

# JOBSHEET 10

## RESTFUL API

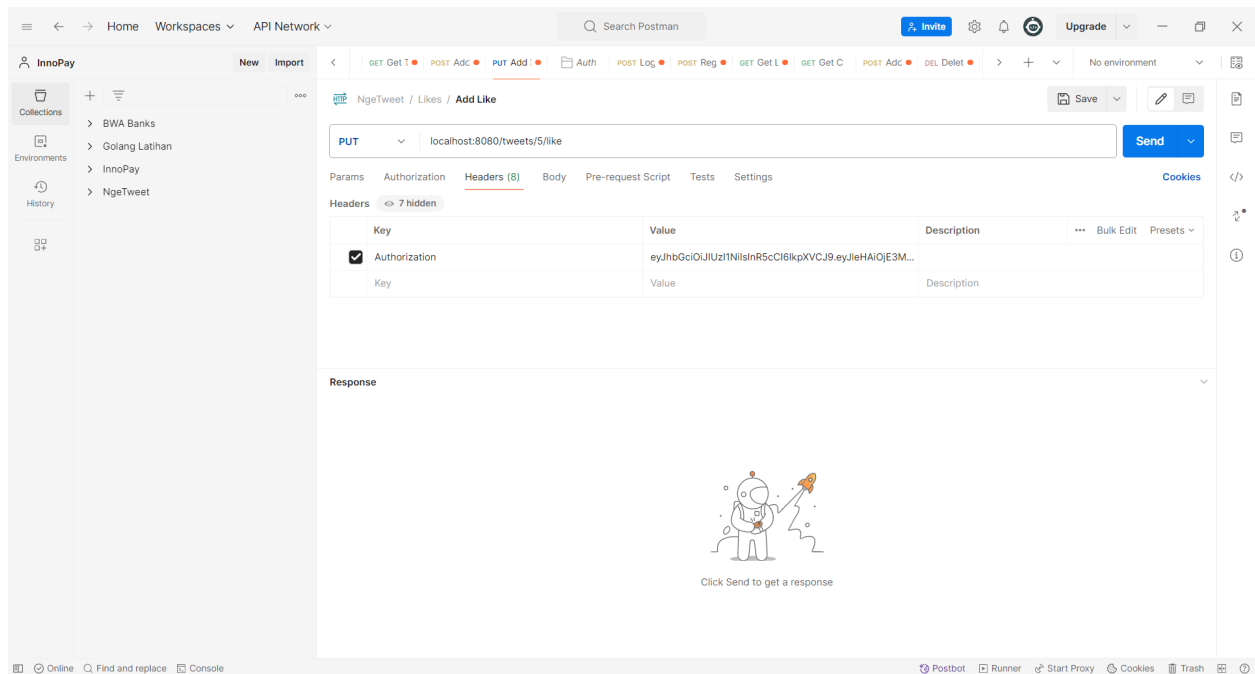
Nama : Raffy Jamil Octavialdy

Kelas : TI - 2H

NIM : 2241720082

### 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 D:\Semester 4\Pemrograman Web Lanjut\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 tymon/jwt-auth (2.1.1)
  - Installing stella-maris/clock (0.1.7): Extracting archive
  - Installing lcobucci/clock (2.3.0): Extracting archive
  - Installing lcobucci/jwt (4.0.4): Extracting archive
  - Installing tymon/jwt-auth (2.1.1): Extracting archive
Generating optimized autoload files
```

3. Setelah berhasil menginstall JWT, lanjutkan dengan publish konfigurasi file dengan perintah berikut: `php artisan vendor:publish --provider="Tymon\JWTAuth\Providers\LaravelServiceProvider"`

```
PS D:\Semester 4\Pemrograman Web Lanjut\PWL_POS> php artisan vendor:publish --provider="Tymon\JWTAuth\Providers\LaravelServiceProvider"

INFO Publishing assets.

Copying file [D:\Semester 4\Pemrograman Web Lanjut\PWL_POS\vendor\tymon\jwt-auth\config\config.php] to [D:\Semester 4\Pemrograman Web Lanjut\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`.

```
PS D:\Semester 4\Pemrograman Web Lanjut\PWL_POS> php artisan jwt:secret
jwt-auth secret [af17tUyZgEy1VmpKgQX55AZ79glSMITxn7nWwjen0h12YODuWmQyQnk80Tuc145] 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

```
class UserModel extends Authenticatable implements JWTSubject
{
    use HasFactory;

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

    public function getJWTCustomClaims()
    {
        return [];
    }
}
```

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.

```
PS D:\Semester 4\Pemrograman Web Lanjut\PWL_POS> php artisan make:controller Api/RegisterController
```

```
INFO Controller [D:\Semester 4\Pemrograman Web Lanjut\PWL_POS\app\Http\Controllers\Api\RegisterController.php] created successfully.
```

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

```
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->error(), 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 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.

The screenshot shows the Postman interface. At the top, a 'New Request' tab is active for 'PWL\_POS'. The request method is set to 'POST' and the URL is 'localhost:8000/api/register'. Below the URL bar, there are tabs for 'Params', 'Authorization', 'Headers (8)', 'Body', 'Pre-request Script', 'Tests', and 'Settings'. The 'Query Params' section is visible, showing a table with columns 'Key', 'Value', and 'Description'. The 'Body' tab is selected, showing a JSON response in 'Pretty' format. The response status is '422 Unprocessable Content' with a time of '116 ms' and size of '507 B'. The JSON response contains error messages for 'username', 'nama', 'password', and 'level\_id' fields.

Key	Value	Description
Key	Value	Description

```
1 {
2   "username": [
3     "The username field is required."
4   ],
5   "nama": [
6     "The nama field is required."
7   ],
8   "password": [
9     "The password field is required."
10  ],
11  "level_id": [
12    "The level id field is required."
13  ]
14 }
```

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.

PWL\_POS / New Request Save Send

POST localhost:8000/api/register

Params Authorization Headers (8) **Body** Pre-request Script Tests Settings Cookies

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

Key	Value	Description	Bulk Edit
<input checked="" type="checkbox"/> username	Text penggunasatu		
<input checked="" type="checkbox"/> nama	Text Pengguna 1		
<input checked="" type="checkbox"/> password	Text 12345		
<input checked="" type="checkbox"/> password_confirmation	Text 12345		
<input checked="" type="checkbox"/> level_id	Text 2		
Key	Text Value	Description	

Body Cookies Headers (10) Test Results Status: 201 Created Time: 528 ms Size: 487 B Save as example

Pretty Raw Preview Visualize JSON Copy Search

```
1  {
2    "success": true,
3    "user": {
4      "username": "penggunasatu",
5      "nama": "Pengguna 1",
6      "password": "$2y$12$zARexM/Icm1TgPCQbJ2hqeDA5RmLIVZkYN49dbJk80F/15tHfean6",
7      "level_id": "2",
8      "user_id": 29
9    }
10 }
```

## 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 D:\Semester 4\Pemrograman Web Lanjut\PWL_POS> php artisan make:controller Api/LoginController
```

```
INFO Controller [D:\Semester 4\Pemrograman Web Lanjut\PWL_POS\app\Http\Controllers\Api>LoginController.php] created successfully.
```

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

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

        // if validations 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,
            'message' => auth()->user(),
            'token' => $token
        ], 200);
    }
}
```

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


The screenshot shows the Postman interface for a POST request to `localhost:8000/api/login`. The request is saved and has a 'Send' button. Below the request bar, the 'Query Params' section is empty. The 'Body' tab is selected, showing a JSON response in 'Pretty' format. The response status is 422 Unprocessable Content, with a time of 1720 ms and a size of 421 B. The response body contains the following JSON:




```
{
  "username": [
    "The username field is required."
  ],
  "password": [
    "The password field is required."
  ]
}
```

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.

[illegible]

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


**PWL\_POS / Login**

 Save
 


**POST**
localhost:8000/api/login




Params
Authorization
Headers (9)
**Body**
Pre-request Script
Tests
Settings

Cookies


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

Key	Value	Description
<input checked="" type="checkbox"/> username	Text penggunasat	
<input checked="" type="checkbox"/> password	Text 12345	
Key	Value	Description

**Body**
Cookies (2)
Headers (10)
Test Results

 Status: 401 Unauthorized
 Time: 225 ms
 Size: 380 B
  Save as example
 

Pretty
Raw
Preview
Visualize
JSON



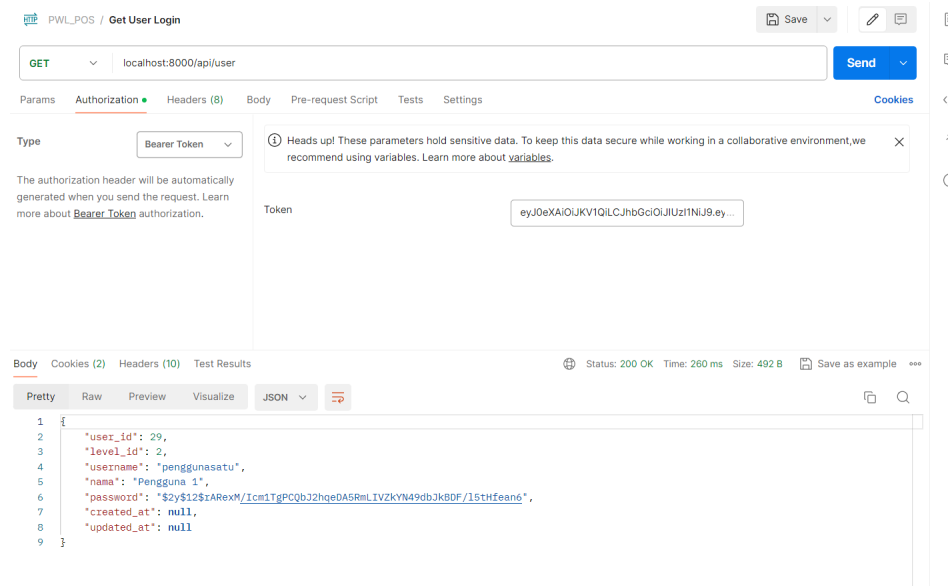
```

1 {
2   "success": false,
3   "message": "Username atau Password Anda salah"
4 }

```



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.



### Praktikum 3 – Membuat RESTful API Logout

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

```
JWT_SHOW_BLACKLIST_EXCEPTION=true
```

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

```
PS D:\Semester 4\Pemrograman Web Lanjut\PWL_POS> php artisan make:controller Api/LogoutController

INFO Controller [D:\Semester 4\Pemrograman Web Lanjut\PWL_POS\app\Http\Controllers\Api\LogoutController.php] created successfully.
```

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

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

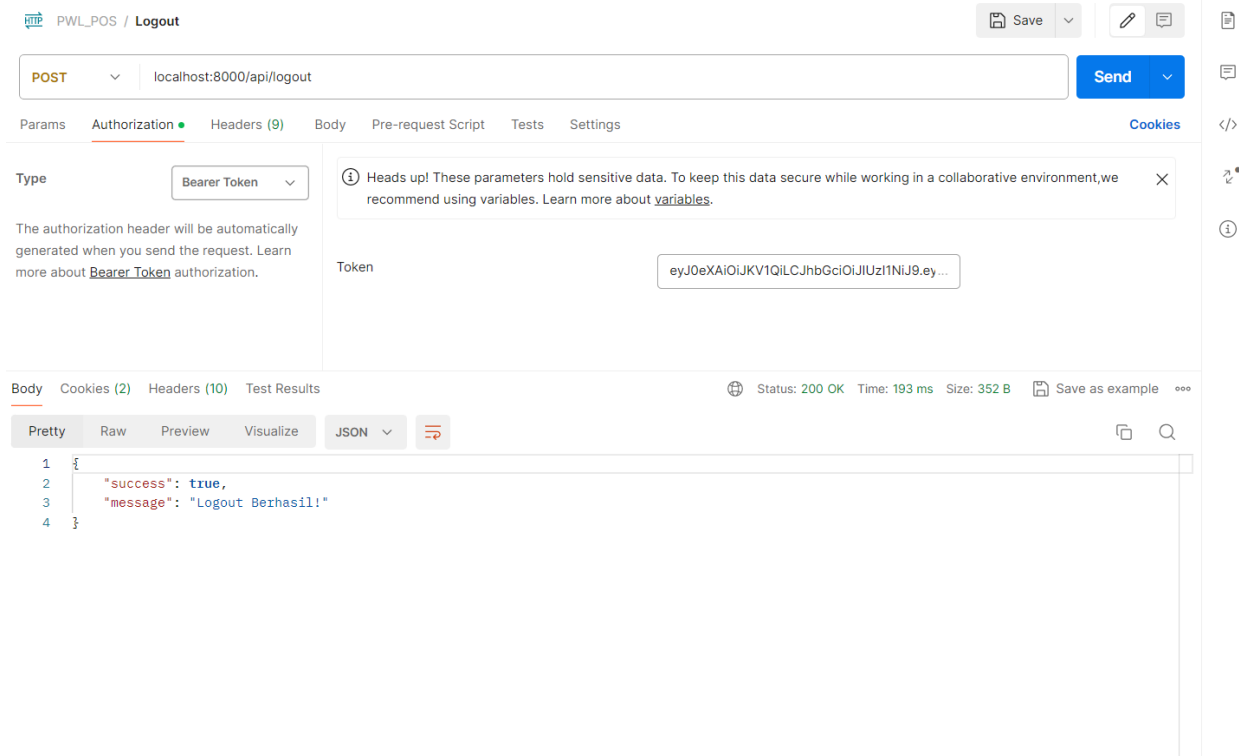
        if ($removeToken) {
            // return response JSON
            return response()->json([
                'success' => true,
                'message' => 'Logout Berhasil!',
            ]);
        }
    }
}
```

4. Lalu kita tambahkan routes pada api.php

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

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.



## 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 D:\Semester 4\Pemrograman Web Lanjut\PWL_POS> php artisan make:controller Api/LevelController  
INFO Controller [D:\Semester 4\Pemrograman Web Lanjut\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.

```
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.

```
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.

REST client interface showing a GET request to `localhost:8000/api/levels`. The response status is 200 OK, with a time of 294 ms and a size of 718 B. The response body is a JSON array of three level objects:

```
1 {
2   {
3     "level_id": 1,
4     "level_kode": "ADM",
5     "level_nama": "Administrator",
6     "created_at": null,
7     "updated_at": null
8   },
9   {
10    "level_id": 2,
11    "level_kode": "MNG",
12    "level_nama": "Manager",
13    "created_at": null,
14    "updated_at": null
15  },
16  {
17    "level_id": 3,
18    "level_kode": "STF",
19    "level_nama": "Staff/Kasir",
20    "created_at": null,
21    "updated_at": null
22  }
23 }
```

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

REST client interface showing a POST request to `localhost:8000/api/levels`. The request body is form-data with the following fields:

Key	Value	Description
<input checked="" type="checkbox"/> level_kode	Text: SPV	
<input checked="" type="checkbox"/> level_nama	Text: Supervisor	
Key	Text: Value	Description

The response status is 201 Created, with a time of 273 ms and a size of 457 B. The response body is a JSON object:

```
1 {
2   "level_kode": "SPV",
3   "level_nama": "Supervisor",
4   "updated_at": "2024-04-23T16:38:28.000000Z",
5   "created_at": "2024-04-23T16:38:28.000000Z",
6   "level_id": 7
7 }
```

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

The screenshot shows a REST client interface with the following details:

- URL:** `localhost:8000/api/levels/7`
- Method:** GET
- Status:** 200 OK
- Time:** 190 ms
- Size:** 454 B

The response body is displayed in JSON format:

```
[
  {
    "level_id": 7,
    "level_kode": "SPV",
    "level_nama": "Supervisor",
    "created_at": "2024-04-23T16:38:28.000000Z",
    "updated_at": "2024-04-23T16:38:28.000000Z"
  }
]
```

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.

The screenshot shows a REST client interface with the following details:

- URL:** `localhost:8000/api/levels/7?level_kode=SPR`
- Method:** PUT
- Status:** 200 OK
- Time:** 177 ms
- Size:** 454 B

The response body is displayed in JSON format:

```
[
  {
    "level_id": 7,
    "level_kode": "SPR",
    "level_nama": "Supervisor",
    "created_at": "2024-04-23T16:38:28.000000Z",
    "updated_at": "2024-04-23T16:43:25.000000Z"
  }
]
```

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

The screenshot displays a REST client interface with the following components:

- URL Bar:** Shows the method **DELETE** and the endpoint `localhost:8000/api/levels/7`. A **Send** button is located to the right.
- Query Params:** A table with columns **Key**, **Value**, and **Description**. It contains one entry with **Key** and **Value** both set to `Key`.
- Response Section:** Includes tabs for **Body**, **Cookies (2)**, **Headers (10)**, and **Test Results**. The **Body** tab is active, showing a JSON response in **Pretty** format.
- Status Bar:** Displays **Status: 200 OK**, **Time: 211 ms**, and **Size: 349 B**. It also includes a **Save as example** button.

The JSON response in the body is as follows:

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

## TUGAS

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

### - m\_user

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

    public function store(Request $request)
    {
        $user = UserModel::create([
            'username' => $request->username,
            'nama' => $request->nama,
            'password' => bcrypt($request->password),
            'level_id' => $request->level_id,
        ]);
        return response()->json($user, 201);
    }

    public function show($user)
    {
        return UserModel::where('user_id', $user)->first();
    }

    public function update(Request $request)
    {
        try {
            $user = UserModel::find(auth()->user()->user_id);

            $data = $request->only('level_id', 'username', 'nama', 'password');

            if ($request->username != $user->username) {
                $isExistUsername = UserModel::where('username', $request->username)->exists();
                if ($isExistUsername) {
                    return response(['message' => 'Username already taken'], 409);
                }
            }

            if ($request->password) {
                $data['password'] = bcrypt($request->password);
            }

            $user->update($data);
            return $user;
        } catch (\Throwable $th) {
            return response()->json(['message' => $th->getMessage()], 500);
        }
    }
}
```

## - m\_kategori dan m\_barang

```
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 $kategori)
    {
        return KategoriModel::find($kategori);
    }

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

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

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

```
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($barang)
    {
        return BarangModel::where('barang_id', $barang)->first();
    }

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

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

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



HomeWorkspacesAPI Network

Search Postman

Invite

Upgrade

InnoPay

NewImport

Collections

PWLPoS

Level

User

Get All User

Create User

Show User

Edit User

Delete User

Kategori

Get All Kategori

Create Kategori

Show Kategori

Edit Kategori

Delete Kategori

Barang

Get All Barang

Create Barang

Show Barang

Edit Barang

Delete Barang

Register

Login

Get User Login

Logout

Environments

History

PWLPoS / User / Get All User

GETlocalhost:8000/api/users

Send

Params

Authorization

Headers (6)

Body

Pre-request Script

Tests

Settings

Cookies

Query Params

Key	Value	Description
Key	Value	Description

Body

Cookies

Headers (10)

Test Results

Status: 200 OKTime: 1965 msSize: 2.16 KB

Save as example

Pretty

Raw

Preview

Visualize

JSON

Raw

```
1 {
2   {
3     "user_id": 1,
4     "level_id": 1,
5     "username": "admin",
6     "nama": "Administrator",
7     "password": "$2y$12$AQZPZz2idefqvXlGFr5HUux3oN.jallwMMXpwTLhhjov5vq/dFPT0",
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$12$TnMAZzjsL.RJuWnw/8TdCOXjb6JhQtILufw0BTwoAeUYKqD9e165",
17    "created_at": null,
18    "updated_at": null
19  },
20 }
```

Online

Find and replace

Console

Postbot

Runner

Start Proxy

Cookies

Trash