Jurusan Teknologi Informasi Politeknik Negeri Malang
JOBSHEET 10 RESTFUL API
Mata Kuliah Pemrograman Web Lanjut (PWL)

Nama	:	Candra Ahmad Dani
Nim	:	2341720187
Kelas	:	TI-2A

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

```php
<?php

namespace App\Models;

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

120 references | 0 implementations
class UserModel extends Authenticatable implements JWTSubject
{
    0 references | 0 overrides
    public function getJWTIdentifier(): mixed
    {
        return $this->getKey();
    }
    0 references | 0 overrides
    public function getJWTCustomClaims(): array
    {
        return [];
    }
```

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

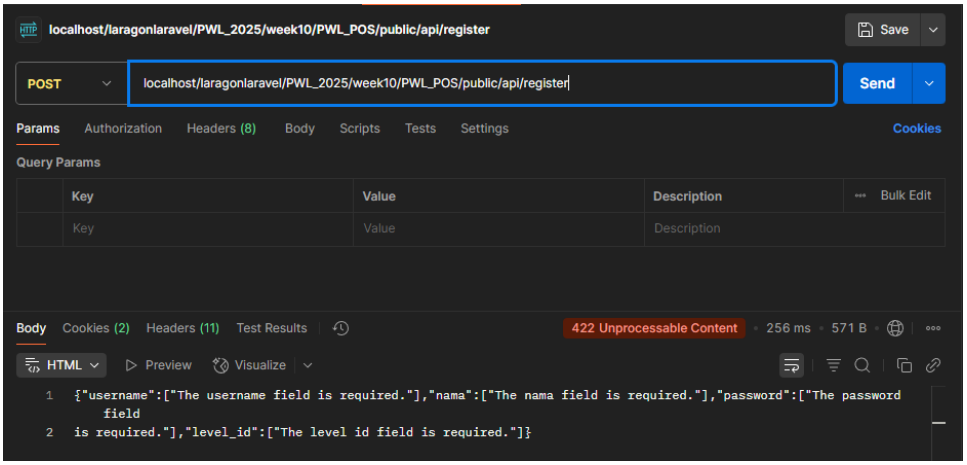
```php
<?php

namespace App\Http\Controllers\Api;

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

class RegisterController extends Controller
{
    public function __invoke(Request $request)
    {
        //set validator
        $validator = Validator::make($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);
        }
```

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

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

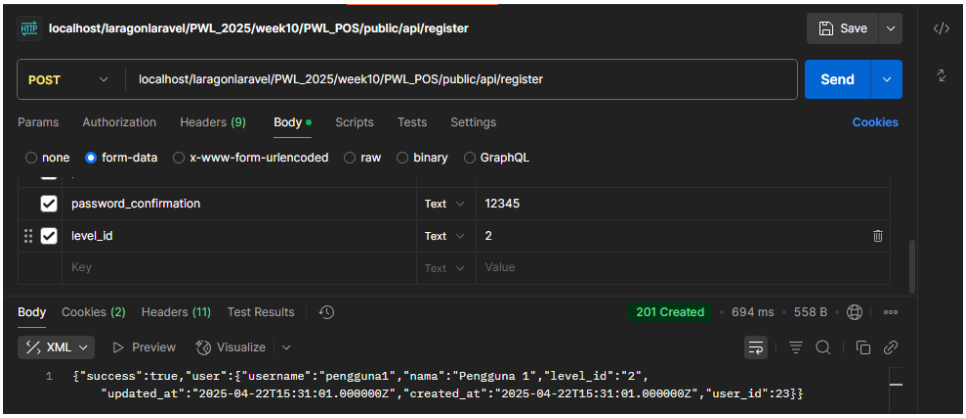
```php
Route::post('/register', App\Http\Controllers\Api\RegisterController::class)->name('register');
```

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.

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

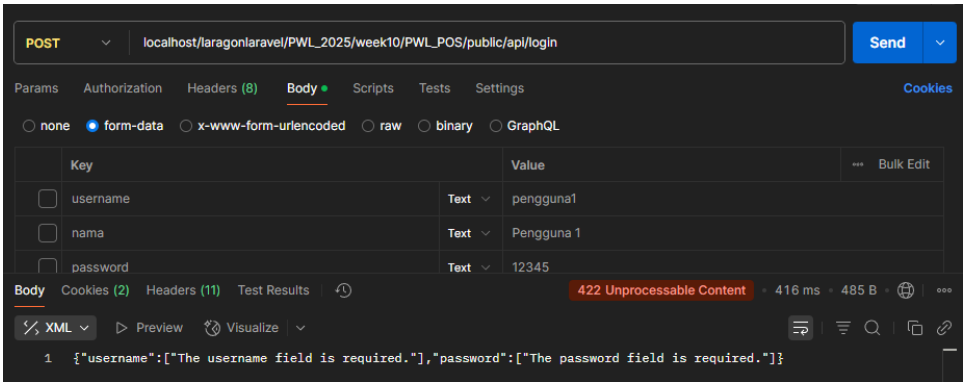
```php
<?php

namespace App\Http\Controllers\Api;

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

class LoginController extends Controller
```

```php
{
    public function __invoke(Request $request)
    {
        //set validator
        $validator = Validator::make($request->all(), [
            'username' => 'required',
            'password' => 'required'
        ]);
        //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,
            'user' => auth()->guard('api')->user(),
            'token' => $token
        ], 200);
    }
}
```

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

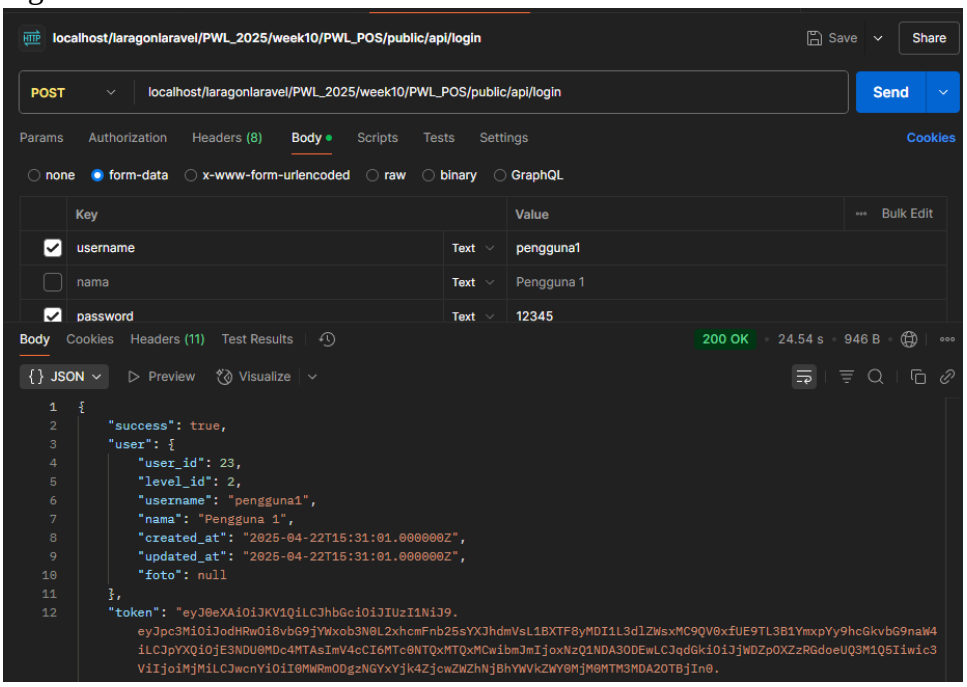
```php
Route::post('/register', App\Http\Controllers\Api\RegisterController::class)->name('register');
Route::post('/login', App\Http\Controllers\Api\LoginController::class)->name('login');
Route::middleware('auth:api')->get('/user', function (Request $request): mixed {
    return $request->user();
});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

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.



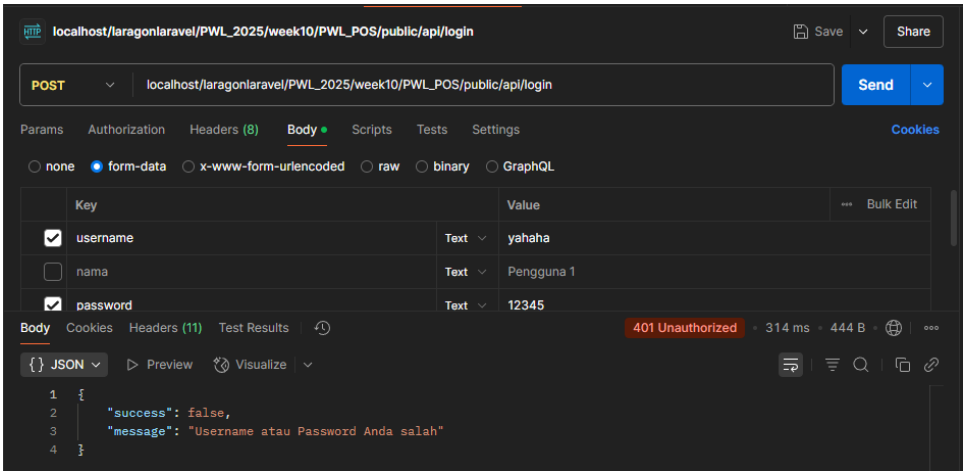
Jika berhasil akan muncul error validasi seperti gambar di atas. Lakukan percobaan yang sama dan berikan screenshoot hasil percobaan Anda.

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

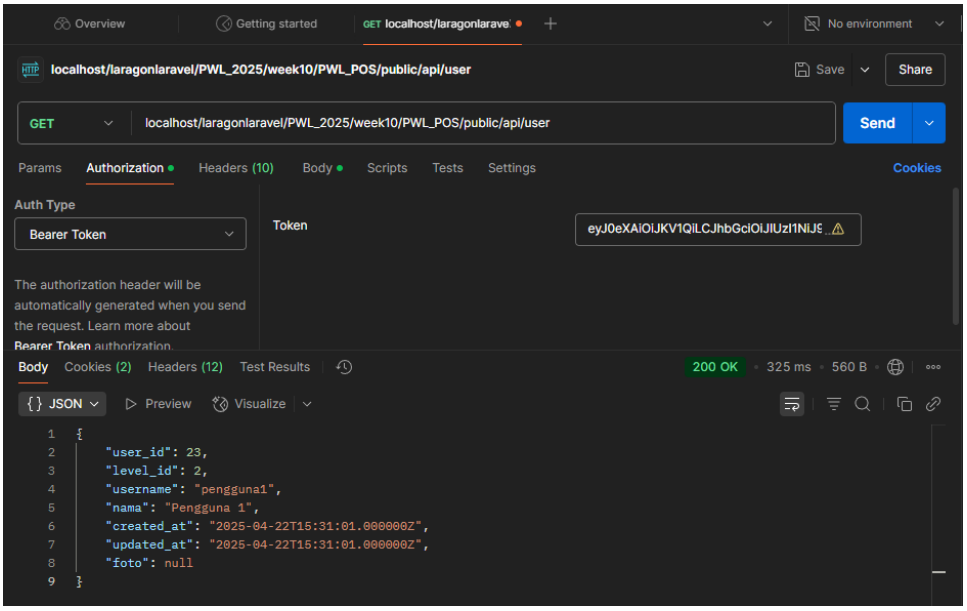


Lakukan percobaan yang sama dan berikan screenshoot hasil percobaan Anda.

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



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 dengan memasukkan token yang diperoleh saat login ke dalam header Authorization sebagai Bearer Token, user dapat mengakses informasi login mereka.

8. Lakukan commit perubahan file pada Github.

Praktikum 3 – Membuat RESTful API Logout

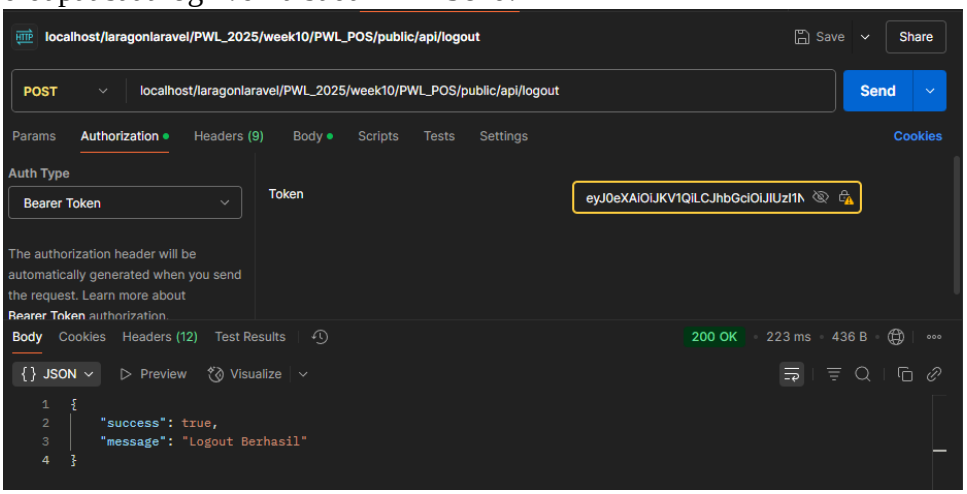
1. 1. Tambahkan kode berikut pada file .env
   JWT_SHOW_BLACKLIST_EXCEPTION=true
2. Buat Controller baru dengan nama LogoutController.
   php artisan make:controller Api/LogoutController
3. Buka file tersebut dan ubah kode menjadi seperti berikut.

```php
<?php

namespace App\Http\Controllers\Api;

use Illuminate\Http\Request;
use Tymon\JWTAuth\Facades\JWTAuth;
use App\Http\Controllers\Controller;

1 reference | 0 implementations
class LogoutController extends Controller
{
    0 references | 0 overrides
    public function __invoke(Request $request): JsonResponse|mixed
    {
        $removeToken = JWTAuth::invalidate(JWTAuth::getToken());
        if ($removeToken) {
            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.

Praktikum 4 – Implementasi CRUD dalam RESTful API

1. Pertama, buat controller untuk mengolah API pada data level.
   php artisan make:controller Api/LevelController
2. Setelah berhasil, buka file tersebut dan tuliskan kode seperti berikut yang berisi fungsi CRUDnya.

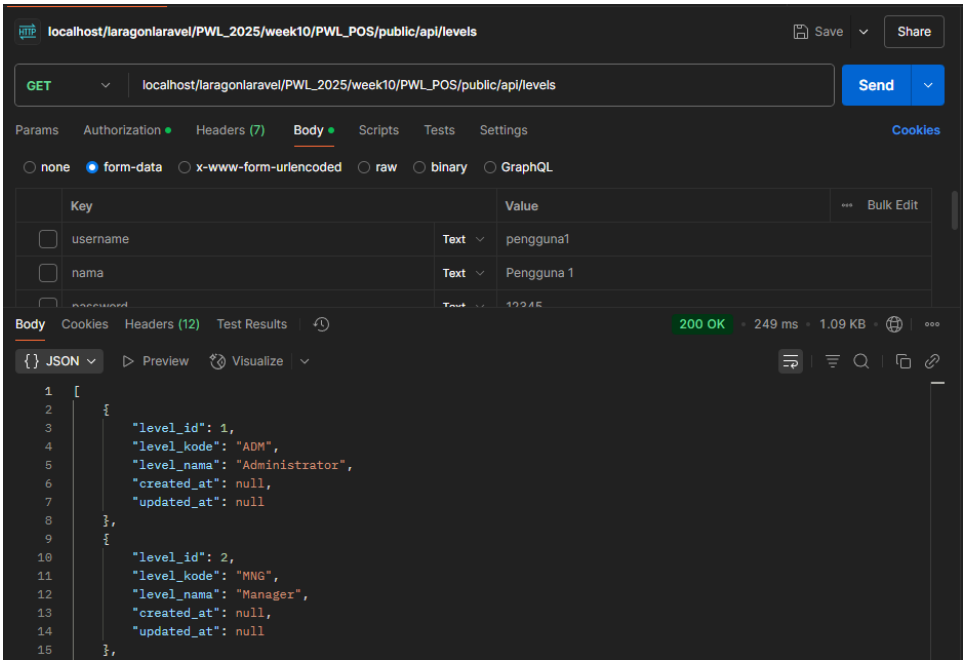
```php
<?php

namespace App\Http\Controllers\Api;

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

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

```php
    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.

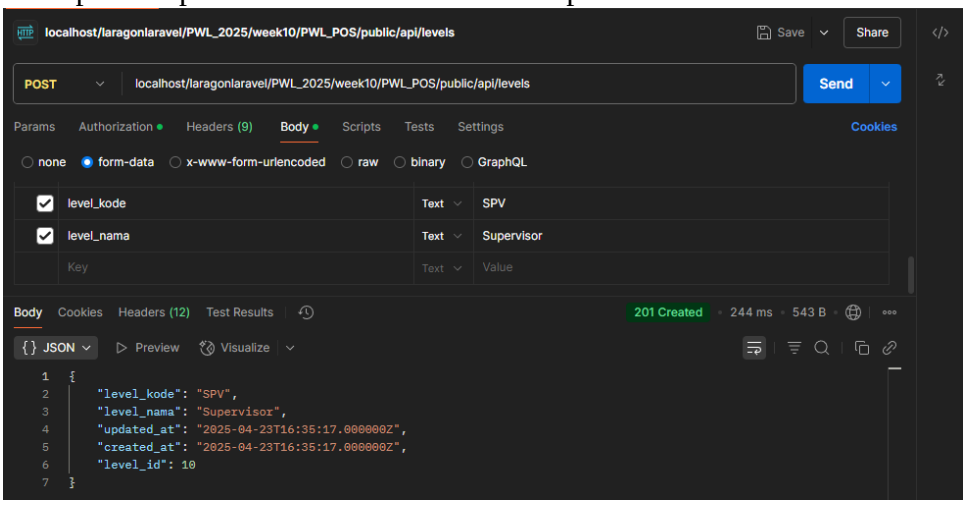
```php
Route::get('levels', [LevelController::class, 'index']);
Route::post('levels', [LevelController::class, 'store']);
Route::get('levels/{level}', [LevelController::class, 'show']);
Route::put('levels/{level}', [LevelController::class, 'update']);
Route::delete('levels/{level}', [LevelController::class,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

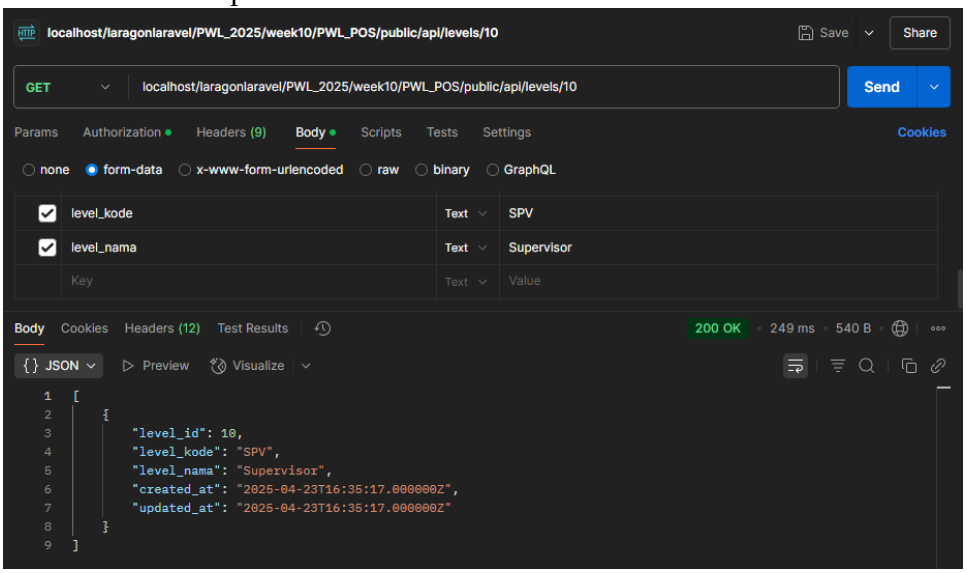


Akan menampilkan data menggunakan api/levels.

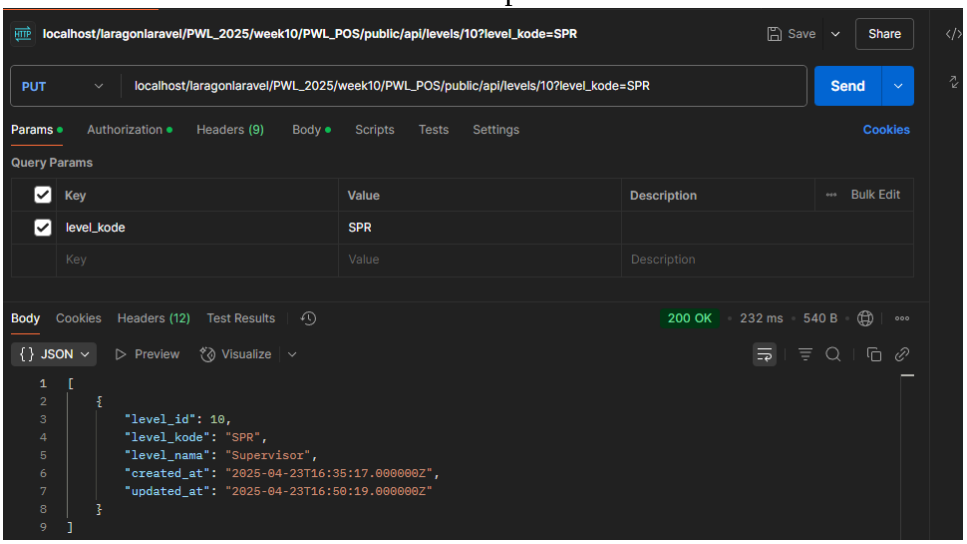
5. Kemudian, lakukan percobaan penambahan data dengan URL : localhost/PWL_POS-main/public/api/levels dan method POST seperti di bawah ini.



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



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



8. PUT digunakan untuk mengubah data yang sudah ada. Pada levels/{id}, PUT mengganti data level berdasarkan ID dengan data baru yang dikirim lewat Body.
9. Lakukan commit perubahan file pada Github.

TUGAS

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

- UserController

```php
<?php

namespace App\Http\Controllers\Api;

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

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

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

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

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

    public function destroy(UserModel $user)
    {
        $user->delete();
        return response()->json([
            'success' => true,
            'message' => 'Data Terhapus',
        ]);
    }
}
```

- KategoriController

```php
<?php

namespace App\Http\Controllers\Api;

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

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

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

    public function show(KategoriModel $kategori)
    {
        return KategoriModel::find($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',
        ]);
    }
}
```

- BarangController

```php
<?php

namespace App\Http\Controllers\Api;

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

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

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

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

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

    public function destroy(BarangModel $barang)
    {
        $barang->delete();
        return response()->json([
            'success' => true,
            'message' => 'Data Terhapus',
        ]);
    }
}
```

- Api.php

```php
Route::get('users', [UserController::class, 'index']);
Route::post('users', [UserController::class, 'store']);
Route::get('users/{user}', [UserController::class, 'show']);
Route::put('users/{user}', [UserController::class, 'update']);
Route::delete('users/{user}', [UserController::class, 'destroy']);

Route::get('kategoris', [KategoriController::class, 'index']);
Route::post('kategoris', [KategoriController::class, 'store']);
Route::get('kategoris/{kategori}', [KategoriController::class, 'show']);
Route::put('kategoris/{kategori}', [KategoriController::class, 'update']);
Route::delete('kategoris/{kategori}', [KategoriController::class, 'destroy']);

Route::get('barangs', [BarangController::class, 'index']);
Route::post('barangs', [BarangController::class, 'store']);
Route::get('barangs/{barang}', [BarangController::class, 'show']);
Route::put('barangs/{barang}', [BarangController::class, 'update']);
Route::delete('barangs/{barang}', [BarangController::class, 'destroy']);
```