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	:	Karina Ika Indasa

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`

```
C:\laragon\www\PWL_2025\Week11\JS11>php artisan make:migration add_image_to_m_user_table

  INFO  Migration [C:\laragon\www\PWL_2025\Week11\JS11\database\migrations/2025_05_04_121024_add_image_to_m_user_table.
php] created successfully.
```

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

JS11 > database > migrations > 2025_05_04_121024_add_image_to_m_user_table.php > ...

```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(table: 'm_user', callback: function (Blueprint $table): void {
            $table->string(column: 'image');
        });
    }

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

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

```
C:\laragon\www\PWL_2025\Week11\JS11>php artisan migrate

  INFO  Running migrations.

  2025_05_04_121024_add_image_to_m_user_table ..... 131ms DONE
```

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

JS11 > app > Models > UserModel.php > PHP > UserModel

```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;
use Tymon\JWTAuth\Contracts\JWTSubject;
use Illuminate\Database\Eloquent\Casts\Attribute;

43 references | 0 implementations
class UserModel extends Authenticatable implements JWTSubject
{
    use HasFactory;

    0 references | 0 overrides
    public function getJWTIdentifier(): mixed
    {
        return $this->getKey();
    }

    0 references | 0 overrides
    public function getJWTCustomClaims(): array
    {
        return [];
    }

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

    0 references
    protected $hidden = ['password'];

    0 references
    protected $casts = ['password' => 'hashed'];
```

```php
    /**
     * Relasi ke tabel level
     */
    0 references | 0 overrides
    public function level(): BelongsTo
    {
        return $this->belongsTo(related: LevelModel::class, foreignKey: 'level_id', ownerKey: 'level_id');
    }

    /**
     * Mendapatkan nama role
     */
    0 references | 0 overrides
    public function getRoleName(): string
    {
        return $this->level->level_nama;
    }

    /**
     * Cek apakah user memiliki role tertentu
     */
    0 references | 0 overrides
    public function hasRole($role): bool
    {
        return $this->level->level_kode == $role;
    }

    /**
     * Mendapatkan nama role
     */
    0 references | 0 overrides
    public function getRole(): mixed
    {
        return $this->level->level_kode;
    }

    0 references | 0 overrides
    public function getProfilePictureUrlAttribute(): string
    {
        return $this->profile_picture
            ? asset(path: 'storage/profile_pictures/' . $this->profile_picture)
            : asset(path: 'images/default-profile.png');
    }
    0 references | 0 overrides
    protected function image(): Attribute
    {
        return Attribute::make(
            get: fn($image): string|UrlGenerator => url(path: '/storage/posts/' . $image),
        );
    }
}
```

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;
use Illuminate\Support\Facades\Validator;

3 references | 0 implementations
class RegisterController extends Controller
{
    0 references | 0 overrides
    public function __invoke(Request $request): JsonResponse|mixed
    {
        // Set validation
        $validator = Validator::make(data: $request->all(), rules: [
            '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(data: $validator->errors(), status: 422);
        }

        // Create user
        $user = UserModel::create(attributes: [
            'username' => $request->username,
            'nama' => $request->nama,
            'password' => bcrypt(value: $request->password),
            'level_id' => $request->level_id,
            'image' => $request->image
        ]);

        // Return response JSON if user is created
        if ($user) {
            return response()->json(data: [
                'success' => true,
                'user' => $user,
            ], status: 201);
        }

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

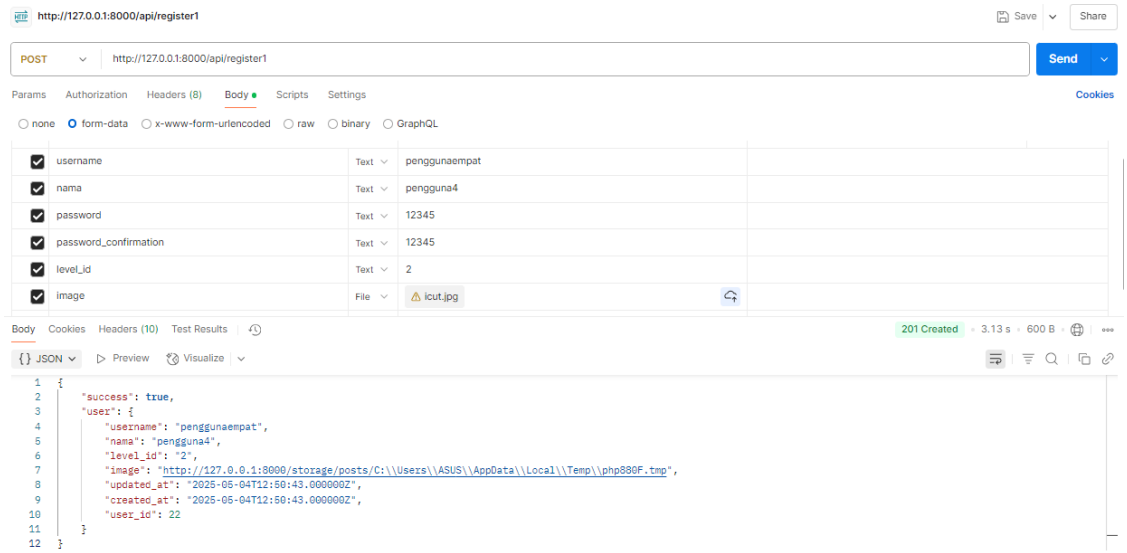
7. Anda dapat menambahkan detail untuk spesifikasi image pada validator

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

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

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

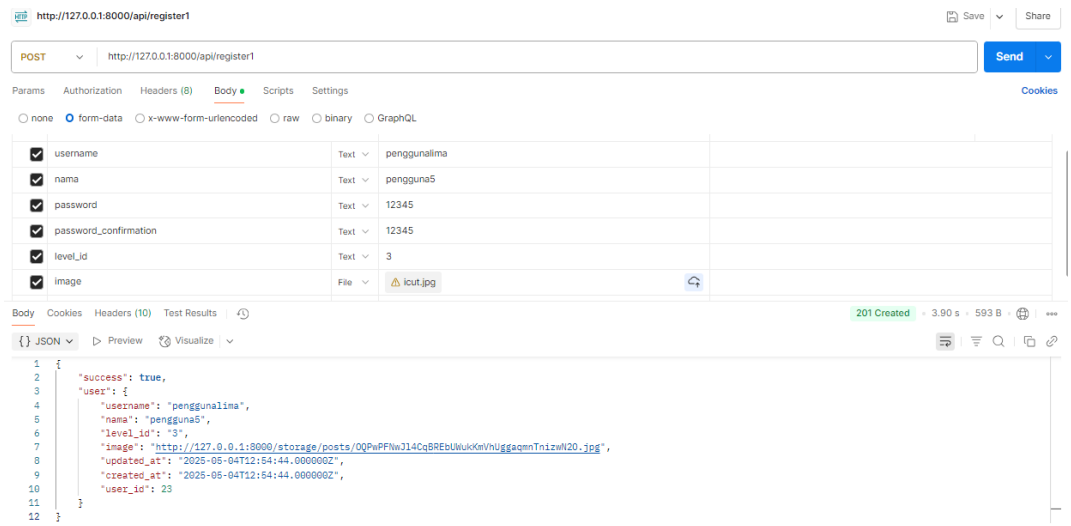
9. Simpan dan akses pada aplikasi Postman, atur pada Body isi manual Key dan Valuenya pada Key image tambahkan value File dan upload gambar
http://127.0.0.1:8000/api/register1 dengan method POST dan klik send



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

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

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



Perbedaan:

- Sebelum `"image": "http://127.0.0.1:8000/storage/posts/C:\\Users\\ASUS\\AppData\\Local\\Temp\\php880F.tmp",`
- Sesudah `"image": "http://127.0.0.1:8000/storage/posts/OQPwPFNwJl4CqBREbUWukKmVhUggaqmnTnizwN2O.jpg",`

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. Implementasi Eloquent Accessor m_barang
   - Membuat migration

```
C:\laragon\www\PWL_2025\Week11\JS11>php artisan make:migration add_image_to_m_barang_table

  INFO  Migration [C:\laragon\www\PWL_2025\Week11\JS11\database\migrations/2025_05_04_130820_add_image_to_m_barang_table.php] created successfully.
```

   - Tambahkan kode pada migration yang sudah dibuat

```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(table: 'm_barang', callback: function (Blueprint $table): void {
            $table->string(column: 'image')->nullable(); // boleh null
        });
    }

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

   - Jalankan migration

```
C:\laragon\www\PWL_2025\Week11\JS11>php artisan migrate

  INFO  Running migrations.

  2025_05_04_130820_add_image_to_m_barang_table ..... 446ms DONE
```

   - Tambahkan kode pada BarangModel.php

```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(): mixed{
        return $this->getKey();
    }

    public function getJWTCustomClaims(): array{ 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(related: KategoriModel::class, foreignKey: 'kategori_id', ownerKey: 'kategori_id');
    }

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

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

- Tambahkan pada Api\BarangController.php

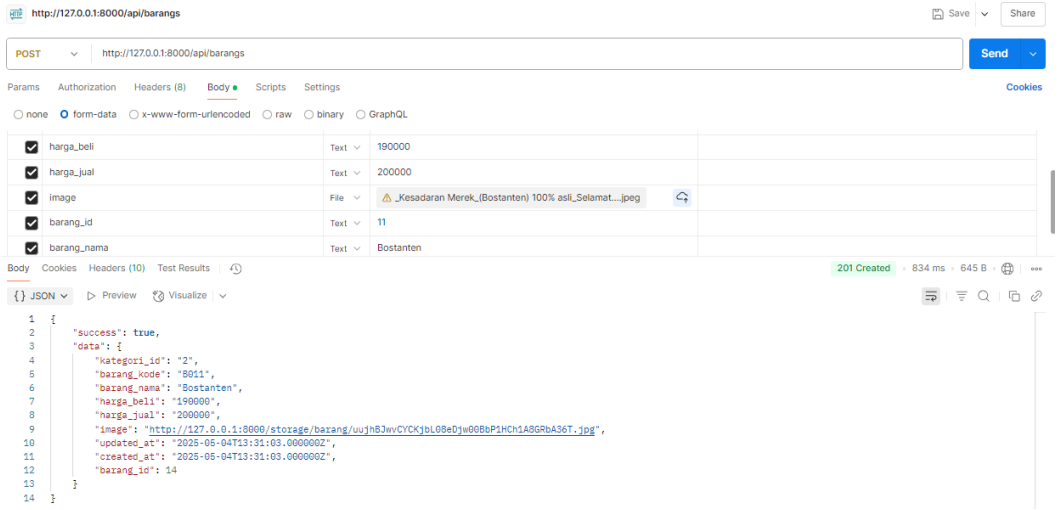
```php
    public function destroy($id): JsonResponse|mixed
    {
        $barang = BarangModel::find(id: $id);

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

        $barang->delete();

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

- Hasil
  POST



2. Implementasi Eloquent Accessor transaksi
   - Membuat controller

```
C:\laragon\www\PWL_2025\Week11\JS11>php artisan make:controller Api/PenjualanController

  INFO  Controller [C:\laragon\www\PWL_2025\Week11\JS11\app\Http\Controllers\Api\PenjualanController.php] created successfully.
```

   - Tambahkan kode pada PenjualanController

```php
JS11 > app > Http > Controllers > Api > PenjualanController.php > PHP > PenjualanController
<?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;

0 references | 0 implementations
class PenjualanController extends Controller
{
    0 references | 0 overrides
    public function index(): JsonResponse|mixed
    {
        $data = PenjualanModel::with(relations: ['user', 'details.barang'])->get();
        return response()->json(data: $data);
    }

    0 references | 0 overrides
    public function show($transaksi): JsonResponse|mixed
    {
        $penjualan = PenjualanModel::with(relations: ['user', 'details.barang'])->findOrFail(id: $transaksi);
        return response()->json(data: $penjualan);
    }
```

```php
0 references | 0 overrides
public function store(Request $request): JsonResponse|mixed
{
    $v = Validator::make(data: $request->all(), rules: [
        '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 ($v->fails()) {
        return response()->json(data: ['success'=>false,'errors'=>$v->errors()], status: 422);
    }

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

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

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

```php
0 references | 0 overrides
public function update(Request $request, $transaksi): JsonResponse|mixed
{
    $trx = PenjualanModel::findOrFail(id: $transaksi);

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

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

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

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

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

0 references | 0 overrides
public function destroy($transaksi): JsonResponse|mixed
{
    $trx = PenjualanModel::findOrFail(id: $transaksi);
    //cek file exists
    if ($trx->image && Storage::exists(path: 'public/transaksi/'.$trx->image)) {
        Storage::delete(paths: 'public/transaksi/'.$trx->image);
    }
    $trx->delete();
    return response()->json(data: ['success'=>true,'message'=>'Transaksi dihapus']);
}
}
```

- Membuat migration

```
C:\laragon\www\PWL_2025\Week11\JS11>php artisan make:migration add_image_to_t_prenjualan_table

  INFO  Migration [C:\laragon\www\PWL_2025\Week11\JS11\database\migrations/2025_05_04_134451_add_image_to_t_prenjualan_table.php]
created successfully.
```

- Tambahkan kode pada migration yang sudah dibuat

JS11 > database > migrations > 2025_05_04_134451_add_image_to_t_prenjualan_table.php > ...

```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(table: 't_penjualan', callback: function (Blueprint $table): void {
            $table->string(column: 'image')->nullable()->after(column: 'penjualan_tanggal');
        });
    }

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

};
```

- Jalankan migration

```
C:\laragon\www\PWL_2025\Week11\JS11>php artisan migrate

  INFO  Running migrations.

  2025_05_04_134451_add_image_to_t_prenjualan_table ..... 729ms DONE
```

- Tambahkan kode pada PenjualanModel.php

JS11 > app > Models > PenjualanModel.php > PHP Intelephense > PenjualanModel > user

```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;

20 references | 0 implementations
class PenjualanModel extends Model
{

    0 references | 0 overrides
    public function getJWTIdentifier(): mixed{
        return $this->getKey();
    }

    0 references | 0 overrides
    public function getJWTCustomClaims(): array{
        return [];
    }
```

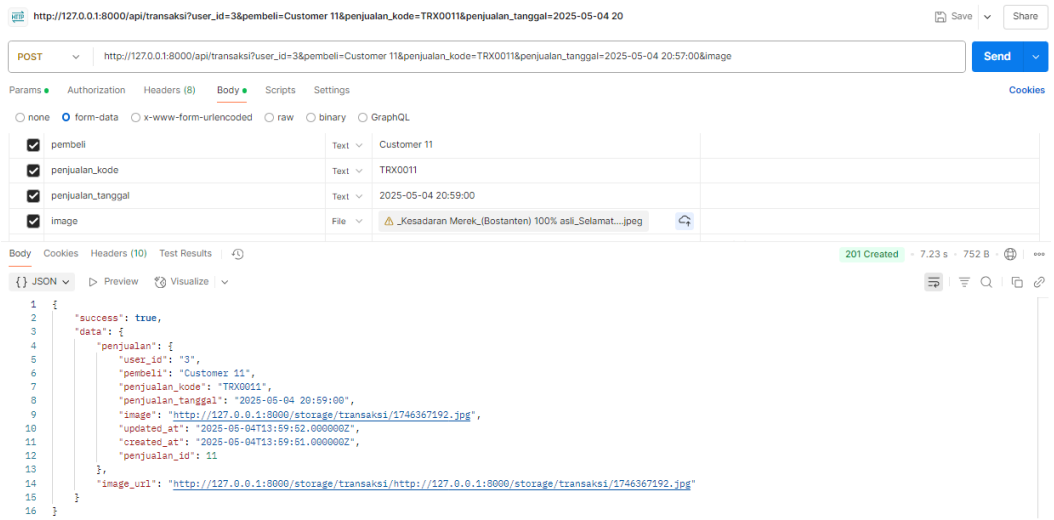
- Tambahkan kode pada routes api.php

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

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

- Hasil

POST



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