**LAPORAN PRAKTIKUM INTERNET OF THINGS (IoT)** Fakultas Vokasi, Universitas Brawijaya

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

*Wildan Aridh Takhfif  
 Fakultas Vokasi, Universitas Brawijaya  
 Email: apipkph@gmail.*

**Abstract**

This experiment aims to build and test a RESTful API using Laravel 11 and make it accessible via the public internet using Ngrok. The practical work includes setting up Laravel 11, configuring a MySQL database, and creating a model, resource, and API controller for sensor transaction data. Postman was used to verify GET and POST methods for retrieving and inserting data into the database. Finally, Ngrok was employed to expose the local Laravel development server over the internet so that it could be accessed remotely. The results demonstrate that the API is fully operational and accessible via both local and public URLs, making it suitable for integration with IoT devices or simulators such as Wokwi.

*Keywords—Internet of Things, Laravel, Ngrok, API, Postman*

**1. Introduction (Pendahuluan)**

**1.1 Latar Belakang** Dalam pengembangan sistem IoT, API menjadi komponen vital untuk pertukaran data antar perangkat. Laravel 11 menyediakan kerangka kerja yang efisien untuk membangun API, sedangkan Ngrok memudahkan akses ke aplikasi yang berjalan secara lokal dari jaringan publik. Praktikum ini difokuskan pada pembuatan API untuk transaksi sensor, yang nantinya dapat diintegrasikan dengan perangkat IoT atau simulator.

**1.2 Tujuan Eksperimen**

* Menginstal dan mengkonfigurasi Laravel 11 untuk pembuatan API.
* Membuat model, migrasi, resource, dan API controller untuk data transaksi sensor.
* Menguji API menggunakan Postman untuk operasi GET dan POST.
* Mengonline-kan API melalui Ngrok agar dapat diakses secara publik.

**2. Methodology (Metodologi)**

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

* **Software:** Laravel 11, Composer, PHP, MySQL, Postman, Ngrok
* **Database:** MySQL (Database: *iot\_25*)
* **Tools Pengujian:** Postman untuk testing API

**2.2 Implementation Steps (Langkah Implementasi)**

1. **Instalasi Laravel 11:**

Jalankan perintah:  
 composer create-project --prefer-dist laravel/laravel:^11.0 laravel-11

cd laravel-11

1. **Konfigurasi Database:**
   * Buat database dengan nama iot\_25 melalui phpMyAdmin.

Ubah file .env sesuai dengan pengaturan database:  
 DB\_CONNECTION=mysql

DB\_HOST=127.0.0.1

DB\_PORT=3306

DB\_DATABASE=iot\_25

DB\_USERNAME=root

DB\_PASSWORD=caberg2010

DB\_CHARSET=utf8mb4

DB\_COLLATION=utf8mb4\_unicode\_ci

1. **Pembuatan Model & Migrasi:**

Buat model TransaksiSensor dengan perintah:  
 php artisan make:model TransaksiSensor -m

* + Ubah file migrasi (contoh: 2025\_02\_21\_074123\_create\_transaksi\_sensors\_table.php) dengan mendefinisikan tabel transaksi\_sensor beserta kolomnya.

1. **Menjalankan Migrasi:**

Eksekusi perintah:  
 php artisan migrate

1. **Membuat Resource:**

Jalankan perintah:  
 php artisan make:resource TransaksiSensorResource

* + Ubah isi file resource di app/Http/Resources/TransaksiSensorResource.php agar mengembalikan data dengan format JSON.

1. **Membuat API Controller:**

Jalankan perintah:  
 php artisan make:controller Api/TransaksiSensorController

* + Implementasikan fungsi index, store, show, update, dan destroy di controller untuk operasi CRUD pada data transaksi sensor.

1. **Menambahkan Route API:**

Edit file routes/api.php untuk mendefinisikan endpoint API:  
 use App\Http\Controllers\Api\TransaksiSensorController;

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

1. **Testing API dengan Postman:**
   * Jalankan Laravel dengan perintah php artisan serve.
   * Gunakan Postman untuk mengakses endpoint http://127.0.0.1:8000/api/posts dengan metode GET untuk mengambil data dan metode POST untuk menambahkan data baru.
2. **Mengonlinekan API dengan Ngrok:**
   * Unduh dan install Ngrok dari<https://dashboard.ngrok.com/signup>.

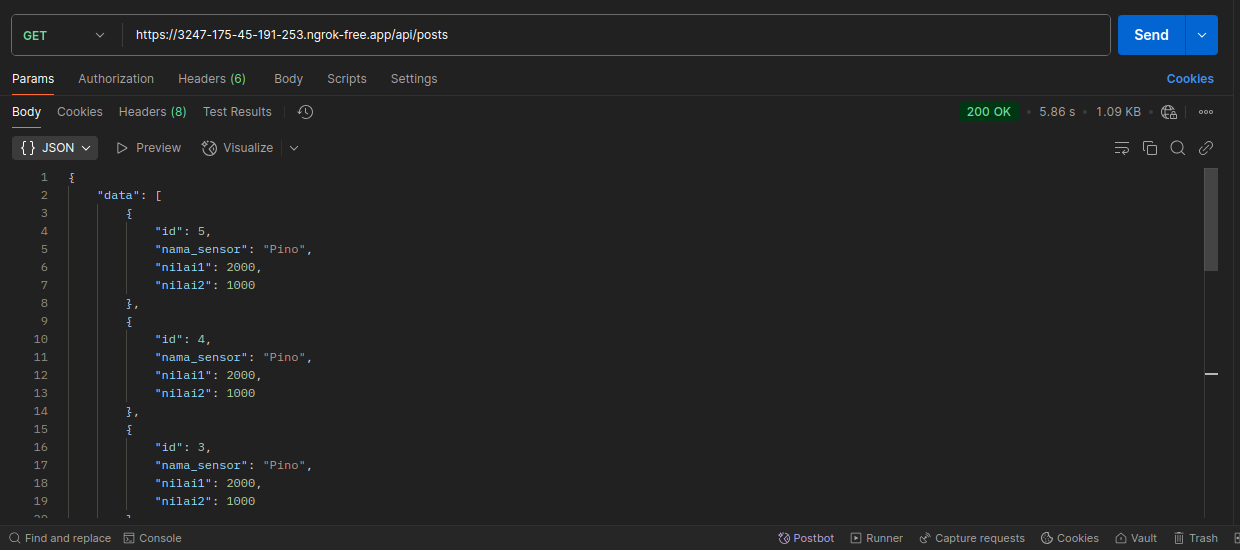
Jalankan perintah:  
 ngrok http 8000

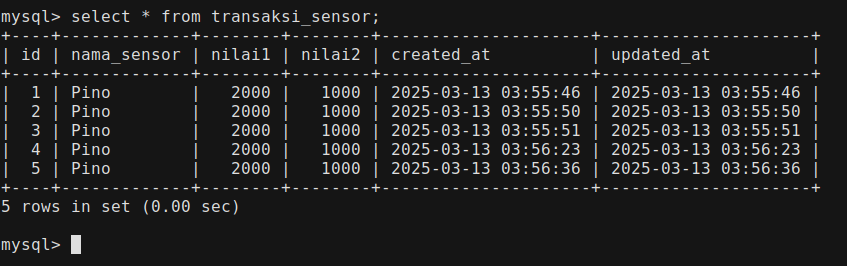
* + Gunakan URL publik yang diberikan (misalnya, https://xxxx.ngrok-free.app/api/posts) untuk mengakses API dari perangkat IoT atau simulator.

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

**3.1 Hasil Eksperimen**

Setelah mengikuti langkah-langkah implementasi, API berhasil dibuat dan dapat mengembalikan data dari tabel transaksi\_sensor dalam format JSON. Contoh output dari Postman:





Pengujian POST juga berhasil menambahkan data baru ke database (bisa dicek secara manual di phpMyAdmin). Selain itu, dengan Ngrok, API yang berjalan secara lokal dapat diakses melalui URL publik, memungkinkan pengujian dari perangkat IoT atau simulasi seperti Wokwi.

**4. Appendix (Lampiran, jika diperlukan)**

**4.1 Kode Program** Berikut adalah contoh kode utama pada API Controller dan Resource:

*TransaksiSensorResource.php:*

namespace App\Http\Resources;

use Illuminate\Http\Resources\Json\JsonResource;

class TransaksiSensorResource extends JsonResource

{

public function toArray($request)

{

return [

'id' => $this->id,

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

'nilai1' => $this->nilai1,

'nilai2' => $this->nilai2,

];

}

}

*TransaksiSensorController.php:*

namespace App\Http\Controllers\Api;

use Illuminate\Http\Request;

use App\Models\TransaksiSensor;

use App\Http\Controllers\Controller;

use App\Http\Resources\TransaksiSensorResource;

class TransaksiSensorController extends Controller

{

public function index()

{

$transaksiSensors = TransaksiSensor::latest()->paginate(5);

return TransaksiSensorResource::collection($transaksiSensors);

}

public function store(Request $request)

{

$validatedData = $request->validate([

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

'nilai1' => 'required|integer',

'nilai2' => 'required|integer',

]);

$transaksiSensor = TransaksiSensor::create($validatedData);

return new TransaksiSensorResource($transaksiSensor);

}

public function show($id)

{

$transaksiSensor = TransaksiSensor::findOrFail($id);

return new TransaksiSensorResource($transaksiSensor);

}

public function update(Request $request, $id)

{

$validatedData = $request->validate([

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

'nilai1' => 'required|integer',

'nilai2' => 'required|integer',

]);

$transaksiSensor = TransaksiSensor::findOrFail($id);

$transaksiSensor->update($validatedData);

return new TransaksiSensorResource($transaksiSensor);

}

public function destroy($id)

{

$transaksiSensor = TransaksiSensor::findOrFail($id);

$transaksiSensor->delete();

return response()->json(['message' => 'Deleted successfully'], 204);

}

}