|  |  |  |
| --- | --- | --- |
| Mata Kuliah | **:** | Pemrograman Web Lanjut (PWL) |
| Program Studi | **:** | D4 – Teknik Informatika **/** D4 – Sistem Informasi Bisnis |
| Semester | **:** | 4 (empat) / 6 (enam) |
| Pertemuan ke- | **:** | 10 (tujuh) |

# JOBSHEET 10 RESTFUL API

Sebelumnya kita sudah membahas mengenai *authentication, authorization,* dan *middleware* pada Laravel. Dimana kita telah membuat fungsi login, register, logout, serta pemilihan role dan penerapan session pada halaman web. Pada pertemuan kali ini, kita akan mempelajari penerapan RESTFUL API di dalam project Laravel.

Sebelum kita masuk materi, kita buat dulu project baru yang akan kita gunakan untuk membangun aplikasi sederhana dengan topik *Point of Sales (PoS),* sesuai dengan **Studi Kasus PWL.pdf**.

Jadi kita bikin project Laravel 10 dengan nama **PWL\_POS.**

*Project* **PWL\_POS** akan kita gunakan sampai pertemuan 12 nanti, sebagai project yang akan kita pelajarai

# RESTFUL API

Representational State Transfer (REST) adalah gaya arsitektur perangkat lunak yang mendefinisikan seperangkat prinsip untuk merancang jaringan aplikasi terdistribusi. RESTful API adalah aplikasi pemrograman antarmuka yang mengikuti prinsip-prinsip REST untuk mentransfer data antara klien dan server.

RESTful API adalah salah satu arsitektur dalam API (*Application Program Interface*) yang menggunakan request HTTP untuk mengakses data. Data diakses dengan menggunakan HTTP method GET, PUT, POST dan DELETE yang merujuk pada operasi pembacaan, pembaruan, pembuatan dan penghapusan pada resource. Selain HTTP method, dalam RESTful atau REST digunakan juga HTTP response untuk mendefinisikan respon data yang dikembalikan. Format respon yang umum digunakan berupa JSON (Javascript Object Notation).

# JSON Web Token (JWT)

JWT adalah singkatan dari JSON Web Token. Ini adalah standar terbuka (RFC 7519) yang mendefinisikan format token yang kompak dan mandiri untuk mentransfer klaim antara dua pihak. JWT sering digunakan dalam otentikasi dan pertukaran informasi yang aman di lingkungan yang tidak terpercaya, seperti internet.

JWT terdiri dari tiga bagian yang dipisahkan oleh titik ("."): header, payload, dan signature. Setiap bagian ini terdiri dari data JSON yang dienkripsi menggunakan algoritma tertentu dan kemudian disatukan untuk membentuk token yang lengkap. Header berisi jenis token dan tipe algoritma yang digunakan untuk enkripsi. Payload berisi klaim atau informasi yang ingin disampaikan. Signature digunakan untuk memverifikasi bahwa token belum berubah dan datanya berasal dari sumber yang dipercayai.

JWT sering digunakan dalam sistem otentikasi dan otorisasi modern, seperti aplikasi web dan layanan web API, karena fleksibilitasnya dalam menyampaikan informasi terenkripsi secara ringkas.

Kita dapat menggunakan JWT untuk:

* + Authentication

Ketika pengguna melakukan authentication dan mendapatkan token, maka setiap permintaan berikutnya akan menyertakan token tersebut, dan memungkinkan pengguna untuk mengakses route, service, dan resources yang diizinkan.

* + Pertukaran informasi

JSON Web Token adalah cara yang baik untuk mengirimkan informasi antar pihak dengan aman. Dengan token yang sudah ditandatangani dengan algoritma RSA, maka kita bisa tahu siapa yang melakukan request tersebut.

Berikut adalah cara kerja JWT :

JWT (JSON Web Token) adalah cara untuk mentransfer informasi antara dua pihak secara aman sebagai objek JSON. Ini terdiri dari tiga bagian: header, payload, dan signature. Setelah pengguna berhasil autentikasi, server menghasilkan token JWT yang disematkan dalam permintaan HTTP. Server kemudian memvalidasi token untuk memberikan akses ke sumber daya yang diminta. Ini memberikan autentikasi yang aman dan stateless tanpa memerlukan penyimpanan status sesi di server.

**Praktikum 1** – Membuat RESTful API Register

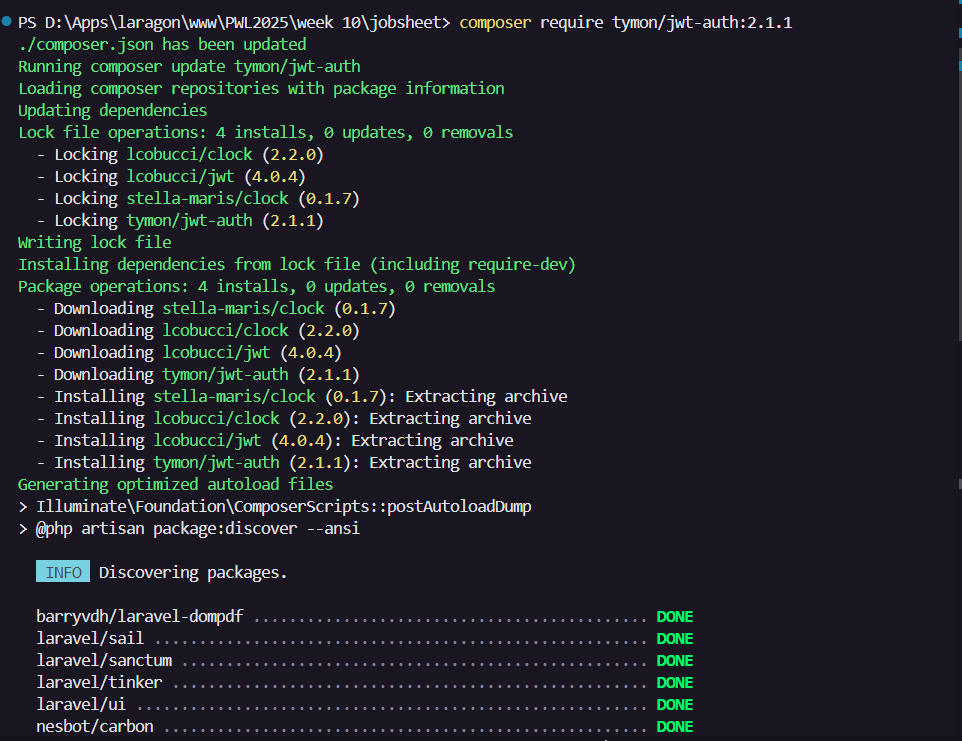
1. Sebelum memulai membuat REST API, terlebih dahulu download aplikasi Postman di <https://www.postman.com/downloads>.

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

1. Lakukan instalasi JWT dengan mengetikkan perintah berikut:

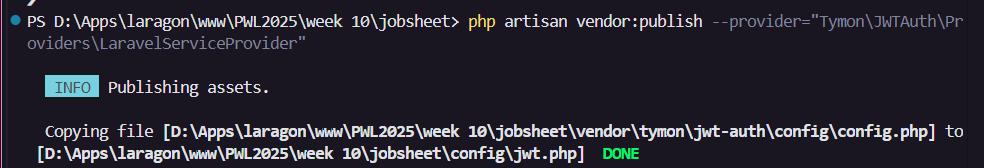
composer require tymon/jwt-auth:2.1.1

Pastikan Anda terkoneksi dengan internet.



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

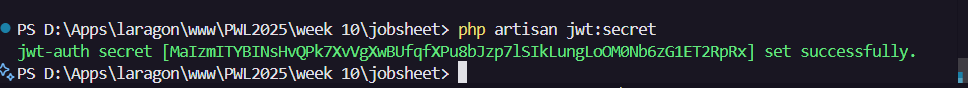
php artisan vendor:publish -- provider="Tymon\JWTAuth\Providers\LaravelServiceProvider"



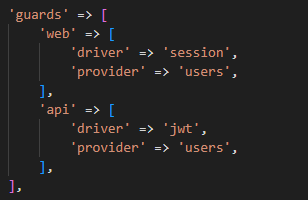
1. Jika perintah di atas berhasil, maka kita akan mendapatkan 1 file baru yaitu config/jwt.php. Pada file ini dapat dilakukan konfigurasi jika memang diperlukan.
2. Setelah itu jalankan peintah berikut untuk membuat secret key JWT.

php artisan jwt:secret

Jika berhasil, maka pada file .env akan ditambahkan sebuah baris berisi nilai key JWT\_SECRET.

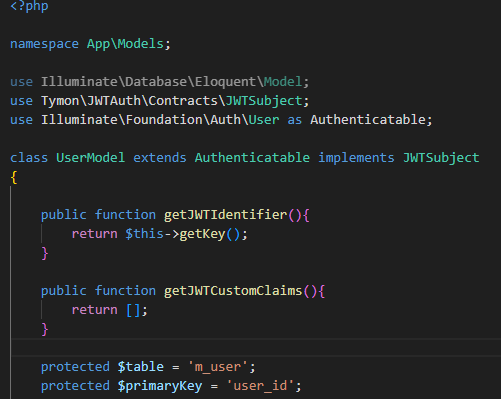


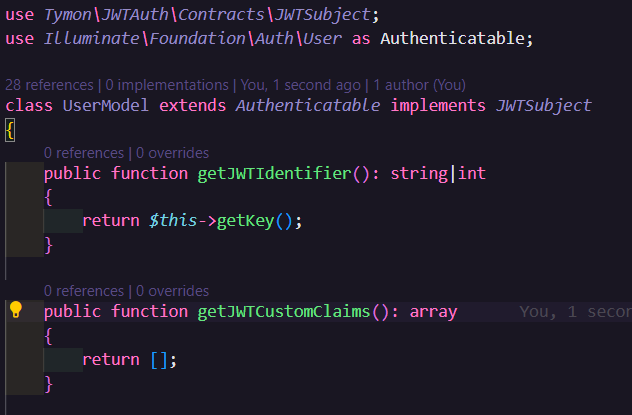
1. Selanjutnya lakukan konfigurasi guard API. Buka config/auth.php. Ubah bagian ‘guards’ menjadi seperti berikut.





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

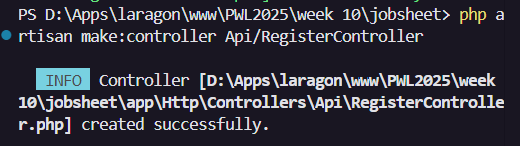




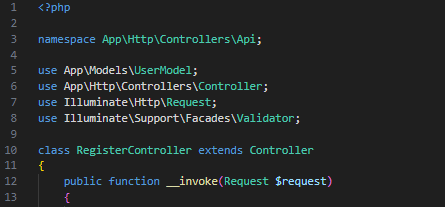
1. Berikutnya kita akan membuat controller untuk register dengan menjalankan peintah berikut.

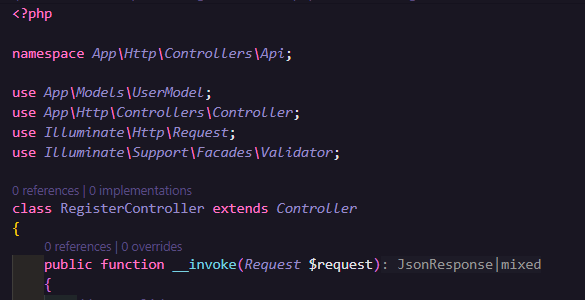
php artisan make:controller Api/RegisterController

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

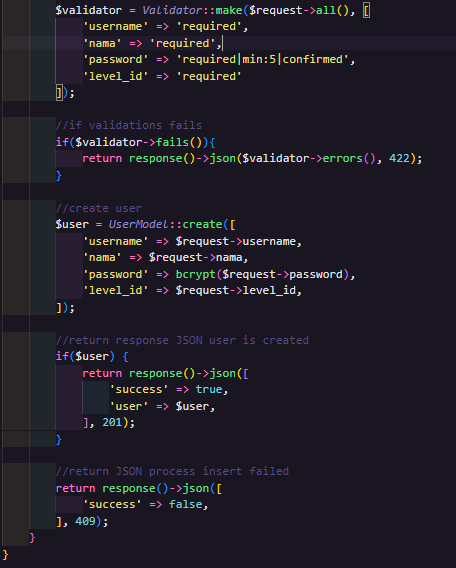


1. Buka file tersebut, dan ubah kode menjadi seperti berikut.

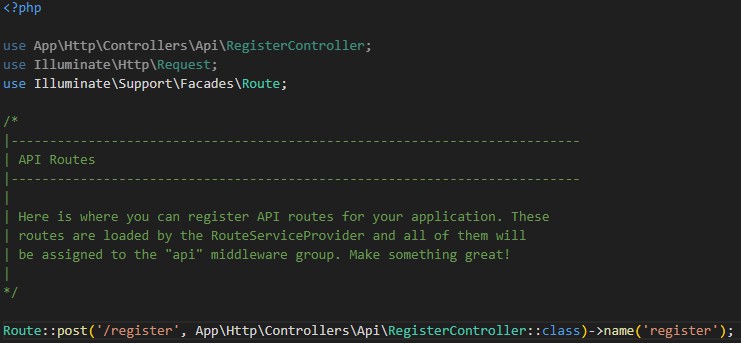


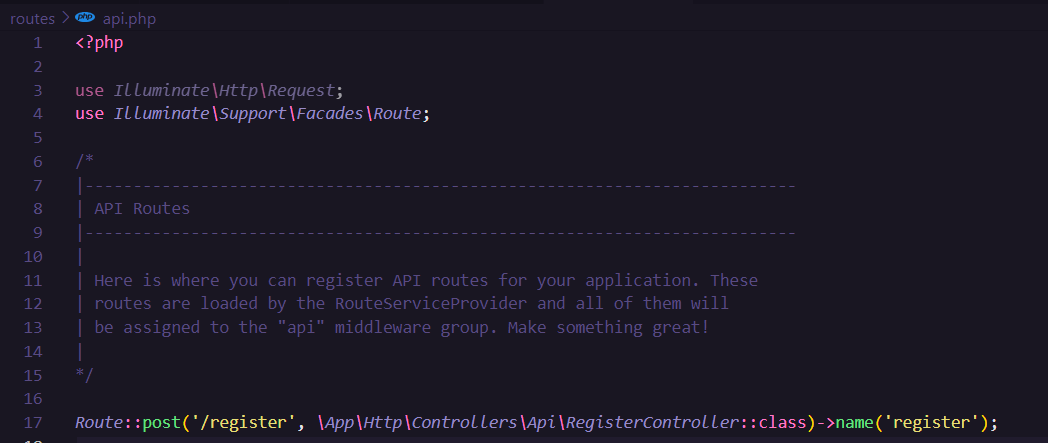




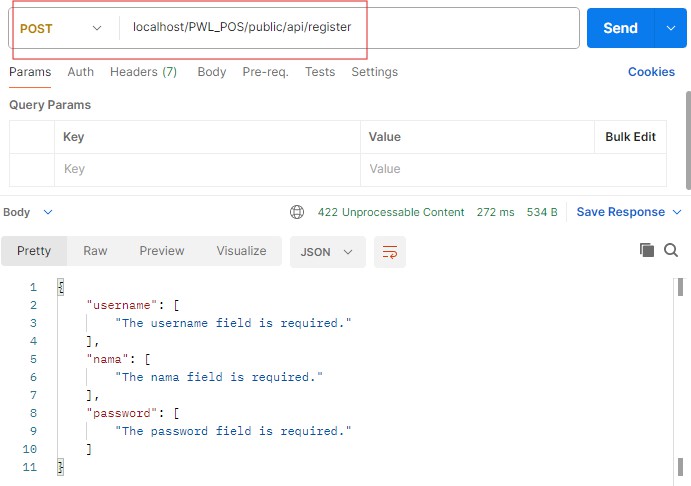


1. Selanjutnya buka routes/api.php, ubah semua kode menjadi seperti berikut.





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

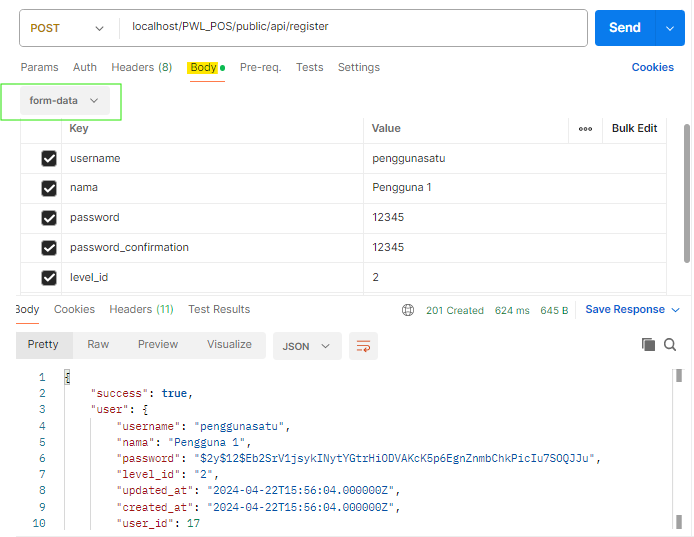
Buka aplikasi Postman, isi URL localhost/PWL\_POS/public/api/register serta method POST. Klik Send.

Jika berhasil akan muncul error validasi seperti gambar di atas.

Lakukan percobaan yang sama dan berikan screenshoot hasil percobaan Anda.

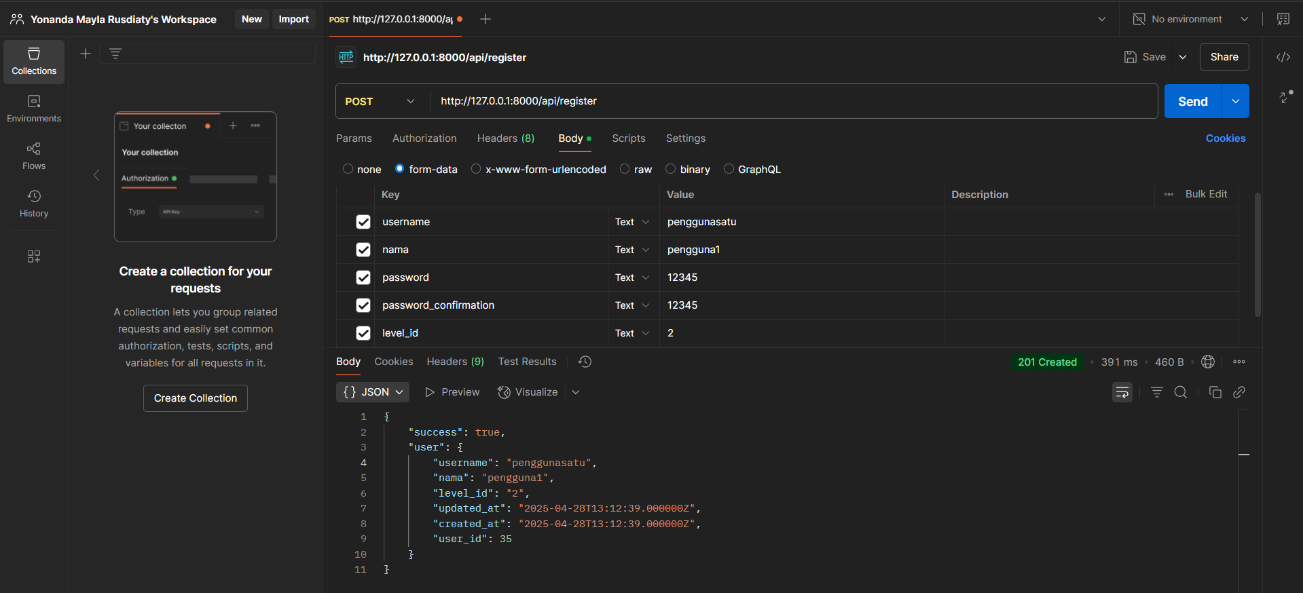


1. Sekarang kita coba masukkan data. Klik tab Body dan pilih form-data. Isikan key sesuai dengan kolom data, serta isikan data registrasi menggunakan nilai yang Anda inginkan.



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:** Jika kita mengirim data (seperti username, password) ke alamat /api/register dengan metode POST, maka aplikasi akan memproses pendaftaran pengguna baru.

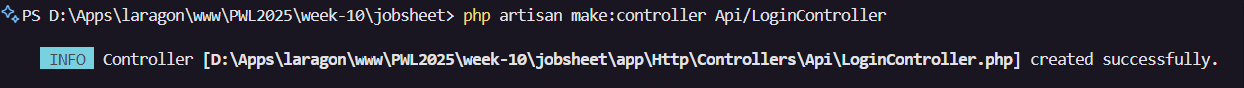
1. Lakukan commit perubahan file pada Github.

**Praktikum 2** – Membuat RESTful API Login

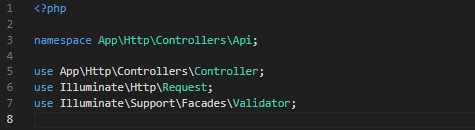
1. Kita buat file controller dengan nama LoginController.

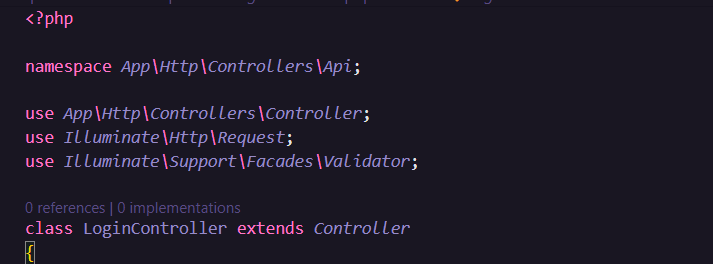
php artisan make:controller Api/LoginController

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

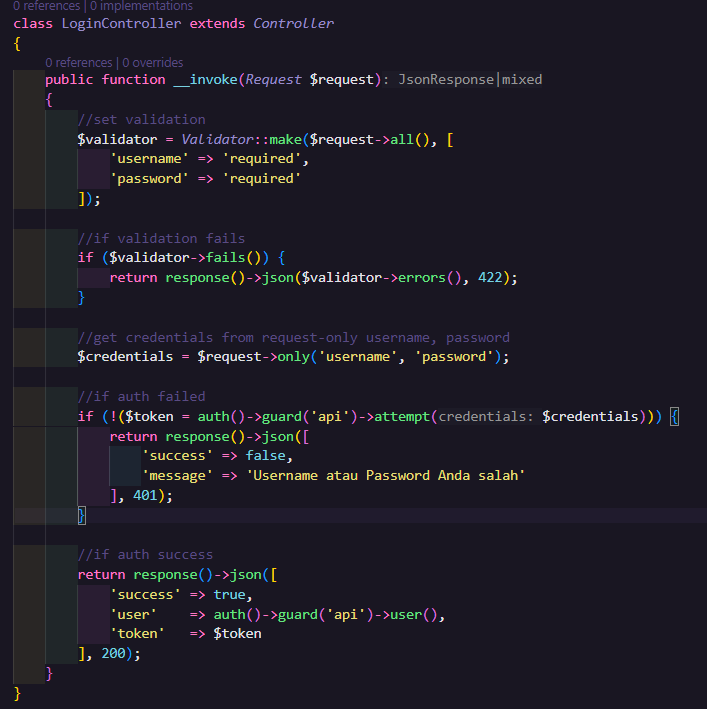


1. Buka file tersebut, dan ubah kode menjadi seperti berikut.

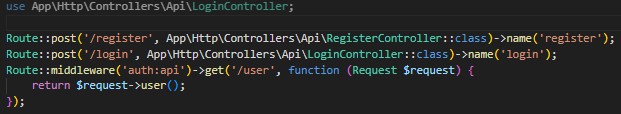


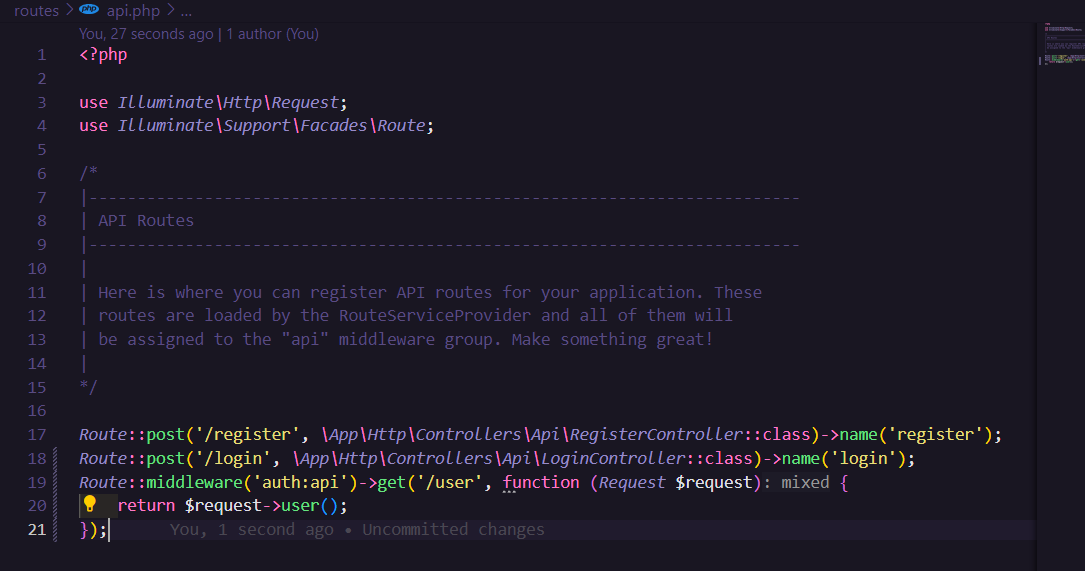




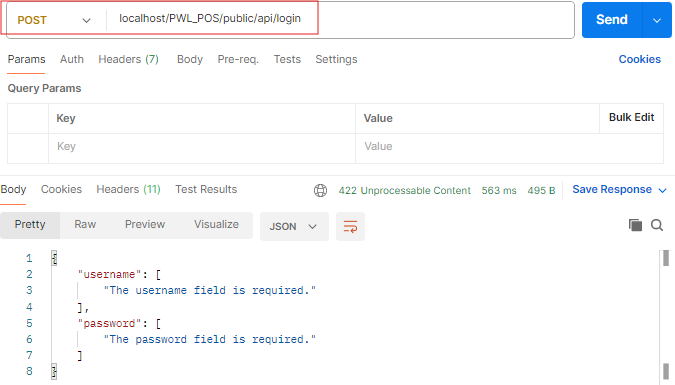


1. Berikutnya tambahkan route baru pada file api.php yaitu /login dan /user.



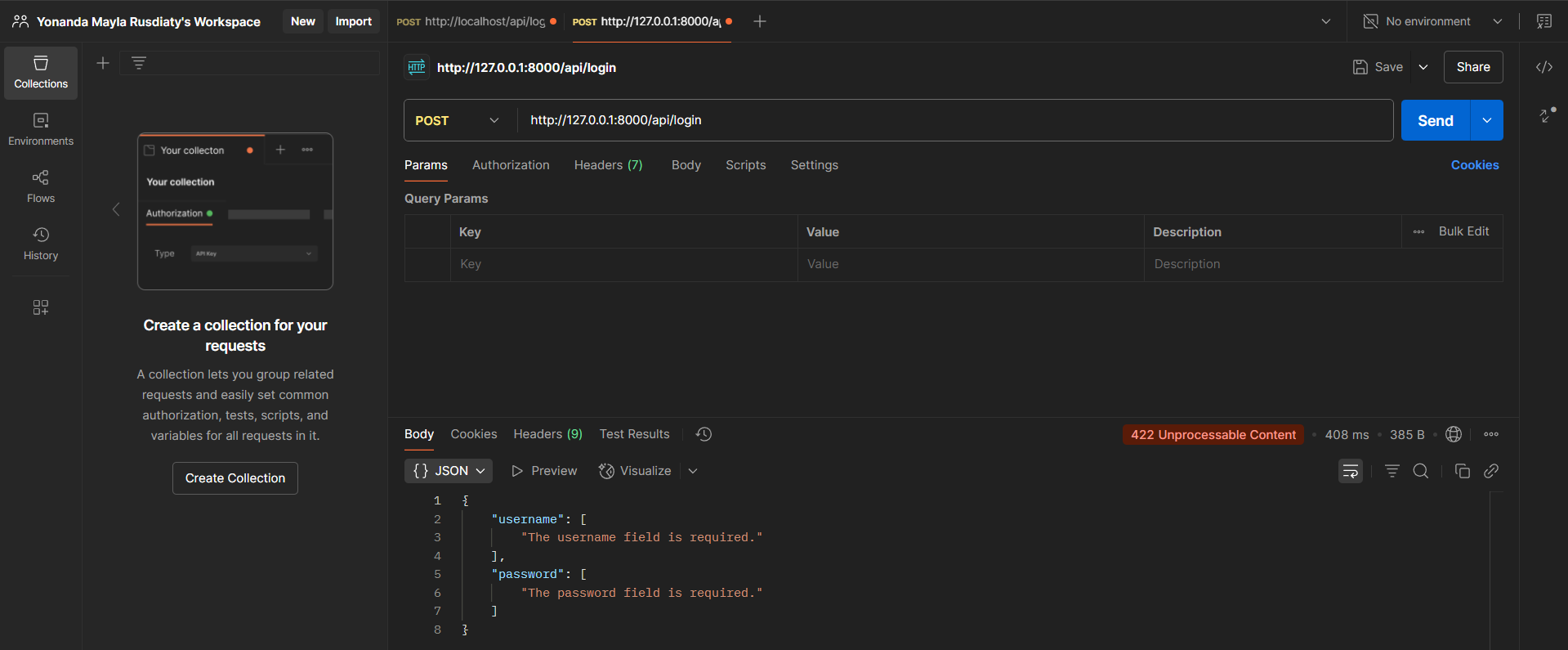


1. Jika sudah, kita akan melakukan uji coba REST API melalui aplikasi Postman. Buka aplikasi Postman, isi URL localhost/PWL\_POS/public/api/login serta method POST. Klik Send.

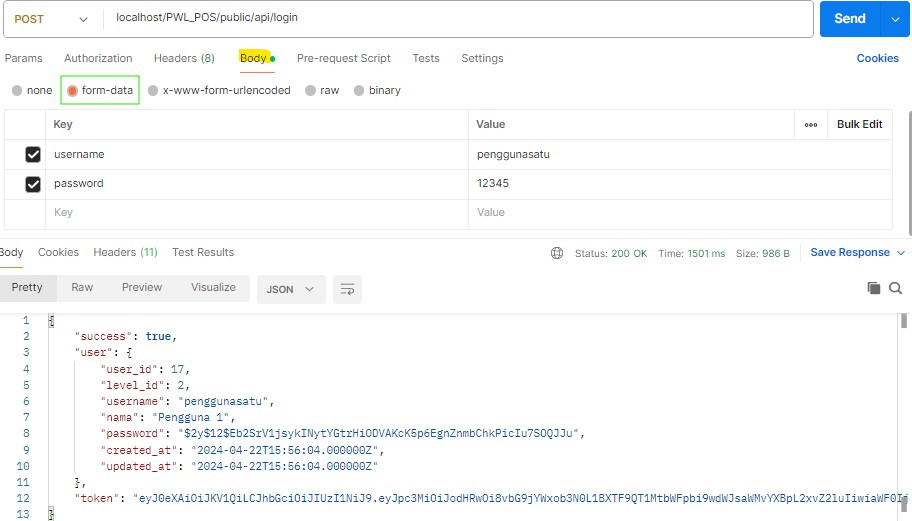


Jika berhasil akan muncul error validasi seperti gambar di atas.

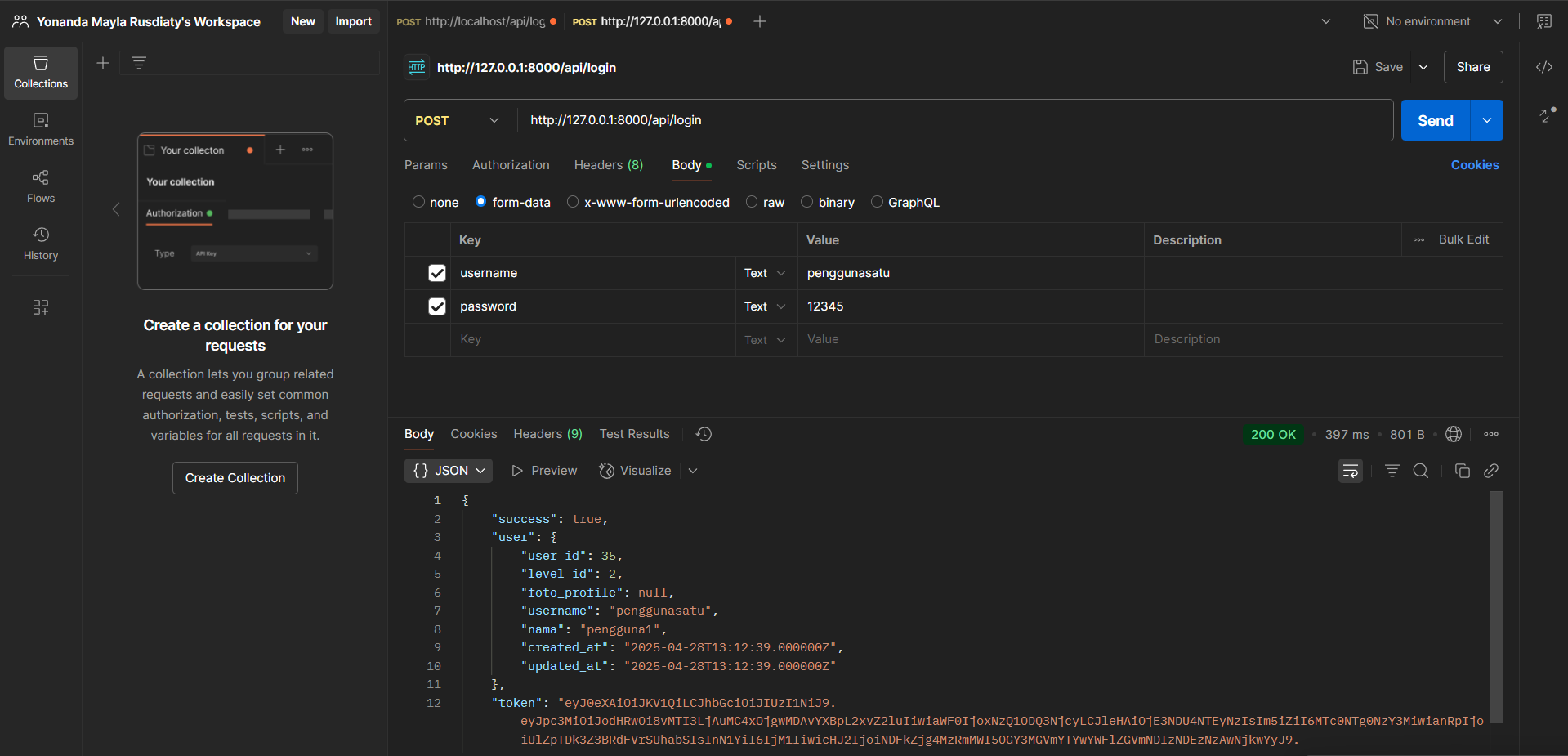
Lakukan percobaan yang sama dan berikan screenshoot hasil percobaan Anda.



1. 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.

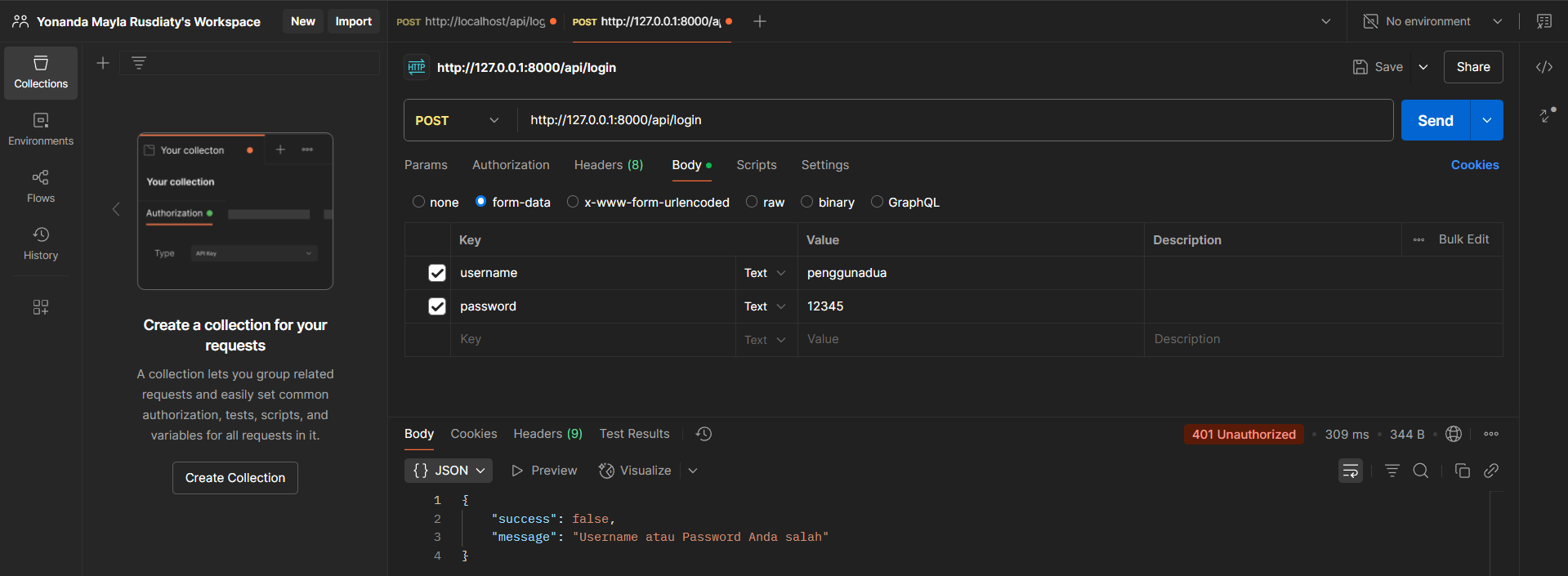


Lakukan percobaan yang sama dan berikan screenshoot hasil percobaan Anda.



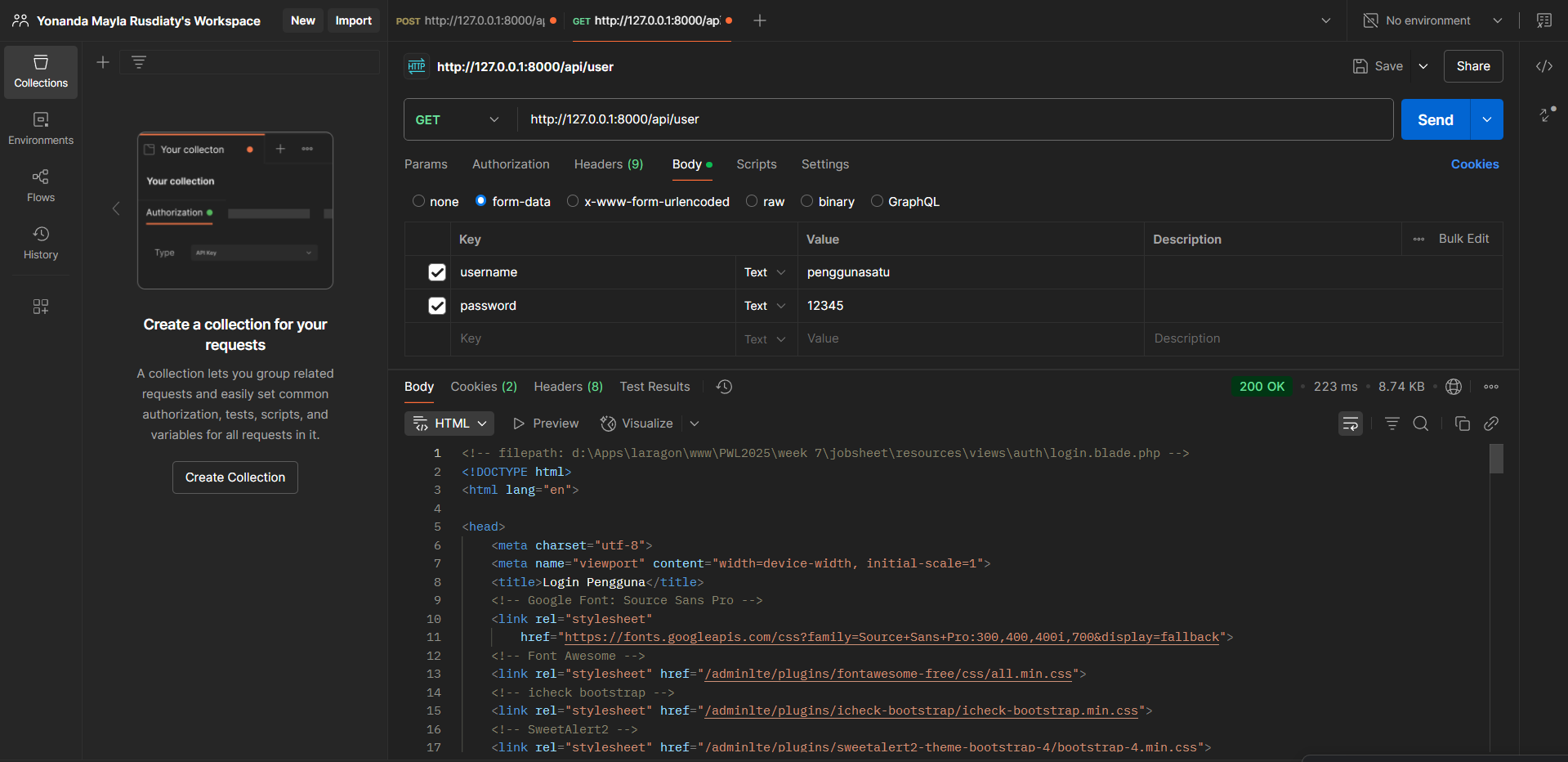
Jika kita mengirim username dan password ke alamat /api/login dengan metode POST, aplikasi akan memeriksa data tersebut dan memberikan token akses jika benar.

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



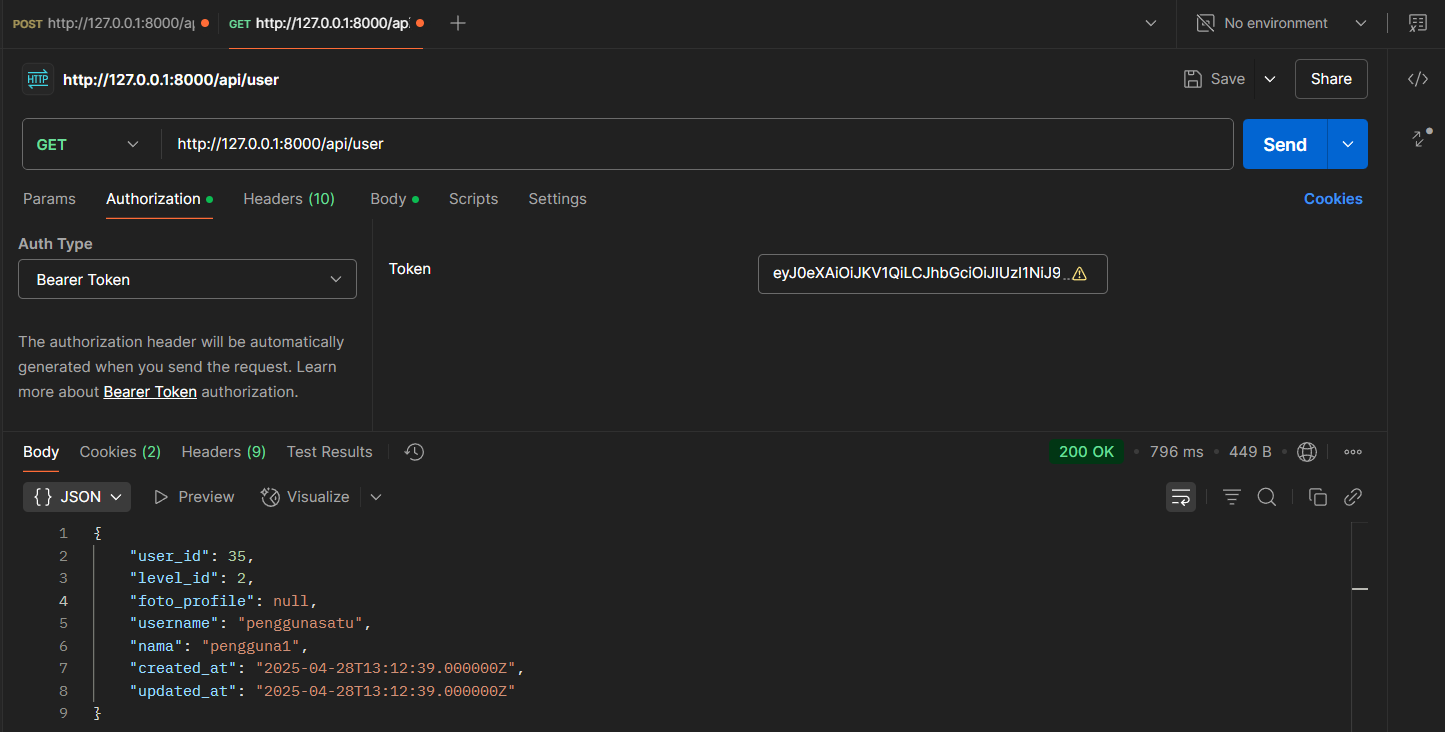
1. Coba kembali melakukan login dengan data yang benar. Sekarang mari kita coba menampilkan data user yang sedang login menggunakan URL localhost/PWL\_POS/public/api/user dan method GET.

Jelaskan hasil dari percobaan tersebut.



**Jawab:**

Token menyimpan informasi identitas pengguna yang telah berhasil login. Saat kita mengakses method get pada url <http://127.0.0.1:8000/api/user>, hal ini berarti kita ingin menampilkan data diri sendiri di alamat /api/user tetapi harus sudah login dulu. Namun di return postmannya hanya bisa menampilkan halaman login karena token belum dimasukkan. Begini tampilan jika kita sudah memasukkan tokennya:



Setelah login, token disimpan dan ditambahkan ke header request berikutnya. Tanpa token, kita akan mendapatkan error 401 Unauthorized saat mencoba mengakses endpoint terproteksi

1. Lakukan commit perubahan file pada Github.

**Praktikum 3** – Membuat RESTful API Logout

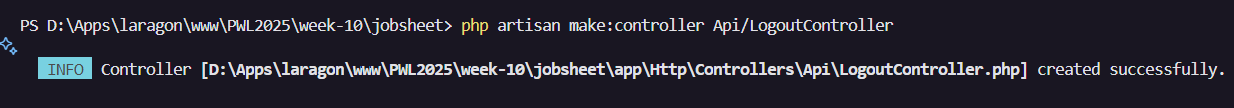
1. Tambahkan kode berikut pada file .env

JWT\_SHOW\_BLACKLIST\_EXCEPTION=true



1. Buat Controller baru dengan nama LogoutController.

php artisan make:controller Api/LogoutController



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

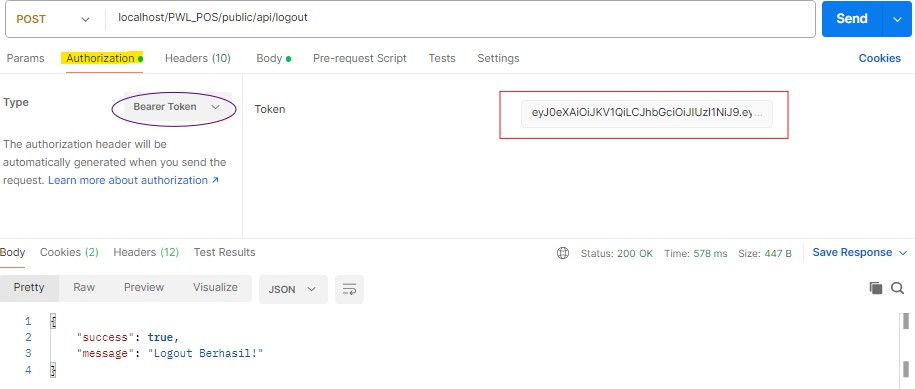




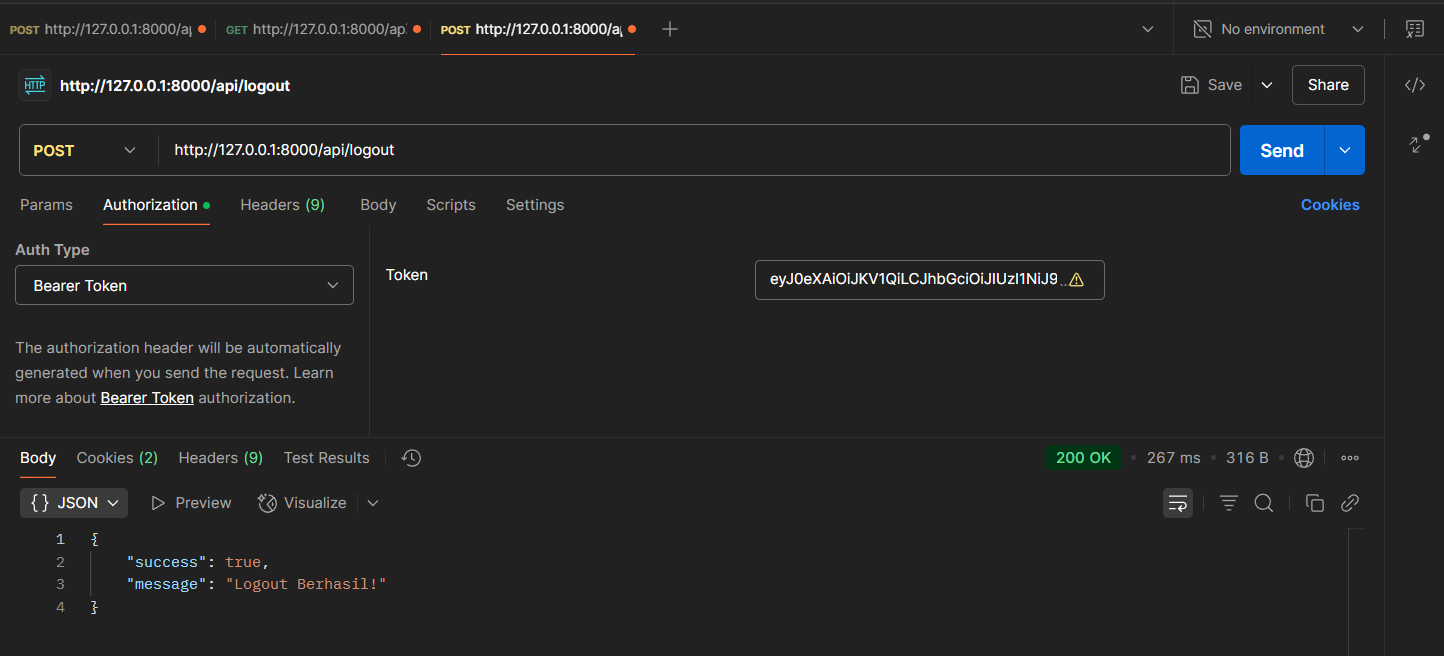
1. Lalu kita tambahkan routes pada api.php





1. 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.
2. 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:** Jika kita ingin keluar dari sistem, kirim permintaan ke alamat /api/logout dengan metode POST. Ini akan membuat token kita sebelumnya tidak bisa digunakan lagi. JWT\_SHOW\_BLACKLIST\_EXCEPTION=true, mencegah penggunaan token yang sudah tidak valid

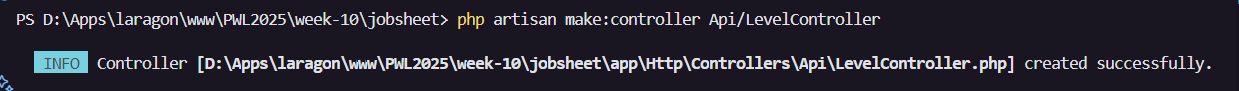
1. Lakukan commit perubahan file pada Github.

**Praktikum 4** – Implementasi CRUD dalam RESTful API

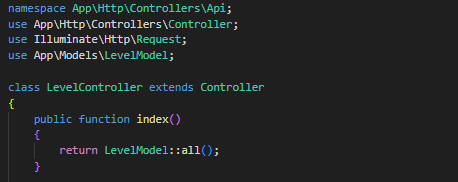
Pada praktikum ini kita akan menggunakan tabel m\_level untuk dimodifikasi menggunakan RESTful API.

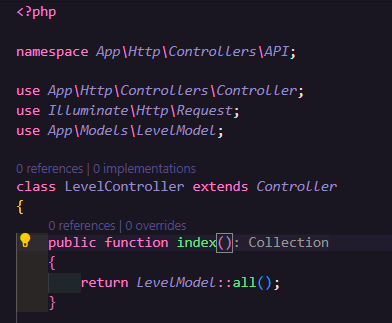
1. Pertama, buat controller untuk mengolah API pada data level.

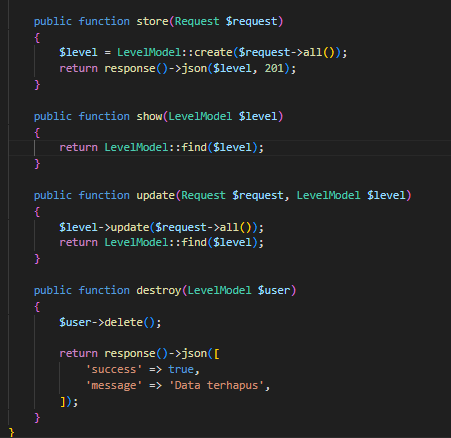
php artisan make:controller Api/LevelController

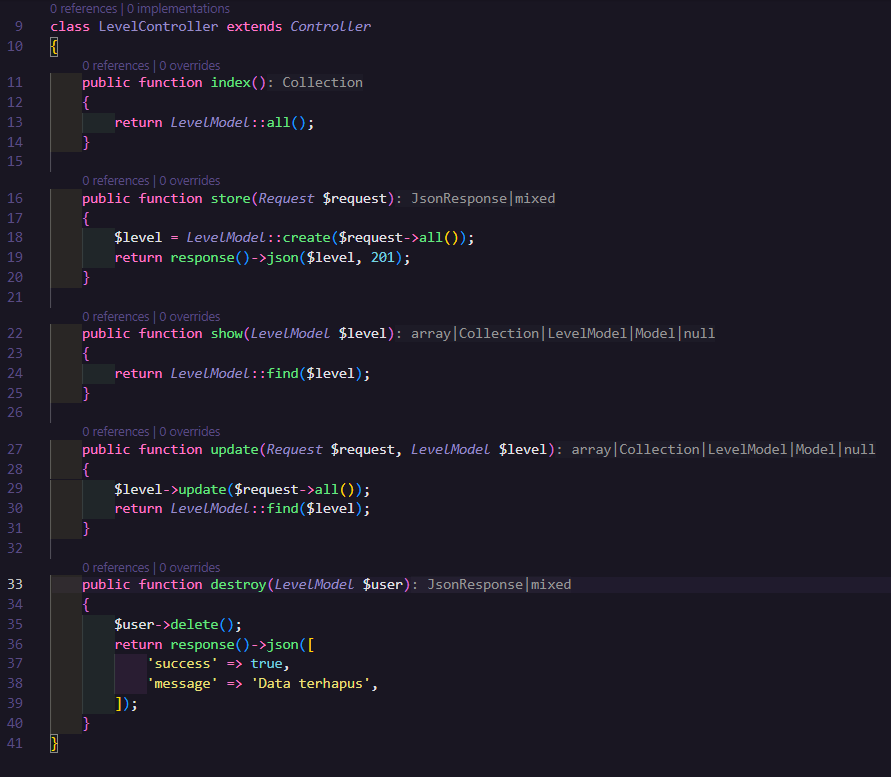


1. Setelah berhasil, buka file tersebut dan tuliskan kode seperti berikut yang berisi fungsi CRUDnya.

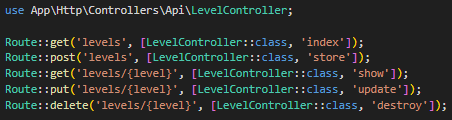






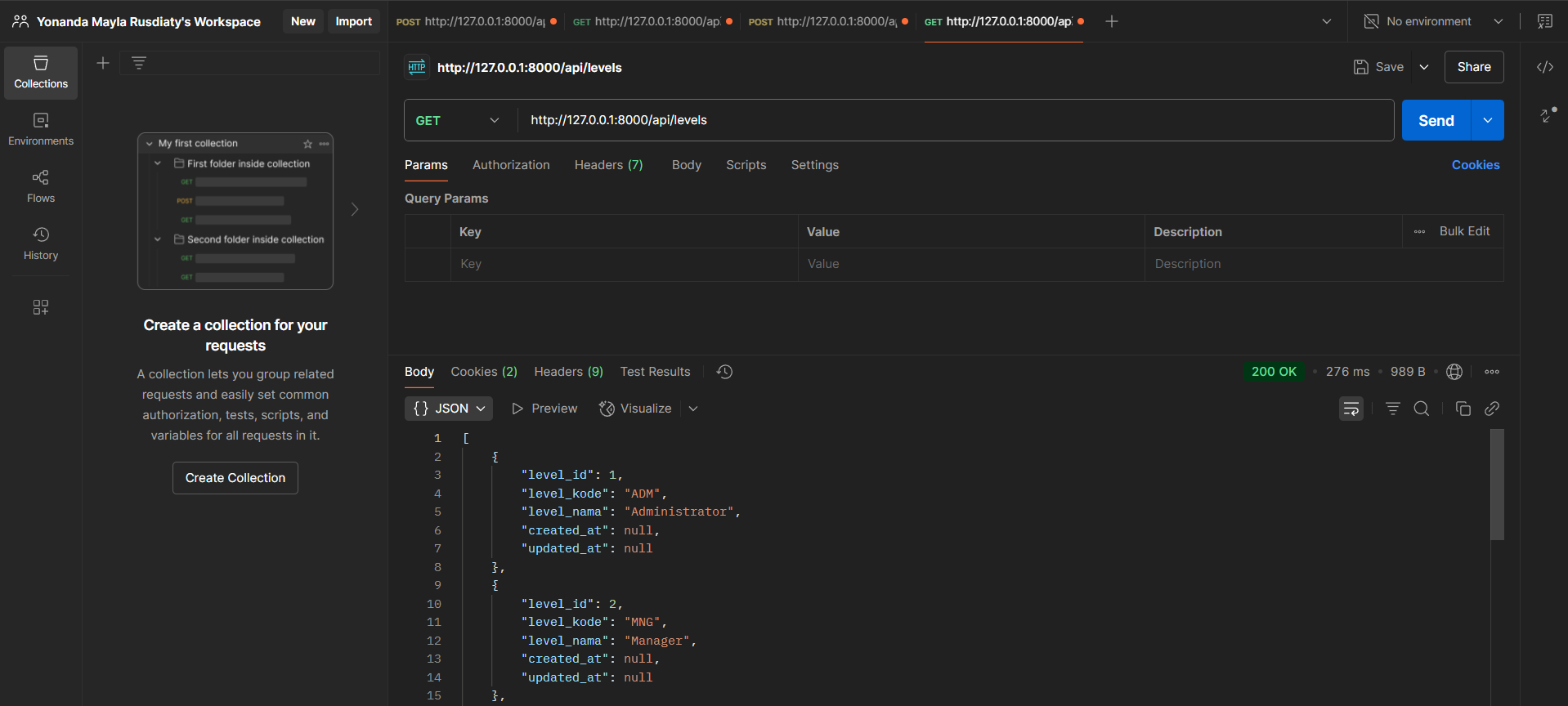


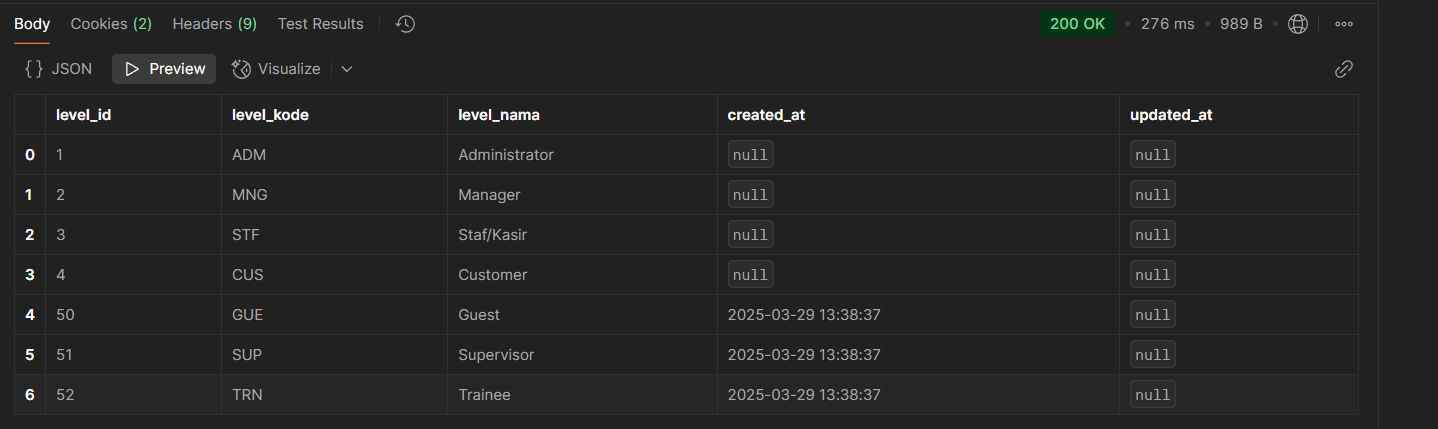
1. Kemudian kita lengkapi routes pada api.php.



1. Jika sudah. Lakukan uji coba API mulai dari fungsi untuk menampilkan data. Gunakan URL: localhost/PWL\_POS-main/public/api/levels dan method GET.

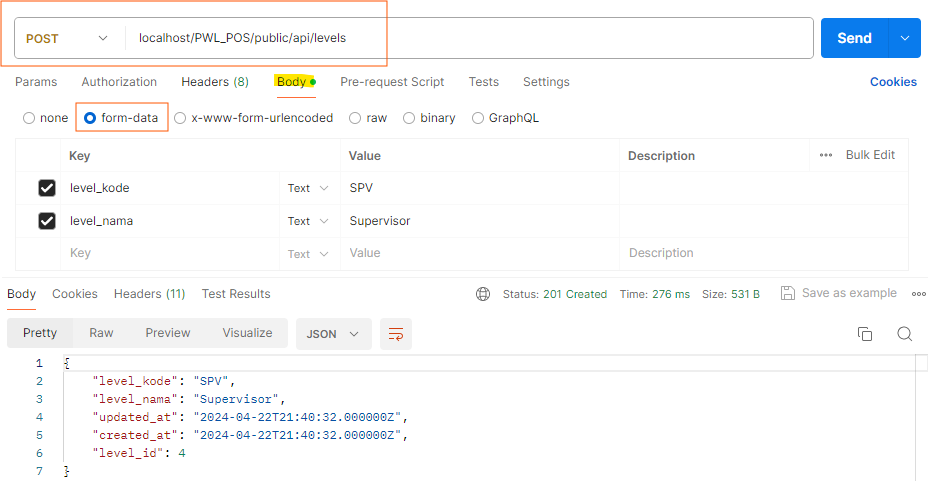
Jelaskan dan berikan screenshoot hasil percobaan Anda.



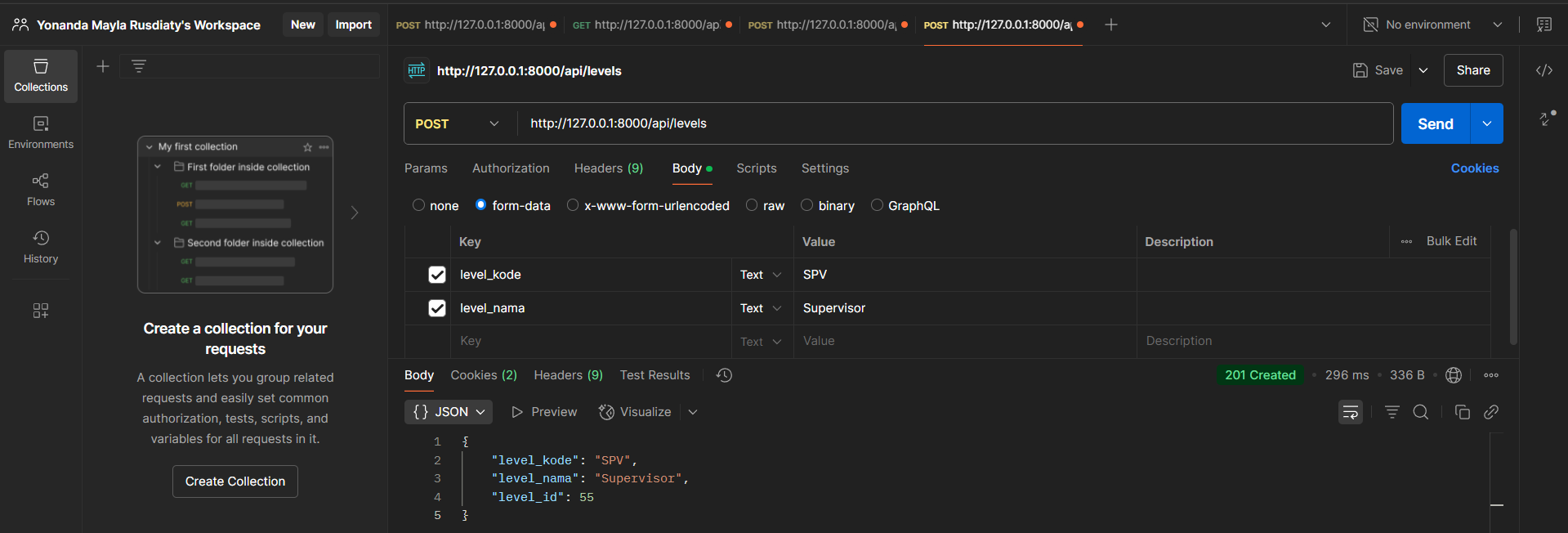


**Jawab:** Hasil yang didapatkan dari Postman saat mengakses endpoint GET /api/levels adalah data dalam format JSON yang menampilkan semua level pengguna dalam sistem

1. 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:** Disini kita sedang mengirimkan permintaan (request) ke API untuk membuat data baru di endpoint /api/levels, yaitu dengan kode SPV as a Supervisor

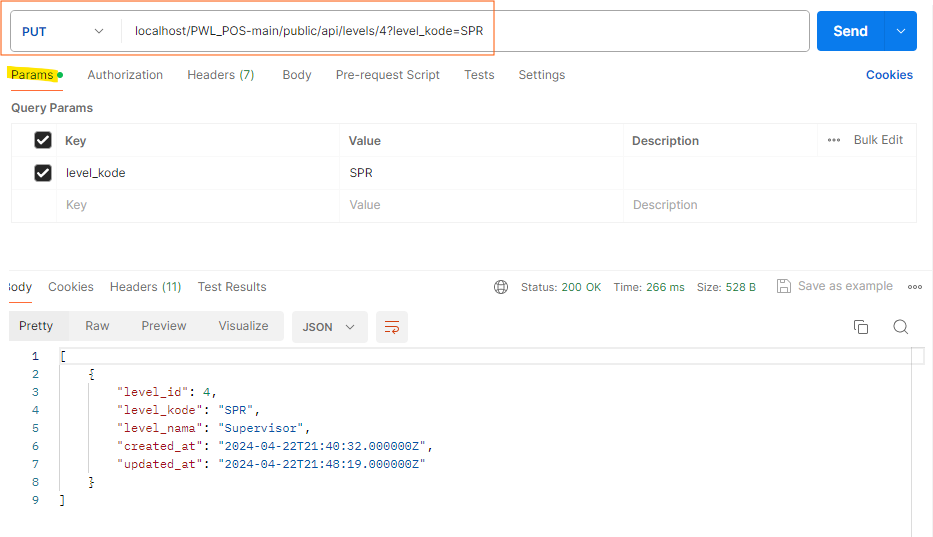
1. Berikutnya lakukan percobaan menampilkan detail data.

Jelaskan dan berikan screenshoot hasil percobaan Anda.

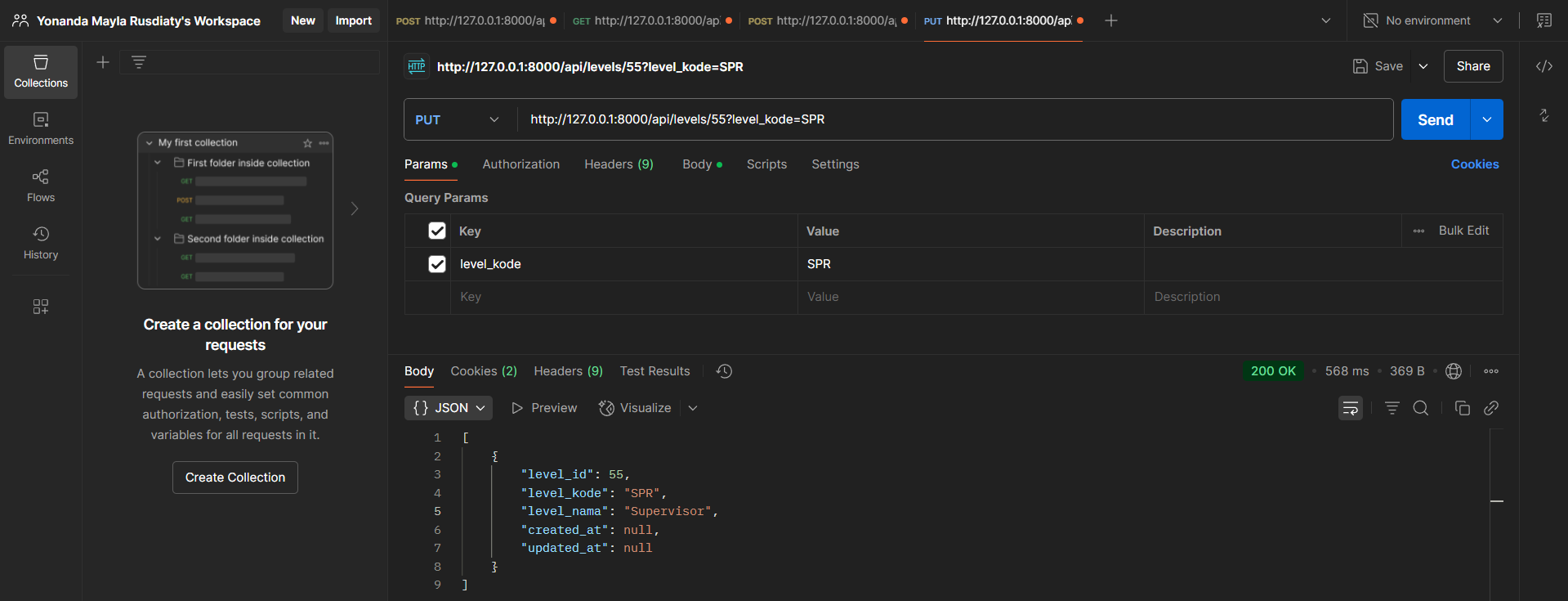


Disini kita sedang menampilkan detail data dengan endpoint /api/levels/ /{level}. Karena saya ingin menampilkan detail data dari supervisor, maka endpointnya adalah <http://127.0.0.1:8000/api/levels/55>. Dan hasilnya seperti pada gambar.

1. 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.

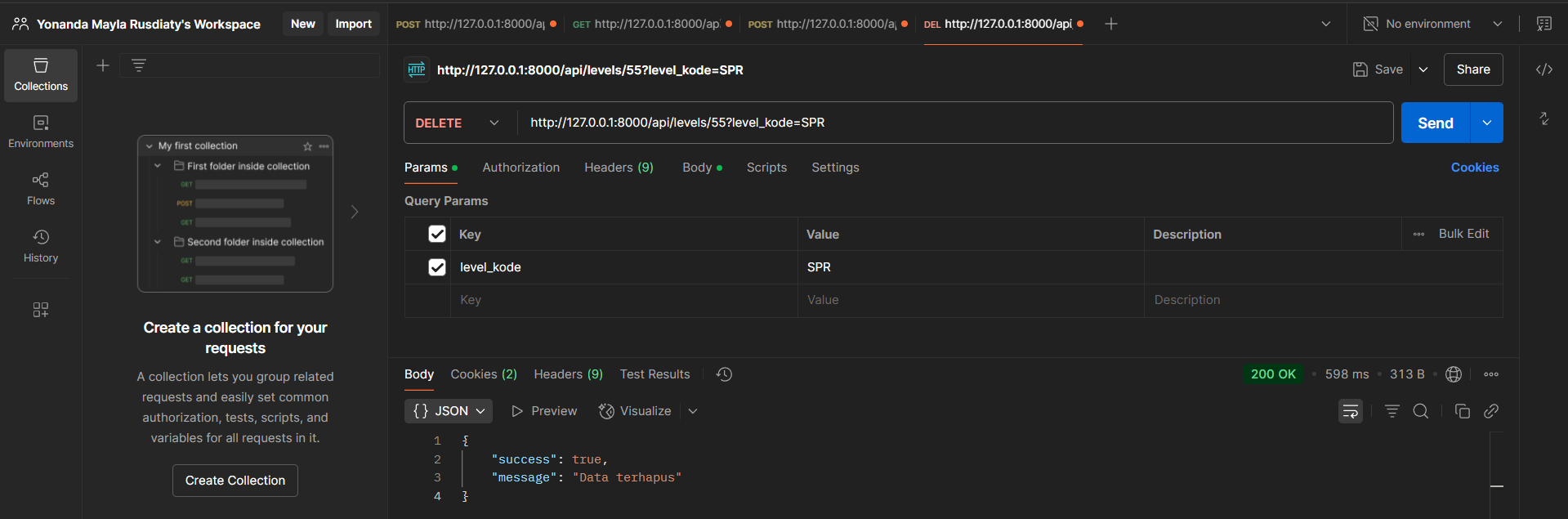


Jelaskan dan berikan screenshoot hasil percobaan Anda.



Data di Query Params bersifat terlihat di URL dan terbatas panjangnya. Params biasanya digunakan untuk GET seperti filter, dan searching.

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



Disini saya menghapus level/55 menggunakan endpoint <http://127.0.0.1:8000/api/levels/55?level_kode=SPR> dengan method DELETE dan menggunakan query params karena yg dibutuhkan hanya id saja

1. Lakukan commit perubahan file pada Github.

# TUGAS

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

# JAWAB

# Tabel : m\_user

# Pertama, buat controller untuk mengolah API pada data user

# 

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

# 

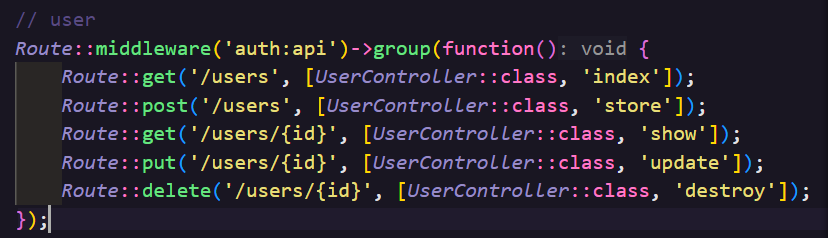
# 

# 

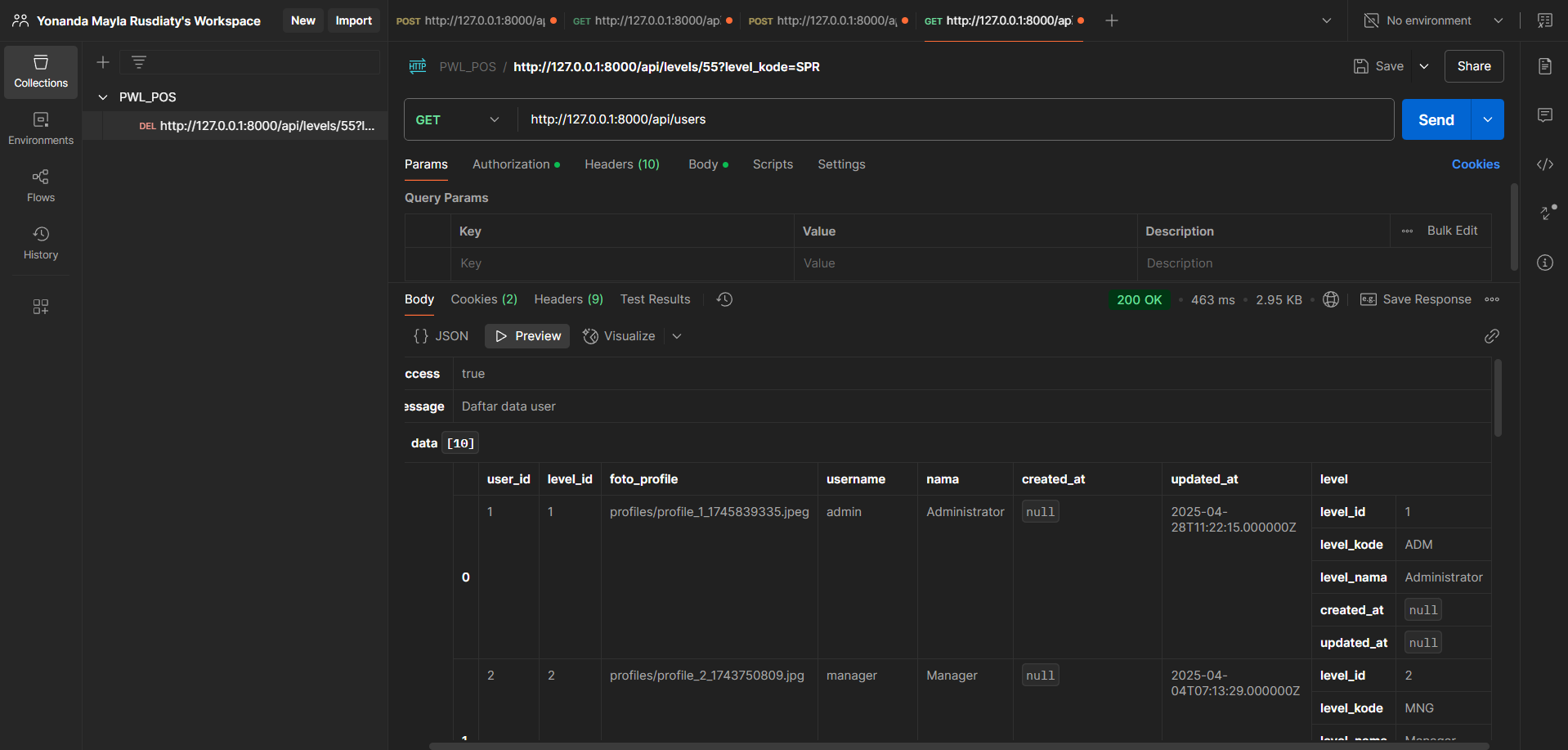
# 

# 

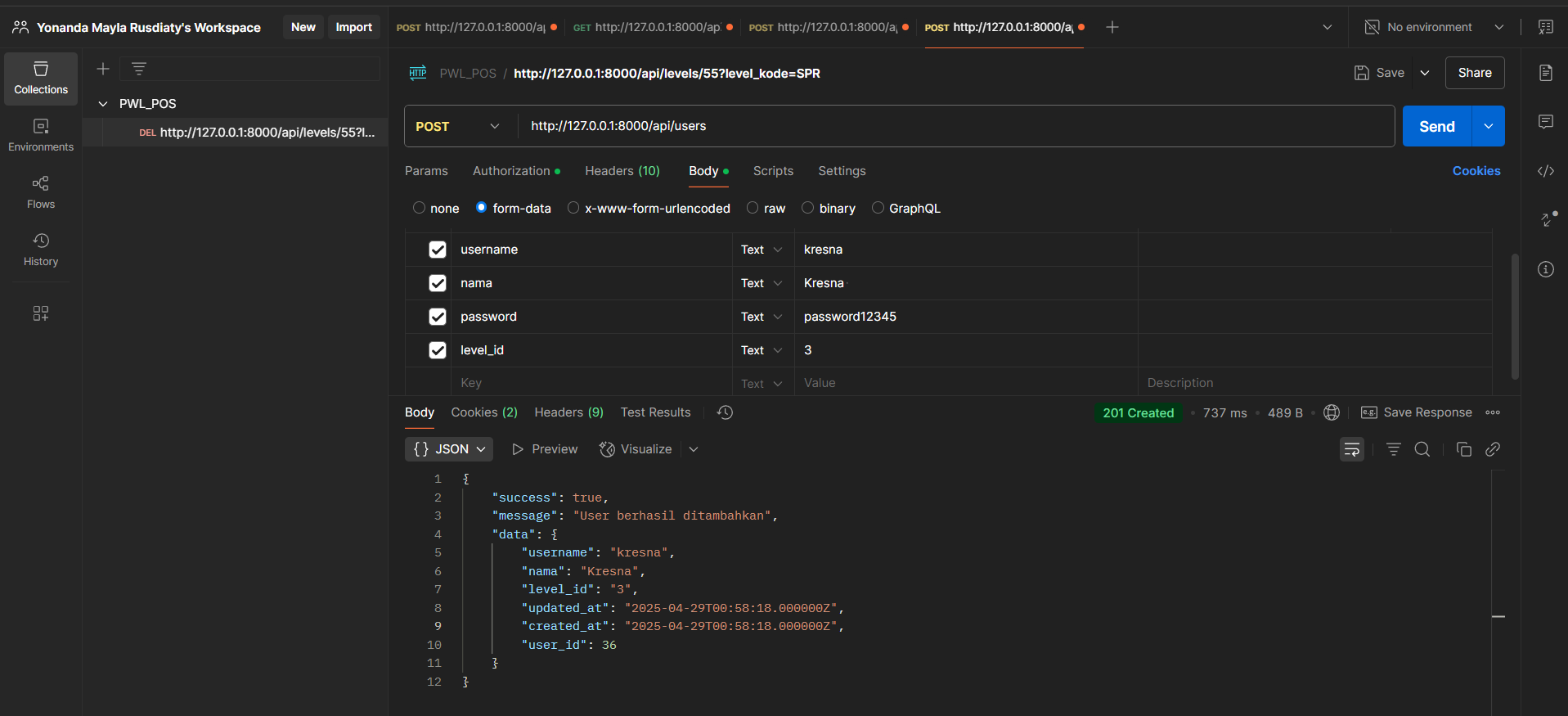
1. Kemudian kita lengkapi routes pada api.php.



1. Jika sudah. Lakukan uji coba API mulai dari fungsi untuk menampilkan data. Gunakan URL: localhost/PWL\_POS-main/public/api/users dan method GET.



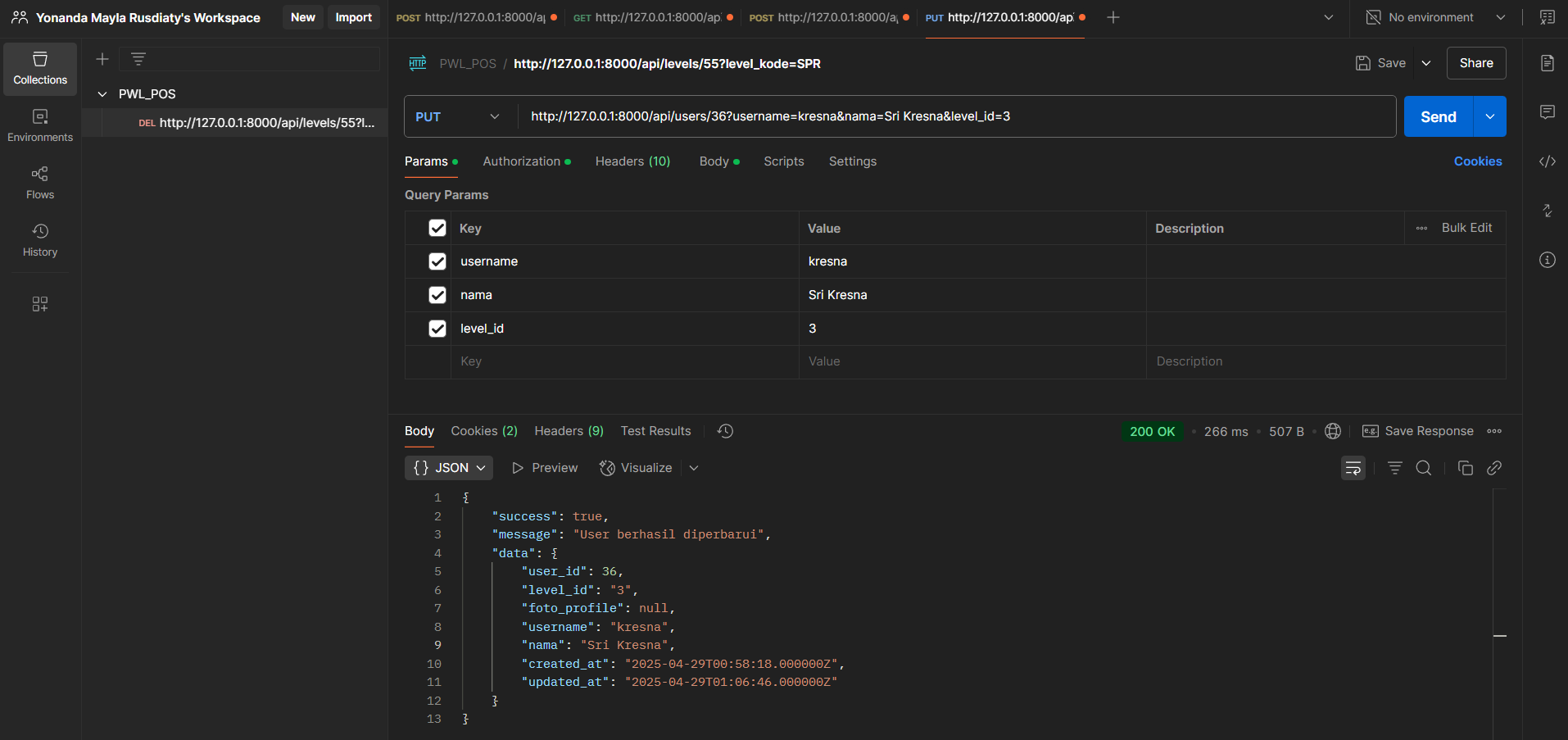
1. Kemudian, lakukan percobaan penambahan data dengan URL : localhost/PWL\_POS- main/public/api/users dan method POST seperti di bawah ini.



# Berikutnya lakukan percobaan menampilkan detail data.

# 

1. 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.



# Terakhir lakukan percobaan hapus data

# 

# Tabel: m\_kategori

# Pertama, buat controller untuk mengolah API pada data kategori

# 

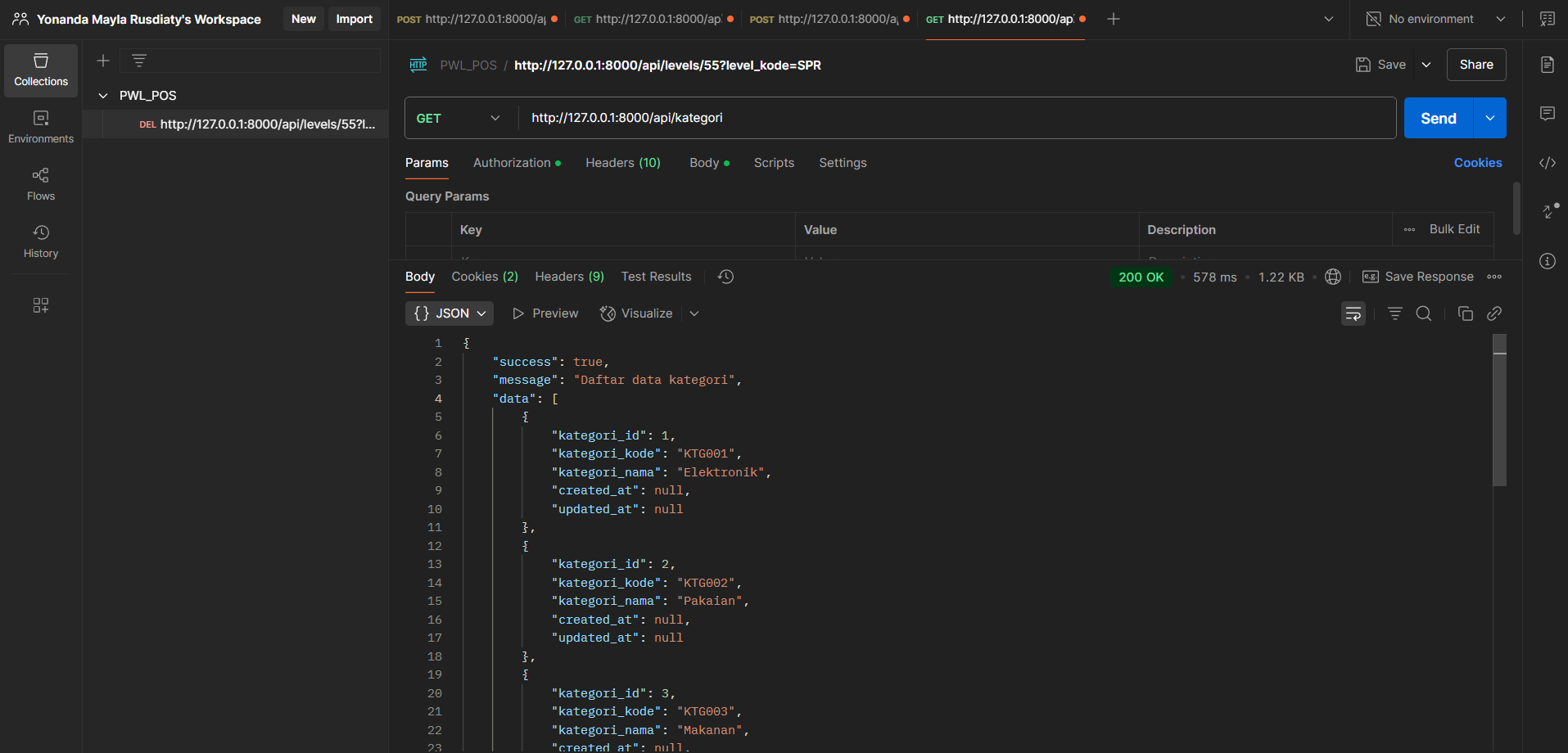
# Setelah berhasil, buka file tersebut dan tuliskan kode seperti berikut yang berisi fungsi CRUDnya

# 

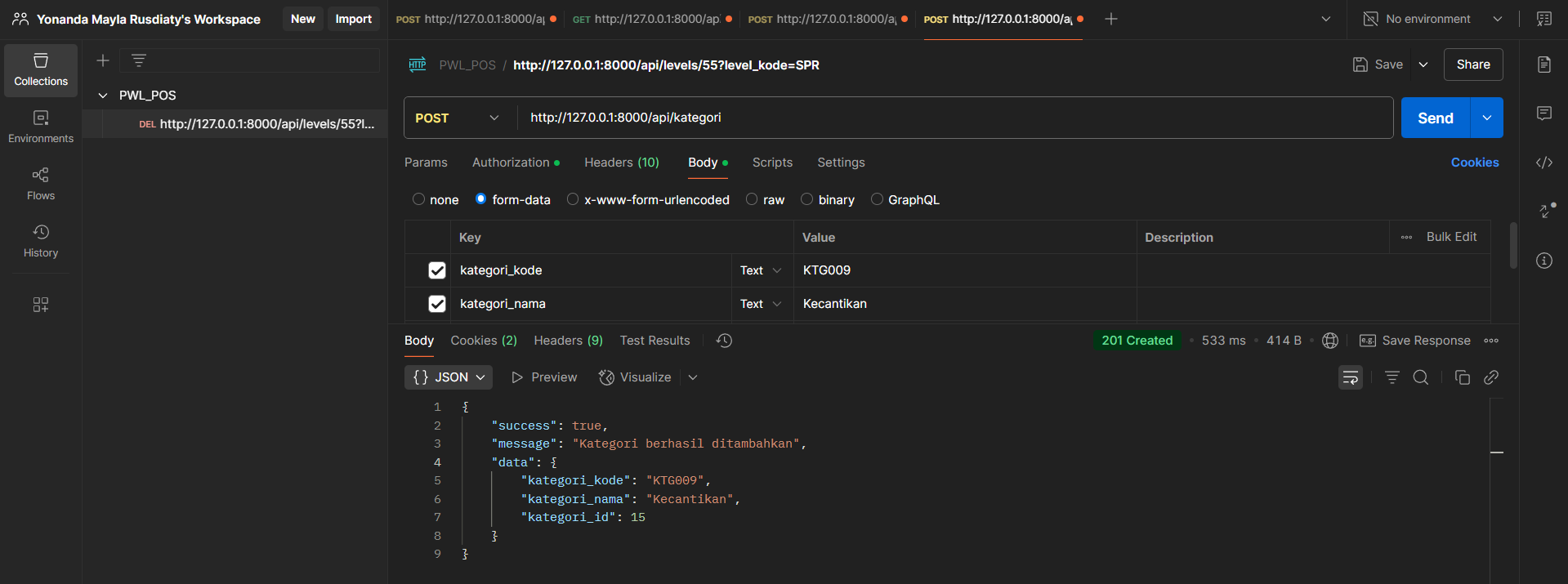
1. Kemudian kita lengkapi routes pada api.php.



1. Jika sudah. Lakukan uji coba API mulai dari fungsi untuk menampilkan data. Gunakan URL: localhost/PWL\_POS-main/public/api/kategori dan method GET.



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



# Berikutnya lakukan percobaan menampilkan detail data.

# 

1. 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.



# Terakhir lakukan percobaan hapus data

# 

# Tabel: m\_barang

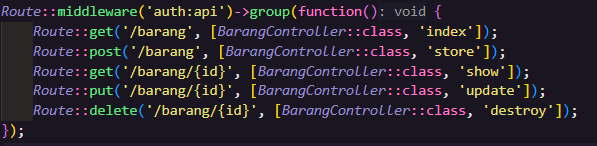
# Pertama, buat controller untuk mengolah API pada data barang

# 

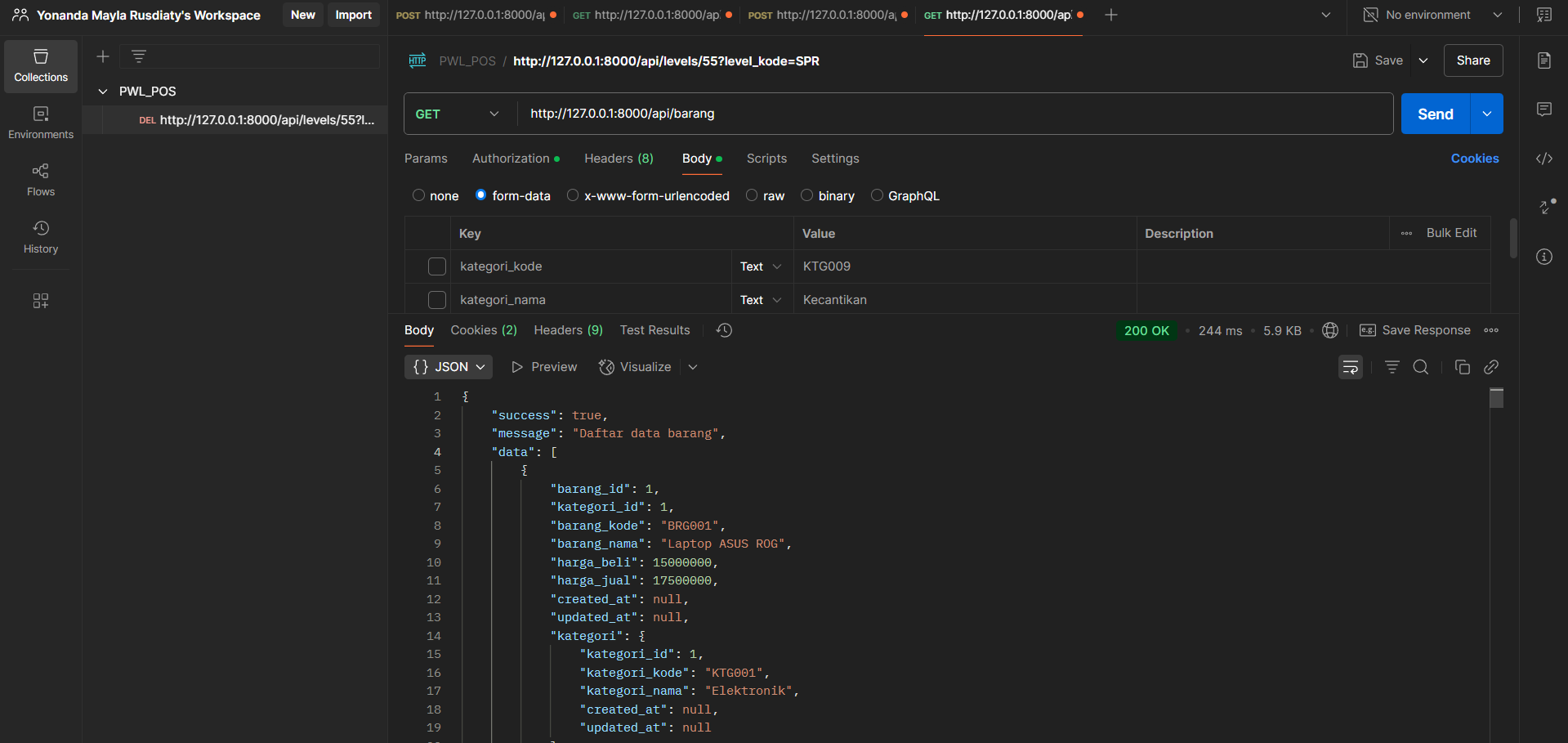
# Setelah berhasil, buka file tersebut dan tuliskan kode seperti berikut yang berisi fungsi CRUDnya

# 

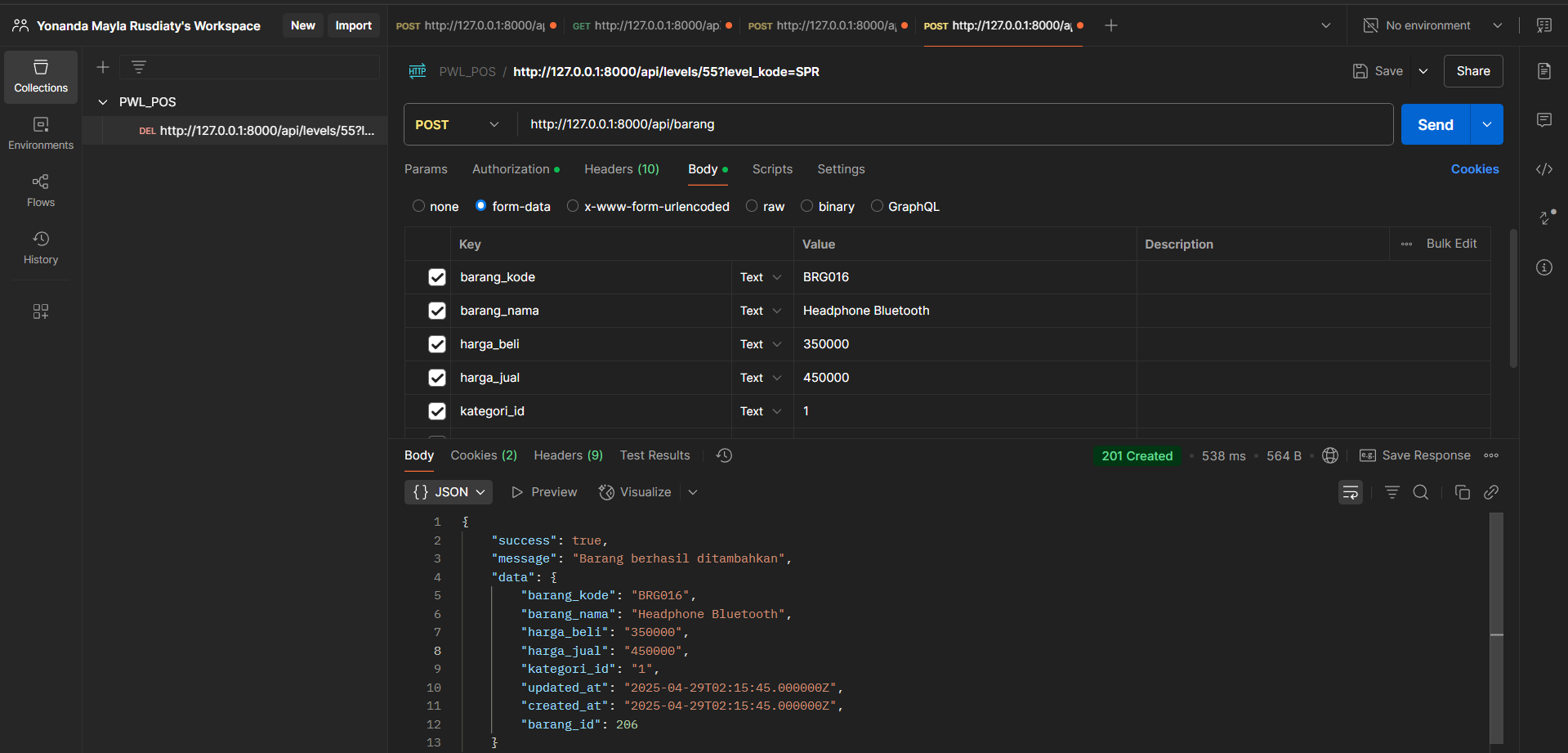
1. Kemudian kita lengkapi routes pada api.php.



1. Jika sudah. Lakukan uji coba API mulai dari fungsi untuk menampilkan data. Gunakan URL: localhost/PWL\_POS-main/public/api/barang dan method GET.



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



# Berikutnya lakukan percobaan menampilkan detail data.

# 

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



# Terakhir lakukan percobaan hapus data

# 

*\*\*\* Sekian, dan selamat belajar \*\*\**