LAPORAN PRAKTIKUM INTERNET OF THINGS

(IoT)

FAKULTAS VOKASI, UNIVERSITAS BRAWIJAYA



**Praktik Pembuatan API   
Menggunakan Laravel 11 dan Ngrok**

*Wildan Taufiqurrahman*

Fakultas Vokasi, Universitas Brawijaya

*Email: wildantaw@gmail.com*

**Abstrak**

Pada praktikum kali ini yaitu membuat API menggunakan platform Laravel 11 dan NGRok. Pada praktikum ini saya berhasil membuat nya dan program yang saya buat sesuai tujuan yaitu mengetes API di postman.

Keywords : Postman, Laravel 11 dan API.

1. Introduction (Pendahuluan)

* 1. Latar Belakang

Pembuatan praktikum ini dilakukan untuk memahami bagaimana membuat API menggunakan Laravel 11 dimana code editor yang di gunakan adalah Visual Studio Code.

Tujuan Eksperimen Tujuan eksperimen ini adalah:

1. Memahami cara membuat API di Laravel 11.
2. Melakukan simulasi test API menggunakan Postman.

2. Methodology (Metodologi)

2.1 Tools & Materials (Alat dan Bahan)

Adapun perangkat yang digunakan dalam praktikum ini adalah sebagai berikut:

* Laravel 11
* Visual Studio Code
* Chrome

2.2 Implementation Steps (Langkah Implementasi)

Adapun Langkah-langkah dalam penyusunan sistem, pengkodean, dan pengujian adalah sebagai berikut:

1. Buka terminal dan jalankan: composer create-project --prefer-dist laravel/laravel:^11.0 laravel-11
2. Kemudian masuk ke folder project cd Laravel-11
3. Buat database di php my admin dengan naman iot\_25
4. Kemudian setting env
5. Buat migrasi
6. Isi file migrasi sesuai kode yang tertera
7. Buat model kemudian ubah sesuai ketentuan
8. Jalankan migrasi
9. Buat resource utuk api
10. Ubah file TransaksiResource.php sesuai lampiran.
11. Buat controller untuk api
12. Ubah isi controller api sesuai lampiran.
13. Tambahkan route API.
14. Jalankan route list.
15. Kemudian start server Laravel.
16. Testing API menggunakan postman dengan metode get dan post.
17. Kemudian online kan API dengan NGROk.
18. Integrasikan ESP32 via wokwi.

3. Results and Discussion (Hasil dan Pembahasan)

3.1 Experimental Results (Hasil Eksperimen) Hasil eksperimen menunjukkan keberhasilan dalam Mengetes API. Data yang diperoleh:

* Simulasi berjalan sesuai dengan kode yang telah dibuat.
* API menampilkan respon Ketika kita GET maupun POST.

1. Appendix (Lampiran, jika diperlukan)\

**Kode Program**

Migration:  
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::create('transaksi\_sensor', function (Blueprint $table) {

$table->id('id')->startingValue(1); // Auto Increment dimulai dari 1

$table->string('nama\_sensor', 255);

$table->integer('nilai1');

$table->integer('nilai2');

$table->timestamps();

});

}

public function down(): void

{

Schema::dropIfExists('transaksi\_sensor');

}

};

Model:  
namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;

use Illuminate\Database\Eloquent\Model;

class TransaksiSensor extends Model

{

use HasFactory;

protected $table = 'transaksi\_sensor';

protected $fillable = [

'nama\_sensor',

'nilai1',

'nilai2',

];

}

**Resource API**

**namespace App\Http\Resources;**

**use Illuminate\Http\Request;**

**use Illuminate\Http\Resources\Json\JsonResource;**

**class TransaksiSensorResource extends JsonResource**

**{**

**public function toArray($request): array**

**{**

**return [**

**'id' => $this->id,**

**'nama\_sensor' => $this->nama\_sensor,**

**'nilai1' => $this->nilai1,**

**'nilai2' => $this->nilai2,**

**];**

**}**

**}**

**CONTROLLER API**

**namespace App\Http\Controllers\Api;**

**use App\Models\TransaksiSensor;**

**use App\Http\Controllers\Controller;**

**use App\Http\Resources\TransaksiSensorResource;**

**use Illuminate\Http\Request;**

**class TransaksiSensorController extends Controller**

**{**

**public function index()**

**{**

**return TransaksiSensorResource::collection(TransaksiSensor::latest()->paginate(5));**

**}**

**public function store(Request $request)**

**{**

**$validatedData = $request->validate([**

**'nama\_sensor' => 'required|string|max:255',**

**'nilai1' => 'required|integer',**

**'nilai2' => 'required|integer',**

**]);**

**$sensor = TransaksiSensor::create($validatedData);**

**return new TransaksiSensorResource($sensor);**

**}**

**public function show($id)**

**{**

**return new TransaksiSensorResource(TransaksiSensor::findOrFail($id));**

**}**

**public function update(Request $request, $id)**

**{**

**$sensor = TransaksiSensor::findOrFail($id);**

**$sensor->update($request->all());**

**return new TransaksiSensorResource($sensor);**

**}**

**public function destroy($id)**

**{**

**TransaksiSensor::findOrFail($id)->delete();**

**return response()->json(['message' => 'Data berhasil dihapus'], 204);**

**}**

**}**

**FILE ROUTE API**

**use App\Http\Controllers\Api\TransaksiSensorController;**

**Route::apiResource('/posts', TransaksiSensorController::class);**







