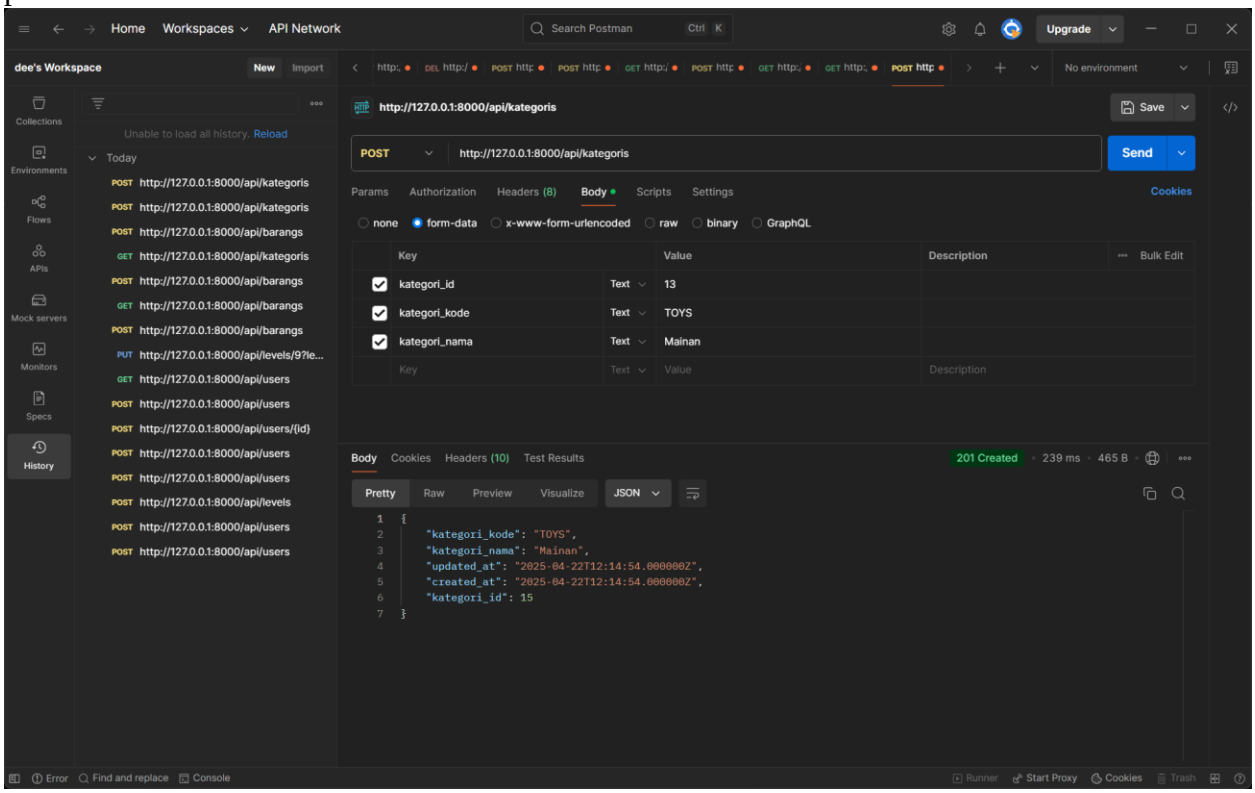


Get/id

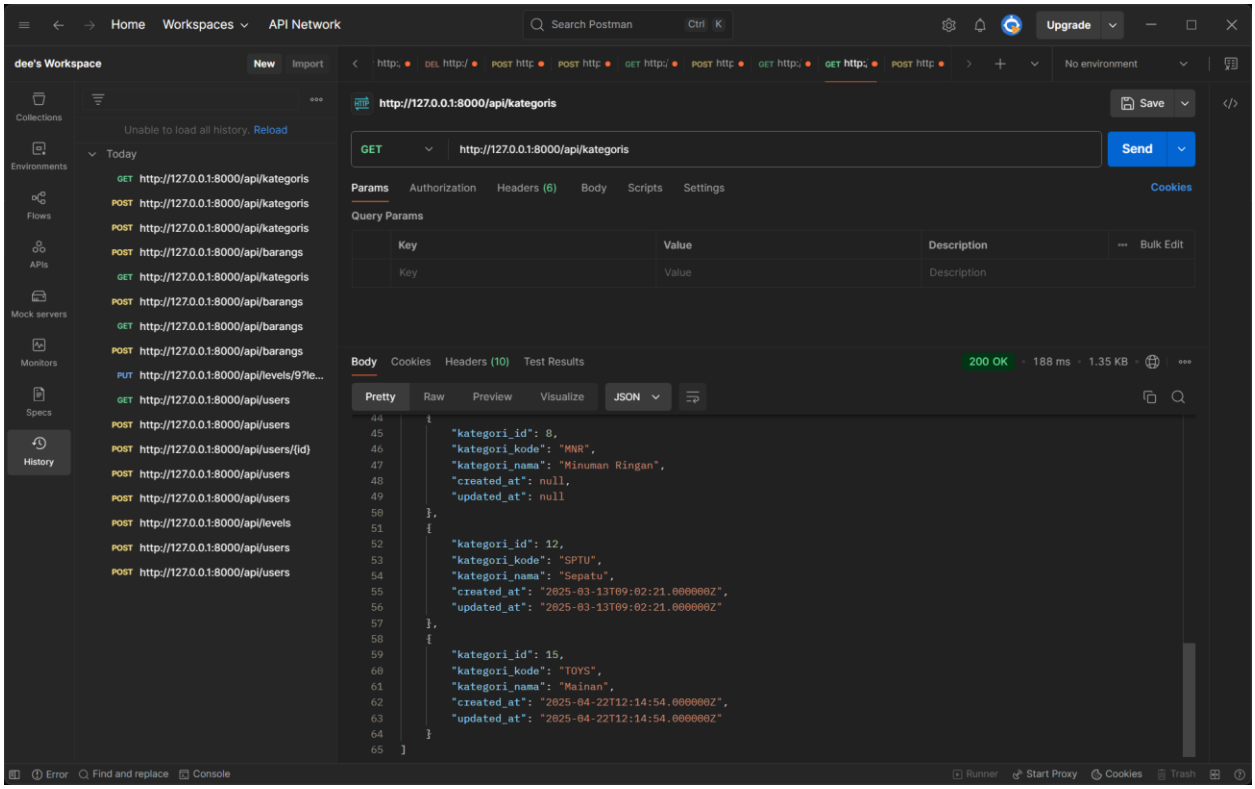


- kategori

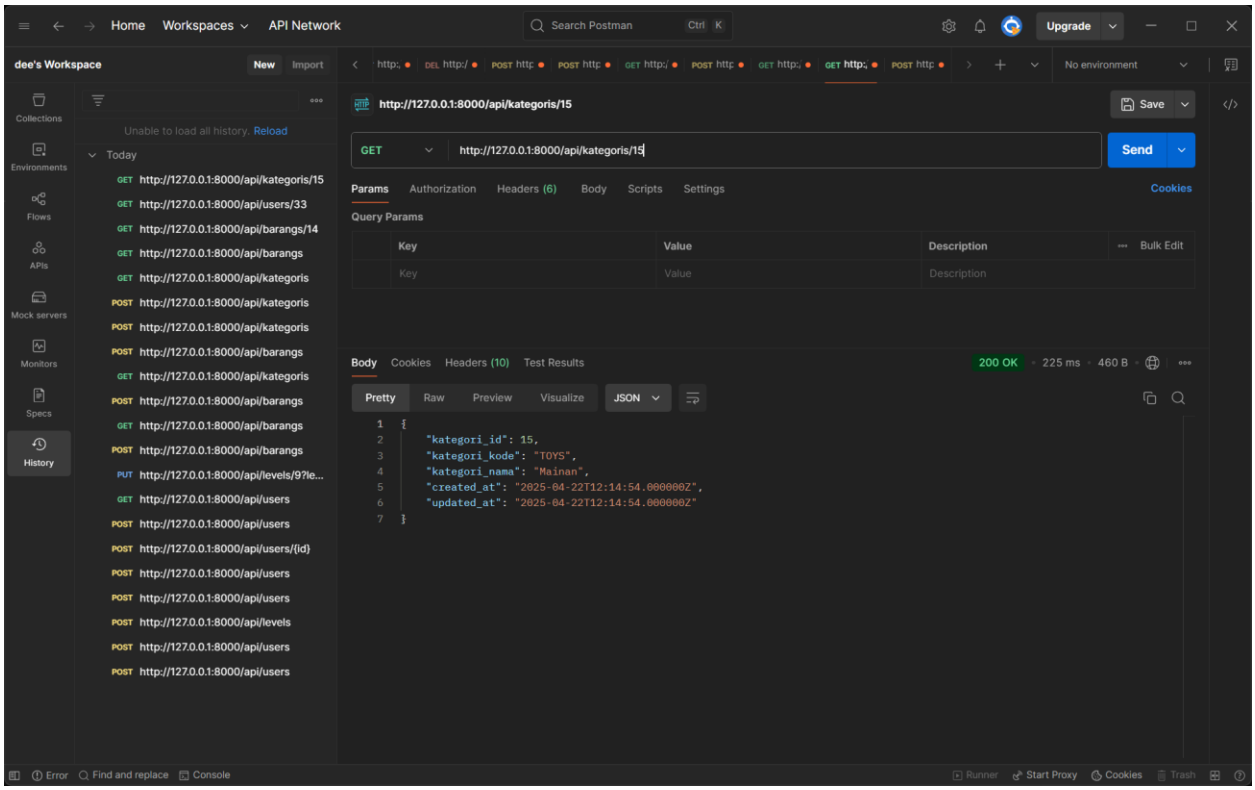
post



Get


Home
Workspaces
API Network
Search Postman
Ctrl K
Upgrade
dee's Workspace
New
Import
Collections
Environments
Flows
APIs
Mock servers
Monitors
Specs
History
Unable to load all history. Reload
Today
GET http://127.0.0.1:8000/api/kategoris
POST http://127.0.0.1:8000/api/kategoris
POST http://127.0.0.1:8000/api/kategoris
POST http://127.0.0.1:8000/api/barangs
GET http://127.0.0.1:8000/api/kategoris
POST http://127.0.0.1:8000/api/barangs
GET http://127.0.0.1:8000/api/barangs
POST http://127.0.0.1:8000/api/barangs
PUT http://127.0.0.1:8000/api/levels/9?le...
GET http://127.0.0.1:8000/api/users
POST http://127.0.0.1:8000/api/users
POST http://127.0.0.1:8000/api/users/{id}
POST http://127.0.0.1:8000/api/users
POST http://127.0.0.1:8000/api/users
POST http://127.0.0.1:8000/api/levels
POST http://127.0.0.1:8000/api/users
POST http://127.0.0.1:8000/api/users
No environment
http://127.0.0.1:8000/api/kategoris
Save
GET
http://127.0.0.1:8000/api/kategoris
Send
Params
Authorization
Headers (6)
Body
Scripts
Settings
Cookies
Query Params
Key
Value
Description
Bulk Edit
Body
Cookies
Headers (10)
Test Results
200 OK
188 ms
1.35 KB
Pretty
Raw
Preview
Visualize
JSON
{
"kategori_id": 8,
"kategori_kode": "MNR",
"kategori_nama": "Minuman Ringan",
"created_at": null,
"updated_at": null
},
{
"kategori_id": 12,
"kategori_kode": "SPTU",
"kategori_nama": "Sepatu",
"created_at": "2025-03-13T09:02:21.000000Z",
"updated_at": "2025-03-13T09:02:21.000000Z"
},
{
"kategori_id": 15,
"kategori_kode": "TOYS",
"kategori_nama": "Mainan",
"created_at": "2025-04-22T12:14:54.000000Z",
"updated_at": "2025-04-22T12:14:54.000000Z"
}
]
Error
Find and replace
Console
Runner
Start Proxy
Cookies
Trash

Get/id


Home
Workspaces
API Network
Search Postman
Ctrl K
Upgrade
dee's Workspace
New
Import
Collections
Environments
Flows
APIs
Mock servers
Monitors
Specs
History
Unable to load all history. Reload
Today
GET http://127.0.0.1:8000/api/kategoris/15
GET http://127.0.0.1:8000/api/users/33
GET http://127.0.0.1:8000/api/barangs/14
GET http://127.0.0.1:8000/api/barangs
GET http://127.0.0.1:8000/api/kategoris
POST http://127.0.0.1:8000/api/kategoris
POST http://127.0.0.1:8000/api/kategoris
POST http://127.0.0.1:8000/api/barangs
GET http://127.0.0.1:8000/api/kategoris
POST http://127.0.0.1:8000/api/barangs
GET http://127.0.0.1:8000/api/barangs
POST http://127.0.0.1:8000/api/barangs
PUT http://127.0.0.1:8000/api/levels/9?le...
GET http://127.0.0.1:8000/api/users
POST http://127.0.0.1:8000/api/users
POST http://127.0.0.1:8000/api/users/{id}
POST http://127.0.0.1:8000/api/users
POST http://127.0.0.1:8000/api/users
POST http://127.0.0.1:8000/api/levels
POST http://127.0.0.1:8000/api/users
POST http://127.0.0.1:8000/api/users
No environment
http://127.0.0.1:8000/api/kategoris/15
Save
GET
http://127.0.0.1:8000/api/kategoris/15
Send
Params
Authorization
Headers (6)
Body
Scripts
Settings
Cookies
Query Params
Key
Value
Description
Bulk Edit
Body
Cookies
Headers (10)
Test Results
200 OK
225 ms
460 B
Pretty
Raw
Preview
Visualize
JSON
{
"kategori_id": 15,
"kategori_kode": "TOYS",
"kategori_nama": "Mainan",
"created_at": "2025-04-22T12:14:54.000000Z",
"updated_at": "2025-04-22T12:14:54.000000Z"
}
Error
Find and replace
Console
Runner
Start Proxy
Cookies
Trash