LAPORAN PRAKTIKUM INTERNET OF THINGS (IoT)

Fakultas Vokasi, Universitas Brawijaya

**Praktik Pembuatan API**

**Menggunakan Laravel 11 dan Ngrok**

***Imran Darajati***

***Fakultas Vokasi, Universitas Brawijaya***

***Email :*** [***imran.darajati@gmail.com***](mailto:imran.darajati@gmail.com)

**Abstrac**

Praktik ini bertujuan untuk mempelajari cara membuat API menggunakan Laravel 11 serta menggunakan Ngrok agar API dapat diakses dari internet. API yang dibuat memungkinkan POST data ke database serta GET data dari database melalui Postman dengan URL yang diberikan oleh Ngrok. Laravel 11 digunakan sebagai framework backend, sementara MySQL digunakan sebagai database. Hasil praktik menunjukkan bahwa Laravel 11 dapat menangani request POST untuk menyimpan data serta request GET untuk menampilkan data melalui Ngrok URL di Postman.

*Kata Kunci: API, Laravel 11, Ngrok, Postman, GET Request, POST Request, Database, VS Code.*

1. **Pendahuluan**

**1.1 Latar Belakang**

Dalam pengembangan aplikasi modern, API (Application Programming Interface) menjadi penghubung utama antara frontend dan backend. Laravel 11 sebagai framework PHP terbaru menawarkan kemudahan dalam pembuatan API.

Namun, API yang dijalankan secara lokal tidak dapat langsung diuji dari luar jaringan komputer pengembang. Oleh karena itu, Ngrok digunakan untuk menyediakan URL publik yang memungkinkan API dapat diuji secara langsung melalui Postman. Dalam praktik ini, API yang dibuat dapat menyimpan data ke database (POST) dan menampilkan data yang telah disimpan (GET) menggunakan Laravel 11, MySQL, Postman, dan Ngrok.

**1.2 Tujuan Eksperimen**

Tujuan dari eksperimen ini adalah :

1. Mempelajari pembuatan API menggunakan Laravel 11.
2. Menggunakan Ngrok untuk membuka akses API ke internet.
3. Melakukan request POST untuk menyimpan data ke database melalui Postman.
4. Melakukan request GET untuk menampilkan data dari database melalui Postman.
5. **Metodologi**

**2.1 Alat dan Bahan**

* Software yang Digunakan:

1. Visual Studio Code (VS Code) → Sebagai IDE untuk pengembangan API Laravel.
2. XAMPP → Untuk menjalankan MySQL sebagai database backend.
3. Composer → Untuk mengelola dependensi Laravel.
4. Ngrok → Untuk membuat tunneling URL agar API Laravel bisa diakses secara publik.
5. Postman → Untuk melakukan GET dan POST request ke API Laravel melalui Ngrok.

**2.2 Langkah Implementasi**

1. Instalasi Laravel 11

* Buka VS Code dan jalankan terminal.
* Instal Laravel 11 dengan perintah :

composer create-project --prefer-dist laravel/laravel IoT\_25

* Masuk ke folder proyek Laravel :

cd IoT\_25

* Jalankan server Laravel :

php artisan serve

1. Konfigurasi Database

* Buat file model TransaksiSensor.php dengan cara menjalankan perintah :

php artisan make:model TransaksiSensor

* Edit file app/Models/TransaksiSensor.php :

<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;

use Illuminate\Database\Eloquent\Model;

class TransaksiSensor extends Model

{

use HasFactory;

/\*\*

\* The table associated with the model.

\*

\* @var string

\*/

protected $table = 'transaksi\_sensor';

/\*\*

\* The attributes that are mass assignable.

\*

\* @var array

\*/

protected $fillable = [

'nama\_sensor',

'nilai1',

'nilai2',

];

/\*\*

\* The attributes that should be hidden for arrays.

\*

\* @var array

\*/

protected $hidden = [];

/\*\*

\* The attributes that should be cast.

\*

\* @var array

\*/

protected $casts = [];

}

* Buka file .env dan sesuaikan dengan database MySQL :

DB\_CONNECTION=mysql

DB\_HOST=127.0.0.1

DB\_PORT=3306

DB\_DATABASE=IoT\_25

DB\_USERNAME=root

DB\_PASSWORD=

* Jalankan perintah untuk membuat tabel posts :

php artisan make:migration create\_transaksi\_sensor\_table

* Edit file migration database/migrations/xxxx\_xx\_xx\_xxxxxx\_create\_transaksi\_sensor\_table.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::create('transaksi\_sensor', function (Blueprint $table) {

$table->id('id')->startingValue(1); // Menetapkan AUTO\_INCREMENT dimulai dari 1

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

$table->integer('nilai1', false)->length(255); // int(255)

$table->integer('nilai2', false)->length(255); // int(255)

$table->timestamps(); // Menambahkan created\_at dan updated\_at

});

}

/\*\*

\* Reverse the migrations.

\*/

public function down(): void

{

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

}

};

* Jalankan migrasi untuk membuat tabel di database :

php artisan migrate

1. Membuat API untuk GET dan POST Data ke Database

* Buat controller API :

php artisan make:controller Api/TransaksiSensorController

* Edit app/Http/Controllers/TransaksiSensorController.php untuk menangani POST request (simpan data) dan GET request (ambil data) :

<?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

{

/\*\*

\* index

\*

\* @return \Illuminate\Http\Response

\*/

public function index()

{

// Get all transactions from TransaksiSensor model, paginated

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

// Return a collection of transactions as a resource

return TransaksiSensorResource::collection($transaksiSensors);

}

/\*\*

\* Store a newly created resource in storage.

\*

\* @param \Illuminate\Http\Request $request

\* @return \Illuminate\Http\Response

\*/

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);

}

/\*\*

\* Display the specified resource.

\*

\* @param int $id

\* @return \Illuminate\Http\Response

\*/

public function show($id)

{

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

return new TransaksiSensorResource($transaksiSensor);

}

/\*\*

\* Update the specified resource in storage.

\*

\* @param \Illuminate\Http\Request $request

\* @param int $id

\* @return \Illuminate\Http\Response

\*/

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);

}

/\*\*

\* Remove the specified resource from storage.

\*

\* @param int $id

\* @return \Illuminate\Http\Response

\*/

public function destroy($id)

{

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

$transaksiSensor->delete();

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

}

}

* Jalankan server Laravel kembali :

php artisan serve

1. Menghubungkan API dengan Ngrok

* Jalankan Ngrok untuk membuka akses API Laravel ke internet:

ngrok http 8000

* Ngrok akan memberikan URL publik, misalnya:

<https://abcd1234.ngrok.io>

* API sekarang dapat diakses melalui Ngrok URL, contohnya:

<https://abcd1234.ngrok.io/api/posts>

1. Menguji API dengan Postman

* POST Request (Simpan Data)
* Pilih metode POST.
* Massukkan URL Ngrok :

<https://abcd1234.ngrok.io/api/posts>

* Di bagian Body, pilih raw dan gunakan format JSON :

{

"nama\_sensor": "SENSOR GD",

"nilai1": 123,

"nilai2": 321

}

* Klik **Send** untuk menyimpan data ke database.
* Respon yang diterima :

{

"data": {

"id": 12,

"nama\_sensor": "SENSOR GD",

"nilai1": 123,

"nilai2": 321

}

}

* GET Request (Ambil Data)
* Pilih metode POST.
* Massukkan URL Ngrok :

<https://abcd1234.ngrok.io/api/posts>

* Klik **Send** untuk mengambil semua data dari database.
* Respon yang diterima :

{

"data": [

{

"id": 13,

"nama\_sensor": "SENSOR A",

"nilai1": 123,

"nilai2": 321

}

],

"links": {

"first": "http://68c5-36-71-117-80.ngrok-free.app/api/posts?page=1",

"last": "http://68c5-36-71-117-80.ngrok-free.app/api/posts?page=1",

"prev": null,

"next": null

},

"meta": {

"current\_page": 1,

"from": 1,

"last\_page": 1,

"links": [

{

"url": null,

"label": "&laquo; Previous",

"active": false

},

{

"url": "http://68c5-36-71-117-80.ngrok-free.app/api/posts?page=1",

"label": "1",

"active": true

},

{

"url": null,

"label": "Next &raquo;",

"active": false

}

],

"path": "http://68c5-36-71-117-80.ngrok-free.app/api/posts",

"per\_page": 5,

"to": 1,

"total": 1

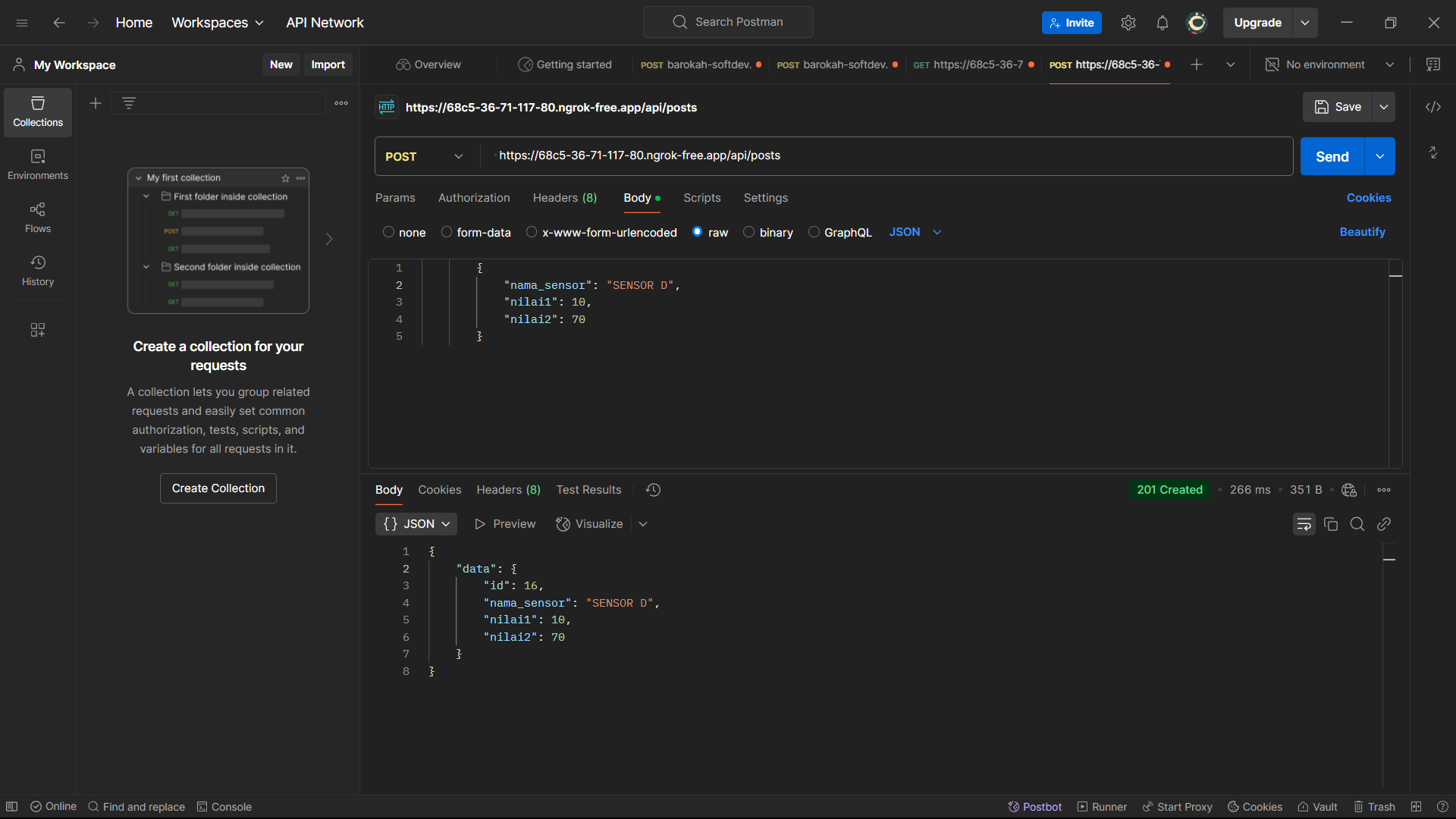
}

}

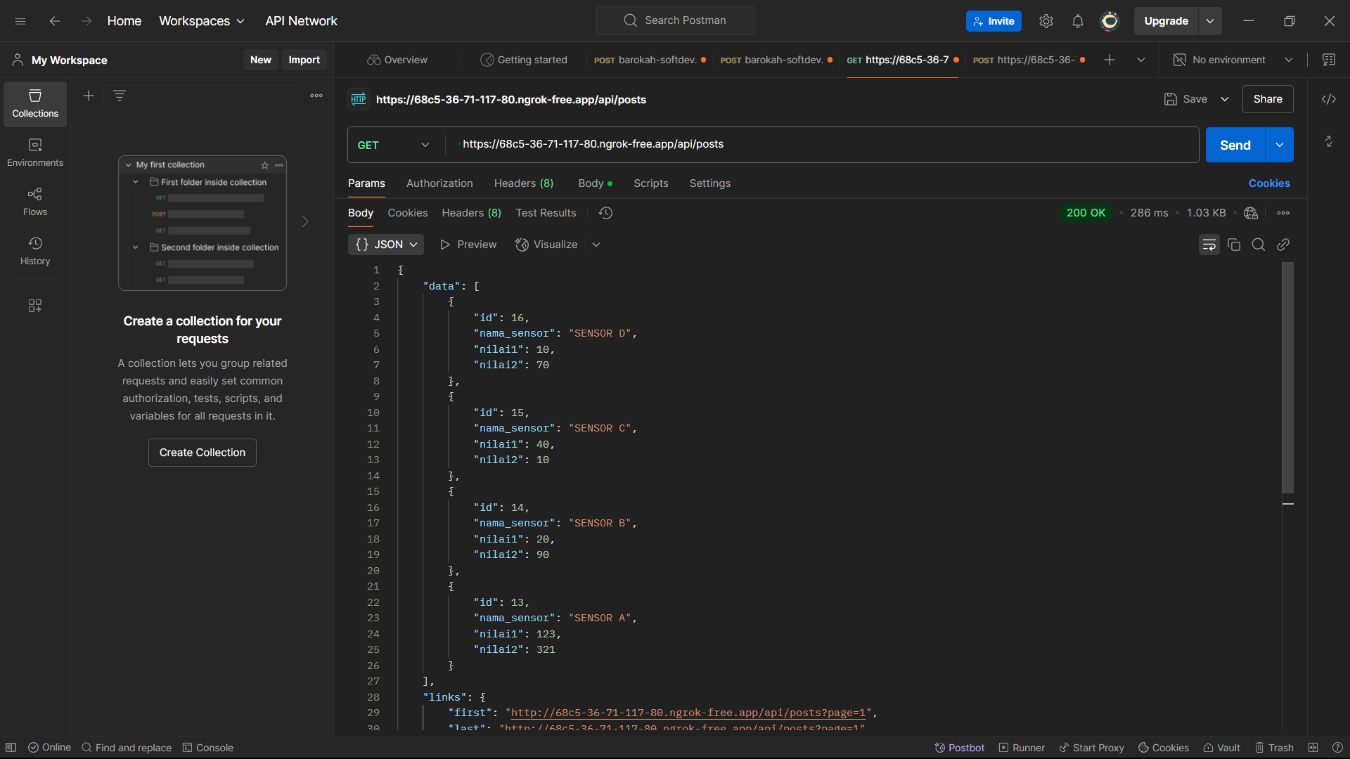
1. **Hasil dan Pembahasan**

Berikut adalah tampilan hasil pengujian API Laravel 11 dengan Postman dan Ngrok:

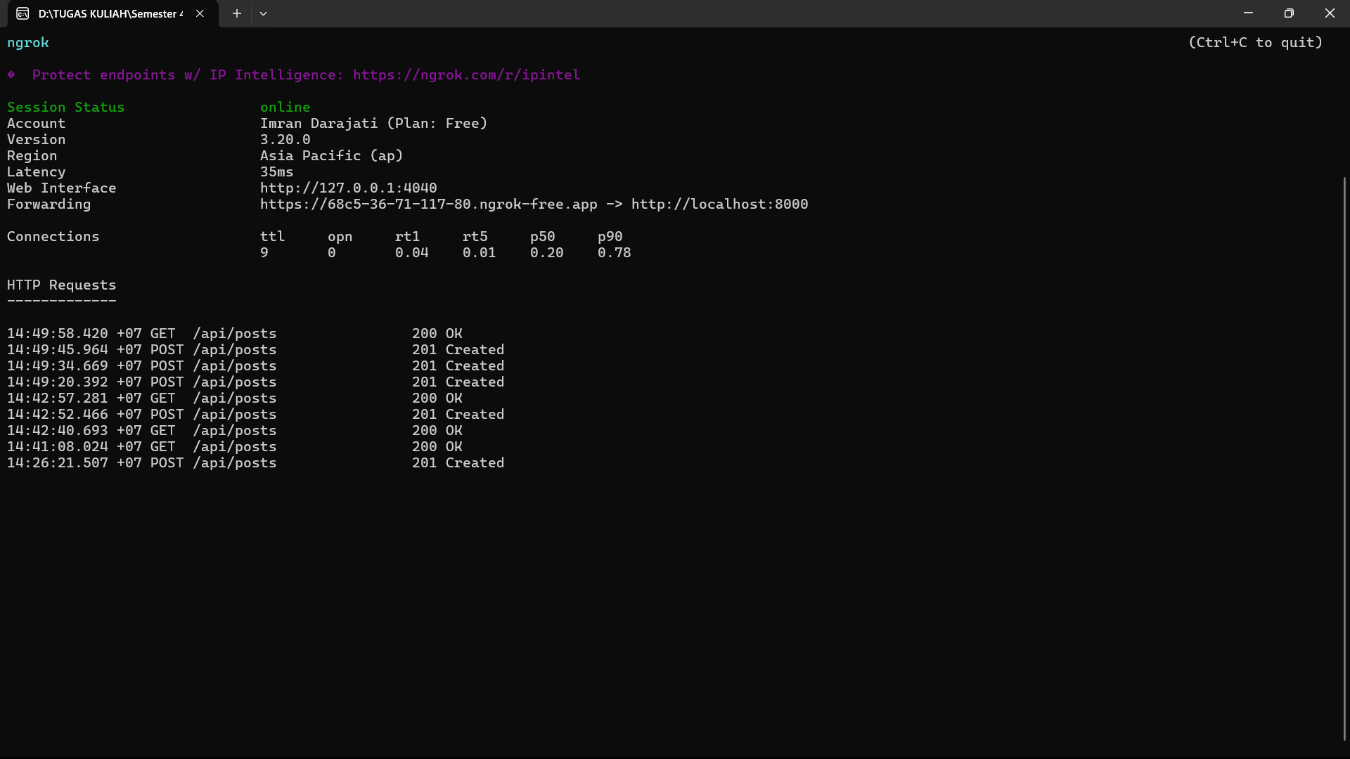
1. Hasil pengujian POST request di Postman untuk menyimpan data ke database.



1. Hasil pengujian GET request di Postman untuk mengambil data dari database.



1. Tampilan URL Ngrok yang aktif dan digunakan untuk mengakses API Laravel secara online.



1. Tampilan database MySQL yang menunjukkan data yang berhasil disimpan melalui API