



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
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

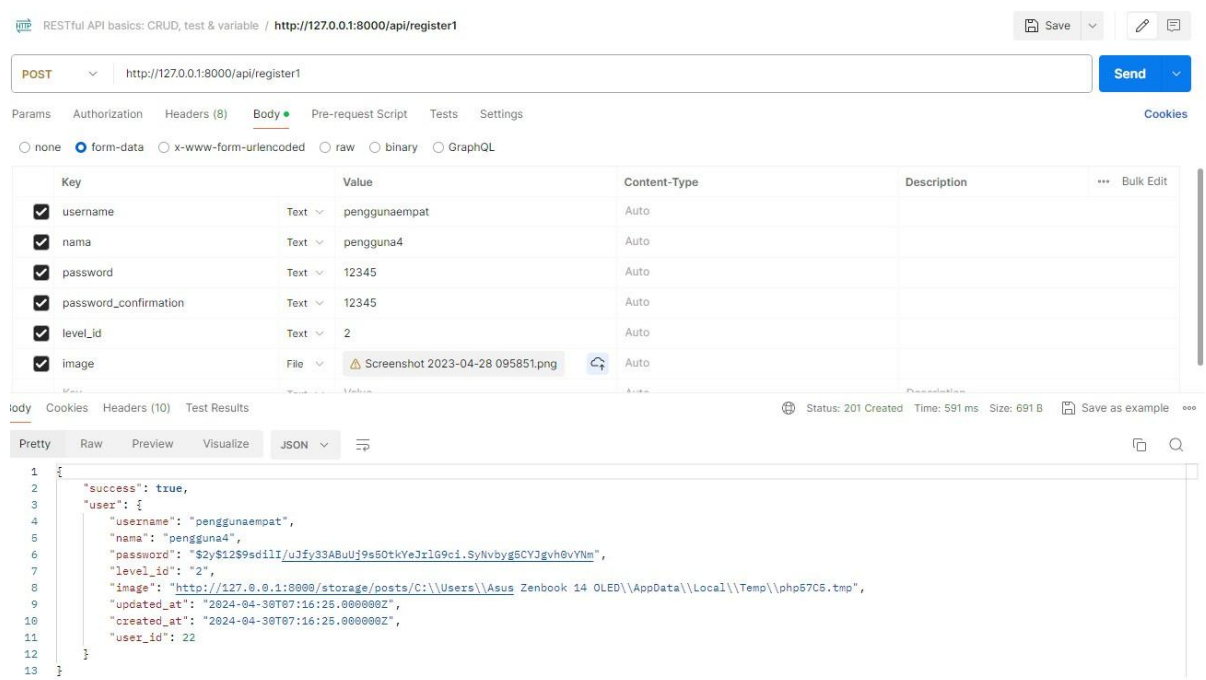
```
'image' => 'required|image|mimes:jpeg,png,jpg,gif,svg|max:2048',
```

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

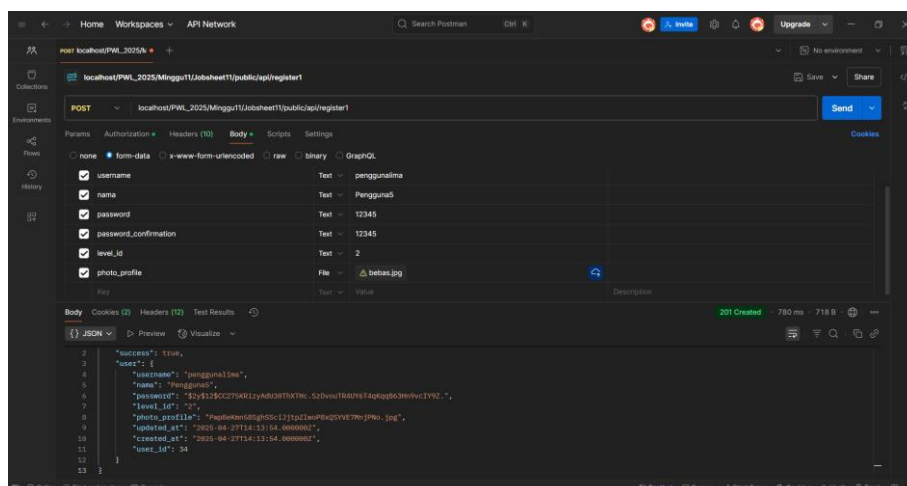
<http://127.0.0.1:8000/api/register1> dengan method POST dan klik send



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

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

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





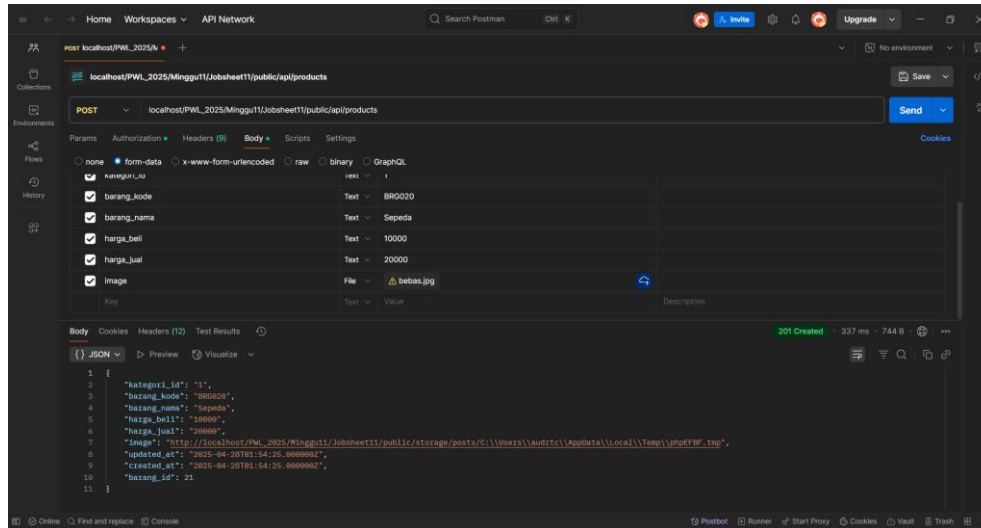
## TUGAS

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

```
database > migrations > 2025_04_28_013154_add_image_to_m_barang_table.php > class
1  <?php
2
3  use Illuminate\Database\Migrations\Migration;
4  use Illuminate\Database\Schema\Blueprint;
5  use Illuminate\Support\Facades\Schema;
6
7  return new class extends Migration
8  {
9      /**
10       * Run the migrations.
11       */
12       public function up(): void
13       {
14           Schema::table('m_barang', function (Blueprint $table) {
15               $table->string('image');
16           });
17       }
18
19       /**
20        * Reverse the migrations.
21        */
22       public function down(): void
23       {
24           Schema::table('m_barang', function (Blueprint $table) {
25               $table->dropColumn('image');
26           });
27       }
28  };
```

```
1  <?php
2
3  namespace App\Models;
4
5  use Illuminate\Database\Eloquent\Casts\Attribute;
6  use Illuminate\Database\Eloquent\Factories\HasFactory;
7  use Illuminate\Database\Eloquent\Model;
8  use Illuminate\Database\Eloquent\Relations\BelongsTo;
9
10 class BarangModel extends Model
11 {
12     use HasFactory;
13
14     protected $table = 'm_barang';
15     protected $primaryKey = 'barang_id';
16     protected $fillable = ['kategori_id', 'barang_kode', 'barang_nama', 'harga_beli', 'harga_jual', 'image'];
17
18     public function kategori(): BelongsTo
19     {
20         return $this->belongsTo(KategoriModel::class, 'kategori_id', 'kategori_id');
21     }
22
23     public function stok()
24     {
25         return $this->hasOne(StokModel::class, 'barang_id', 'barang_id');
26     }
27
28     public function image(): Attribute
29     {
30         return Attribute::make(
31             get: fn ($image) => url('/storage/posts/'.$image),
```



```
{
  "barang_id": 21,
  "kategori_id": 1,
  "barang_kode": "BRG020",
  "barang_nama": "Sepeda",
  "harga_beli": 10000,
  "harga_jual": 20000,
  "created_at": "2025-04-28T01:54:25.000000Z",
  "updated_at": "2025-04-28T01:54:25.000000Z",
  "image": "http://localhost/PWL_2025/Minggu11/Jobsheet11/public/storage/posts/C:\\Users\\audrtc\\AppData\\Local\\Temp\\phpEFBF.tmp"
}
```

## t\_penjualan

```
1 <?php
2
3 use Illuminate\Database\Migrations\Migration;
4 use Illuminate\Database\Schema\Blueprint;
5 use Illuminate\Support\Facades\Schema;
6
7 return new class extends Migration
8 {
9     /**
10      * Run the migrations.
11      */
12     public function up(): void
13     {
14         Schema::table('t_penjualan', function (Blueprint $table) {
15             $table->string('image');
16         });
17     }
18
19     /**
20      * Reverse the migrations.
21      */
22     public function down(): void
23     {
24         Schema::table('t_penjualan', function (Blueprint $table) {
25             $table->dropColumn('image');
26         });
27     }
28 };
```





```
1 <?php
2
3 namespace App\Models;
4
5 use Illuminate\Database\Eloquent\Casts\Attribute;
6 use Illuminate\Database\Eloquent\Factories\HasFactory;
7 use Illuminate\Database\Eloquent\Model;
8 use Illuminate\Database\Eloquent\Relations\BelongsTo;
9 use Illuminate\Database\Eloquent\Relations\HasMany;
10
11 class PenjualanModel extends Model
12 {
13     use HasFactory;
14
15     protected $table = 't_penjualan';
16     protected $primaryKey = 'penjualan_id';
17     protected $fillable = ['user_id', 'pembeli', 'penjualan_kode', 'penjualan_tanggal', 'image'];
18     protected $casts = ['penjualan_tanggal' => 'datetime', 'created_at' => 'datetime', 'updated_at' => 'datetime'];
19     public $timestamps = true;
20
21     public function user(): BelongsTo
22     {
23         return $this->belongsTo(UserModel::class, 'user_id', 'user_id');
24     }
25
26     public function penjualanDetail(): HasMany
27     {
28         return $this->hasMany(PenjualanDetailModel::class, 'penjualan_id', 'penjualan_id');
29     }
30
31     public function image(): Attribute
32     {
33         return Attribute::make(
34             get: fn ($image) => url('/storage/posts/'.$image),
35         );
36     }
37 }
```

```
Route::get('sales', [PenjualanController::class, 'index']);
Route::post('sales', [PenjualanController::class, 'store']);
Route::get('sales/{penjualan}', [PenjualanController::class, 'show']);
Route::put('sales/{penjualan}', [PenjualanController::class, 'update']);
Route::delete('sales/{penjualan}', [PenjualanController::class, 'destroy']);
```

```
1 <?php
2
3 namespace App\Http\Controllers\Api;
4
5 use App\Http\Controllers\Controller;
6 use App\Models\PenjualanModel;
7 use Illuminate\Http\Request;
8
9 class PenjualanController extends Controller
10 {
11     public function index()
12     {
13         return PenjualanModel::all();
14     }
15     public function store(Request $request)
16     {
17         $penjualan = PenjualanModel::create($request->all());
18         return response()->json($penjualan, 201);
19     }
20
21     public function show($penjualan)
22     {
23         return PenjualanModel::find($penjualan);
24     }
25
26     public function update(Request $request, $penjualan)
27     {
28         $data = PenjualanModel::find($penjualan);
29         $data->update($request->all());
30         return PenjualanModel::find($penjualan);
31     }
32
33     public function destroy($penjualan)
34     {
35         $data = PenjualanModel::find($penjualan);
36         $data->delete();
37         return response()->json([
38             'success' => true,
39             'message' => 'Data terhapus',
40         ]);
41     }
42 }
```

---

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