PRAKTIKUM PEMROGRAMAN WEB LANJUT

TUGAS PERTEMUAN 10

RESTFUL API

DISUSUN OLEH :

GILANG PURNOMO **NIM:2341720042**

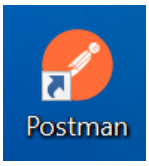
PROGRAM STUDI D-IV TEKNIK INFORMATIKA

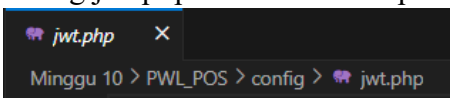
JURUSAN TEKNOLOGI INFORMASI

POLITEKNIK NEGERI MALANG

APRIL 2025

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/jt-auth:2.1.1
   Pastikan Anda terkoneksi dengan internet.
   ```
   PS C:\laragon\www\Web_Lanjut\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
     - 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)
   ```
3. Setelah berhasil menginstall JWT, lanjutkan dengan publish konfigurasi file dengan perintah berikut:
   php artisan vendor:publish --provider="Tymon\JWTAuth\Providers\LaravelServiceProvider"
   ```
   INFO  Publishing assets.

   Copying file [C:\laragon\www\Web_Lanjut\Minggu 10\PWL_POS\vendor\tymon\jwt-auth\config\config.php] to [C:\laragon\www\Web_Lanjut\Minggu 10\PWL_POS\config\jwt.php]  DONE

   PS C:\laragon\www\Web_Lanjut\Minggu 10\PWL_POS>
   ```
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.
   

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
   ```
   PS C:\laragon\www\Web_Lanjut\Minggu 10\PWL_POS> php artisan jwt:secret
   jwt-auth secret [413rCq73xe54RJ5YDnYwjIRYaQdgqdMsbWP4eyeXM9r4OQNwX6xLf0dALiJTRIT3] 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'
        ],
    ],
```

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

```
use Tymon\JWTAuth\Contracts\JWTSubject;
use Illuminate\Database\Eloquent\Relations\BelongsTo;
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';
```

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.

```
PS C:\laragon\www\Web_Lanjut\Minggu 10\PWL_POS> php artisan make:controller Api/RegisterController

  INFO  Controller [C:\laragon\www\Web_Lanjut\Minggu 10\PWL_POS\app\Http\Controllers\Api\RegisterController.php] created successfully.
```

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

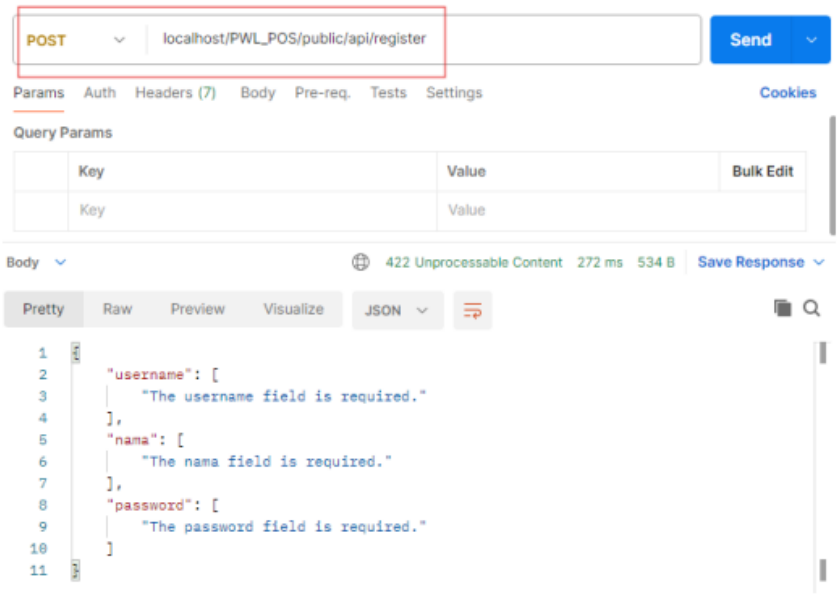
```
<?php

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)
    {
```

```php
        //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);
    }
}
```

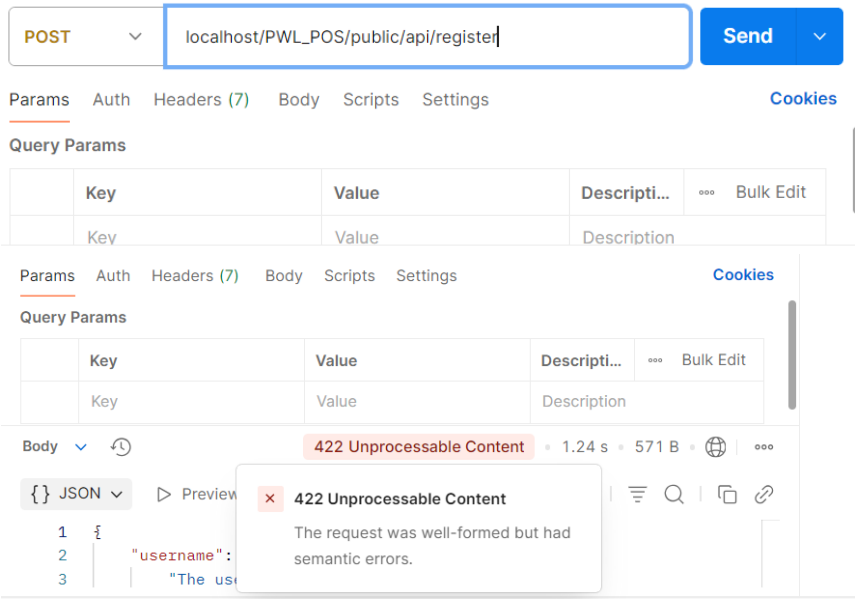
10. Selanjutnya buka routes/api.php, ubah semua kode menjadi seperti berikut.

```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');
```

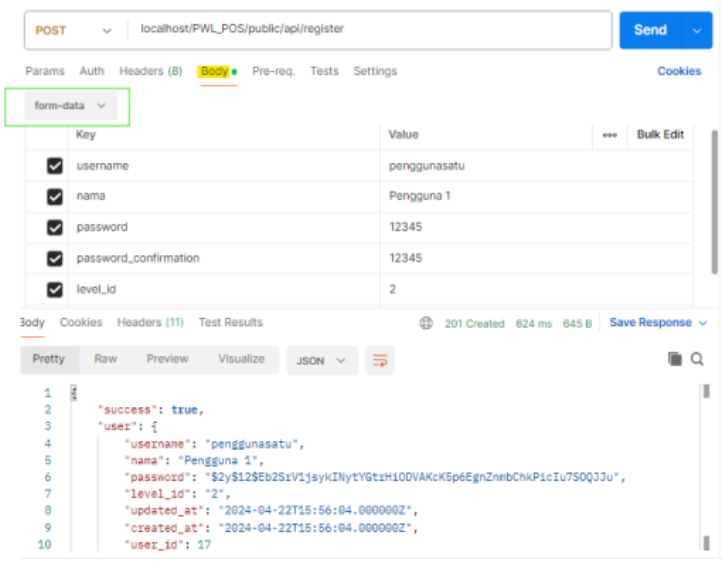
11. Jika sudah, kita akan melakukan uji coba REST API melalui aplikasi Postman. Buka aplikasi Postman, isi URL localhost/PWL_POS/public/api/register serta method POST. Klik Send.



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

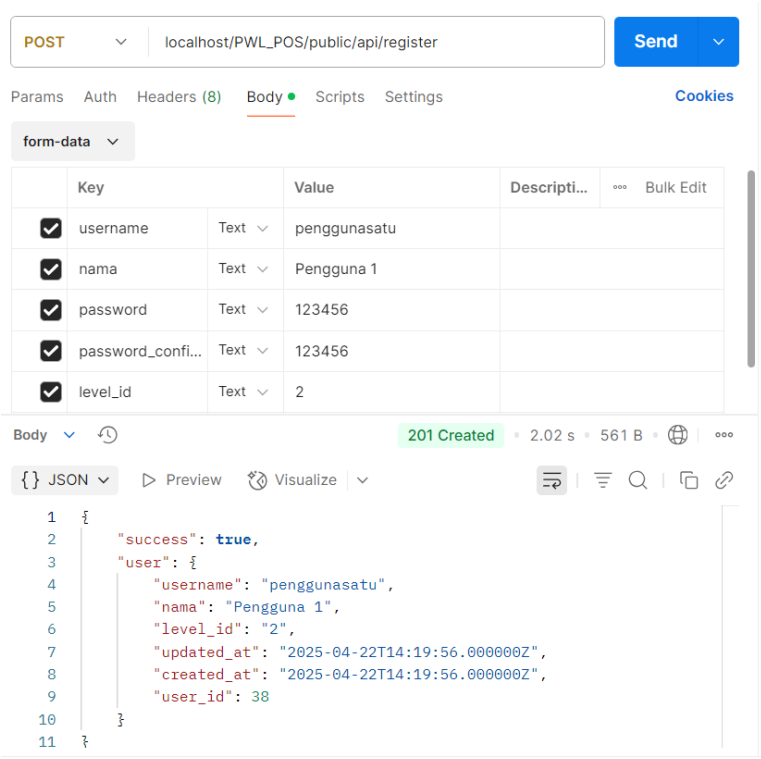


12. Sekarang kita coba masukkan data. Klik tab Body dan pilih form-data. Isikan key sesuai dengan kolom data, serta isikan data registrasi menggunakan nilai yang Anda inginkan.



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.



13. Lakukan commit perubahan file pada Github.
    Link commit :
    https://github.com/Gilangp/Pemrograman_Web_Lanjut/commit/1c0de2bb944cceda6b9abbf1a19fc5f122dba9c0

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.

```
PS C:\laragon\www\Web_Lanjut\Minggu 10\PWL_POS> php artisan make:controller Api/LoginController

   INFO  Controller [C:\laragon\www\Web_Lanjut\Minggu 10\PWL_POS\app\Http\Controllers\Api\LoginController.php] created successfully.
```

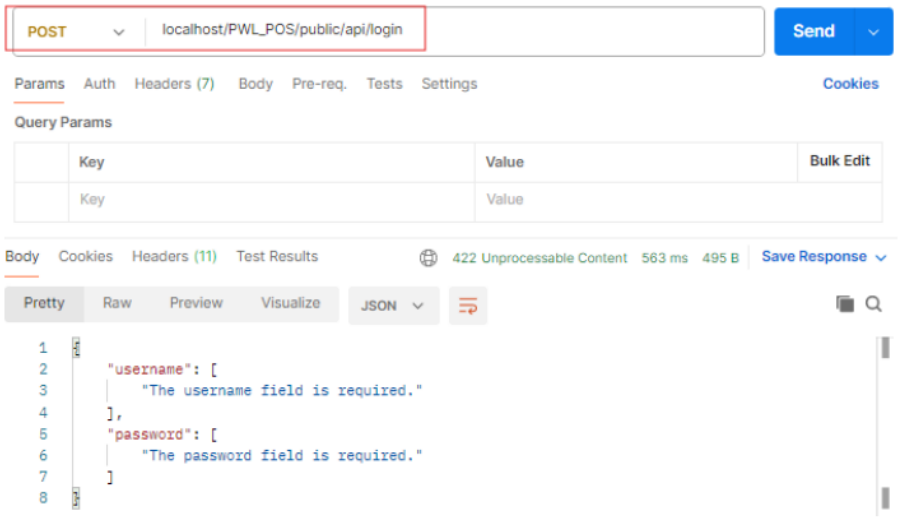
2. Buka file tersebut, dan ubah kode menjadi seperti berikut.

```php
<?php

namespace App\Http\Controllers\Api;

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

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);
    }
}
```

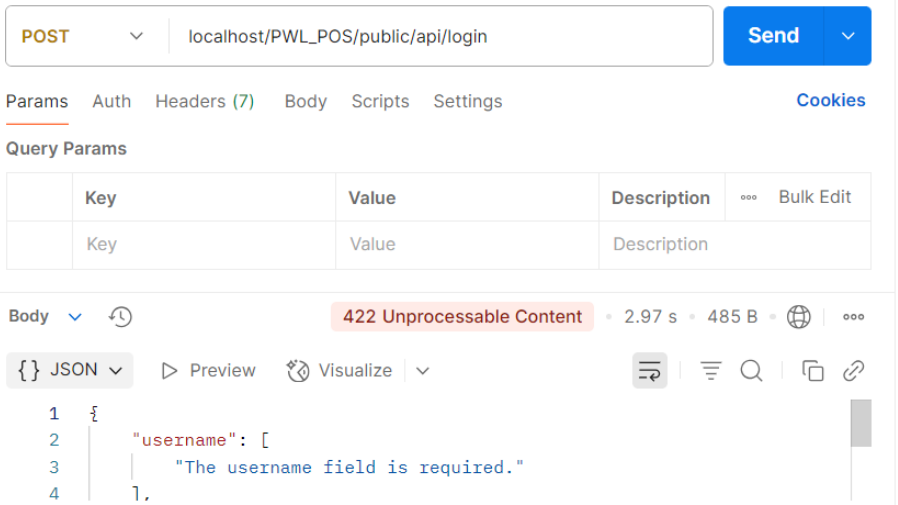
3. Berikutnya tambahkan route baru pada file api.php yaitu /login dan /user.

```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();
});
```

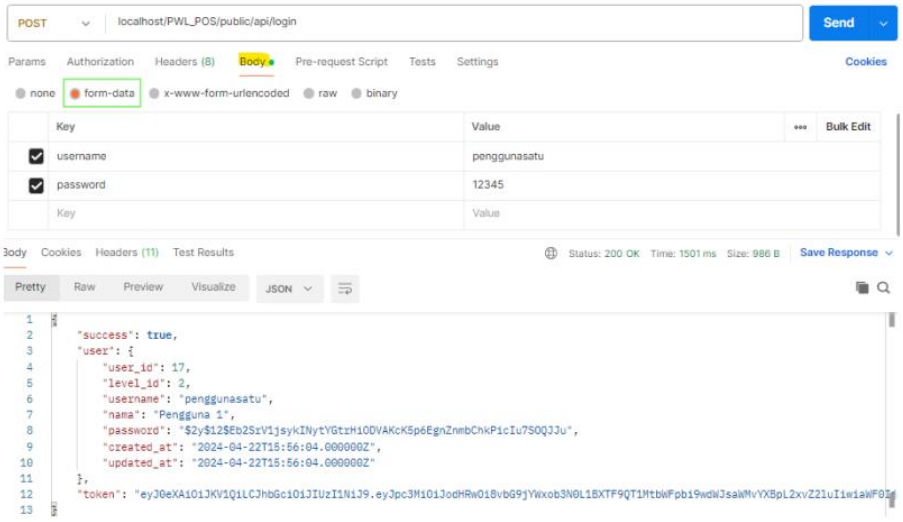
4. Jika sudah, kita akan melakukan uji coba REST API melalui aplikasi Postman. Buka aplikasi Postman, isi URL localhost/PWL_POS/public/api/login serta method POST. Klik Send



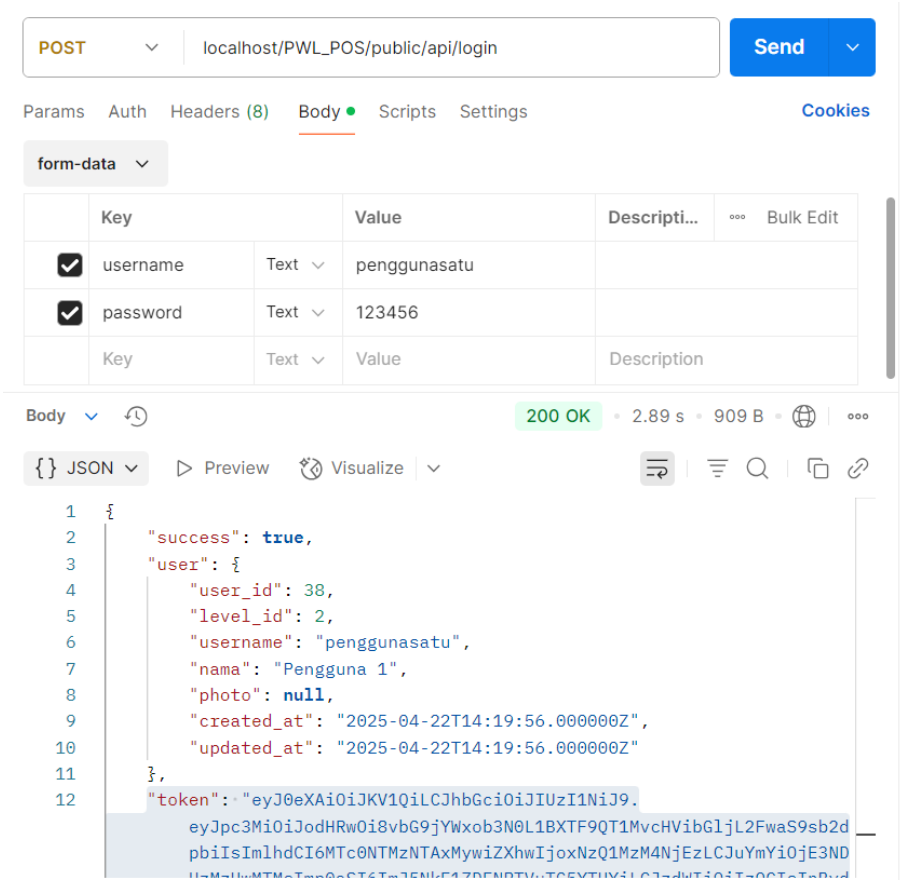
Jika berhasil akan muncul error validasi seperti gambar di atas.
Lakukan percobaan yang sama dan berikan screenshoot hasil percobaan Anda.



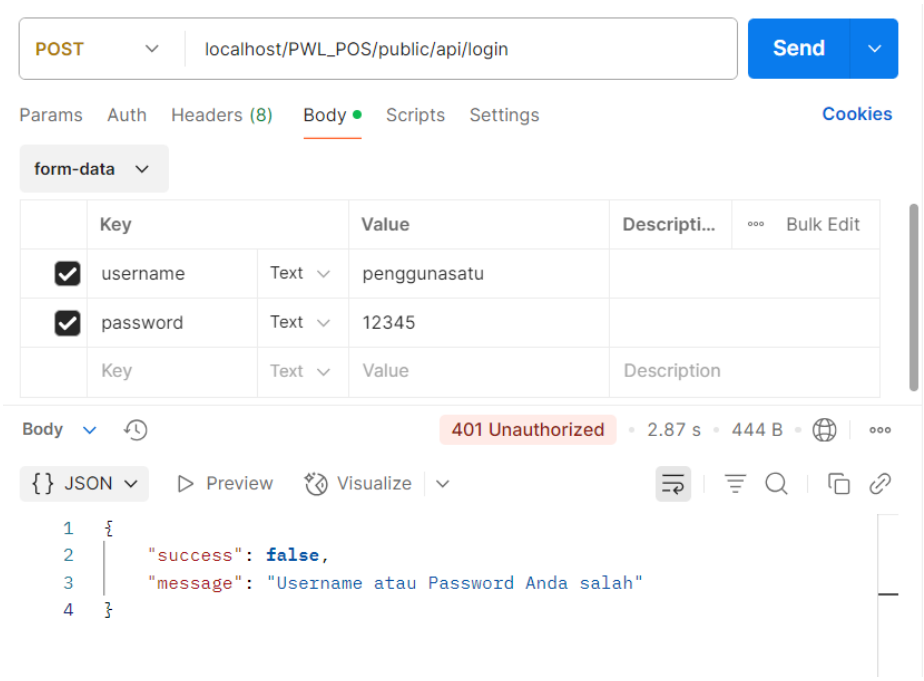
5. Selanjutnya, isikan username dan password sesuai dengan data user yang ada pada database. Klik tab Body dan pilih form-data. Isikan key sesuai dengan kolom data, serta isikan data user. Klik tombol Send, jika berhasil maka akan keluar tampilan seperti berikut. Copy nilai token yang diperoleh pada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



Lakukan percobaan yang sama dan berikan screenshoot hasil percobaan Anda.

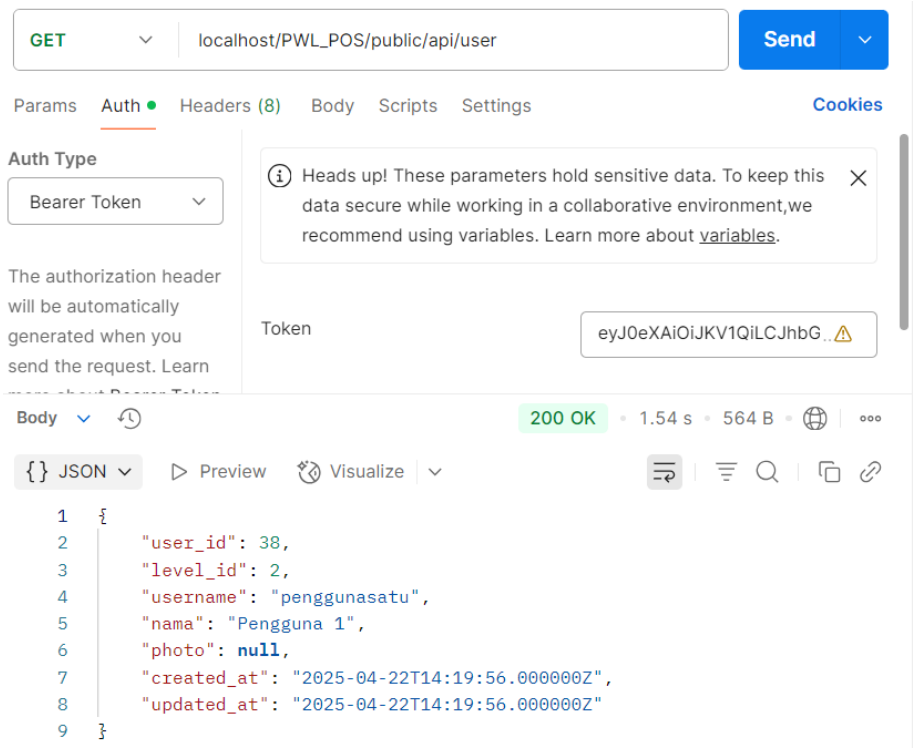


6. Lakukan percobaan yang untuk data yang salah dan berikan screenshoot hasil percobaan Anda.



Harusnya passowordnya ada 6 digit dan diatas menginputkan 5 digit maka terjadi error karena password tidak sesuai data.

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.



Jadi pertama membuat request get dengan endpoint api/user, lalu menggunakan metode Auth dengan tipe Bearer Token dan menginputkan token yang sudah didapat saat login untuk mengetahui informasi login.

8. Lakukan commit perubahan file pada Github.
   Link Commit :
   https://github.com/Gilangp/Pemrograman_Web_Lanjut/commit/0cb32efe2fb1615d9789a0d0700a36918f5074bb

Praktikum 3 – Membuat RESTful API Logout

1. Tambahkan kode berikut pada file .env
   JWT_SHOW_BLACKLIST_EXCEPTION=true

```
61  JWT_SECRET=413rCq73xe54RJ5YDnYwjIRYaQdgqdMsbWP4eyeXM9r4OQNwX6xLf0dALiJTRIT3
62  JWT_SHOW_BLACKLIST_EXCEPTION=true
```

2. Buat Controller baru dengan nama LogoutController.
   php artisan make:controller Api/LogoutController

```
PS C:\laragon\www\Web_Lanjut\Minggu 10\PWL_POS> php artisan make:controller Api/LogoutController

  INFO  Controller [C:\laragon\www\Web_Lanjut\Minggu 10\PWL_POS\app\Http\Controllers\Api\LogoutController.php] created successfully.
```

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

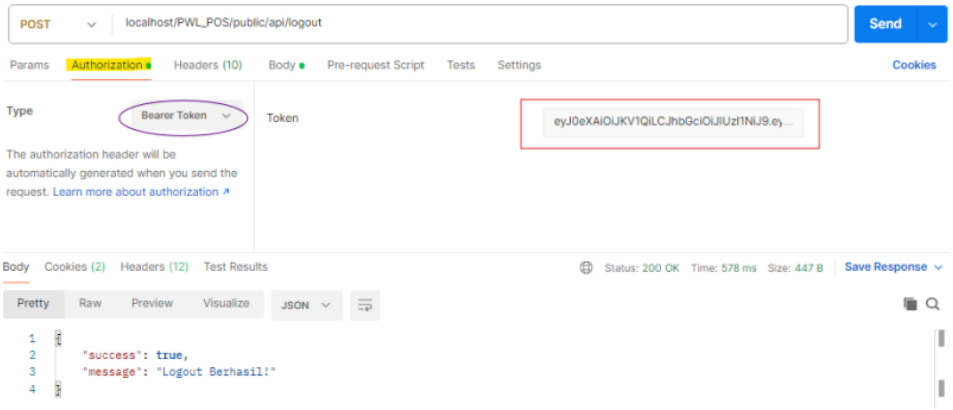
```php
<?php

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!',
            ]);
        }
    }
}
```

4. Lalu kita tambahkan routes pada api.php

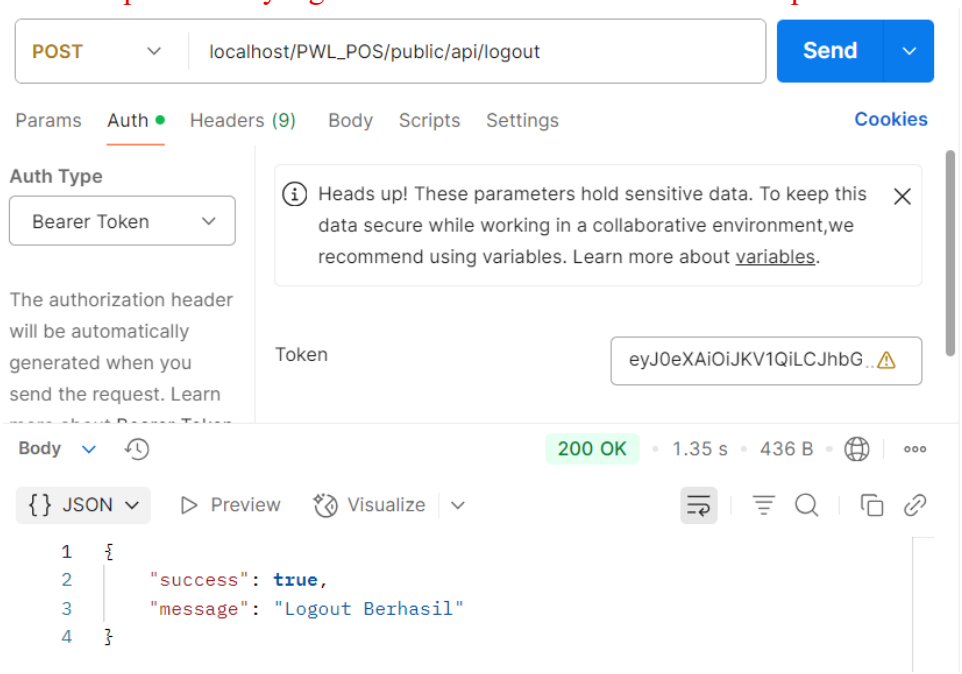
```php
Route::post('/logout', App\Http\Controllers\Api\LogoutController::class)->name('logout');
```

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.

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.



7. Lakukan commit perubahan file pada Github.
   Link Commit :
   https://github.com/Gilangp/Pemrograman_Web_Lanjut/commit/742c5cd472535b8a93fdc901f7e9981903d2ed1f

Praktikum 4 – Implementasi CRUD dalam RESTful API

1. Pertama, buat controller untuk mengolah API pada data level.
   php artisan make:controller Api/LevelController

```
PS C:\laragon\www\Web_Lanjut\Minggu 10\PWL_POS> php artisan make:controller Api/LevelController

   INFO  Controller [C:\laragon\www\Web_Lanjut\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.

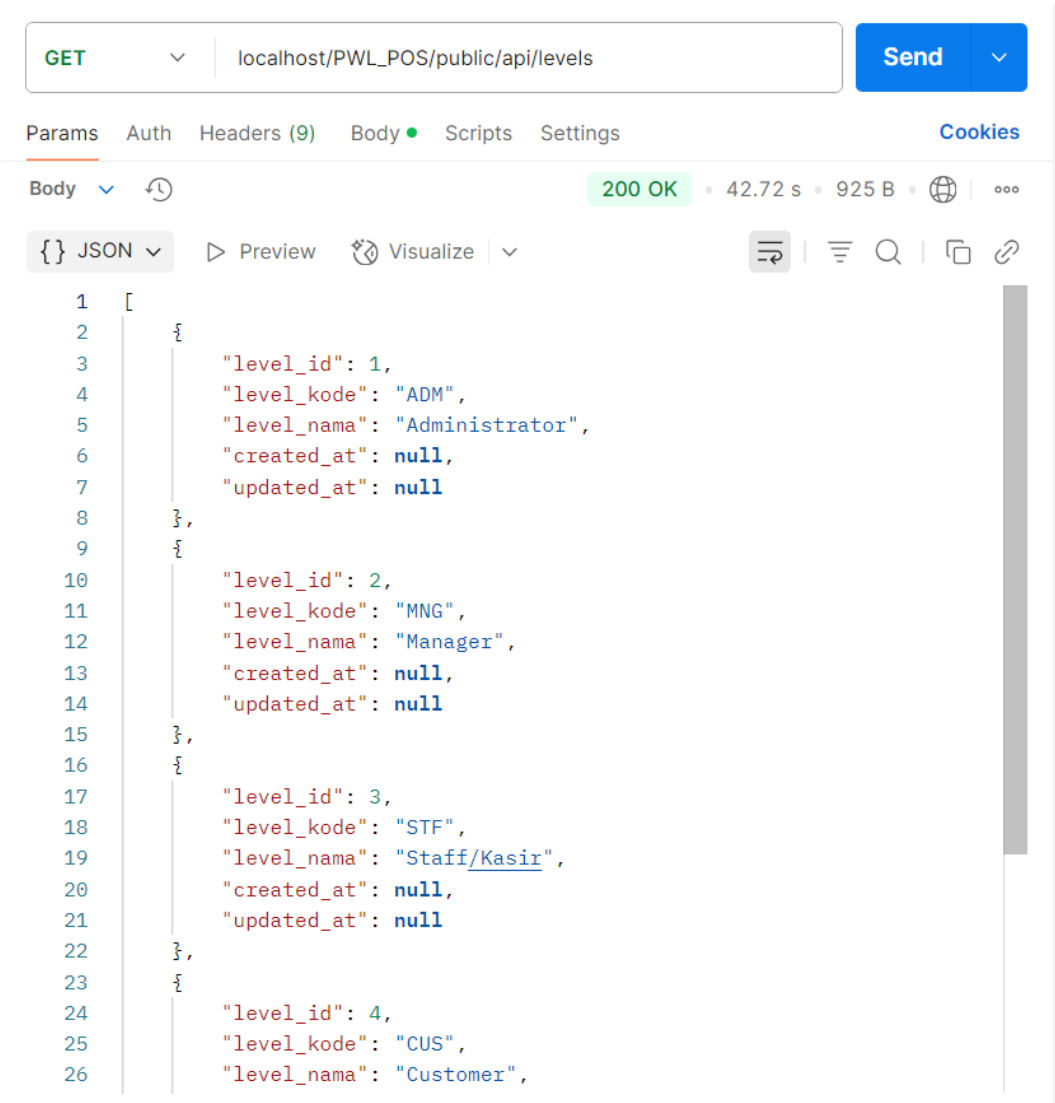
```php
namespace App\Http\Controllers\Api;
use App\Http\Controllers\Controller;
use Illuminate\Http\Request;
use App\Models\LevelModel;

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

```php
    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);
    }

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

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

3. Kemudian kita lengkapi routes pada api.php.

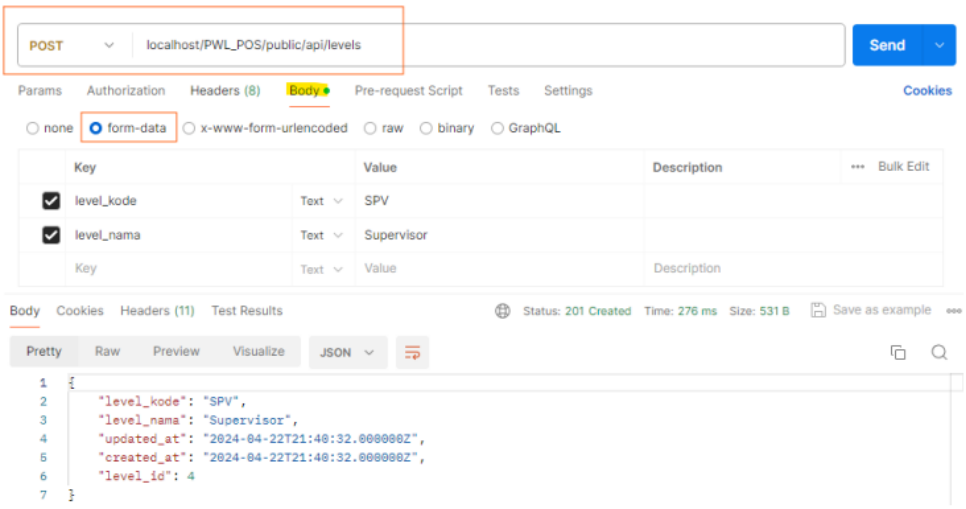
```php
use App\Http\Controllers\Api\LevelController;

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']);
```

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.

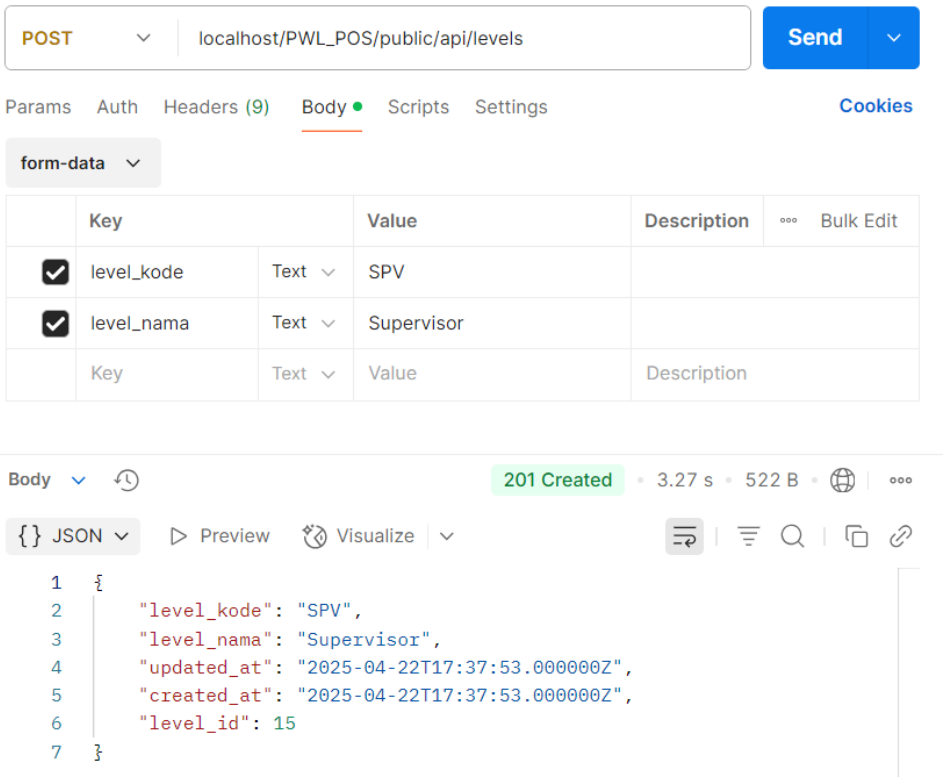


Jadi menggunakan get untuk mengambil data dengan api/levels dan ditampilkan isi datanya.

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

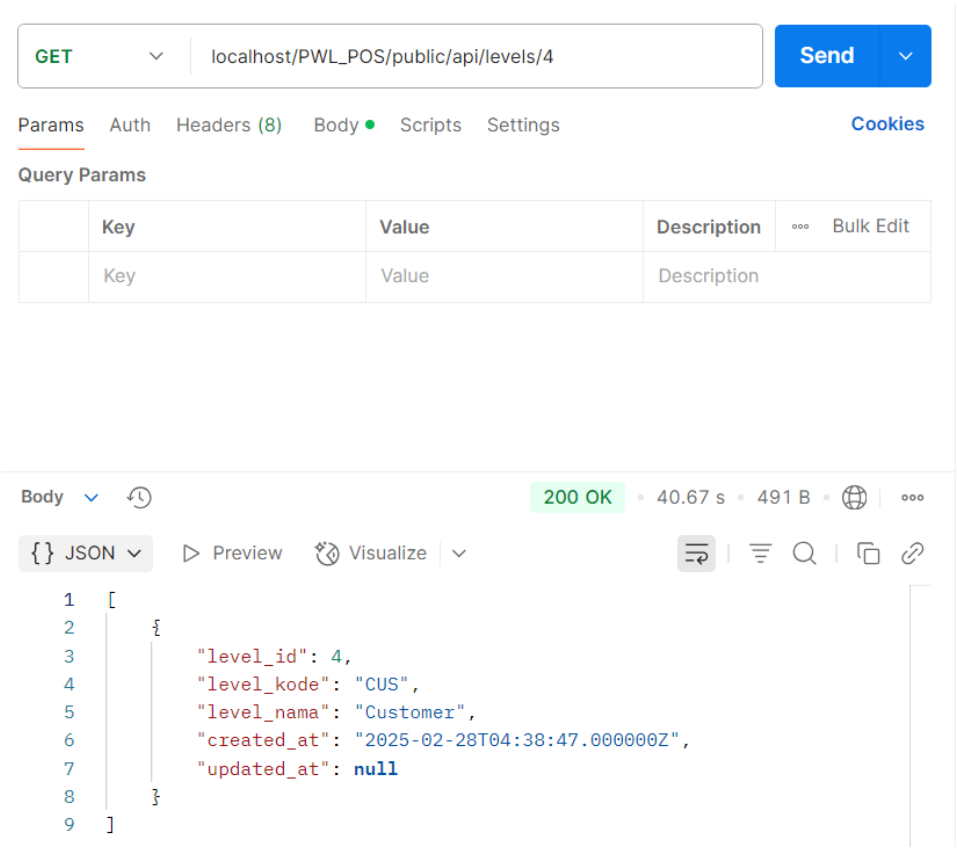


Jelaskan dan berikan screenshoot hasil percobaan Anda.



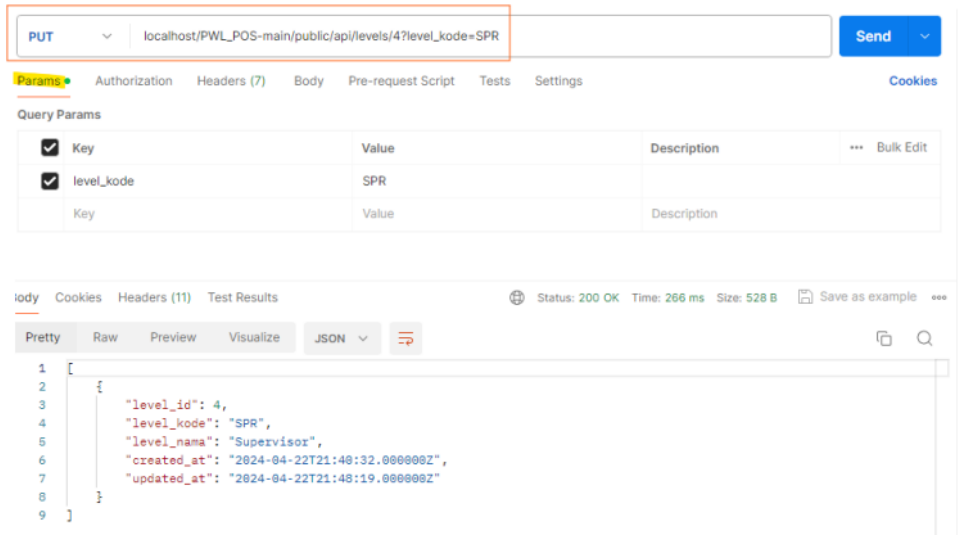
Jadi menggunakan post untuk membuat data baru pada api/levels dengan isi form tersebut.

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

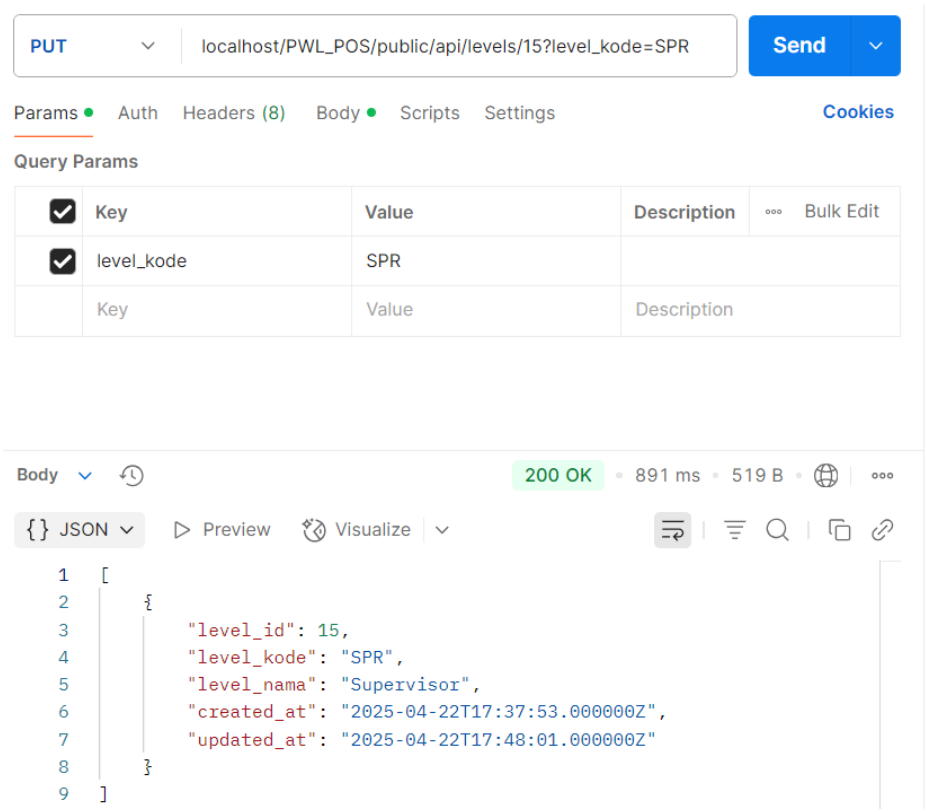


Menggunakan get untuk mengambil data berdasarkan api/levels/{id}agar bisa melihat detail data berdasarkan idnya.

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

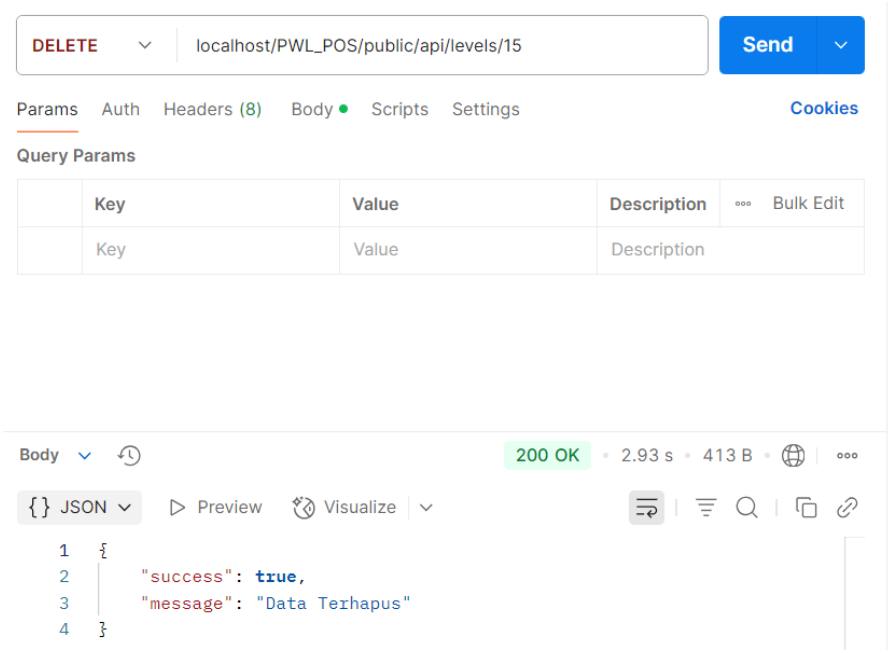


Jelaskan dan berikan screenshoot hasil percobaan Anda.



Jadi menggunakan put untuk memperbarui data berdasarkan api/levels/{id} dan parameter level_kode untuk inputan data baru lalu akan ditampilkan dengan data yang barunya.

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



Menggunakan delete untuk menghapus data dengan api/levels/{id}, jadi berdasarkan level_idnya.

9. Lakukan commit perubahan file pada Github.
   Link Commit :
   https://github.com/Gilangp/Pemrograman_Web_Lanjut/commit/4a85899adb38dc3e7005e12db020ad571bcd3914

TUGAS

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

Link Commit :
https://github.com/Gilangp/Pemrograman_Web_Lanjut/commit/83575608c4d94575b7181ba06d392a7f9ea513f1