JOBSHEET 11

LAPORAN TUGAS

Diajukan untuk Memenuhi Tugas
Pemograman Web Lanjut

Jurusan:

TEKNOLOGI INFORMASI

Program Studi:

TEKNIK INFORMATIKA

Disusun oleh

1. Afrizal Dwi Septian (2241720122) / 01 / 2H

POLITEKNIK NEGERI MALANG
TEKNIK INFORMATIKA

2024

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`

```
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.
```

```
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_05_15_055039_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');
        });
    }
};
```

```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`

```
C:\laragon\www\PWL_POS>php artisan migrate

  INFO  Running migrations.

  2024_05_15_055039_add_image_to_m_user_table ..... 124ms DONE
```

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),
);
    }

}
```

app > Models > UserModel.php > ...

```php
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Model;
use Illuminate\Database\Eloquent\Relations\BelongsTo;
use App\Models\LevelModel;
use Illuminate\Foundation\Auth\User as userAuthentication;
use Tymon\JWTAuth\Contracts\JWTSubject;
use Illuminate\Database\Eloquent\Casts\Attribute;

class UserModel extends userAuthentication 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 $this->belongsTo( related: LevelModel::class, foreignKey: 'level_id', ownerKey: 'level_id');
    }
    protected function image(): Attribute
    {
        return Attribute::make( get: get: fn ($image) => 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;
 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);
    }
```

```
}
```

app > Http > Controllers > Api > RegisterController.php > 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;

class RegisterController extends Controller
{
    public function __invoke(Request $request)
    {
        $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 ($validator->fails()) {
            return response()->json( data: $validator->errors(), status: 422);
        }

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

        if($user){ return response()->json( data: [
            'success' => true,
            'user' => $user,
        ], status: 201);
    }
        return response()->json( data: [
            'status' => false,
        ], status: 409);
    }
}
```

7. Anda dapat menambahkan detail untuk spesifikasi image pada validator

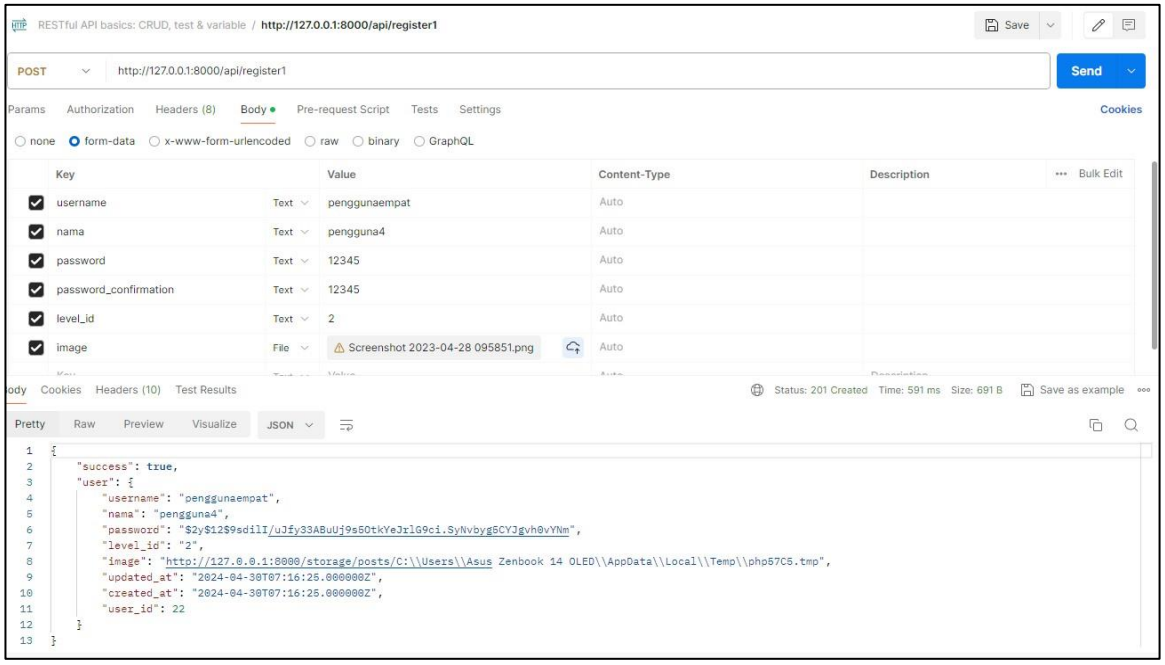
```
'image'      => 'required|image|mimes:jpeg,png,jpg,gif,svg|max:2048',
```

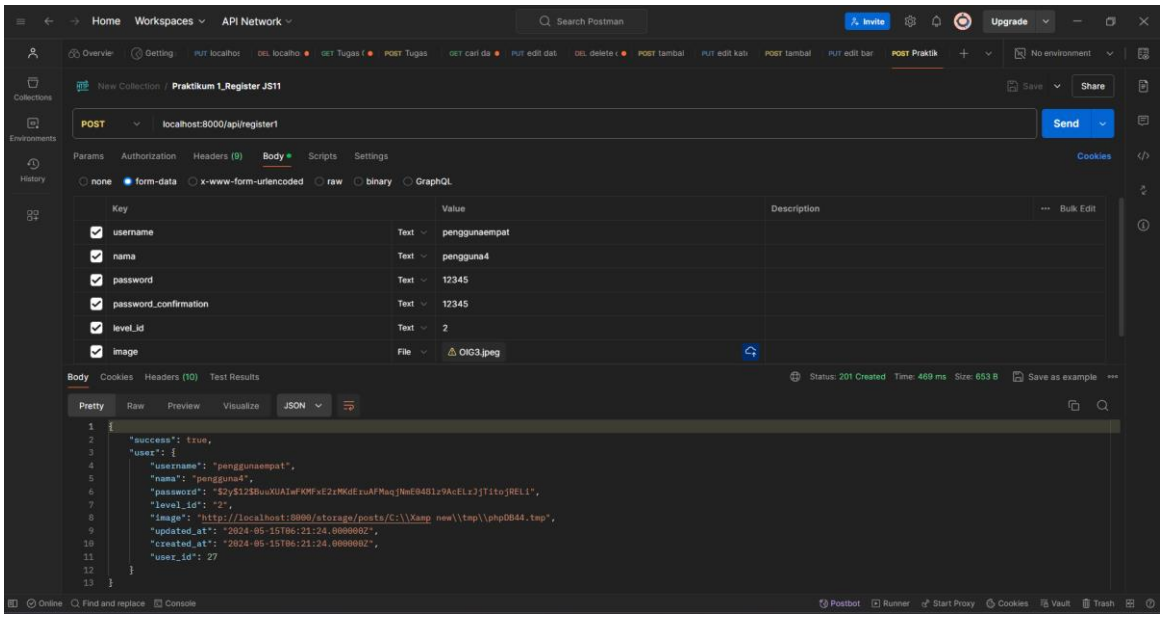
```php
public function __invoke(Request $request)
{
    $validator = Validator::make( data: $request->all(), rules: [
        'username' => 'required',
        'nama' => 'required',
        'password' => 'required|min:5|confirmed',
        'level_id' => 'required',
        'image' => 'required|image|mimes:jpeg,png,jpg,gif,svg|max:2048'
    ]);
```

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

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



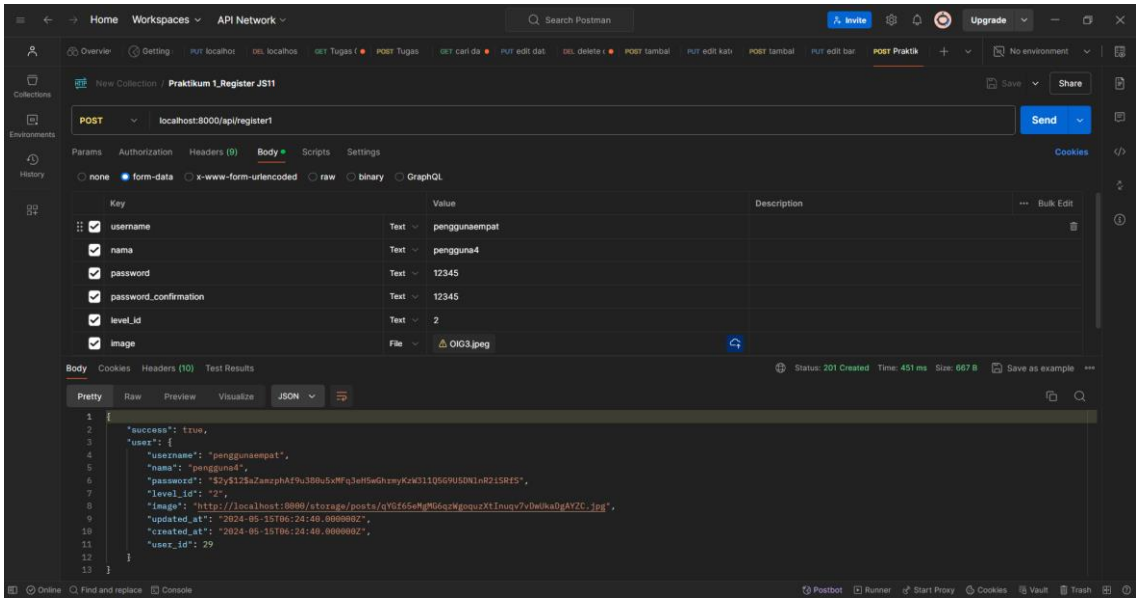


8. Pada Controllers/Api/RegisterController bagian create user ganti dengan

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

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

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



Jawab: Perbedaanya terletak pada nama gambarnya, pada saat pertama nama gambarnya disimpan dengan nama asli sedangkan setelah merubah kodenya seperti diatas nama dari gambar yang kita upload menjadi acak karena hashName.

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.

No	Screenshoot

1	**Kode**
	`C:\laragon\www\PWL_POS>php artisan make:migration add_image_to_m_barang_table`
	`INFO Migration [C:\laragon\www\PWL_POS\database\migrations/2024_05_15_064652_add_image_to_m_barang_table.php] created successfully.`
	`C:\laragon\www\PWL_POS>php artisan migrate`
	`INFO Running migrations.`
	`2024_05_15_064652_add_image_to_m_barang_table ..... 87ms DONE`

database > migrations > 2024_05_15_064652_add_image_to_m_barang_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_barang', callback: function (Blueprint $table) {
            $table->string( column: 'image');
        });
    }

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

app > Models > 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\Relations\HasMany;
use App\Models\KategoriModel;
use Illuminate\Database\Eloquent\Casts\Attribute;

class BarangModel extends Model
{
    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 $this->belongsTo( related: KategoriModel::class, foreignKey: 'kategori_id', ownerKey: 'kategori_id');
    }
    protected function image(): Attribute
    {
        return Attribute::make(
            get: get: fn ($image) => url( path: '/storage/posts/' . $image),
        );
    }

}
```

```php
    public function update(Request $request, BarangModel $barang)
    {
        $request->validate([
            'barang_kode' => 'required|string|min:3|unique:m_barang,barang_kode',
            'barang_nama' => 'required|string|max:100',
            'harga_beli' => 'required|integer',
            'harga_jual' => 'required|integer',
            'kategori_id' => 'required|integer',
            'image' => 'required|image|mimes:jpeg,png,jpg,gif,svg|max:2048'
        ]);

        $barang->update([
            'barang_kode' => $request->barang_kode,
            'barang_nama' => $request->barang_nama,
            'harga_beli' => $request->harga_beli,
            'harga_jual' => $request->harga_jual,
            'kategori_id' => $request->kategori_id,
            'image' => $request->image->hashName()
        ]);

        return BarangModel::find($barang);
    }

    public function destroy(BarangModel $barang)
    {
        $barang->delete();
        return response()->json([
            'success' => true,
            'message' => 'Data Berhasil Dihapus'
        ]);
    }
}
```

app > Http > Controllers > Api > BarangController.php > BarangController

```php
<?php

namespace App\Http\Controllers\Api;

use App\Http\Controllers\Controller;
use Illuminate\Http\Request;
use App\Models\BarangModel;

class BarangController extends Controller
{
    public function index()
    {
        return BarangModel::all();
    }

    public function store(Request $request)
    {
        $request->validate([
            'barang_kode' => 'required|string|min:3|unique:m_barang,barang_kode',
            'barang_nama' => 'required|string|max:100',
            'harga_beli' => 'required|integer',
            'harga_jual' => 'required|integer',
            'kategori_id' => 'required|integer',
            'image' => 'required|image|mimes:jpeg,png,jpg,gif,svg|max:2048'
        ]);

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

        return response()->json(data: $barang, status: 201);
    }

    public function show(BarangModel $barang)
    {
        return response()->json(data: $barang, status: 200);

    }
```

2	**Hasil**