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

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

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

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

Dengan begitu dapat melakukan import Eloquent Attribute dengan

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

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

```
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_2025\PWL_WEEK11\PWL_POS> php artisan make:migration add_image_to_m_user_table

INFO  Migration [C:\laragon\www\PWL_2025\PWL_WEEK11\PWL_POS\database\migrations/2025_04_27_112716_add_image_to_m_user_table.php] created successfully.
```

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');
        });
    }
};

```

4. Lakukan jalankan update migrasi dengan cara: `php artisan migrate`

```
S C:\laragon\www\PWL_2025\PWL_WEEK11\PWL_POS> php artisan migrate
```

```
INFO  Running migrations.

2025_04_27_112716_add_image_to_m_user_table ..... 63ms DONE
```

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

```php
<?php
namespace App\Models;

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

class UserModel extends Authenticatable implements JWTSubject
{
    use HasFactory;

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

    public function getJWTCustomClaims(){
        return [];
    }

    protected $table = 'm_user';
    protected $primaryKey = 'user_id';
    protected $fillable = ['username', 'password', 'nama', 'level_id', 'profile_picture', 'created_at', 'updated_at', 'image']; //tambahan

    protected $hidden = ['password']; // jangan di tampilkan saat select

    protected $casts = ['password' => 'hashed']; // casting password agar otomatis di hash

    /**
     * Relasi ke tabel level
     */
    public function level(): BelongsTo
    {
        return $this->belongsTo(LevelModel::class, 'level_id', 'level_id');
    }

    /**
     * Mendapatkan nama role
     */
    public function getRoleName(): string
    {
        return $this->level->level_nama;
    }

    /**
     * Cek apakah user memiliki role tertentu
     */
    public function hasRole($role): bool
    {
        return $this->level->level_kode == $role;
    }

     /**
     * Mendapatkan nama role
     */
    public function getRole()
    {
        return $this->level->level_kode;
    }

    public function getProfilePictureUrlAttribute()
    {
        return $this->profile_picture
            ? asset('storage/profile_pictures/'.$this->profile_picture)
            : asset('images/default-profile.png');
    }
    protected function image(): Attribute
    {
        return Attribute::make(
            get: fn ($image) => url('/storage/posts/' . $image),
        );
    }
}
```

6. Lakukan modifikasi pada Controllers/Api/RegisterController l

```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 validation
        $validator = Validator::make($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'
        ]);

        // If validation 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 if user is created
        if ($user) {
            return response()->json([
                'success' => true,
                'user'    => $user,
            ], 201);
        }

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

7. Anda dapat menambahkan detail untuk spesifikasi image pada validator

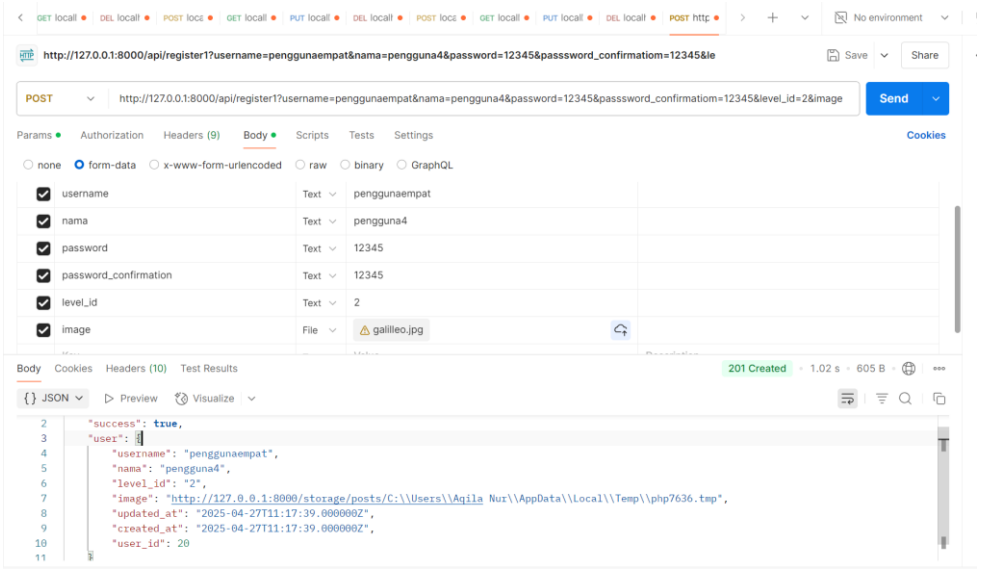
```php
'image' => 'required|image|mimes:jpeg,png,jpg,gif,svg|max:2048'
```

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

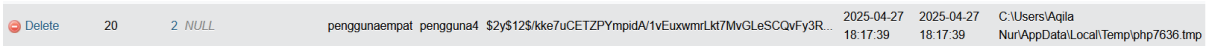
```php
Route::post('/register1', App\Http\Controllers\Api\RegisterController::class)->name('register1');
```

8. 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

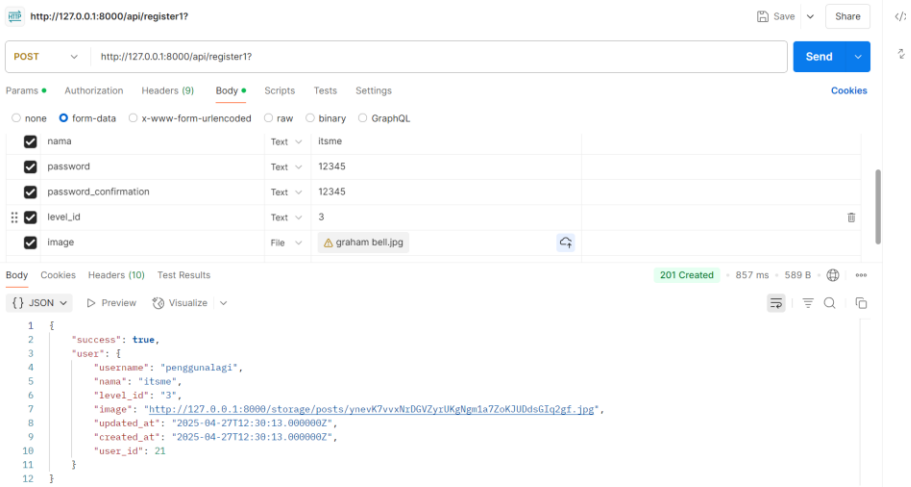


Pada database :



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

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



Perbedaan :

- Sebelum di modif :

```
C:\Users\Aqila Nur\AppData\Local\Temp\php7636.tmp
```

- Setelah di modif :

```
ynevK7vvxNrDGVZyrUKgNgm1a7ZoKJUDdsGlq2gf.jpg
```

Penjelasan :

- Jika menggunakan *'image' => request->image* maka langsung ambil file yang di-*upload* tanpa diproses apa pun. Yang tersimpan ke database bukan nama file (contohnya gambar.jpg), tapi object file pathnya.
- Jika menggunakan *'image' => $request->image->hashName()* maka akan mengambil file ➔ simpan dulu ke storage (storage/app/public/users/) ➔ baru simpan nama filenya ke database. File gambarnya akan masuk ke folder storage/app/public/users. Di database, hanya akan menyimpan nama file-nya aja, contoh *graham bell.jpg*, jadi lebih ringan dan aman.

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.

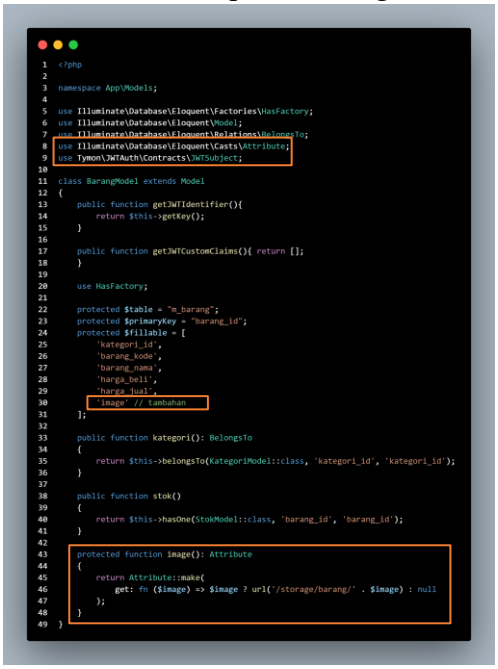
1. Tabel Barang
   - Membuat migrasi

```
PS C:\laragon\www\PWL_2025\PWL_WEEK11\PWL_POS> php artisan make:migration add_image_to_m_barang_table

INFO  Migration [C:\laragon\www\PWL_2025\PWL_WEEK11\PWL_POS\database\migrations/2025_04_27_200941_add_image_to_m_barang_table.php] created successfully.
```

   - Tambahkan kode pada migration yang telah di buat

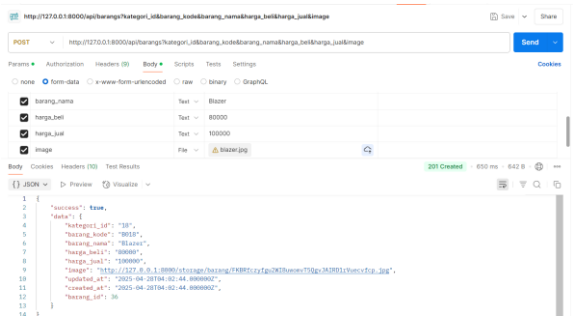
```php
<?php

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

return new class extends Migration
{
    public function up(): void
    {
        Schema::table('m_barang', function (Blueprint $table) {
            $table->string('image')->nullable(); // boleh null
        });
    }

    public function down(): void
    {
        Schema::table('m_barang', function (Blueprint $table) {
            $table->dropColumn('image');
        });
    }
};
```

- Tambahkan kode pada BarangModel

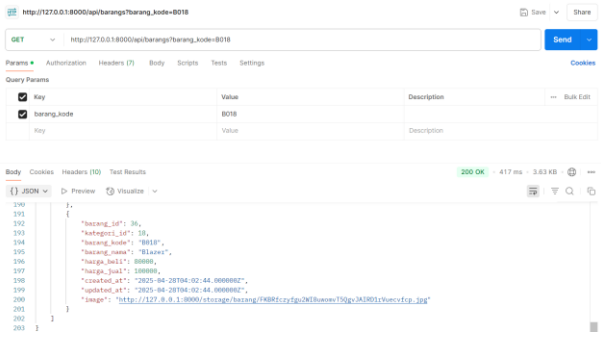


```php
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Model;
use Illuminate\Database\Eloquent\Relations\BelongsTo;
use Illuminate\Database\Eloquent\Casts\Attribute;
use Tymon\JWTAuth\Contracts\JWTSubject;

class BarangModel extends Model
{
    public function getJWTIdentifier(){
        return $this->getKey();
    }

    public function getJWTCustomClaims(){ return []; }

    use HasFactory;

    protected $table = "m_barang";
    protected $primaryKey = "barang_id";
    protected $fillable = [
        'kategori_id',
        'barang_kode',
        'barang_nama',
        'harga_beli',
        'harga_jual',
        'image' // tambahan
    ];

    public function kategori(): BelongsTo
    {
        return $this->belongsTo(KategoriModel::class, 'kategori_id', 'kategori_id');
    }

    public function stok()
    {
        return $this->hasOne(StokModel::class, 'barang_id', 'barang_id');
    }

    protected function image(): Attribute
    {
        return Attribute::make(
            get: fn ($image) => $image ? url('/storage/barang/' . $image) : null
        );
    }
}
```

- Create data barang (POST)



- Memanggil data barang (GET)



2. Tabel Transaksi
   - Membuat Controller pada folder API/PenjualanController

```
PS C:\laragon\www\PWL_2025\PWL_WEEK11\PWL_POS> php artisan make:controller Api/PenjualanController

INFO  Controller [C:\laragon\www\PWL_2025\PWL_WEEK11\PWL_POS\app\Http\Controllers\Api\PenjualanController.php] created successfull
```

   - Tambahkan kode pada PenjualanController

```php
<?php

namespace App\Http\Controllers\Api;

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

class PenjualanController extends Controller
{
    public function index()
    {
        $data = PenjualanModel::with(['user', 'details.barang'])->get();
        return response()->json($data);
    }

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

    public function store(Request $request)
    {
        $v = Validator::make($request->all(), [
            'user_id'          => 'required|exists:m_user,user_id',
            'pembeli'          => 'required|string|max:100',
            'penjualan_kode'   => 'required|string|unique:t_penjualan,penjualan_kode',
            'penjualan_tanggal' => 'required|date_format:Y-m-d H:i:s',
            'image'            => 'nullable|image|mimes:jpg,jpeg,png|max:2048',
        ]);

        if ($v->fails()) {
            return response()->json(['success'=>false,'errors'=>$v->errors()], 422);
        }

        $trx = PenjualanModel::create($v->validated());

        if ($request->hasFile('image')) {
            $fn = time().'.'.$request->image->extension();
            $request->image->storeAs('public/transaksi', $fn);
            $trx->update(['image'=>$fn]);
        }

        return response()->json([
            'success' => true,
            'data' => [
                'penjualan' => $trx,
                'image_url' => $trx->image ? asset('storage/transaksi/'.$trx->image) : null,
            ],
        ], 201);
    }

    public function update(Request $request, $transaksi)
    {
        $trx = PenjualanModel::findOrFail($transaksi);

        $v = Validator::make($request->all(), [
            'pembeli'          => 'sometimes|required|string|max:100',
            'penjualan_kode'   => 'sometimes|required|string|unique:t_penjualan,penjualan_kode,'.$transaksi.',penjualan_id',
            'penjualan_tanggal' => 'sometimes|required|date',
            'image'            => 'nullable|image|mimes:jpg,jpeg,png|max:2048',
        ]);

        if ($v->fails()) {
            return response()->json(['success'=>false,'errors'=>$v->errors()], 422);
        }

        $trx->update($v->validated());

        if ($request->hasFile('image')) {
            if ($trx->image) {
                Storage::delete('public/transaksi/'.$trx->image);
            }
            $fn = time().'.'.$request->image->extension();
            $request->image->storeAs('public/transaksi', $fn);
            $trx->update(['image'=>$fn]);
        }

        return response()->json($trx);
    }

    public function destroy($transaksi)
    {
        $trx = PenjualanModel::findOrFail($transaksi);
        //cek file exists
        if ($trx->image && Storage::exists('public/transaksi/'.$trx->image)) {
            Storage::delete('public/transaksi/'.$trx->image);
        }
        $trx->delete();
        return response()->json(['success'=>true,'message'=>'Transaksi dihapus']);
    }
}
```

- Membuat migrasi add image pada tabel penjualan

```
PS C:\laragon\www\PWL_2025\PWL_WEEK11\PWL_POS> php artisan make:migration add_image_to_t_penjualan_table

  INFO  Migration [C:\laragon\www\PWL_2025\PWL_WEEK11\PWL_POS\database\migrations/2025_04_28_231127_add_image_to_t_penjualan_table.php] created successfully.
```

- Tambahkan program kode pada migrasi

```php
<?php

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

return new class extends Migration
{
    public function up(): void
    {
        Schema::table('t_penjualan', function (Blueprint $table) {
            $table->string('image')->nullable()->after('penjualan_tanggal');
        });
    }

    public function down(): void
    {
        Schema::table('t_penjualan', function (Blueprint $table) {
            $table->dropColumn('image');
        });
    }
};
```

- Run Migrate

```
PS C:\laragon\www\PWL_2025\PWL_WEEK11\PWL_POS> php artisan migrate

INFO  Running migrations.

2025_04_28_231127_add_image_to_t_penjualan_table ..... 115ms DONE
```

- Tambah kode pada Penjualan Model

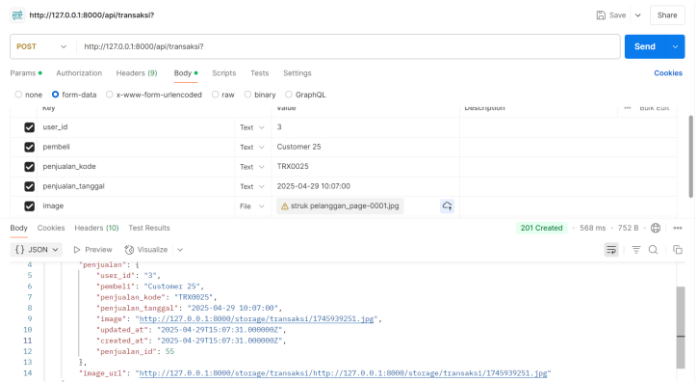
```php
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Model;
use Illuminate\Database\Eloquent\Relations\BelongsTo;
use Illuminate\Database\Eloquent\Relations\HasMany;
use App\Models\UserModel;
use App\Models\DetailPenjualanModel;
use Illuminate\Database\Eloquent\Casts\Attribute;
use Tymon\JWTAuth\Contracts\JWTSubject;
use Illuminate\Foundation\Auth\User as Authenticatable;

class PenjualanModel extends Model
{
    public function getJWTIdentifier(){
        return $this->getKey();
    }

    public function getJWTCustomClaims(){
        return [];
    }

    use HasFactory;
    protected $table = 't_penjualan';
    protected $primaryKey = 'penjualan_id';
    /**
     * The attributes that are mass assignable.
     *
     * @var array
     */
    protected $fillable = ['user_id', 'pembeli', 'penjualan_kode', 'penjualan_tanggal', 'image'];

    //Relasi ke tabel user
    public function user(): BelongsTo
    {
        return $this->belongsTo(UserModel::class, 'user_id', 'user_id');
    }

    public function details(): HasMany
    {
        return $this->hasMany(DetailPenjualanModel::class, 'penjualan_id', 'penjualan_id');}


    // Menambahkan method untuk mendapatkan nama
    public function getUsernameAttribute()
    {
        return $this->user->nama ?? null;  // Ambil nama dari relasi user
    }

    protected function image(): Attribute
    {
        return Attribute::make(
            get: fn ($image) => $image ? url('/storage/transaksi/' . $image) : null
        );
    }
}
```

- Tambahkan kode pada route api.php

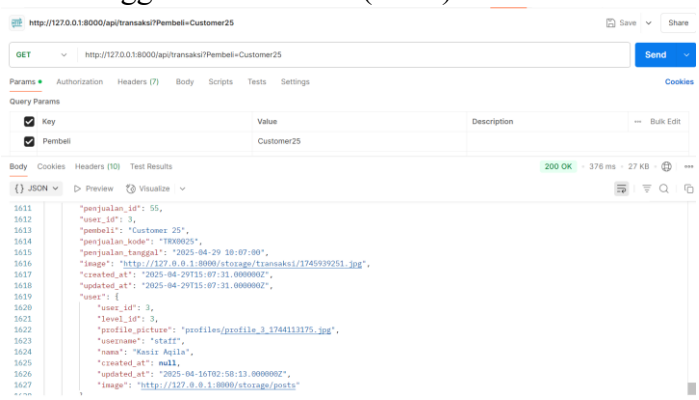
```php
use App\Http\Controllers\Api\PenjualanController;
```

```php
// Route Transaksi
Route::get('transaksi', [PenjualanController::class, 'index']);
Route::get('transaksi/{transaksi}', [PenjualanController::class, 'show']);
Route::post('transaksi', [PenjualanController::class, 'store']);
Route::post('transaksi/{transaksi}', [PenjualanController::class, 'update']);
Route::delete('transaksi/{transaksi}', [PenjualanController::class, 'destroy']);
```

- Create data transaksi (POST)



- Memanggil data transaksi (GET)



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