LAPORAN PRAKTIKUM INTERNET OF THINGS

(IoT):

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



**Micko Junior Putra Pratma**

**233140700111076 (T4C)**

**Email : mickojunior1004@gmail.com**

Fakultas Vokasi

Program studi Teknologi Informasi

Universitas Brawijaya

2025

**Abstrak**

Laravel 11 membawa sejumlah peningkatan performa serta fitur yang lebih optimal dalam pengelolaan API. Salah satu tantangan utama dalam pengembangan API adalah bagaimana mengaksesnya dari perangkat lain atau melakukan pengujian di lingkungan lokal.

**1. Pendahuluan**

* 1. **Latar Belakang**

Praktikum ini dibuat untuk mempelajari cara membangun API menggunakan Laravel 11, dengan Visual Studio Code sebagai code editor yang digunakan.

**1.2 Tujuan eksperimen**

 Mempelajari proses pembuatan API dengan Laravel 11.

 Melakukan pengujian API menggunakan Postman

**2. Methodology (Metodologi)**

**2.1 Tools & Materials (Alat dan Bahan)**

* Laravel 11
* Visual Studio Code
* PHP my admin
* Nrok

**2.2 Implementation Steps (Langkah Implementasi)**

 Buat proyek Laravel dan masuk ke direktori proyek.

 Siapkan database di phpMyAdmin atau MySQL CLI, lalu konfigurasi di file .env.

 Buat dan edit migrasi di database/migrations/, lalu jalankan dengan php artisan migrate.

 Buat model Transaksi di app/Models/ dan sesuaikan strukturnya.

 Buat resource API di app/Http/Resources/TransaksiResource.php.

 Buat controller API di app/Http/Controllers/Api/TransaksiController.php.

 Tambahkan route API di routes/api.php.

 Jalankan php artisan route:list, start server dengan php artisan serve, dan gunakan Ngrok untuk mengakses API secara online.

 Integrasikan dengan ESP32 di Wokwi menggunakan URL dari Ngrok.

**3. Results and Discussion (Hasil dan Pembahasan)**

**3.1 Experimental Results**

****

**4. Appendix**

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);**