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)

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

```
C:\laragon\www\PWL_2025\Week11\Jobsheet>php artisan make:migration add_image_to_m_user_table
```

```
  INFO  Migration [C:\laragon\www\PWL_2025\Week11\Jobsheet\database\migrations/2025_05_06_202627_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`

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;
```

```
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),
);
    }

}
```

6. Lakukan modifikasi pada Controllers/Api/RegisterController

```
<?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'

        ]);

        //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);
    }
}
```

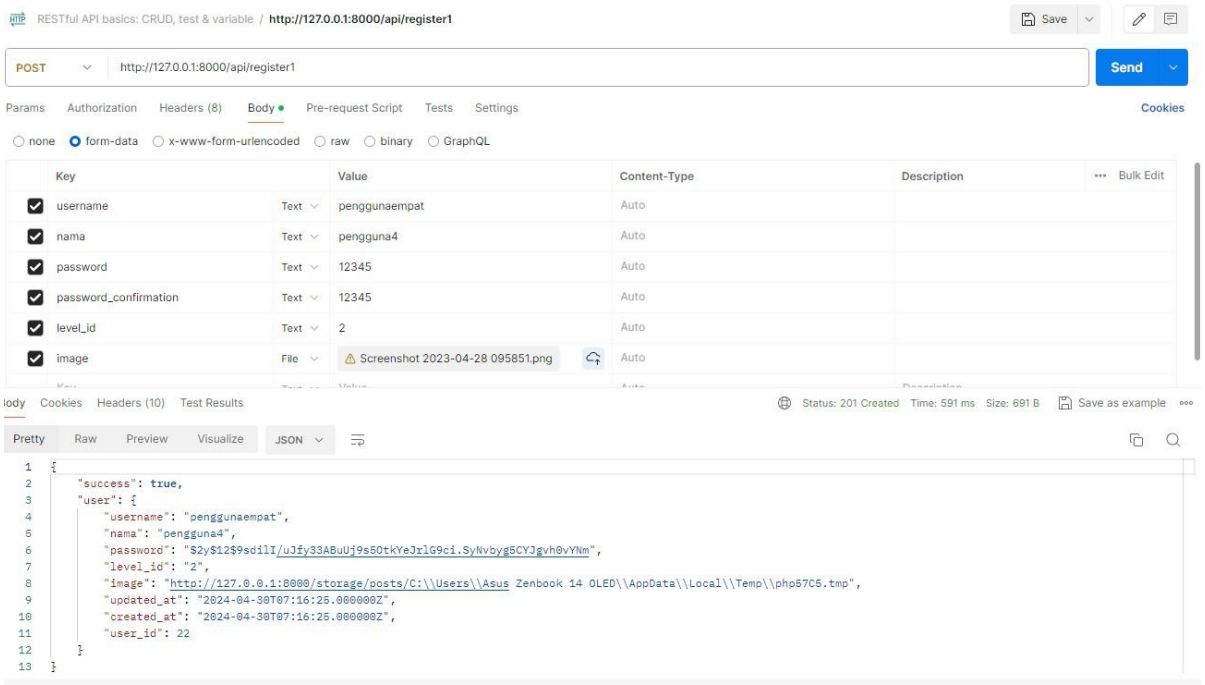
7. Anda dapat menambahkan detail untuk spesifikasi image pada validator

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

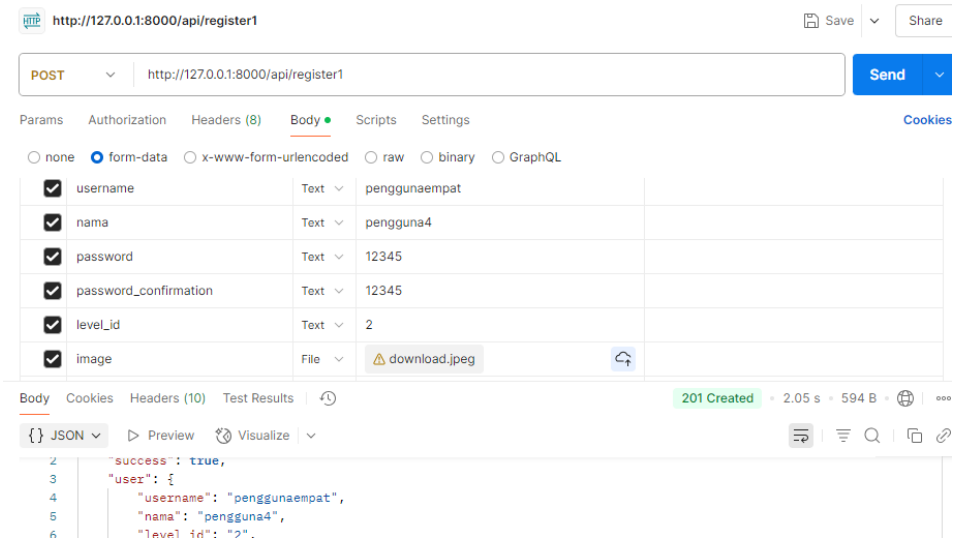
8. Ubah atau tambahkan register1 pada routes/api.php

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

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



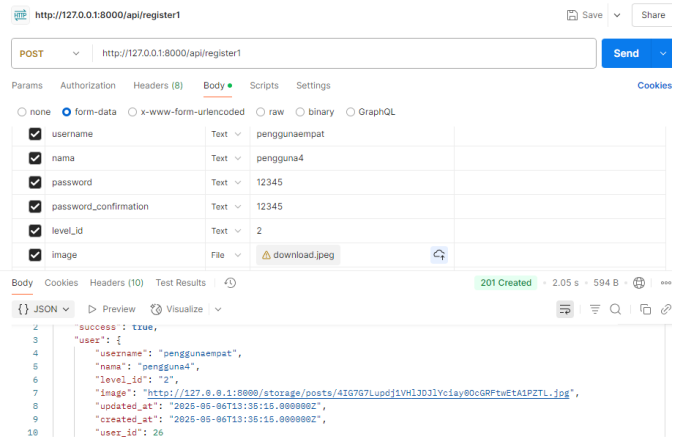
Hasil:



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

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

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



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.

- M_barang
  - Make migration for image

```
C:\laragon\www\PWL_2025\Week11\Jobsheet>php artisan make:migration add_image_to_m_barang_table

  INFO  Migration [C:\laragon\www\PWL_2025\Week11\Jobsheet\database\migrations/2025_05_06_204307_add_image_to_m_barang_table.php] created successfully.
```

  - Modify migration

```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');
        });
    }
};
```

- Migrate

```
INFO  Running migrations.

2025_05_06_204307_add_image_to_m_barang_table ..... 61ms DONE
```

- Edit controller

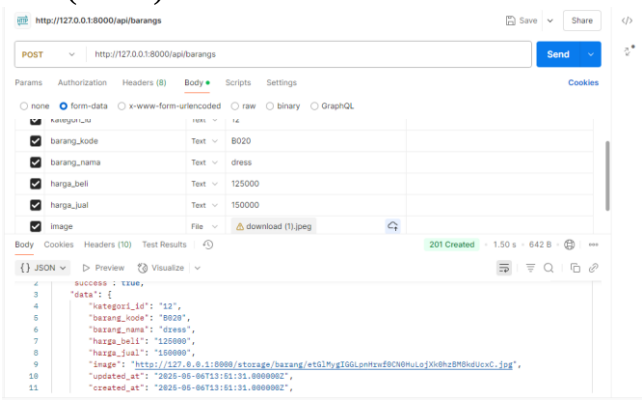
```php
<?php

namespace App\Http\Controllers\Api;

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

class BarangController extends Controller
{
    public function index()
    {
        $barang = BarangModel::all();
        return response()->json([
            'success' => true,
            'data' => $barang,
        ]);
    }

    public function store(Request $request)
    {
        // Validasi
        $validator = Validator::make($request->all(), [
            'kategori_id'  => 'required|exists:m_kategori,kategori_id',
            'barang_kode'  => 'required|unique:m_barang,barang_kode',
            'barang_nama'  => 'required',
            'harga_beli'   => 'required|numeric',
            'harga_jual'   => 'required|numeric',
            'image'        => 'nullable|image|mimes:jpeg,png,jpg,gif,svg|max:2048',
        ]);

        if ($validator->fails()) {
            return response()->json($validator->errors(), 422);
        }

        // Upload image kalau ada
        $imageName = null;
        if ($request->hasFile('image')) {
            $imageName = $request->image->hashName();
            $request->image->storeAs('public/barang', $imageName);
        }

        $barang = BarangModel::create([
            'kategori_id' => $request->kategori_id,
            'barang_kode' => $request->barang_kode,
            'barang_nama' => $request->barang_nama,
            'harga_beli'  => $request->harga_beli,
            'harga_jual'  => $request->harga_jual,
            'image'       => $imageName,
        ]);

        return response()->json([
            'success' => true,
            'data' => $barang,
        ], 201);
    }

    public function show($id)
    {
        $barang = BarangModel::find($id);

        if (!$barang) {
            return response()->json([
                'success' => false,
                'message' => 'Barang tidak ditemukan',
            ], 404);
        }
```

```php
        return response()->json([
            'success' => true,
            'data' => $barang,
        ]);
    }

    public function update(Request $request, $id)
    {
        $barang = BarangModel::find($id);

        if (!$barang) {
            return response()->json([
                'success' => false,
                'message' => 'Barang tidak ditemukan',
            ], 404);
        }

        // Validasi
        $validator = Validator::make($request->all(), [
            'kategori_id'  => 'sometimes|required|exists:m_kategori,kategori_id',
            'barang_kode'  => 'sometimes|required|unique:m_barang,barang_kode,' . $id . ',barang_id',
            'barang_nama'  => 'sometimes|required',
            'harga_beli'   => 'sometimes|required|numeric',
            'harga_jual'   => 'sometimes|required|numeric',
            'image'        => 'nullable|image|mimes:jpeg,png,jpg,gif,svg|max:2048',
        ]);

        if ($validator->fails()) {
            return response()->json($validator->errors(), 422);
        }

        // Upload image kalau ada
        if ($request->hasFile('image')) {
            $imageName = $request->image->hashName();
            $request->image->storeAs('public/barang', $imageName);
            $barang->image = $imageName;
        }

        $barang->update($request->except('image')); // Kecuali image, karena sudah dihandle manual
        $barang->save();

        return response()->json([
            'success' => true,
            'data' => $barang,
        ]);
    }

    public function destroy($id)
    {
        $barang = BarangModel::find($id);

        if (!$barang) {
            return response()->json([
                'success' => false,
                'message' => 'Barang tidak ditemukan',
            ], 404);
        }

        $barang->delete();

        return response()->json([
            'success' => true,
            'message' => 'Barang berhasil dihapus',
        ]);
    }
}
```

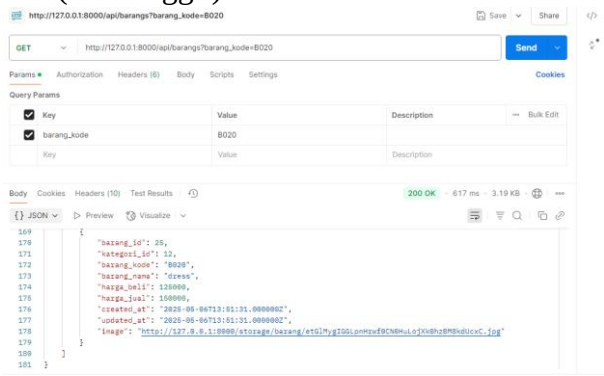
- Edit barangmodel

```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
        );
    }
}
```

- Post (create)



- Get (memanggil)

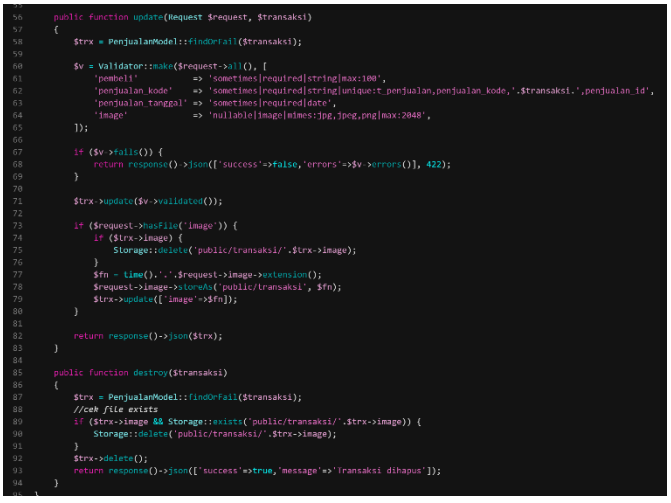


- Transaksi
  - Make controller

```
   INFO  Controller [C:\laragon\www\PWL_2025\Week11\Jobsheet\app\Http\Controllers\Api\PenjualanController.php] created s
uccessfully.
```

  - Modify controller

```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);
```



```php
    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']);
    }
}
```

- Make migration for image

```
C:\laragon\www\PWL_2025\Week11\Jobsheet>php artisan make:migration add_image_to_t_penjualan_table

   INFO  Migration [C:\laragon\www\PWL_2025\Week11\Jobsheet\database\migrations/2025_05_06_210011_add_image_to_t_penjual
an_table.php] created successfully.
```

- Modify migration

```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');
        });
    }

};
```

- Migrate

```
   INFO  Running migrations.

  2025_05_06_210011_add_image_to_t_penjualan_table ..................................................... 62ms DONE
```

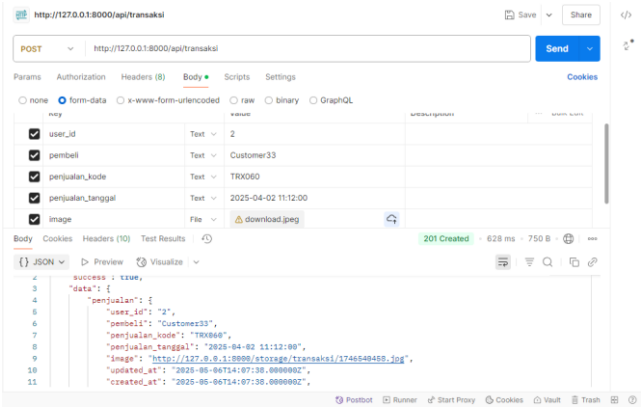
- Edit penjualanmodel

```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
        );
    }
}
```

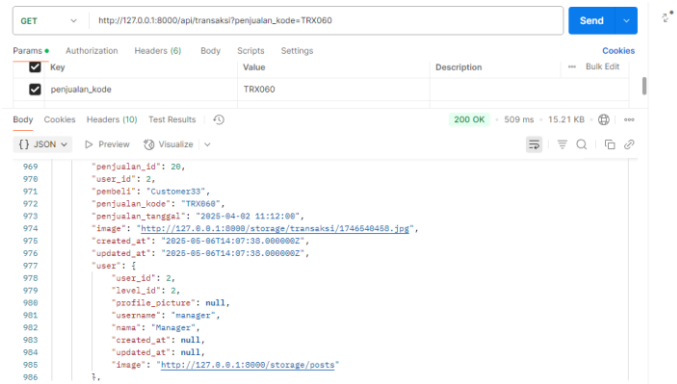
- Edit routes in API

```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']);
```

- Post (create)



- Get(memanggil)



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