

**LAPORAN PRAKTIKUM**

**JOBSHEET 11**

**RESTFUL API 2**

Disusun untuk memenuhi nilai tugas

Mata Kuliah : Pemrograman Web Lanjut



Oleh :

Aqueena Regita Hapsari

2341760096

SIB 2B

03

**PROGRAM STUDI D-IV SISTEM INFORMASI BISNIS**

**POLITEKNIK NEGERI MALANG**

**TAHUN AJARAN 2024/2025**

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 di table 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_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.
```

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

```

<?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 D:\LARAGON\laragon\www\PWL2025\Minggu11> php artisan migrate

  INFO  Running migrations.

 2025_04_28_095410_add_image_to_m_
user_table  24ms  DONE

```

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

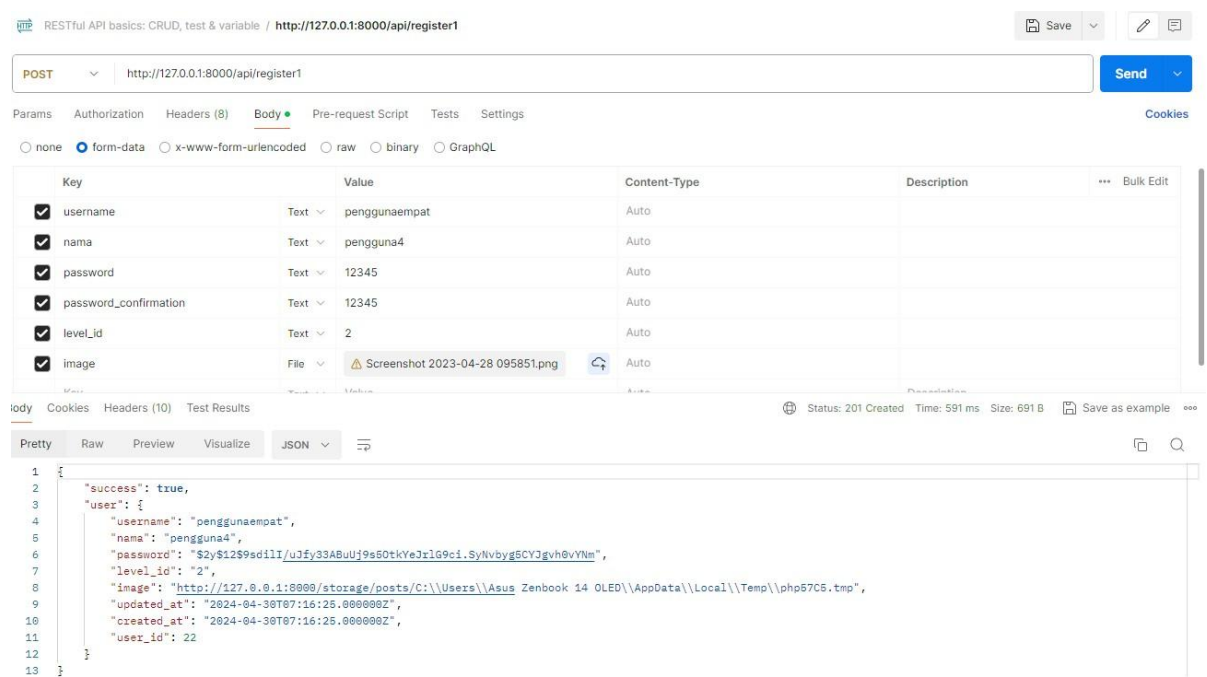
```
'image' => 'required|image|mimes:jpeg,png,jpg,gif,svg|max:2048',
```

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

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

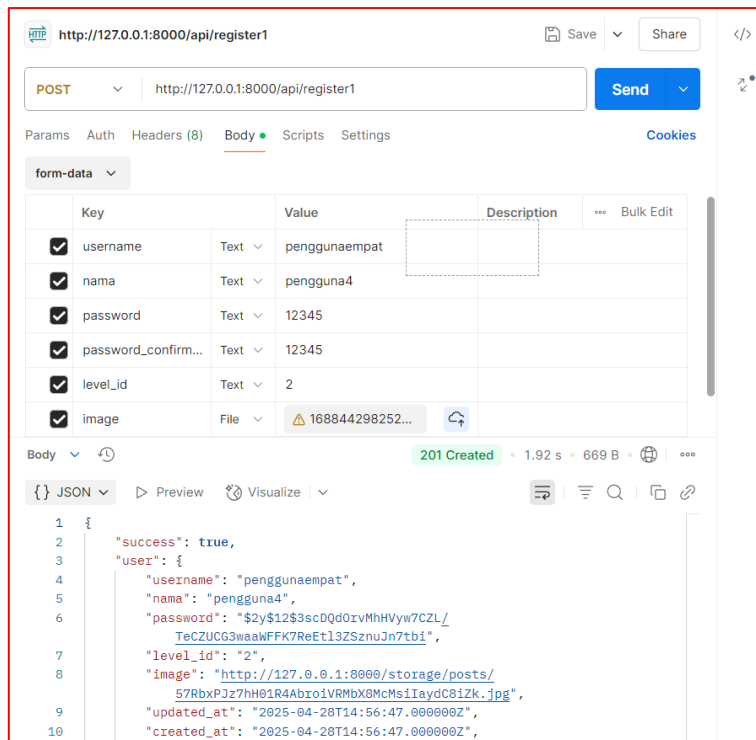
8. Simpan dan akses pada aplikasi Postman, atur pada Body isi manual Key dan Valuenya pada Key image tambahkan value File dan upload gambar

<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(),

// harus ambil dr upload dulu baru bisa pakai $image->hashName()
// upload image
$image = $request->file('image');
$image->storeAs('public/posts', $image->hashName());
//create user
$user = UserModel::create([
    'username' => $request->username,
    'nama' => $request->nama,
    'password' => bcrypt($request->password),
    'level_id' => $request->level_id,
    'image' => $image->hashName()
    // $image->hashName() tidak bisa karena Variabel $image
    belum pernah didefinisikan sbmlnya
]);
```

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

**Sebelum :** Ini langsung ambil value biasa dari form-data tanpa benar-benar meng-upload file ke server. Jadi file gambarnya **tidak** disimpan ke folder storage, cuma namanya saja yang disimpan ke database. File gambar tidak benar-benar ada di server → saat mau load gambarnya nanti, gambar tidak ketemu (404 not found) dan mudah jadi penyebab error saat apk jalan. Maka dari itu modifikasi menjadi hashName(). Ini **upload file gambarnya** ke server **di folder storage/app/public/posts**. Lalu di database simpan nama file uniknya (hasil dari hashName()) untuk menghindari nama file kembar. Dan saat diakses url('/storage/posts/'.\$image), gambarnya bisa tampil.

HTTP <http://127.0.0.1:8000/api/register1> Save Share </>

POST <http://127.0.0.1:8000/api/register1> Send

Params Auth Headers (8) **Body** Scripts Settings Cookies

form-data

	Key	Value	Description	...	Bulk Edit
<input checked="" type="checkbox"/>	username	Text	penggunaempat		
<input checked="" type="checkbox"/>	nama	Text	pengguna4		
<input checked="" type="checkbox"/>	password	Text	12345		
<input checked="" type="checkbox"/>	password_confirm...	Text	12345		
<input checked="" type="checkbox"/>	level_id	Text	2		
<input checked="" type="checkbox"/>	image	File	168844298252...		

Body [Visualize](#) 201 Created = 1.92 s = 669 B

```

1 {
2   "success": true,
3   "user": {
4     "username": "penggunaempat",
5     "nama": "pengguna4",
6     "password": "$2y$12$3scDQd0rvMhHVyw7CZL/
7       TeCZUCG3waawFFK7ReEt13ZSznJn7tbi",
8     "level_id": "2",
9     "image": "http://127.0.0.1:8000/storage/posts/
10      57RbxPJz7hH01R4Abro1VRMbX8McMsiIaydC8iZk.jpg",
11     "updated_at": "2025-04-28T14:56:47.000000Z",
12     "created_at": "2025-04-28T14:56:47.000000Z",
13   }
14 }
```

**Sesudah :**

HTTP <http://127.0.0.1:8000/api/register1> Save Share

POST <http://127.0.0.1:8000/api/register1> Send

Params Auth Headers (8) **Body** Scripts Settings Cookies

form-data

	Key	Value	Description	...	Bulk Edit
<input checked="" type="checkbox"/>	username	Text	penggunatiga		
<input checked="" type="checkbox"/>	nama	Text	pengguna3		
<input checked="" type="checkbox"/>	password	Text	12345		
<input checked="" type="checkbox"/>	password_confirm...	Text	12345		

Body [Visualize](#) 201 Created = 2.21 s = 667 B

```

1 {
2   "success": true,
3   "user": {
4     "username": "penggunatiga",
5     "nama": "pengguna3",
6     "password": "$2y$12$eVUm00hKzxksTzBJIVZn50PE3w3zEaVdTpbKpM2XRPx15YU9uwcHK",
7     "level_id": "3",
8     "image": "http://127.0.0.1:8000/storage/posts/
9       eMLwn0yZccVH0jqMbaboGnAFDgKl6eizLc1AHZVu.jpg",
10     "updated_at": "2025-04-28T15:19:32.000000Z",
11     "created_at": "2025-04-28T15:19:32.000000Z",
12     "user_id": 30
13   }
14 }
```

## 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.

**Jawaban :**

## 1. m\_barang pada transaksi

### a. Api/BarangController

```
<?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);
    }

    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',
    });
}
}

```

## b. BarangModel

```

<?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'
    ];

    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
        );
    }
}

```

### c. Make migration add\_image\_to\_m\_barang\_table

```

<?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');
        });
    }
};

```

### d. PenjualanModel

```

<?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\PenjualanDetailModel;

class PenjualanModel extends Model

```

```

{
    use HasFactory;

    protected $table = 't_penjualan';
    protected $primaryKey = 'penjualan_id';
    protected $guarded = [];

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

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

```

#### e. DetailPenjualan

```

<?php

namespace App\Models;

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

class DetailPenjualan extends Model
{
    use HasFactory;

    protected $table = 't_penjualan_detail'; // ← ganti ini
    protected $primaryKey = 'detail_id';

    protected $fillable = [
        'penjualan_id',
        'barang_id',
        'harga',
        'jumlah',
    ];

    public function barang(): BelongsTo
    {
        return $this->belongsTo(Barang::class, 'barang_id', 'barang_id');
    }
}

```



```
}
```

## f. Penjualan

```
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Model;
use Illuminate\Database\Eloquent\Relations\HasMany;

class Penjualan extends Model
{
    use HasFactory;

    protected $table = 't_penjualan'; // ← ganti ini
    protected $primaryKey = 'penjualan_id';

    protected $fillable = [
        'user_id',
        'pembeli',
        'penjualan_kode',
        'penjualan_tanggal',
    ];

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

## g. Api/PenjualanController

```
<?php

namespace App\Http\Controllers\Api;

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

class PenjualanController extends Controller
{
    // POST - Buat Transaksi
    public function store(Request $request)
    {

```

```

// Validasi input
$validator = Validator::make($request->all(), [
    'user_id' => 'required',
    'pembeli' => 'required',
    'penjualan_kode' => 'required',
    'penjualan_tanggal' => 'required|date',
    'detail' => 'required|array',
    'detail.*.barang_id' => 'required|exists:m_barang,barang_id',
    'detail.*.harga' => 'required|numeric',
    'detail.*.jumlah' => 'required|numeric',
]);

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

// Simpan Penjualan
$penjualan = Penjualan::create([
    'user_id' => $request->user_id,
    'pembeli' => $request->pembeli,
    'penjualan_kode' => $request->penjualan_kode,
    'penjualan_tanggal' => $request->penjualan_tanggal,
]);

// Simpan Detail Penjualan
foreach ($request->detail as $item) {
    DetailPenjualan::create([
        'penjualan_id' => $penjualan->penjualan_id,
        'barang_id' => $item['barang_id'],
        'harga' => $item['harga'],
        'jumlah' => $item['jumlah'],
    ]);
}

return response()->json([
    'success' => true,
    'message' => 'Transaksi berhasil disimpan',
    'data' => $penjualan
], 201);
}

// GET - List Semua Penjualan
public function index()
{
    $penjualan = Penjualan::with(['details.barang'])->get();
}

```

```

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

    // GET - Detail Penjualan by ID
    public function show($id)
    {
        $penjualan = Penjualan::with(['details.barang'])->find($id);

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

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

```

### Penjelasan Singkat :

Pada PenjualanController, terdapat tiga fungsi utama yang digunakan untuk mengelola transaksi penjualan.

Fungsi store(Request \$request) digunakan untuk membuat transaksi baru. Proses diawali dengan melakukan validasi input, memastikan semua data penting seperti user\_id, pembeli, penjualan\_kode, penjualan\_tanggal, serta array detail barang yang dijual sudah diisi dengan benar. Setiap item detail juga divalidasi untuk memastikan barang yang dijual terdaftar dalam database. Setelah validasi berhasil, data transaksi penjualan disimpan ke tabel t\_penjualan. Kemudian, setiap item barang dalam transaksi disimpan ke tabel t\_penjualan\_detail yang berelasi dengan transaksi tersebut. Setelah semua data berhasil disimpan, sistem akan mengirimkan respon JSON yang menandakan bahwa transaksi berhasil dibuat.

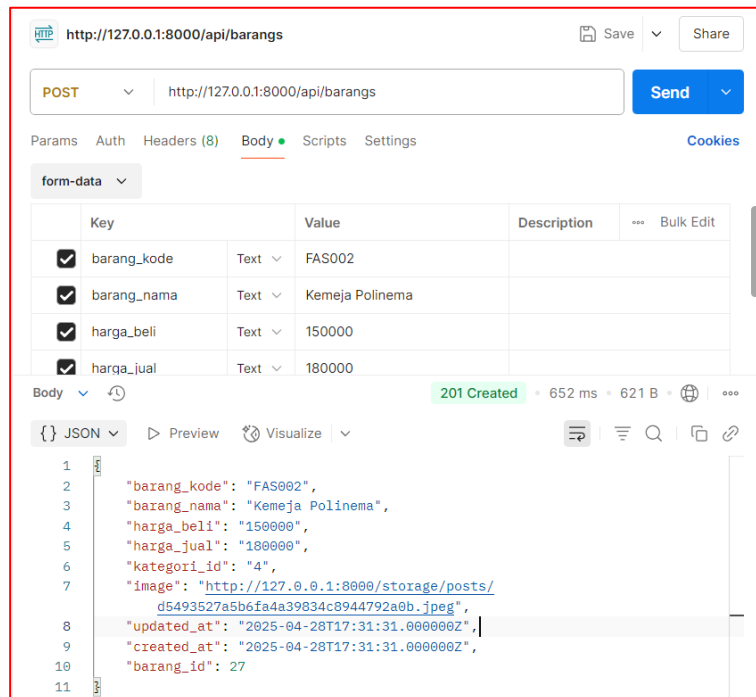
Fungsi index() berfungsi untuk menampilkan semua transaksi penjualan yang sudah ada. Data transaksi yang diambil mencakup juga relasi ke tabel detail penjualan (details) dan relasi ke tabel barang (barang) yang terkait. Data dikembalikan dalam bentuk JSON untuk memudahkan konsumsi API oleh frontend atau aplikasi lain.

Fungsi show(\$id) digunakan untuk menampilkan detail satu transaksi penjualan berdasarkan ID tertentu. Fungsi ini juga membawa data detail barang yang dijual dalam transaksi tersebut. Jika transaksi tidak ditemukan berdasarkan ID yang diberikan, maka

sistem akan mengembalikan respon error 404 yang menunjukkan bahwa data tidak ditemukan.

Secara umum, alur kerja dalam PenjualanController memastikan bahwa transaksi baru dapat disimpan dengan aman dan lengkap, serta data transaksi dapat ditampilkan baik dalam bentuk daftar seluruh transaksi maupun detail transaksi tertentu.

### Hasil :



http://127.0.0.1:8000/api/barangs

POST http://127.0.0.1:8000/api/barangs Send

Params Auth Headers (8) Body Scripts Settings Cookies

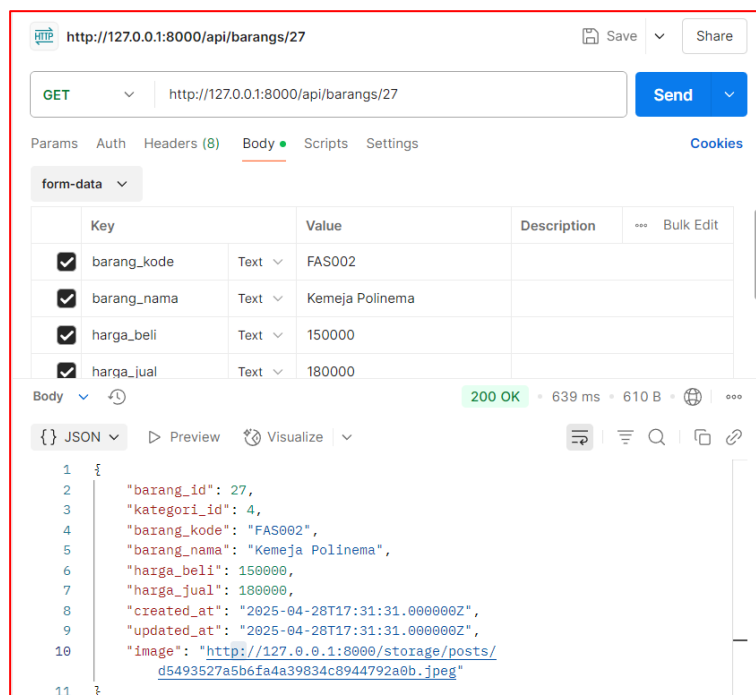
form-data

	Key	Value	Description	Bulk Edit
<input checked="" type="checkbox"/>	barang_kode	Text FAS002		
<input checked="" type="checkbox"/>	barang_nama	Text Kemeja Polinema		
<input checked="" type="checkbox"/>	harga_beli	Text 150000		
<input checked="" type="checkbox"/>	harga_jual	Text 180000		

Body 201 Created • 652 ms • 621 B • Visualize

```

1 {
2   "barang_kode": "FAS002",
3   "barang_nama": "Kemeja Polinema",
4   "harga_beli": "150000",
5   "harga_jual": "180000",
6   "kategori_id": "4",
7   "image": "http://127.0.0.1:8000/storage/posts/d5493527a5b6fa4a39834c8944792a0b.jpeg",
8   "updated_at": "2025-04-28T17:31:31.000000Z",
9   "created_at": "2025-04-28T17:31:31.000000Z",
10  "barang_id": 27
11 }
```



http://127.0.0.1:8000/api/barangs/27

GET http://127.0.0.1:8000/api/barangs/27 Send

Params Auth Headers (8) Body Scripts Settings Cookies

form-data

	Key	Value	Description	Bulk Edit
<input checked="" type="checkbox"/>	barang_kode	Text FAS002		
<input checked="" type="checkbox"/>	barang_nama	Text Kemeja Polinema		
<input checked="" type="checkbox"/>	harga_beli	Text 150000		
<input checked="" type="checkbox"/>	harga_jual	Text 180000		

Body 200 OK • 639 ms • 610 B • Visualize

```

1 {
2   "barang_id": 27,
3   "kategori_id": 4,
4   "barang_kode": "FAS002",
5   "barang_nama": "Kemeja Polinema",
6   "harga_beli": 150000,
7   "harga_jual": 180000,
8   "created_at": "2025-04-28T17:31:31.000000Z",
9   "updated_at": "2025-04-28T17:31:31.000000Z",
10  "image": "http://127.0.0.1:8000/storage/posts/d5493527a5b6fa4a39834c8944792a0b.jpeg"
11 }
```

HTTP client interface showing a GET request to `http://127.0.0.1:8000/api/penjualan`. The response is a 200 OK status, indicating success. The response body is displayed in JSON format, showing a successful transaction record.

Query Params

Key	Value	Description	Bulk Edit
-----	-------	-------------	-----------

Body

200 OK • 1.53 s • 4.55 KB

JSON

```
1 {
2   "success": true,
3   "data": [
4     {
5       "penjualan_id": 1,
6       "user_id": 1,
7       "pembeli": "Andi",
8       "penjualan_kode": "TRX001",
9       "penjualan_tanggal": "2025-03-03 06:53:46",
10      "created_at": null,
11      "updated_at": null,
12      "details": [
13        {
14          "detail_id": 1,
15          "penjualan_id": 1,
16          "barang_id": 1,
17          "harga": 8000000,
18          "jumlah": 1,
19          "created_at": "2025-03-03T06:58:11.000000Z",
20          "updated_at": "2025-03-03T06:58:11.000000Z",
21          "barang": {
22            "barang_id": 1,
23            "kategori_id": 1,
```

HTTP client interface showing a GET request to `http://127.0.0.1:8000/api/penjualan/9`. The response is a 200 OK status, indicating success. The response body is displayed in JSON format, showing a successful transaction record for ID 9.

Query Params

Key	Value	Description	Bulk Edit
-----	-------	-------------	-----------

Body

200 OK • 1.10 s • 497 B

JSON

```
1 {
2   "success": true,
3   "data": {
4     "penjualan_id": 9,
5     "user_id": 3,
6     "pembeli": "Indah",
7     "penjualan_kode": "TRX009",
8     "penjualan_tanggal": "2025-02-23 06:53:46",
9     "created_at": null,
10    "updated_at": null,
11    "details": []
12  }
13 }
```

POST

http://localhost:8000/api/penjualan

Params

Authorization

Headers (10)

Body

Scripts

Settings

☐ none

☐ form-data

☐ x-www-form-urlencoded

☒ raw

☐ binary

☐ Grap

```
1 {
2   "user_id": 1,
3   "pembeli": "Pelanggan 10",
4   "penjualan_kode": "PJL010",
5   "penjualan_tanggal": "2025-04-28",
6   "detail": [
7     {
8       "barang_id": 59,
9       "harga": 6000,
10      "jumlah": 2
11    }
12  ]
13 }
```

Body

Cookies (2)

Headers (10)

Test Results

🔍

{ } JSON

Preview

Visualize

⌵

```
1 {
2   "success": true,
3   "message": "Transaksi berhasil disimpan",
4   "data": {
5     "user_id": 1,
6     "pembeli": "Pelanggan 10",
7     "penjualan_kode": "PJL010",
8     "penjualan_tanggal": "2025-04-28",
9     "updated_at": "2025-04-27T21:35:50.000000Z",
10    "created_at": "2025-04-27T21:35:50.000000Z",
11    "penjualan_id": 12
12  }
13 }
```

☐ Edit

☐ Copy

☐ Delete

12

1 Pelanggan 10 PJL010

2025-04-28 00:00:00

2025-04-27 21:35:50

2025-04-27 21:35:50