KEMENTERIAN PENDIDIKAN, KEBUDAYAAN, RISET, DAN TEKNOLOGI
POLITEKNIK NEGERI MALANG
JURUSAN TEKNOLOGI INFORMASI
Jl. Soekarno Hatta No. 9, Jatimulyo, Lowokwaru, Malang 65141
Telp. (0341) 404424 – 404425, Fax (0341) 404420
http://www.polinema.ac.id

Mata Kuliah	:	Pemrograman Web Lanjut (PWL)
Program Studi	:	D4 – Teknik Informatika / D4 – Sistem Informasi Bisnis
Semester	:	4 (empat) / 6 (enam)
Pertemuan ke-	:	11 (sebelas)

Nama : Muhammad Fairuz Daffa Athallah
Kelas : SIB 2B / 15
NIM : 2341760079

JOBSHEET 11

RESTFUL API 2

Sebelumnya kita sudah membahas mengenai *RESTFUL API dan JSON Web Token (JWT)* pada Laravel. Dimana kita telah membuat RESTful API Register, RESTful API Login, RESTful API Logout, dan implementasi CRUD dalam RESTful API pada halaman web. Pada pertemuan kali ini, kita akan mempelajari penerapan RESTFUL API lanjutan 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. ELOQUENT ACCESSOR

Laravel memiliki fitur yang bernama mutator, accessor dan casting, fitur-fitur ini digunakan untuk melakukan manipulasi data di dalam attribute database dengan sangat mudah. Contohnya pada saat insert data dengan enkripsi ke dalam database dan melakukan deskripsi saat menampilkan dari database secara otomatis.

Accessor dapat mengubah nilai saat attribute atau field eloquent diakses. Untuk mendefinisikan Accessor dapat membuat method di dalam model untuk menentukan attribute yang akan diakses. Nama method yang dibuat harus sama dengan nama attribute yang akan di format: contoh :

Attribute/field pada tabel first_name maka methodnya firstName()

```
protected function firstName(): Attribute
{
    //...
}
```

Jika membuat attribute/field image yang ada ditable m_user kita akan memberikan nilai full path dari direktori dimana file gambar tersebut disimpan.contohnya pada UserModel ditambahkan

```php
protected function image(): Attribute
{
    return Attribute::make(
        get: fn ($image) => url('/storage/posts/' . $image),
    );
}
```

Dengan begitu dapat melakukan import Eloquent Attribute dengan

```php
use Illuminate\Database\Eloquent\Casts\Attribute;
```

Lalu method baru dengan nama image() melakukan return path nama file image itu berada

```php
get: fn ($image) => url('/storage/posts/' . $image),
```

Hasil akhir memanggil attribute image

```
domain.com/storage/posts/nama_file_image.png
```

Praktikum 1 – Implementasi Eloquent Accessor

1. Sebelum memulai pastikan REST API, terlebih dahulu pastikan sudah ter instal aplikasi Postman.

2. Kita akan memodifikasi Table m_user dengan menambahkan column : image, buka terminal lalu ketikkan

   php artisan make:migration add_image_to_m_user_table

   ```
   PS C:\laragon\www\PWL_POS> php artisan make:migration add_image_to_m_user_table

   INFO  Migration [C:\laragon\www\PWL_POS\database\migrations/2024_04_30_040822_add_image_to_m_user_table.php] created successfully.
   ```

   migrate

3. Buka file migrasi tersebut lalu modifikasi seperti ini lalu simpan:

```php
<?php

use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

return new class extends Migration
{
    /**
     * Run the migrations.
     */
    public function up(): void
    {
        Schema::table('m_user', function (Blueprint $table) {
            $table->string('image');
        });
    }

    /**
     * Reverse the migrations.
     */
    public function down(): void
    {
        Schema::table('m_user', function (Blueprint $table) {
            $table->dropColumn('image');
        });
    }
};
```

Migration

UserModel.php M | 2025_04_24_112419_add_image_to_m_user_table.php U

Minggu_11 > PWL_POS > database > migrations > 2025_04_24_112419_add_image_to_m_

```php
<?php

use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

return new class extends Migration
{
    /**
     * Run the migrations.
     */
    public function up(): void
    {
        Schema::table('m_user', function (Blueprint $table) {
            $table->string('image');
        });
    }

    /**
     * Reverse the migrations.
     */
    public function down(): void
    {
        Schema::table('m_user', function (Blueprint $table) {
            $table->dropColumn('image');

        });
    }
};
```

4. Lakukan jalankan update migrasi dengan cara:

```
php artisan migrate
```

```
PS E:\Semester3\Pemrograman_Web\laragon\www\PWL_2025_Daffa\Minggu_11\PWL_POS> php artisan migrate

Warning: PHP Startup: Unable to load dynamic library 'php_sqlsrv.dll' (tried: E:/Semester3/Pemrograman_Web/laragon/bin/php/php-8.3.17-Win32-vs16-x64/
mester3/Pemrograman_Web/laragon/bin/php/php-8.3.17-Win32-vs16-x64/ext\php_php_sqlsrv.dll.dll (The specified module could not be found)) in Unknown on

Warning: PHP Startup: Unable to load dynamic library 'php_pdo_sqlsrv.dll' (tried: E:/Semester3/Pemrograman_Web/laragon/bin/php/php-8.3.17-Win32-vs16-
), E:/Semester3/Pemrograman_Web/laragon/bin/php/php-8.3.17-Win32-vs16-x64/ext\php_php_pdo_sqlsrv.dll.dll (The specified module could not be found)) i

  INFO  Running migrations.

  2025_04_24_112419_add_image_to_m_user_table ..... 418ms DONE
```

5. Lalu lakukan modifikasi models pada App/Models/UserModel.php

```php
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Model;
use Tymon\JWTAuth\Contracts\JWTSubject;
use Illuminate\Database\Eloquent\Casts\Attribute;
use Illuminate\Foundation\Auth\User as Authenticatable;
```

```php
class UserModel extends Authenticatable implements JWTSubject
{

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

    public function getJWTCustomClaims(){
        return [];
    }

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

    protected $fillable = [
        'username',
        'nama',
        'password',
        'level_id',
        'image'//tambahan
    ];

    public function level()
    {
        return $this->belongsTo(LevelModel::class, 'level_id',
'level_id');
    }
    protected function image(): Attribute
    {
        return Attribute::make(
            get: fn ($image) => url('/storage/posts/' . $image),
        );
    }

}
```

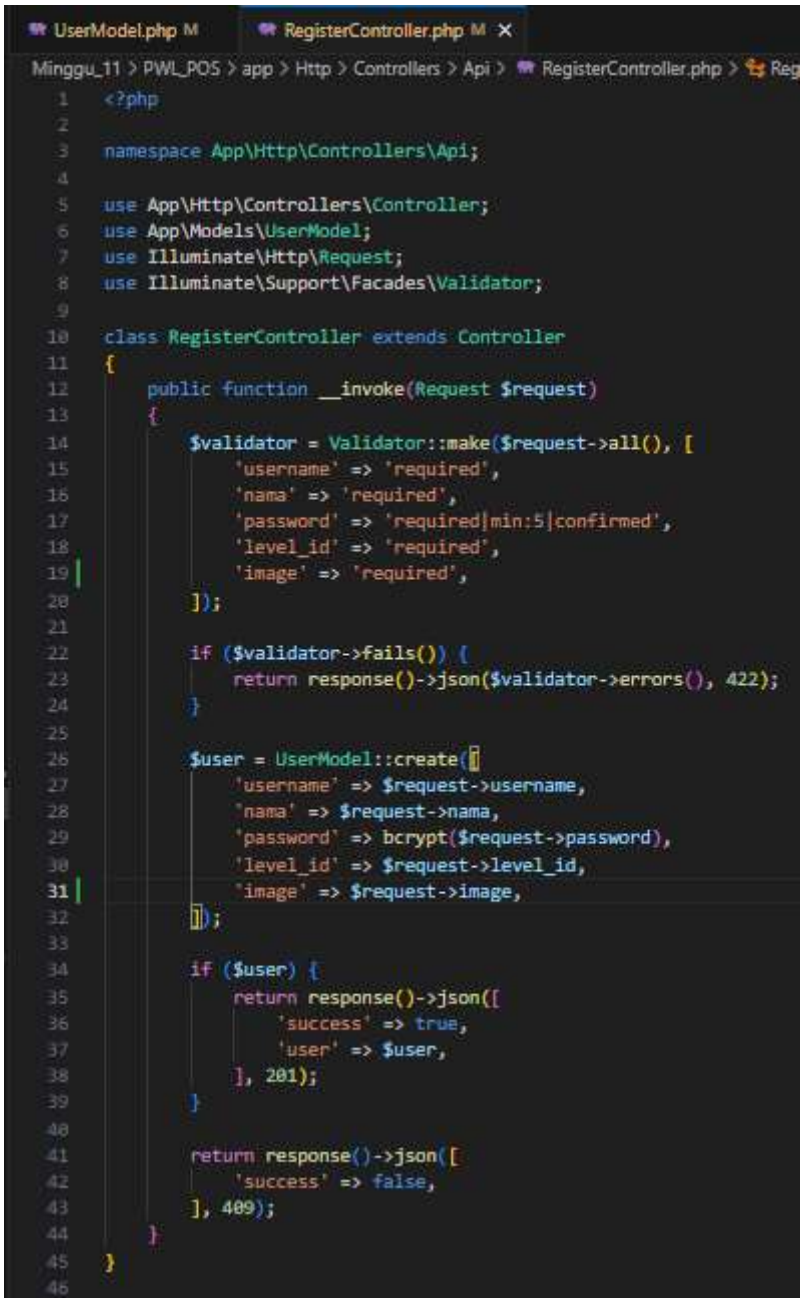
UserModel

6. Lakukan modifikasi pada Controllers/Api/RegisterController

```php
<?php

namespace App\Http\Controllers\Api;

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

```php
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',
            'image' => '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,
            'image' => $request->image
        ]);

        //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);
    }
}
```

RegisterController



7. Anda dapat menambahkan detail untuk spesifikasi image pada validator

```
'image'      => 'required|image|mimes:jpeg,png,jpg,gif,svg|max:2048',
```

```
public function __invoke(Request $request)
{
    $validator = Validator::make($request->all(), [
        'username' => 'required',
        'nama' => 'required',
        'password' => 'required|min:5|confirmed',
        'level_id' => 'required',
        'image' => 'required|image|mimes:jpeg,png,jpg,gif,svg|max:2048',
    ]);
```

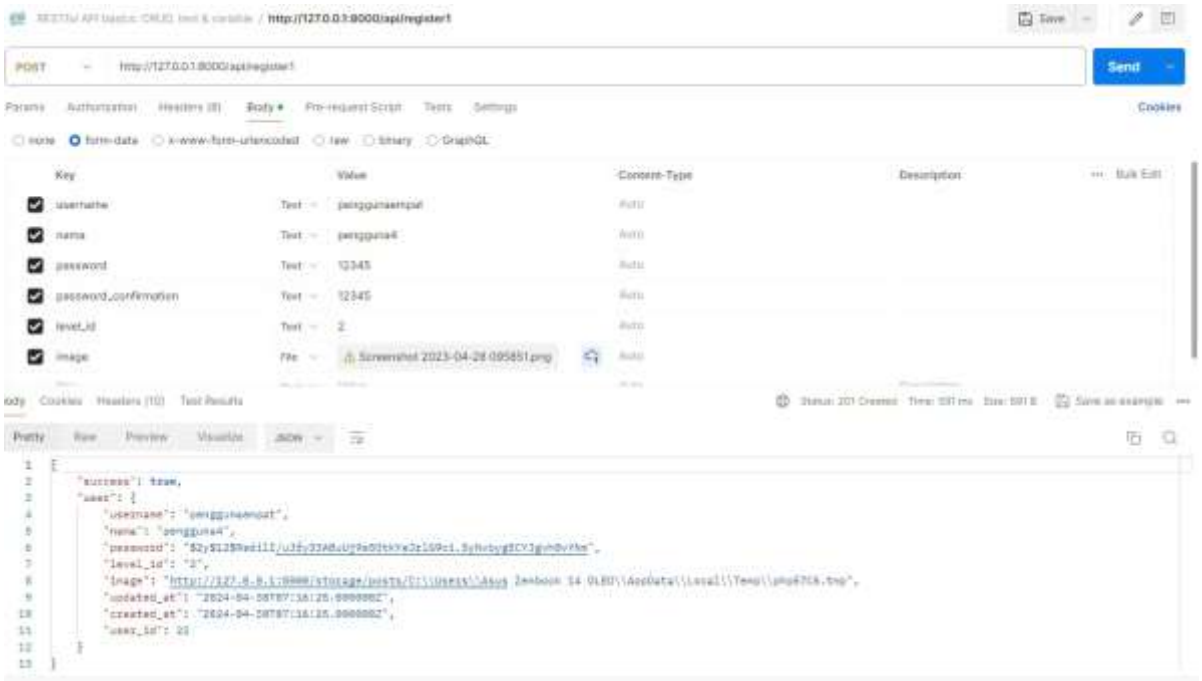
8. Ubah atau tambahkan register1 pada routes/api.php

```
Route::post('/register1', App\Http\Controllers\Api\RegisterController::class)->name('register1');
```

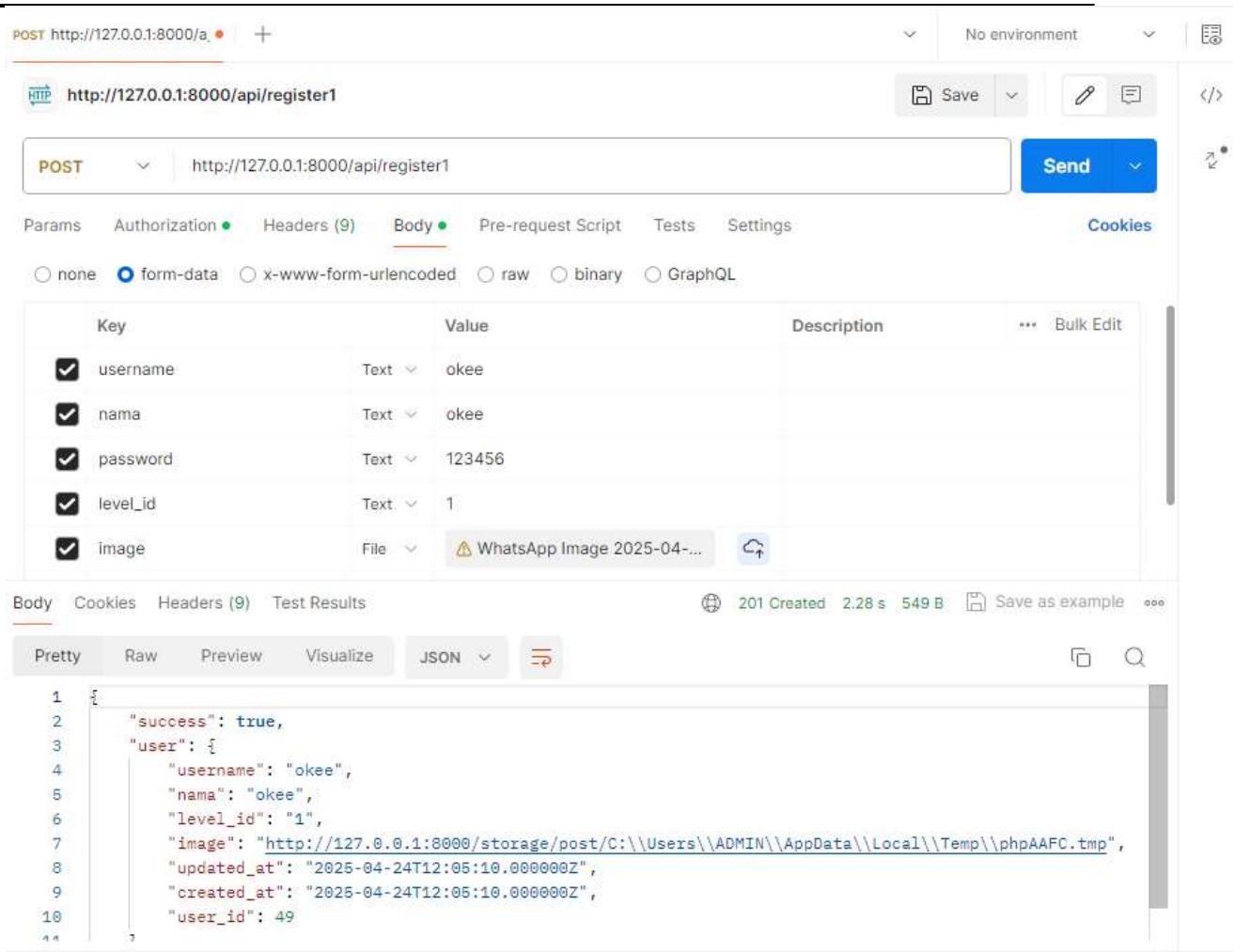
Api.php

```
22 | Route::post('/register1', \App\Http\Controllers\Api\RegisterController::class)->name('register1');
```

9. Simpan dan akses pada aplikasi Postman, atur pada Body isi manual Key dan Valuenya pada Key image tambahkan value File dan upload gambar http://127.0.0.1:8000/api/register1 dengan method POST dan klik send



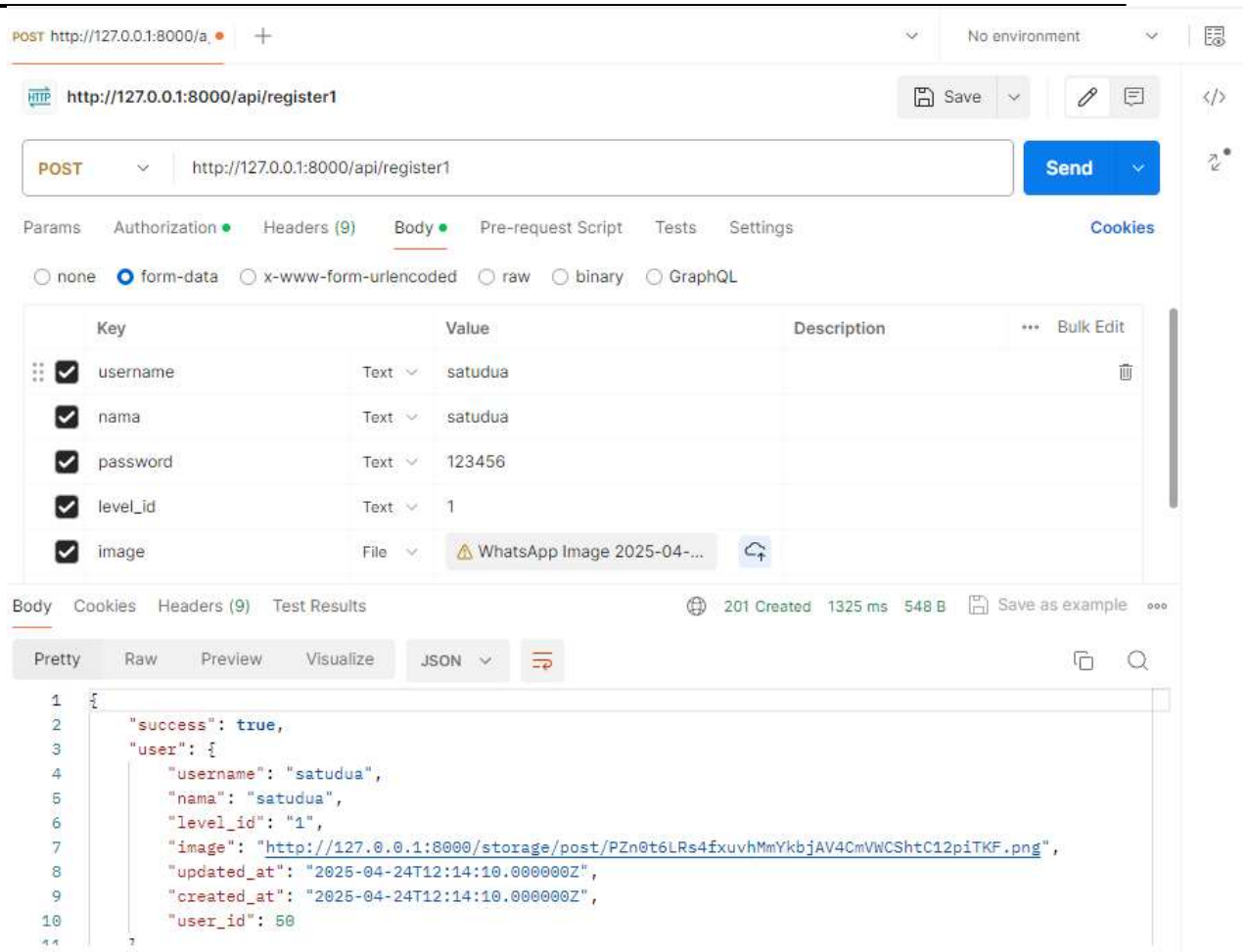
Postman



10. Pada Controllers/Api/RegisterController bagian create user ganti dengan

```
'image'     => $image->hashName(),
```

11. Uji coba dan screenshot hasilnya apa perbedaan dari yang sebelumnya

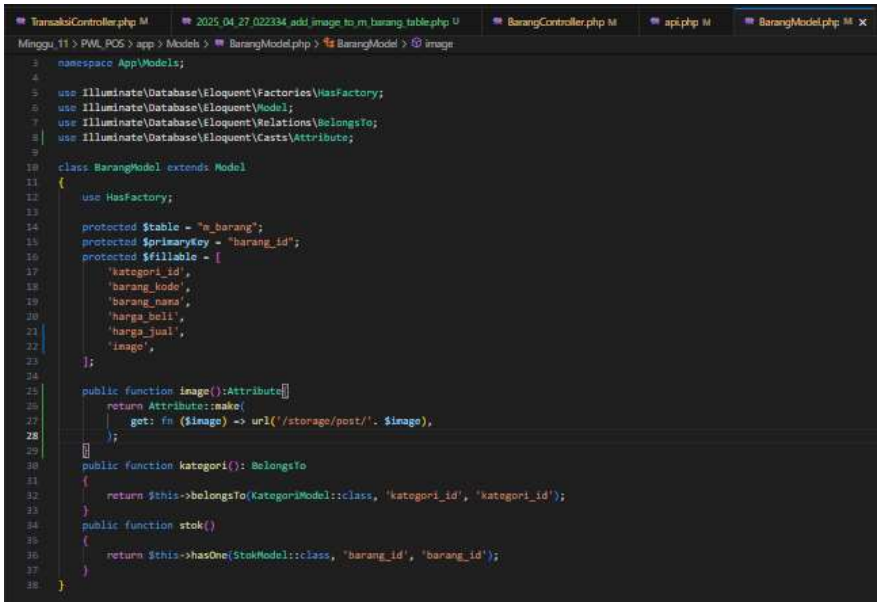


Alamat url dari image akan di enkripsi

TUGAS

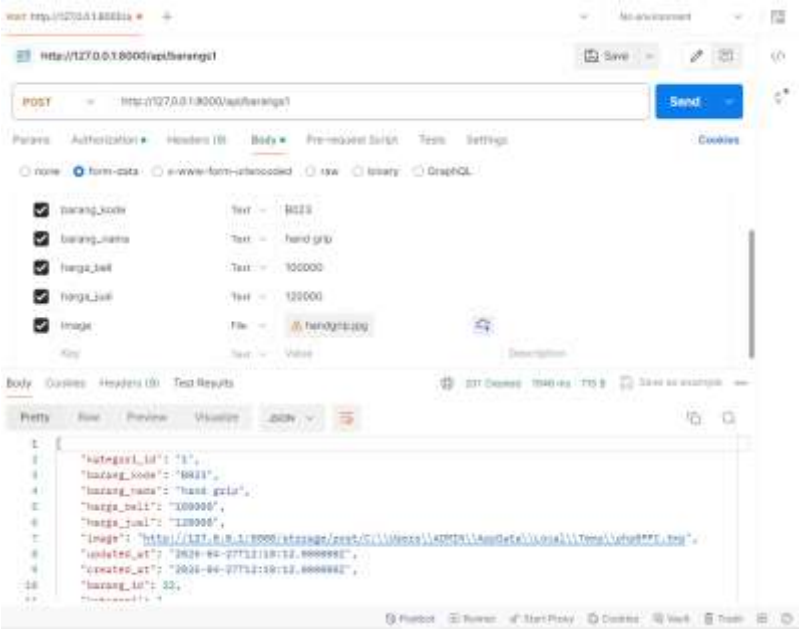
Implementasikan API untuk upload file/gambar pada tabel lainnya yaitu tabel m_barang dan gunakan pada transaksi. Uji coba dengan method GET untuk memanggil data yang sudah di inputkan.

BarangModel

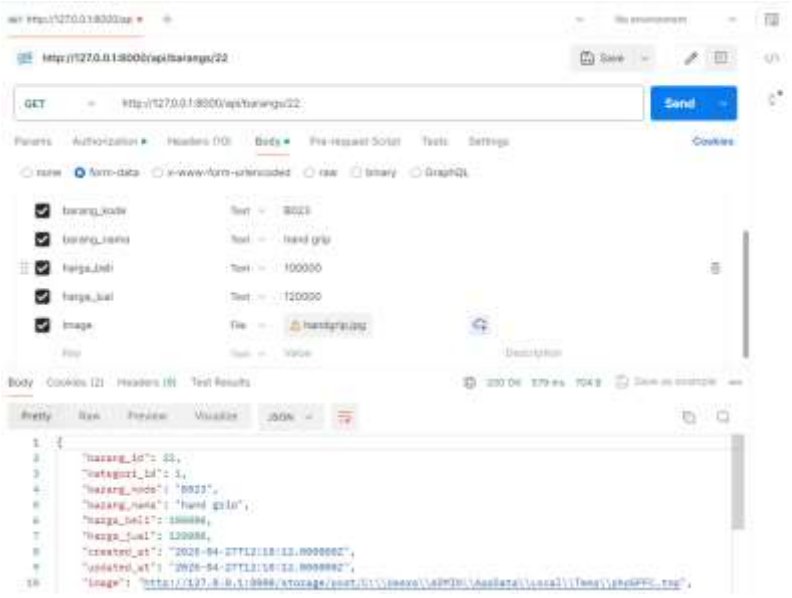


BarangController

CreateData



GetData



Transaksi

DetailPenjualanModel

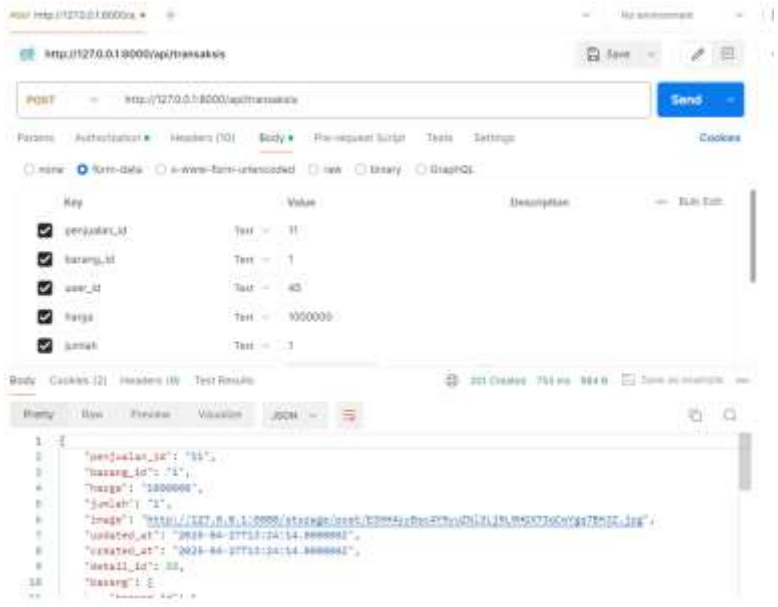
```php
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Casts\Attribute;
use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Model;
use Illuminate\Database\Eloquent\Relations\BelongsTo;

class PenjualanDetailModel extends Model
{
    use HasFactory;

    protected $table = 't_penjualan_detail';
    protected $primaryKey = 'detail_id';
    protected $fillable = [
        'penjualan_id',
        'barang_id',
        'harga',
        'jumlah',
        'image'
    ];

    public function image():Attribute{
        return Attribute::make(
            get: fn ($image) => url('/storage/post/'. $image),
        );
    }

    // Relasi ke header transaksi
    public function sales(): BelongsTo
    {
        return $this->belongsTo(PenjualanModel::class, 'penjualan_id', 'penjualan_id');
    }

    // Relasi ke data barang
    public function barang(): BelongsTo
    {
        return $this->belongsTo(BarangModel::class, 'barang_id', 'barang_id');
    }
}
```

TransaksiController

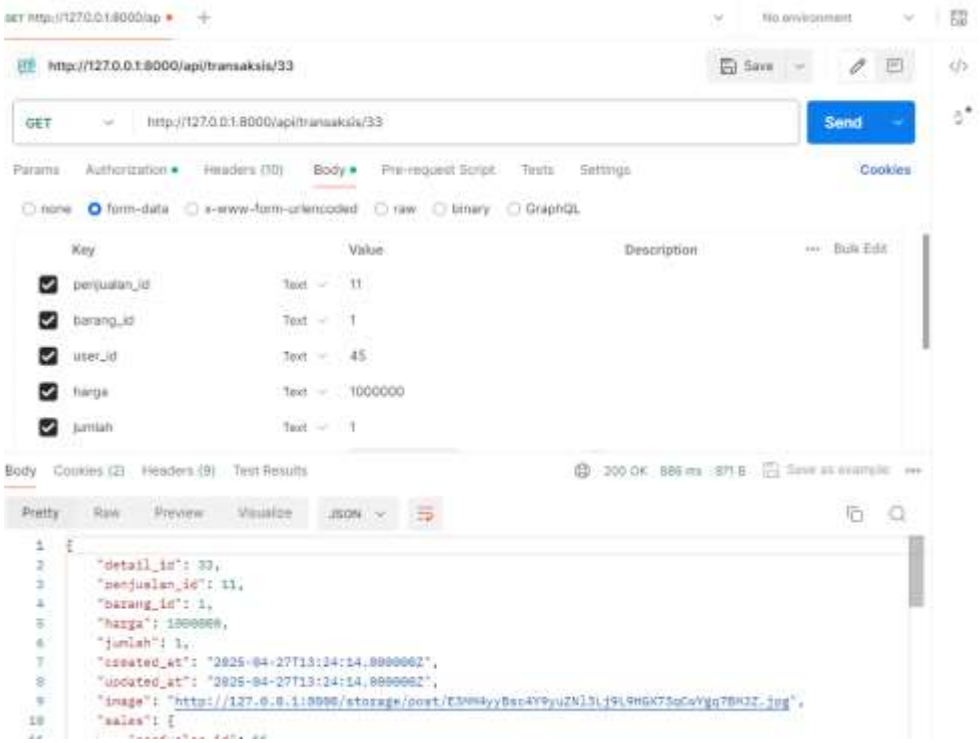
Minggu_11 > PWL_POS > app > Http > Controllers > Api > TransaksiController.php > TransaksiController > store

```php
<?php

namespace App\Http\Controllers\Api;

use App\Http\Controllers\Controller;
use App\Models\PenjualanDetailModel;
use Illuminate\Http\Request;
use App\Models\PenjualanModel;
use Illuminate\Support\Facades\Storage;
use Illuminate\Support\Facades\Validator;

class TransaksiController extends Controller
{
    public function index()
    {
        $data = PenjualanDetailModel::with(['sales', 'barang'])->get();
        return response()->json($data);
    }

    public function show($transaksi)
    {
        $penjualan = PenjualanDetailModel::with(['sales', 'barang'])->findOrFail($transaksi);
        return response()->json($penjualan);
    }

    /** POST /api/transaksi */
    public function store(Request $request)
    {
        $validated = $request->validate([
            'penjualan_id' => 'required|exists:t_penjualan,penjualan_id',
            'barang_id' => 'required|exists:m_barang,barang_id',
            'harga' => 'required|numeric',
            'jumlah' => 'required|numeric',
            'image' => 'nullable|image|mimes:jpeg,png,jpg,gif,svg|max:2048',
        ]);

        if ($request->hasFile('image')) {
            $validated['image'] = basename($request->file('image')->store('post', 'public'));
        }

        $penjualandetail = PenjualanDetailModel::create($validated);

        return response()->json($penjualandetail->load('barang', 'sales'), 201);
    }

    // /** GET /api/transaksi/{transaksi} */
    // public function show($transaksi)
    // {
    //     $trx = PenjualanDetailModel::findOrFail($transaksi);
    //     return response()->json($trx);
    // }

    /** PUT /api/transaksi/{transaksi} */
    public function update(Request $request, $transaksi)
    {
        $trx = PenjualanDetailModel::findOrFail($transaksi);

        $v = Validator::make($request->all(), [
```

StoreData



GetData



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