



S

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 pelajari

A. 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).



B. 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 :

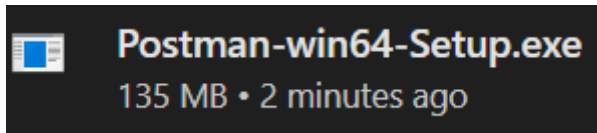
JWT (JSON Web Token) adalah cara untuk mentransfer informasi antara dua pihak secara aman sebagai objek JSON. Ini terdiri dari tiga bagian: header, payload, dan signature. Setelah pengguna berhasil autentikasi, server menghasilkan token JWT yang disematkan dalam permintaan HTTP. Server kemudian memvalidasi 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.



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
```

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.





5. Setelah itu jalankan perintah 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.

```
.env
49  PUSHER_HOST=
50  PUSHER_PORT=443
51  PUSHER_SCHEME=https
52  PUSHER_APP_CLUSTER=mt1
53
54  VITE_APP_NAME="${APP_NAME}"
55  VITE_PUSHER_APP_KEY="${PUSHER_APP_KEY}"
56  VITE_PUSHER_HOST="${PUSHER_HOST}"
57  VITE_PUSHER_PORT="${PUSHER_PORT}"
58  VITE_PUSHER_SCHEME="${PUSHER_SCHEME}"
59  VITE_PUSHER_APP_CLUSTER="${PUSHER_APP_CLUSTER}"
60
61  JWT_SECRET=YP0K6AiJWtrvjiAk1VEihdTbLFDGCq30JOCrcmQxwJxbqtpGAuDjCtOLqkFyhDAj

PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

PS C:\laragon\www\FWL_POS> php artisan jwt:secret
jwt-auth secret [YP0K6AiJWtrvjiAk1VEihdTbLFDGCq30JOCrcmQxwJxbqtpGAuDjCtOLqkFyhDAj] 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',
    ],
],
```

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



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

```
<?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';
}
```

```
<?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();
    }

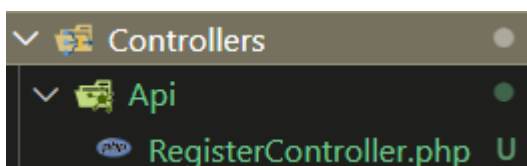
    public function getJWTCustomClaims(){
        return [];
    }

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

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

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

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





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

```
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     public function __invoke(Request $request)
13     {
14         //set validation
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 validations 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 }
```



```
<?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

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');
```

```
<?php

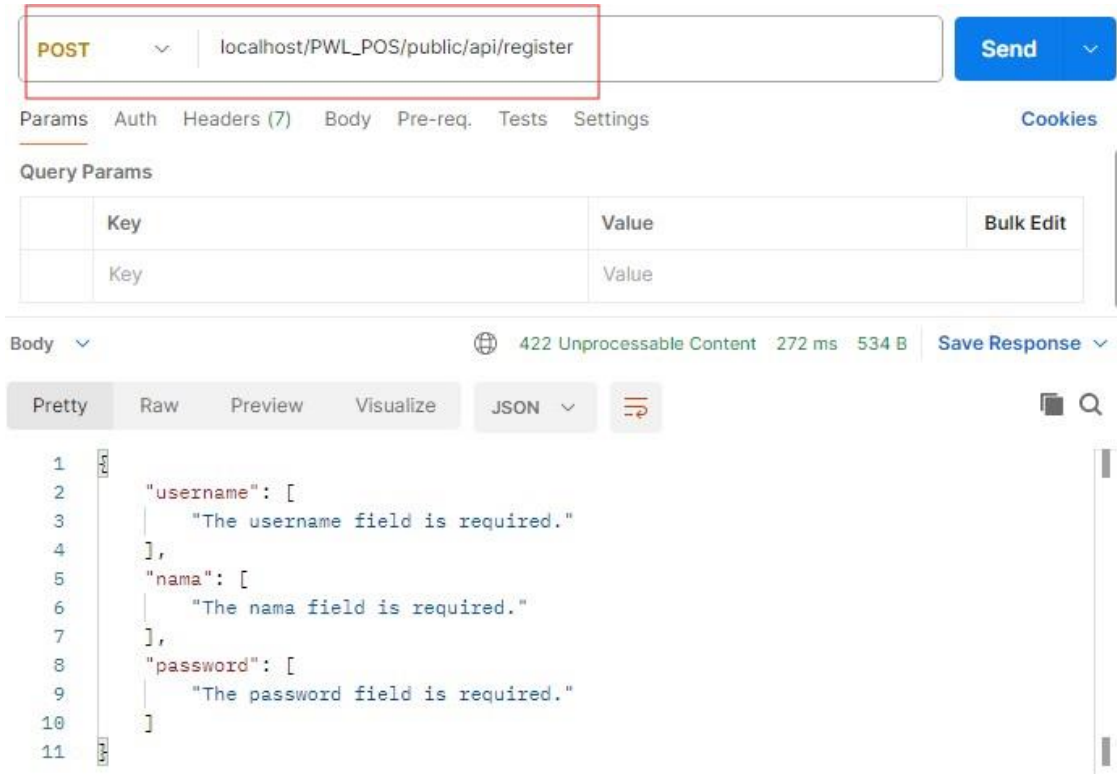
use App\Http\Controllers\Api\RegisterController;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Route;

/*
|-----
| 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');
```

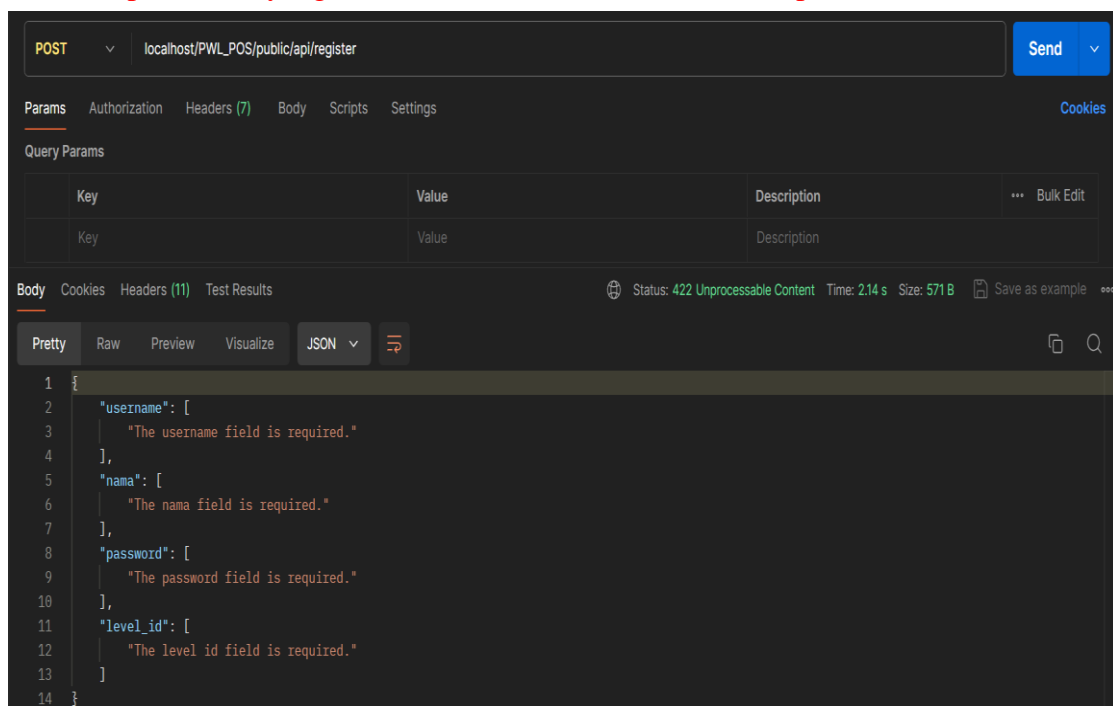



11. Jika sudah, kita akan melakukan uji coba REST API melalui aplikasi Postman. Buka aplikasi Postman, isi URL localhost/PWL_POS/public/api/register serta method POST. Klik Send.



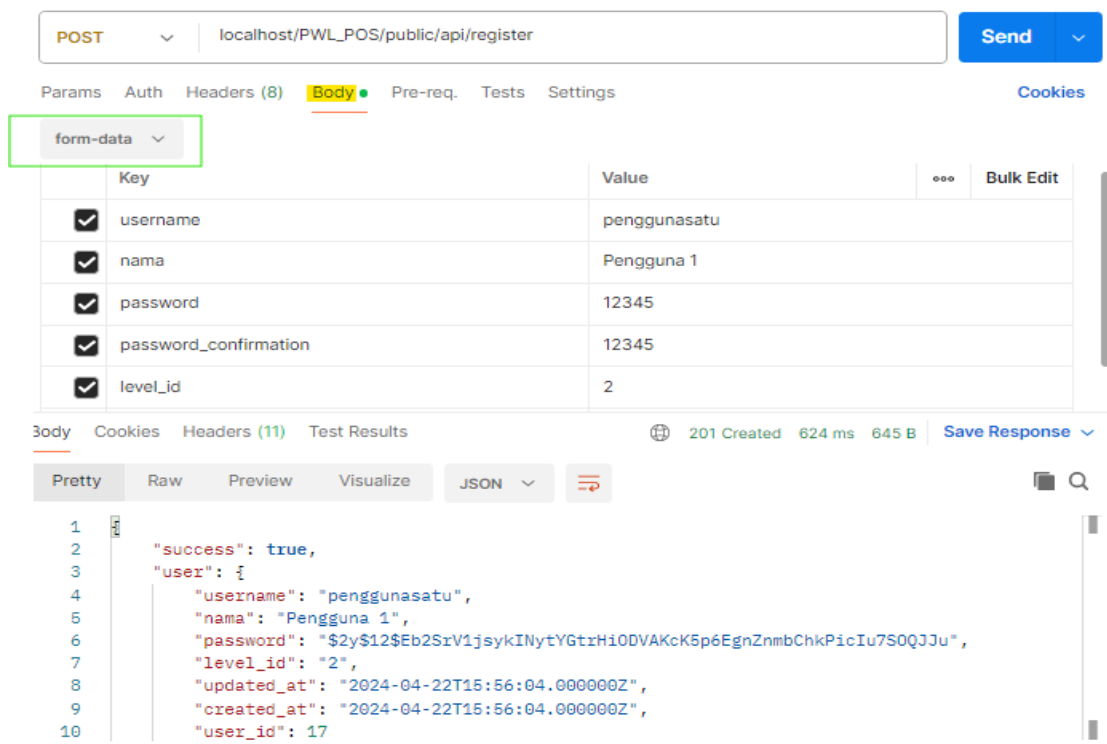
Jika berhasil akan muncul error validasi seperti gambar di atas.

Lakukan percobaan yang sama dan berikan screenshot hasil percobaan Anda.



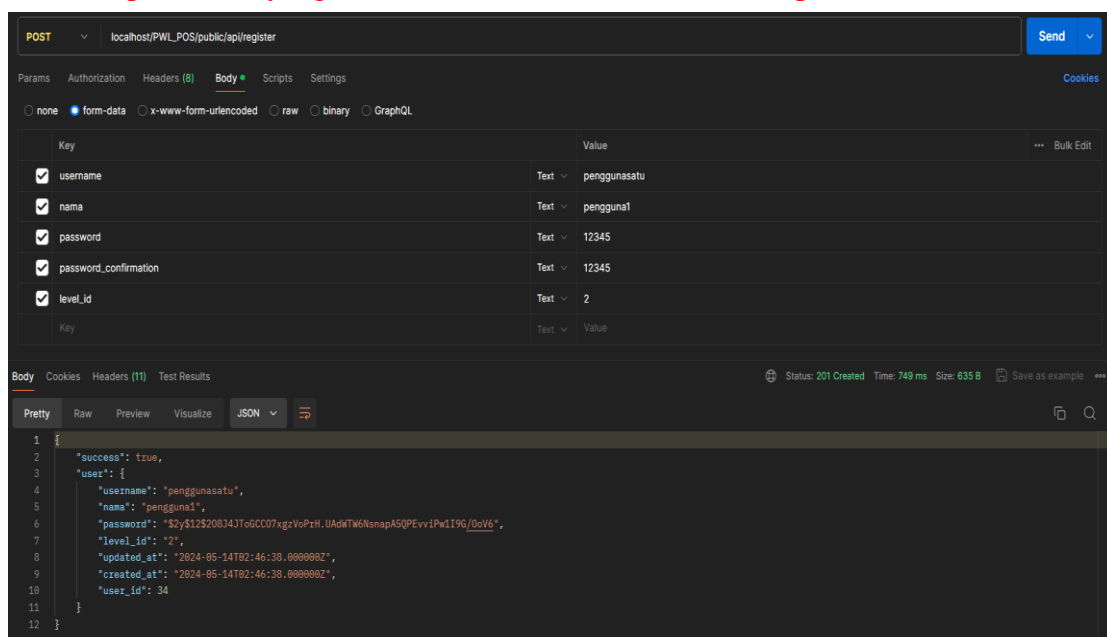


12. Sekarang kita coba masukkan data. Klik tab Body dan pilih form-data. Isikan key sesuai dengan kolom data, serta isikan data registrasi menggunakan nilai yang Anda inginkan.



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

Lakukan percobaan yang sama dan berikan screenshot hasil percobaan Anda.



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`

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

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

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

```
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()->guard('api')->user(),  
39             'token' => $token  
40         ], 200);  
41     }  
42 }
```



```
<?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 fail
        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 salah',
            ], 401);
        }

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

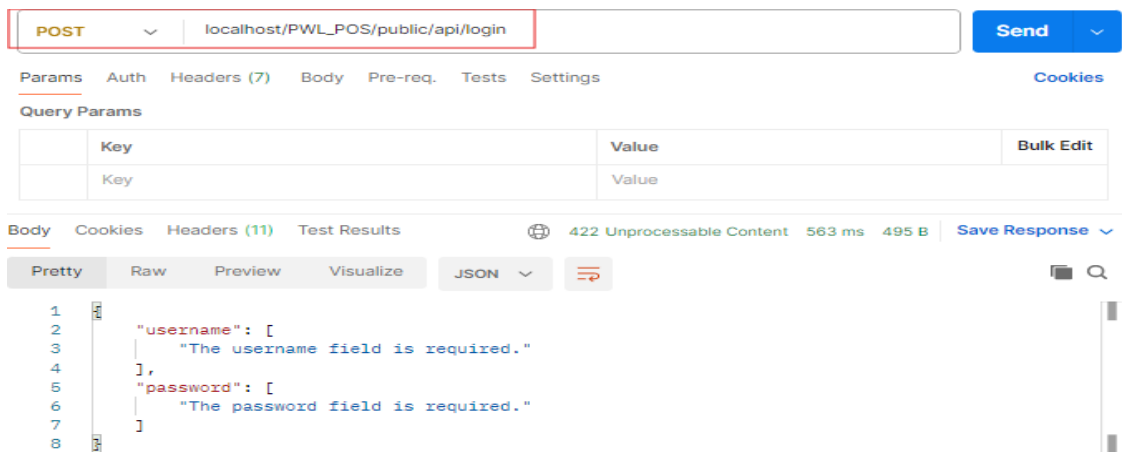


3. Berikutnya tambahkan route baru pada file api.php yaitu /login dan /user.

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

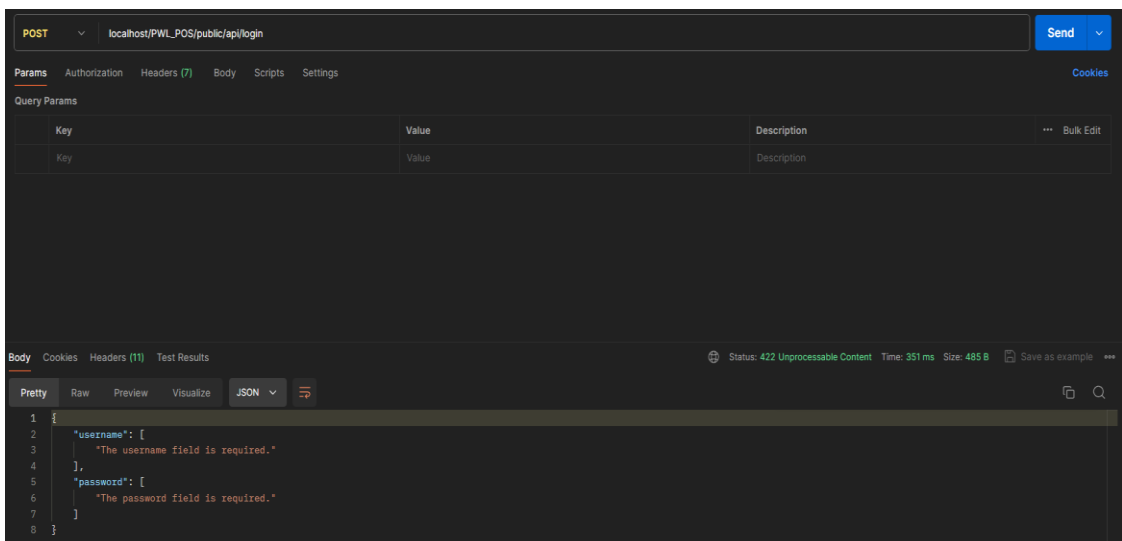
```
Route::post('/register', App\Http\Controllers\Api\RegisterController::class)->name('register');  
Route::post('/login', App\Http\Controllers\Api\LoginController::class)->name('login');  
Route::middleware('auth:api')->get('/user', function (Request $request) {  
    return $request->user();  
});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

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.

The screenshot shows a REST client interface with a POST request to `localhost/PWL_POS/public/api/login`. The request body is set to `form-data` with the following fields:

Key	Value
username	penggunasatu
password	12345

The response status is 200 OK. The response body is a JSON object:

```
1 {
2   "success": true,
3   "user": {
4     "user_id": 17,
5     "level_id": 2,
6     "username": "penggunasatu",
7     "nama": "Pengguna 1",
8     "password": "$2y$12$Eb2SrV1jsykINyTYGtrHi0DVAKcK5p6EgnZnmbChkPicIu7S0QJJU",
9     "created_at": "2024-04-22T15:56:04.000000Z",
10    "updated_at": "2024-04-22T15:56:04.000000Z"
11  },
12  "token": "eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJpc3MiOiJodHRwOi8vbG99YXRob3N0L1BXTF9QT1MtbnVpbi9wdWJsaWVYXSpL2xvZ2luIiwiaWF0Ij0i"
13 }
```

Lakukan percobaan yang sama dan berikan screenshoot hasil percobaan Anda.

The screenshot shows a REST client interface with a POST request to `localhost/PWL_POS/public/api/login`. The request body is set to `form-data` with the following fields:

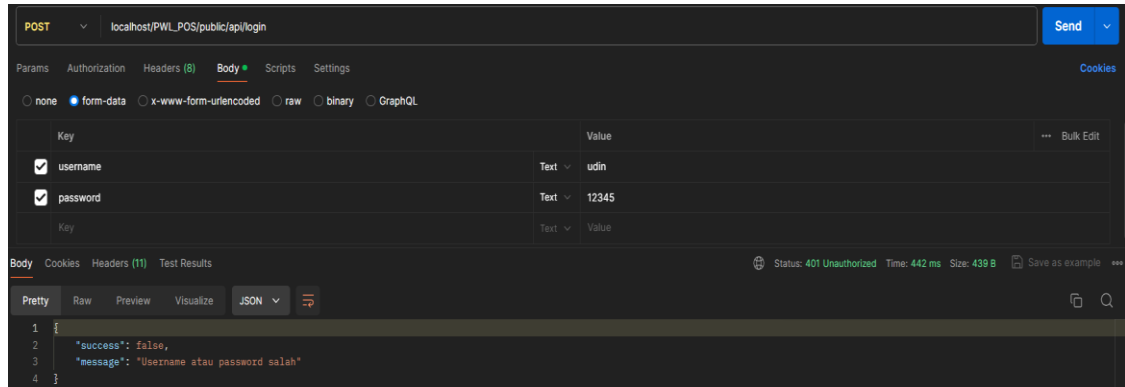
Key	Value
username	penggunasatu
password	12345

The response status is 200 OK. The response body is a JSON object:

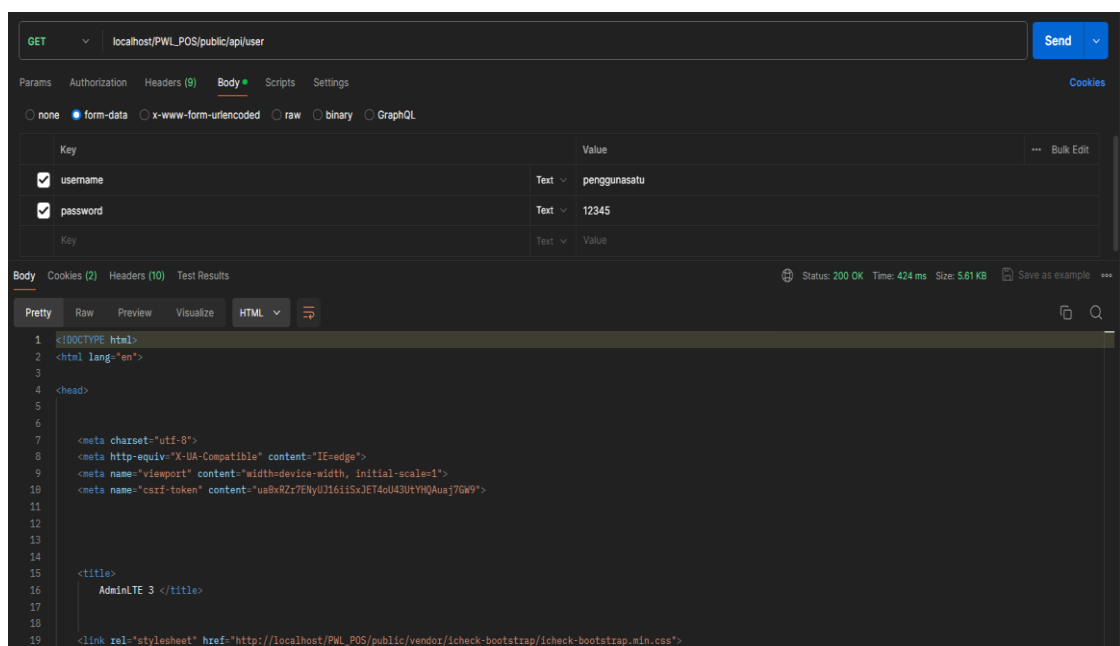
```
1 {
2   "success": true,
3   "user": {
4     "user_id": 35,
5     "level_id": 2,
6     "username": "penggunasatu",
7     "nama": "pengguna1",
8     "password": "$2y$12$PPIyK325WeAGITtASXKI.Az8azCCKBm4JcL6gV0rrv1.Aq2RQda",
9     "created_at": "2024-05-14T02:48:44.888888Z",
10    "updated_at": "2024-05-14T02:48:44.888888Z"
11  },
12  "token": "eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJpc3MiOiJodHRwOi8vbG99YXRob3N0L1BXTF9QT1MtbnVpbi9wdWJsaWVYXSpL2xvZ2luIiwiaWF0Ij0i"
13 }
```



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



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.



Dapat disimpulkan method get dengan url localhost/PWL_POS/public/api/user dapat dijalankan tanpa adanya kendala.

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
```

```
JWT_SECRET=YP0K6AiJWtrvjIAk1VEihdTbLFDGCq30JOCrcmQxwJxbqtpGAuDjCtOLqkFyhDAj  
JWT_SHOW_BLACKLIST_EXCEPTION=true
```

2. Buat Controller baru dengan nama LogoutController.

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

```
PS C:\laragon\www\FWL_POS> php artisan make:controller Api/LogoutController
```

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

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

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



```
<?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());

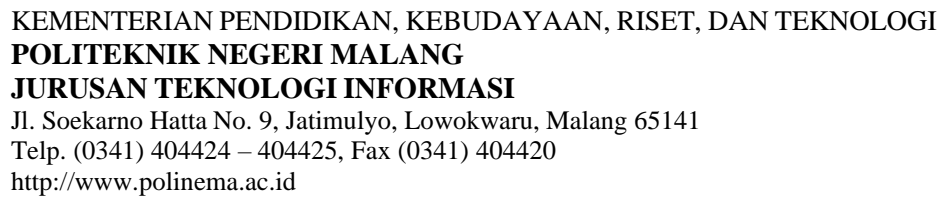
        if ($removeToken){
            // Return response JSON
            return response()->json([
                'success' => true,
                'message' => 'Logout berhasil',
            ]);
        }
    }
}
```

4. Lalu kita tambahkan routes pada api.php

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

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');
```

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.



-
- The screenshot shows the Swagger UI interface for a REST client. The top bar indicates a POST request to the endpoint `localhost/PWL_POS/public/api/logout`. The 'Authorization' tab is selected, showing the 'Type' dropdown set to 'Bearer Token' and the 'Token' field containing a Bearer token. The 'Body' tab is also visible, showing a JSON response with 'success: true' and a message.

The screenshot shows the Postman interface for a POST request to `localhost/PWL_POS/public/apl/logout`. The 'Headers' tab is selected, showing an 'Authorization' header with a Bearer Token. The token value is displayed in a text box, and a tooltip explains that these parameters hold sensitive data and recommends using variables. The 'Body' tab is also visible, showing a JSON response with 'success' and 'message' fields.

- Hal. 18 / 14



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

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

```
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();  
    }  
  
    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',  
        ]);  
    }  
}
```



```
<?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();
    }

    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.

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

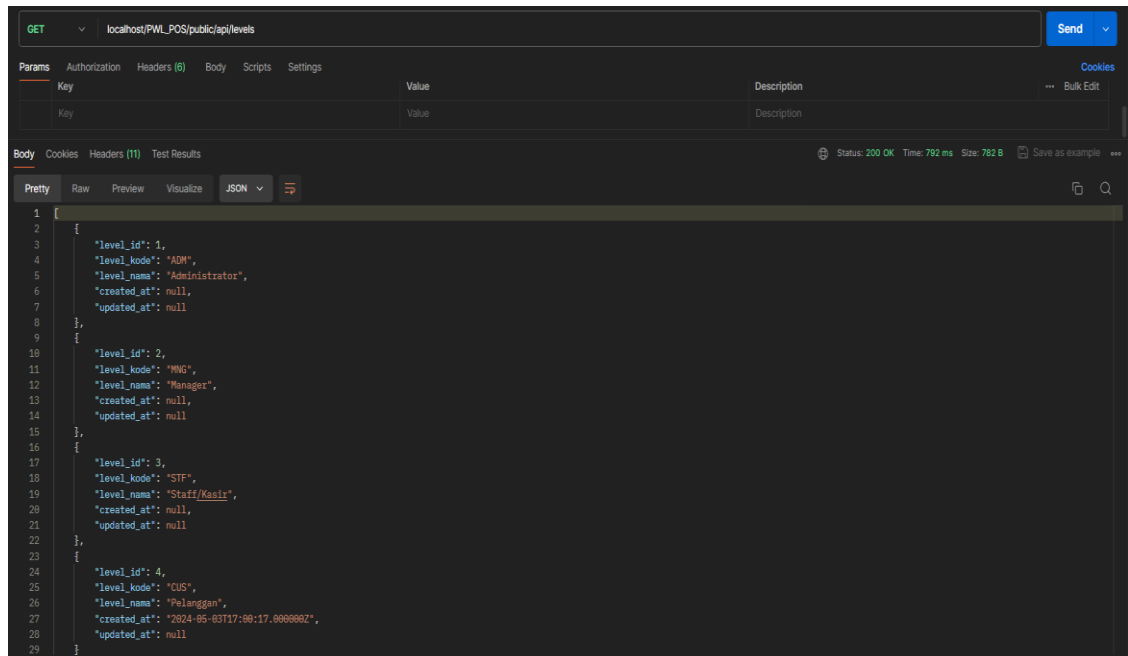
Route::get('levels', [LevelController::class, 'index']);
Route::post('levels', [LevelController::class, 'store']);
Route::get('levels/{level}', [LevelController::class, 'show']);
Route::put('levels/{level}', [LevelController::class, 'update']);
Route::delete('levels/{level}', [LevelController::class,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

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']);
```

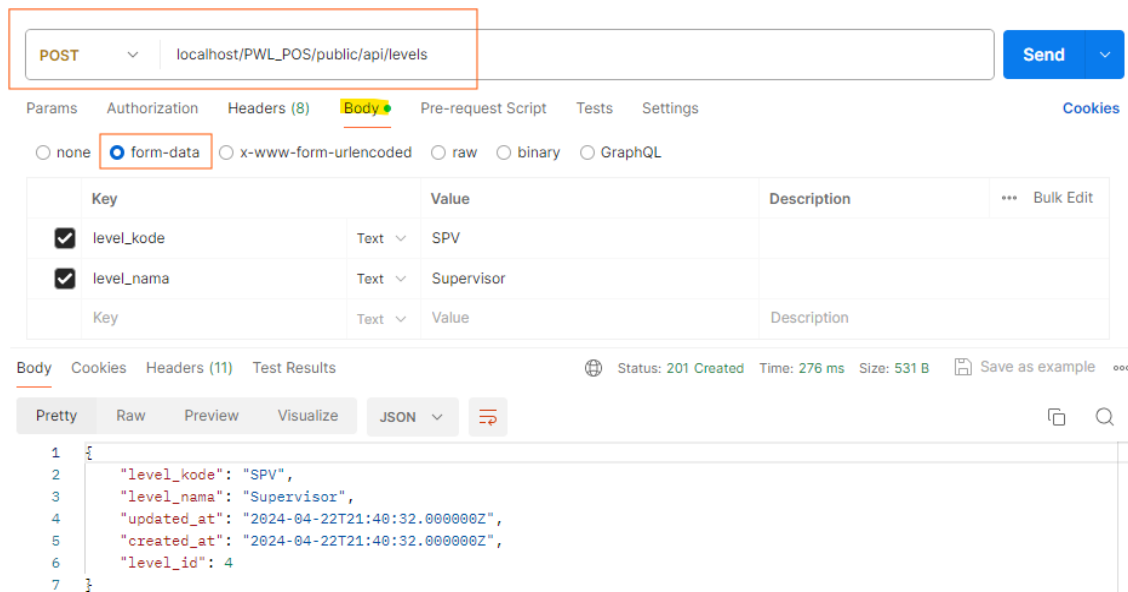


4. Jika sudah. Lakukan uji coba API mulai dari fungsi untuk menampilkan data. Gunakan URL: `localhost/PWL_POS-main/public/api/levels` dan method GET. **Jelaskan dan berikan screenshot hasil percobaan Anda.**



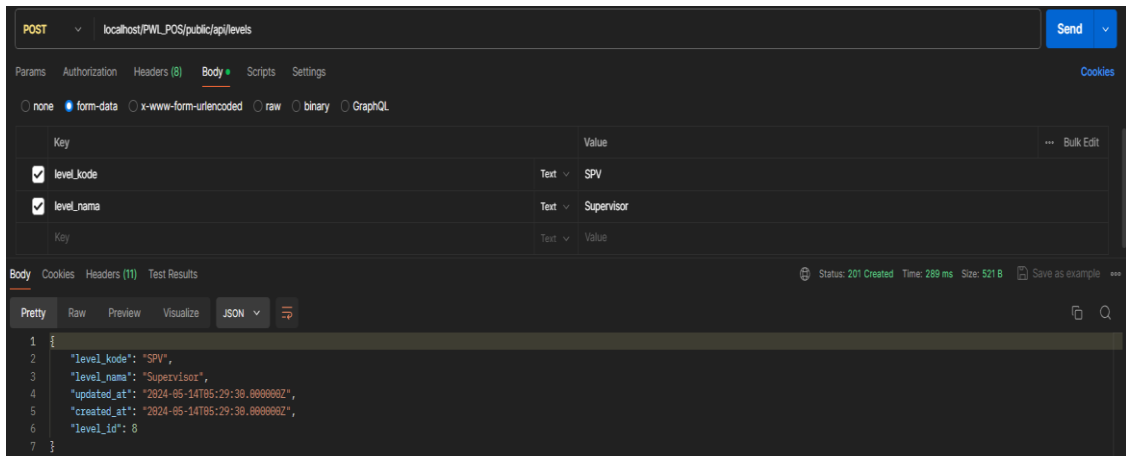
Pada bagian ini, URL dan method berfungsi untuk menampilkan data pada table levels dalam bentuk database.

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



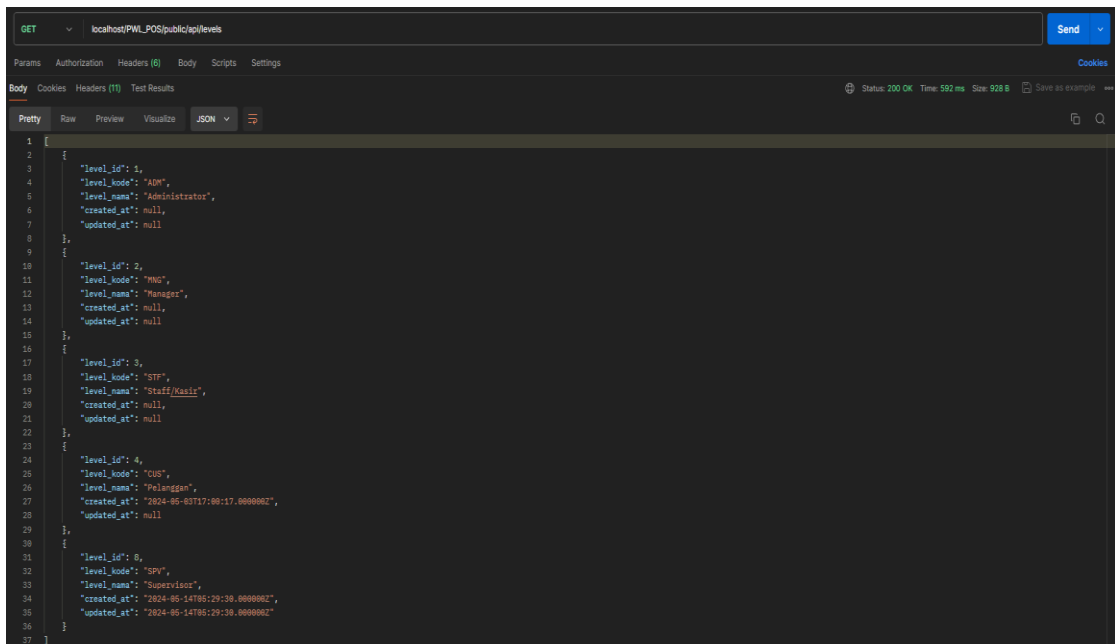


Jelaskan dan berikan screenshoot hasil percobaan Anda.



Pada bagian ini, URL dan method berfungsi untuk memasukan sebuah nilai yaitu level_kode dan nama_kode ke dalam data levels, yang dimana nilainya kita tuliskan di tabel Body -> form-data.

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



Untuk menampilkan detail, kita cara seperti langkah ke-4.



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.

PUT localhost/PWL_POS-main/public/api/levels/4?level_kode=SPR

Params Authorization Headers (7) Body Pre-request Script Tests Settings Cookies

Query Params

Key	Value	Description	Bulk Edit
level_kode	SPR		
Key	Value	Description	

body Cookies Headers (11) Test Results Status: 200 OK Time: 266 ms Size: 528 B Save as example

Pretty Raw Preview Visualize JSON

```
[{"level_id": 4, "level_kode": "SPR", "level_nama": "Supervisor", "created_at": "2024-04-22T21:40:32.000000Z", "updated_at": "2024-04-22T21:48:19.000000Z"}]
```

Jelaskan dan berikan screenshoot hasil percobaan Anda.

PUT localhost/PWL_POS/public/api/levels/8?level_kode=SPR

Params Authorization Headers (7) Body Scripts Settings Cookies

Query Params

Key	Value	Description	Bulk Edit
level_kode	SPR		
Key	Value	Description	

Body Cookies Headers (11) Test Results 200 OK 865 ms 518 B Save as example

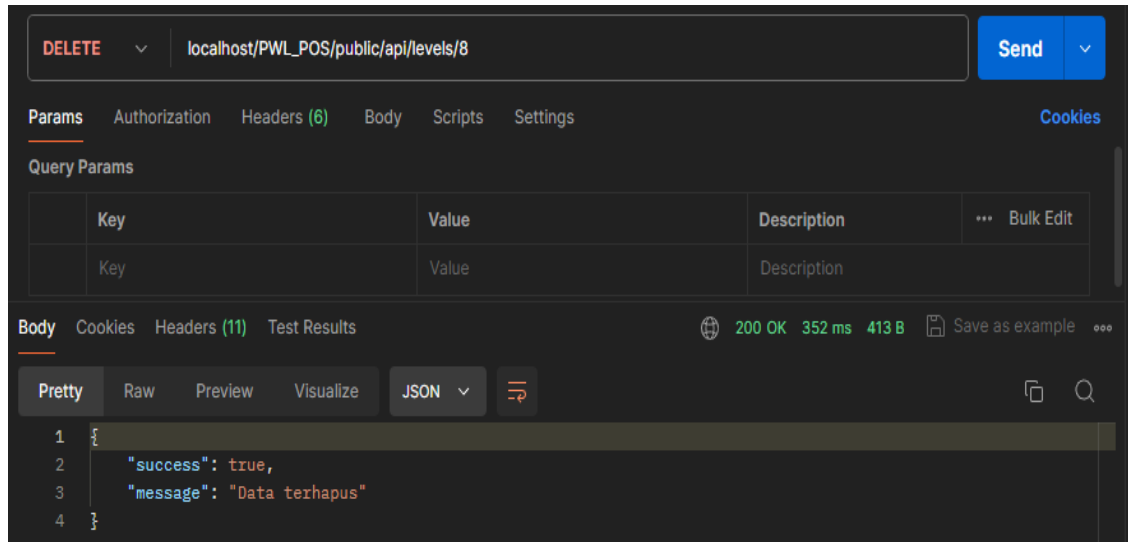
Pretty Raw Preview Visualize JSON

```
[{"level_id": 8, "level_kode": "SPR", "level_nama": "Supervisor", "created_at": "2024-05-14T05:29:30.000000Z", "updated_at": "2024-05-14T05:34:10.000000Z"}]
```

URL dan method diatas berfungsi untuk mengubah sebuah data dengan parameter level_kode=SPR, yang mengisyaratkan perubahan data level dengan ID 8 pada level kodenya menjadi SPR



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



URL dan method diatas berfungsi untuk menghapus sebuah data dengan parameter ID 8, yang mengisyaratkan penghapusan data level pada database.

9. Lakukan commit perubahan file pada Github.



TUGAS

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

1. m_user

- GET

GET localhost/PWL_POS/public/api/users

Status: 200 OK Time: 558 ms Size: 905 B

```
1 {
2   {
3     "user_id": 1,
4     "level_id": 1,
5     "username": "admin",
6     "nama": "Administrator",
7     "password": "$2y$12$J0/wVnzCezHEMIwclDg.teHt8Hr1h7Zisx.n5UAV6NdqQ8R1hIRE0",
8     "created_at": null,
9     "updated_at": null
10  },
11  {
12    "user_id": 2,
13    "level_id": 2,
14    "username": "manager",
15    "nama": "Manager",
16    "password": "$2y$12$Pt6wn7GN3uIKN1navqzdw.ZajLdv883.uMj3.orZ8pPw87n0o4Bk2",
17    "created_at": null,
18    "updated_at": null
19  },
20  {
21    "user_id": 3,
22    "level_id": 3,
23    "username": "staff",
24    "nama": "Staff/Kasir",
25    "password": "$2y$12$atIRlynx0tEnnyvK2AUH7e8qS25doqybQKraln0QoKhpZRErHw0y.",
26    "created_at": null,
27    "updated_at": null
28  }
29 }
```

- POST

POST localhost/PWL_POS/public/api/users

Status: 201 Created Time: 709 ms Size: 602 B

Body: form-data

Key	Value
level_id	2
username	SatriaCS
nama	Satria
password	54321

```
1 {
2   "level_id": "2",
3   "username": "SatriaCS",
4   "nama": "Satria",
5   "password": "$2y$12$ChsuqCBKF93WSpR.La82S.u2qWfuM4vAcPJAuPw85V82vSEC53oT.",
6   "updated_at": "2024-05-14T12:14:39.888888Z",
7   "created_at": "2024-05-14T12:14:39.888888Z",
8   "user_id": 4
9 }
```



- PUT

PUT localhost/PWL_POS/public/api/users/4?nama=Junaldi

Params Authorization Headers (7) Body Scripts Settings Cookies

Query Params

Key	Value	Description
nama	Junaldi	

Body Cookies Headers (11) Test Results

Status: 200 OK Time: 357 ms Size: 596 B Save as example

Pretty Raw Preview Visualize JSON

```
1 {
2   "user_id": 4,
3   "level_id": 2,
4   "username": "SatriaCS",
5   "nama": "Junaldi",
6   "password": "$2y$12$ChsuqCBKF93wS3R.La82S.u2oWfuM4vAcpJAuPw85V82vSEC53oT.",
7   "created_at": "2024-05-14T12:14:39.608866Z",
8   "updated_at": "2024-05-14T12:15:52.608866Z"
9 }
```

- DELETE

DELETE localhost/PWL_POS/public/api/users/4

Params Authorization Headers (6) Body Scripts Settings Cookies

Query Params

Key	Value	Description
Key	Value	Description

Body Cookies Headers (11) Test Results

Status: 200 OK Time: 376 ms Size: 413 B Save as example

Pretty Raw Preview Visualize JSON

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

2. m_kategori

- GET

GET localhost/PWL_POS/public/api/kategori

Params Authorization Headers (6) Body Scripts Settings Cookies

Query Params

Key	Value	Description
Key	Value	Description

Body Cookies Headers (11) Test Results

Status: 200 OK Time: 430 ms Size: 633 B Save as example

Pretty Raw Preview Visualize JSON

```
1 [
2   {
3     "kategori_id": 1,
4     "kategori_kode": "CML",
5     "kategori_nama": "Cemilan",
6     "created_at": "2024-04-05T21:56:24.809080Z",
7     "updated_at": null
8   },
9   {
10    "kategori_id": 2,
11    "kategori_kode": "MNR",
12    "kategori_nama": "Minuman Ringan",
13    "created_at": "2024-04-05T21:57:13.809080Z",
14    "updated_at": null
15  }
16 ]
```



- POST

POST localhost/PWL_POS/public/api/kategori

Params Authorization Headers (8) Body Scripts Settings Cookies

☐ none ☒ form-data ☐ x-www-form-urlencoded ☐ raw ☐ binary ☐ GraphQL

Key	Value
kategori_kode	SNK
kategori_nama	Snack

Body Cookies Headers (11) Test Results

Status: 201 Created Time: 459 ms Size: 525 B Save as example

Pretty Raw Preview Visualize JSON

```
1 {
2   "kategori_kode": "SNK",
3   "kategori_nama": "Snack",
4   "updated_at": "2024-05-14T13:33:51.888888Z",
5   "created_at": "2024-05-14T13:33:51.888888Z",
6   "kategori_id": 3
7 }
```

- PUT

PUT localhost/PWL_POS/public/api/kategori/3?kategori_kode=PJK

Params Authorization Headers (7) Body Scripts Settings Cookies

Query Params

Key	Value	Description
kategori_kode	PJK	

Body Cookies Headers (11) Test Results

Status: 200 OK Time: 452 ms Size: 522 B Save as example

Pretty Raw Preview Visualize JSON

```
1 {
2   "kategori_id": 3,
3   "kategori_kode": "PJK",
4   "kategori_nama": "Snack",
5   "created_at": "2024-05-14T13:33:51.888888Z",
6   "updated_at": "2024-05-14T13:34:45.888888Z"
7 }
```

- DELETE

DELETE localhost/PWL_POS/public/api/kategori/3

Params Authorization Headers (6) Body Scripts Settings Cookies

Query Params

Key	Value	Description
-----	-------	-------------

Body Cookies Headers (11) Test Results

Status: 200 OK Time: 402 ms Size: 413 B Save as example

Pretty Raw Preview Visualize JSON

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



3. m_barang

- GET

GET localhost/PWL_POS/public/api/barangs

Params Authorization Headers (6) Body Scripts Settings Cookies

Query Params

Key	Value	Description
Key	Value	Description

Body Cookies Headers (11) Test Results Status: 200 OK Time: 539 ms Size: 973 B Save as example

Pretty Raw Preview Visualize JSON

```
1 [
2   {
3     "barang_id": 1,
4     "kategori_id": 1,
5     "barang_kode": "BRG001",
6     "barang_nama": "Taro",
7     "harga_beli": 7000,
8     "harga_jual": 10000,
9     "created_at": null,
10    "updated_at": null
11  },
12  {
13    "barang_id": 2,
14    "kategori_id": 2,
15    "barang_kode": "BRG002",
16    "barang_nama": "Fanta",
17    "harga_beli": 4000,
18    "harga_jual": 8000,
19    "created_at": null,
20    "updated_at": null
21  },
22  {
23    "barang_id": 3,
24    "kategori_id": 1,
25    "barang_kode": "BRG003",
26    "barang_nama": "Dorito",
27    "harga_beli": 12000,
28    "harga_jual": 20000,
29    "created_at": null,
30    "updated_at": null
31  },
32 ]
```

- POST

POST localhost/PWL_POS/public/api/barangs

Params Authorization Headers (8) Body Scripts Settings Cookies

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

Key	Value
kategori_id	1
barang_kode	BRG005
barang_nama	Chitao
harga_beli	4000
harga_jual	7000

Body Cookies Headers (11) Test Results Status: 201 Created Time: 502 ms Size: 581 B Save as example

Pretty Raw Preview Visualize JSON

```
1 {
2   "kategori_id": "1",
3   "barang_kode": "BRG005",
4   "barang_nama": "Chitao",
5   "harga_beli": "4000",
6   "harga_jual": "7000",
7   "updated_at": "2024-06-14T13:53:22.000000Z",
8   "created_at": "2024-06-14T13:53:22.000000Z",
9   "barang_id": 5
10 }
```



- PUT

PUT localhost/PWL_POS/public/api/barangs/5barang_nama=Gacoan

Params Authorization Headers (7) Body Scripts Settings Cookies

Query Params

Key	Value	Description
barang_nama	Gacoan	

Body Cookies Headers (11) Test Results

Status: 200 OK Time: 552 ms Size: 570 B Save as example

Pretty Raw Preview Visualize JSON

```
1 {
2   "barang_id": 5,
3   "kategori_id": 1,
4   "barang_kode": "BRG005",
5   "barang_nama": "Gacoan",
6   "harga_beli": 4000,
7   "harga_jual": 7000,
8   "created_at": "2024-05-14T13:53:22.000000Z",
9   "updated_at": "2024-05-14T13:57:26.000000Z"
10 }
```

- DELETE

DELETE localhost/PWL_POS/public/api/barangs/5

Params Authorization Headers (6) Body Scripts Settings Cookies

Query Params

Key	Value	Description
Key	Value	Description

Body Cookies Headers (11) Test Results

Status: 200 OK Time: 527 ms Size: 413 B Save as example

Pretty Raw Preview Visualize JSON

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

*** Sekian, dan selamat belajar ***