**LAPORAN PRAKTIKUM INTERNET OF THINGS (IoT)**

Praktik Pembuatan API Menggunakan Laravel 11 dan Ngrok

*Abdur Rohim Syah Sjadja‘ah*

*Fakultas Vokasi, Universitas Brawijaya*

*Email:* [abdurrohimsyah05@gmail.com](mailto:abdurrohimsyah05@gmail.com)

**Abstrak**

Praktikum ini bertujuan untuk mengembangkan dan mengimplementasikan API menggunakan Laravel 11 dan mengintegrasikannya dengan layanan Ngrok. Ngrok digunakan dalam praktikum ini untuk membuat jalur dari localhost ke internet, sehingga API yang dibangun dapat diakses secara publik. Hal ini sangat berguna untuk pengujian dan demonstrasi, terutama saat berkolaborasi dengan tim atau klien yang memerlukan akses langsung ke aplikasi yang sedang dikembangkan. Dengan mengintegrasikan Ngrok, peserta dapat melakukan pengujian API secara real-time menggunakan alat seperti Postman atau aplikasi frontend lainnya. Pengembangan dilakukan di lingkungan Visual Studio Code (VSCode), dengan memanfaatkan Postman untuk pengujian API, XAMPP sebagai database server, dan terminal Command Prompt (CMD) sebagai alat bantu eksekusi perintah. Dengan memanfaatkan Ngrok, API dapat diakses dari jarak jauh tanpa memerlukan konfigurasi server tambahan, sehingga sangat praktis untuk tujuan pengembangan dan pengujian. Hasil dari praktikum ini menunjukkan bahwa API berbasis Laravel 11 berfungsi secara efektif, memungkinkan pertukaran data yang efisien dan fleksibel.

Kata kunci: API, Ngrok, Laravel 11, XAMPP, VSCode

1. **Introduction (Pendahuluan)**

**1.1 Latar Belakang**

Perkembangan teknologi saat ini memungkinkan  untuk pengaksesan berbagai aplikasi untuk berinteraksi satu sama lain. Salah satu penerapannya adalah pembuatan API yang memungkinkan pertukaran data antara berbagai sistem dan perangkat, sehingga menjadi komponen penting dalam pengembangan aplikasi modern, termasuk dalam sistem IoT, otomasi, dan layanan berbasis cloud yang sudah diterapkan mulai dari bidang industri dan kesehatan. Laravel 11 sebagai framework PHP terbaru, menawarkan berbagai fitur canggih untuk membangun RESTful API yang efisien dan aman. Dalam praktikum ini, peserta akan belajar cara membangun API dengan melakukan langkah-langkah seperti pengaturan autentikasi menggunakan Laravel Sanctum, pembuatan resource dan controller API, serta definisi rute yang diperlukan untuk operasi CRUD (Create, Read, Update, Delete) pada data. Dengan memanfaatkan VSCode, pengembang dapat menuliskan progam file laravel.php dan menjalankan server.

Secara keseluruhan, praktikum ini tidak hanya meningkatkan keterampilan teknis peserta dalam menggunakan Laravel 11 tetapi juga memperkenalkan mereka pada praktik terbaik dalam pengembangan API modern. Melalui pengalaman langsung ini, peserta diharapkan dapat memahami konsep dasar API menggunakan Laravel 11, integrasi dengan Ngrok, serta metode pengujian menggunakan Postman.

**1.2 Tujuan Eksperimen**

Praktikum ini bertujuan untuk memahami konsep dasar API RESTful, termasuk metode HTTP (GET, POST, PUT, DELETE) dan cara kerja komunikasi antara klien dan server. Dengan mengintegrasikan Ngrok, peserta akan dapat membuat API yang dikembangkan di localhost dapat diakses secara publik melalui tunneling. Pengujian API menggunakan Postman juga akan dilakukan untuk memastikan bahwa setiap endpoint berfungsi sesuai dengan spesifikasi yang telah ditentukan. Selain eksperimen ini dapat meningkatkan keterampilan teknis peserta dalam menggunakan Visual Studio Code (VSCode) sebagai alat pengembangan dan debugging aplikasi berbasis Laravel.

1. **Methodology (Metodologi)**
   1. **Tools & Materials**
2. Laptop
3. Internet
4. Web Browser (Chrome)
5. Postman
6. Visual Studio Code
7. CMD
8. XAMPP
9. Composer
10. Ngrok
    1. **Implementation Steps (Langkah Implementasi)**
11. Instalasi Ngrok (<https://ngrok.com/downloads/windows?tab=download>)
12. Instalasi Laravel

Buka Command Prompt dan jalankan perintah berikut untuk menginstal Laravel secara global: composer global require laravel/installer

Buat proyek Laravel baru dengan perintah: composer create-project --prefer-dist laravel/laravel:^11.0 laravel-11

Masuk ke folder proyek Laravel yang baru dibuat: cd laravel-11

Jalankan Laravel untuk memastikan instalasi berhasil: php artisan serve

Sampai muncul port [http://127.0.0.1:8000].

1. Konfigurasi Database
2. Jalankan XAMPP dan aktifkan Apache serta MySQL.

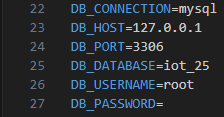
Buka phpMyAdmin di browser <http://localhost/phpmyadmin>

Buat database di phpmyadmin dengan nama **iot\_25**

Sebuah gambar berisi teks, cuplikan layar, Font, garis

Konten yang dihasilkan AI mungkin salah.

Buka VSCode dan cari file .env lalu ubah konfigurasi database seperti di bawah



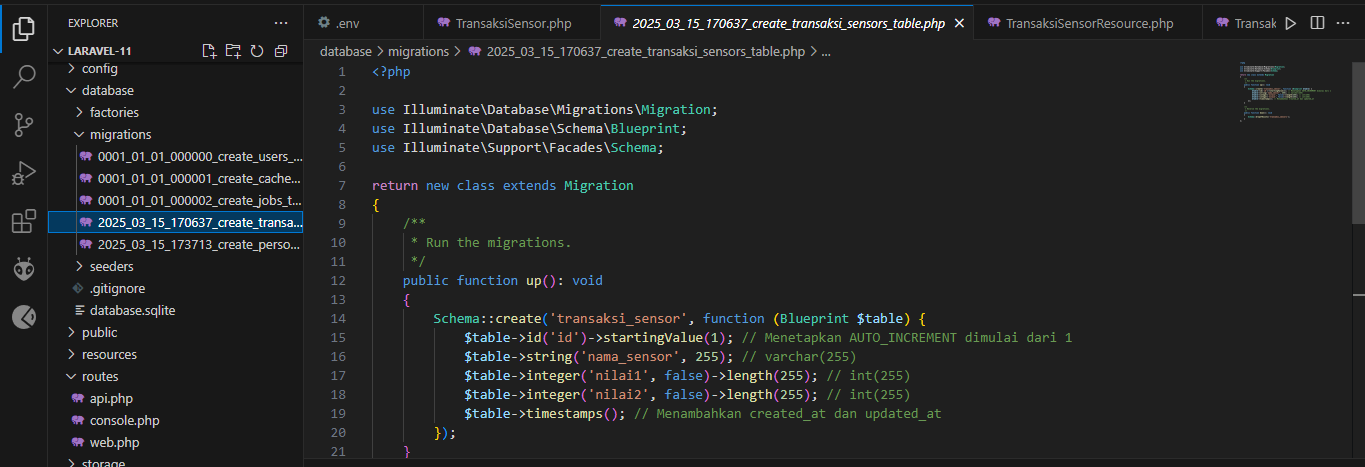
**Start Project**

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

**php artisan make:model TransaksiSensor**







**Kemudian ubah isi file** **database/migrations/2025\_03\_15\_170637\_create\_transaksi\_sensors\_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.

\*/

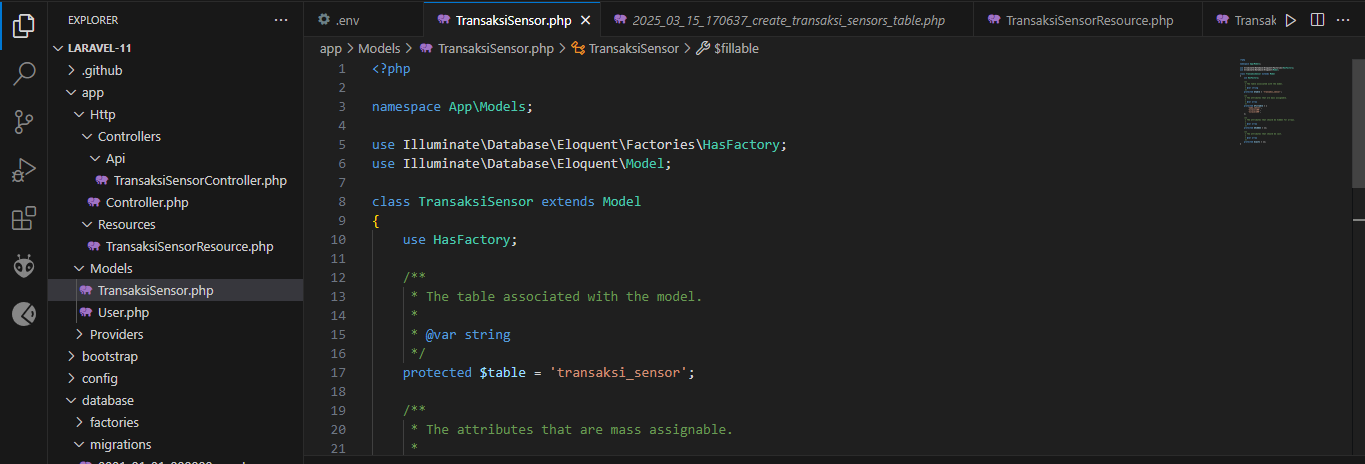
public function down(): void

{

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

}

};



**Kemudian ubah isi 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

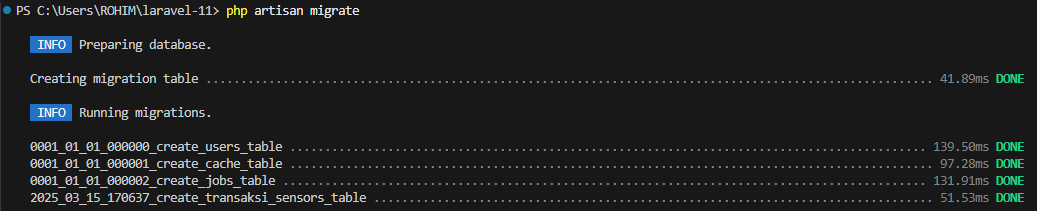
     \*/

    protected $casts = [];

}

**Kemudian jalankan perintah berikut untuk membuat tabel :**

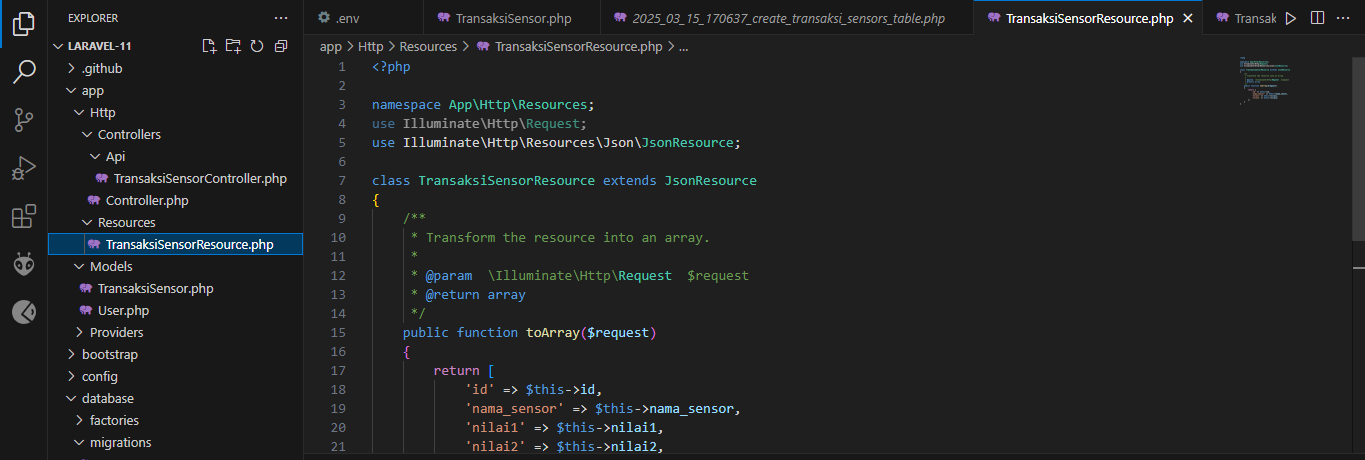
**php artisan migrate**

****

**Buat Resource dengan menjalankan perintah :**

**php artisan make:resource TransaksiSensorResource**

****

****

**Ubah isi file app/Http/Resources/TransaksiSensorResource.php dengan isi file berikut :**

<?php

namespace App\Http\Resources;

use Illuminate\Http\Request;

use Illuminate\Http\Resources\Json\JsonResource;

class TransaksiSensorResource extends JsonResource

{

/\*\*

\* Transform the resource into an array.

\*

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

\* @return array

\*/

public function toArray($request)

{

return [

'id' => $this->id,

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

'nilai1' => $this->nilai1,

'nilai2' => $this->nilai2,

];

}

}

**Buat API controller dengan menjalankan perintah :**

**php artisan make:controller Api/TransaksiSensorController**

****

**Ubah isi file app/Http/Controllers/Api/TransaksiSensorController.php**

<?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();

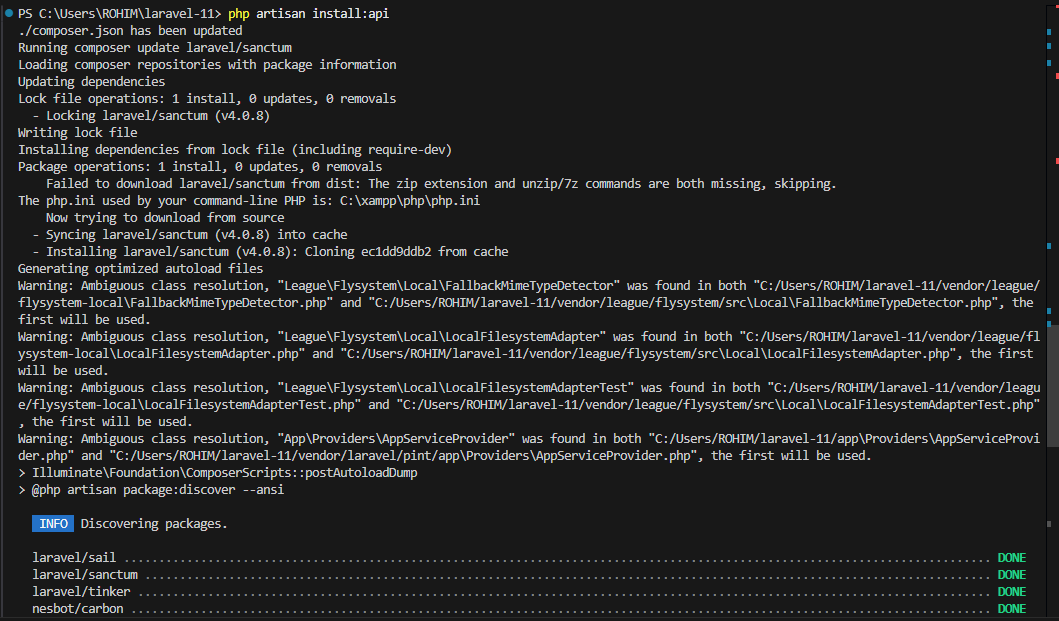
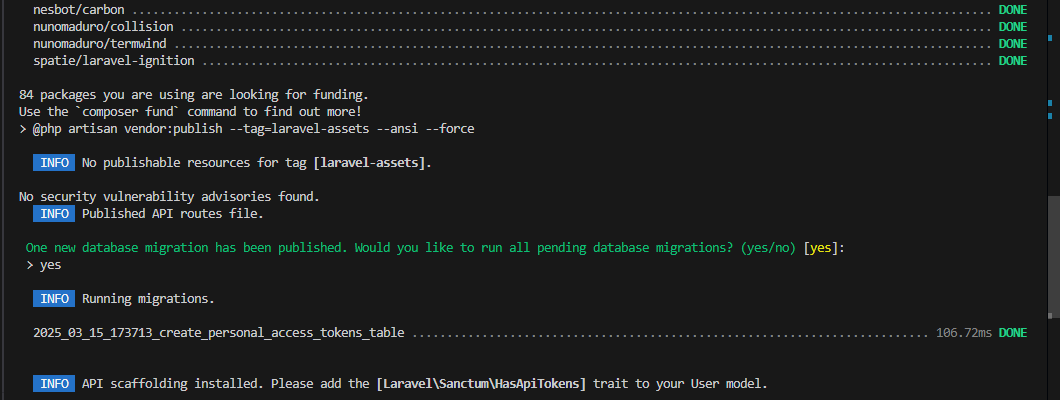
    return response()->json(['message' => 'Deleted successfully'], 204);

}

}

**Buat route khusus API dengan menjalankan perintah :**

**php artisan install:api**

**** ****

**Buka file routes/api.php  dan ubah isi file menjadi :**

<?php

use Illuminate\Auth\Middleware\Authenticate;

use Illuminate\Http\Request;

use Illuminate\Support\Facades\Route;

Route::get('/user', function (Request $request) {

    return $request->user();

})->middleware(Authenticate::using('sanctum'));

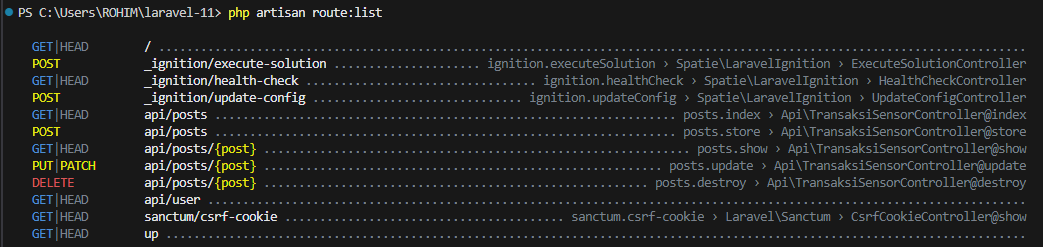
//posts

Route::apiResource('/posts', App\Http\Controllers\Api\TransaksiSensorController::class);

**Kemudian pastikan routes telah terbentuk dengan menjalankan perintah :**

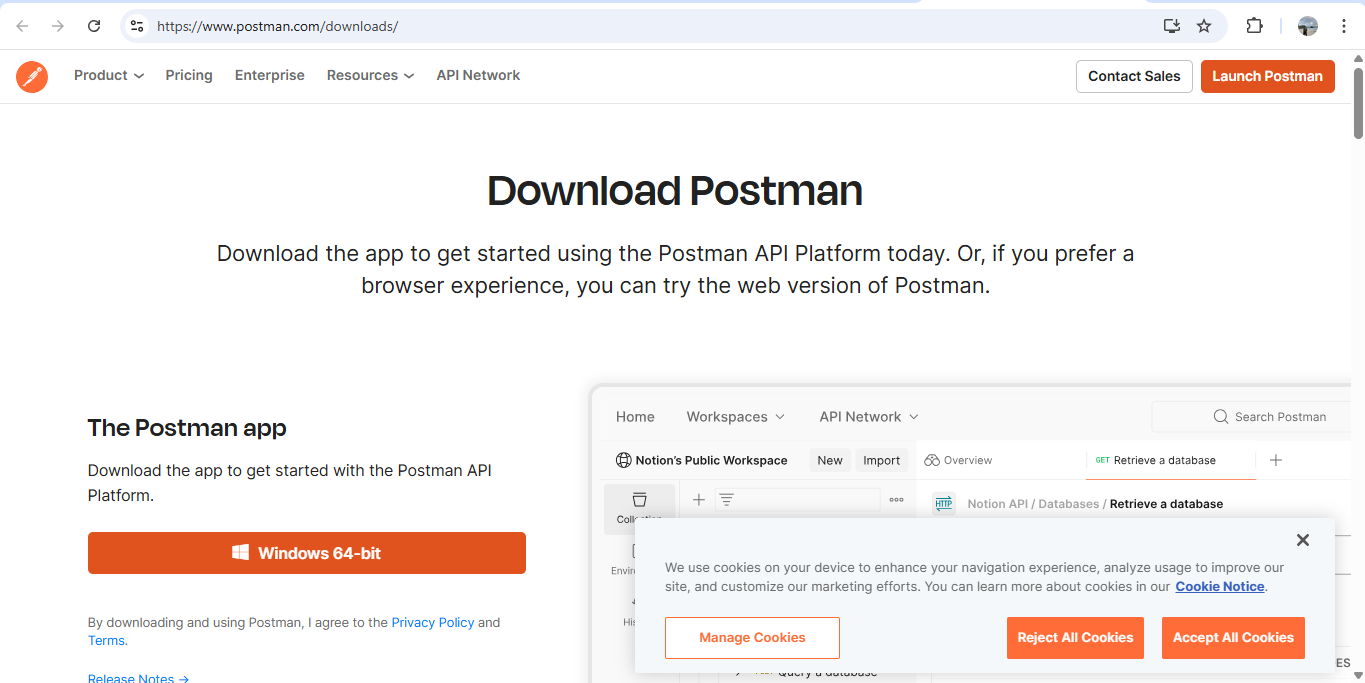
**php artisan route:list**

**Pastikan tampilan seperti di bawah ini**

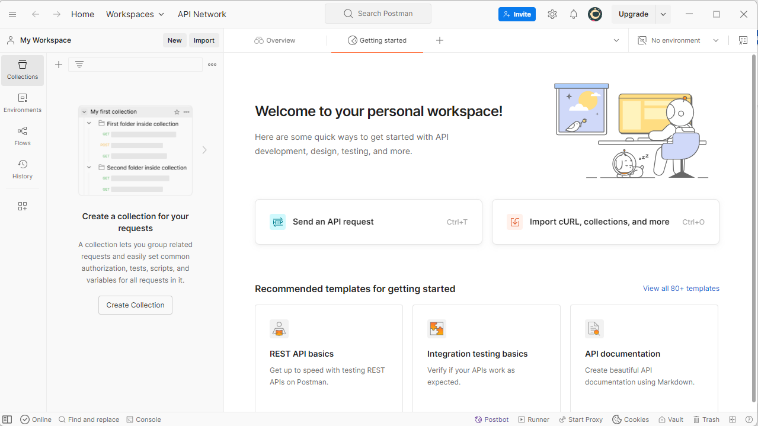
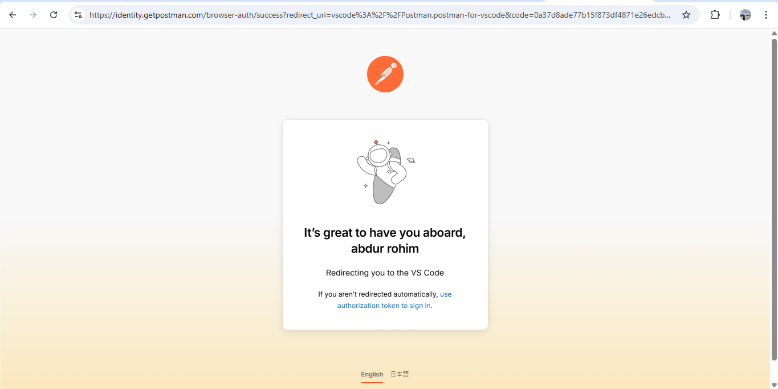
****

**Untuk melakukan testing, gunakan tools postman dengan langkah sebagai berikut :**

Download aplikasi postman pada link berikut :<https://www.postman.com/downloads/>

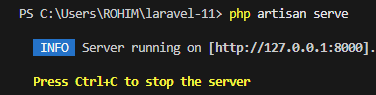
****

**Lakukan prosedur instalasi dan jalankan aplikasi postman**

****

**Untuk melakukan percobaan akses api, pastikan aplikasi laravel dijalankan dengan perintah :**

**php artisan serve**

****

Pastikan telah data yang dimasukkan kedalam tabel di database. Pada contoh berikut, telah ada 2 baris data pada **tabel transaksi\_sensor** pada **database iot\_25**

**Buat Tabel transaksi\_sensor**

CREATE TABLE IF NOT EXISTS transaksi\_sensor (

id INT AUTO\_INCREMENT PRIMARY KEY,

nama\_sensor VARCHAR(50) NOT NULL,

nilai1 INT NOT NULL,

nilai2 INT NOT NULL,

created\_at DATETIME DEFAULT CURRENT\_TIMESTAMP,

updated\_at DATETIME DEFAULT CURRENT\_TIMESTAMP ON UPDATE CURRENT\_TIMESTAMP

);

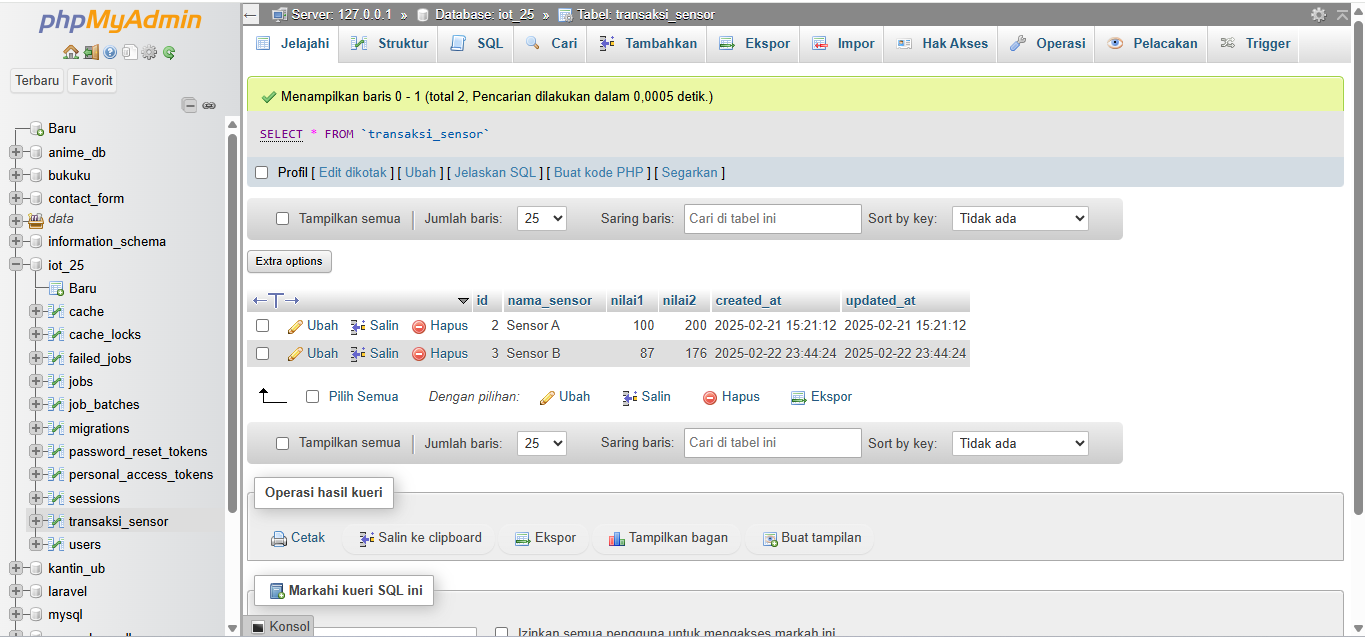
**Tambahkan data**

INSERT INTO transaksi\_sensor (id, nama\_sensor, nilai1, nilai2, created\_at, updated\_at)

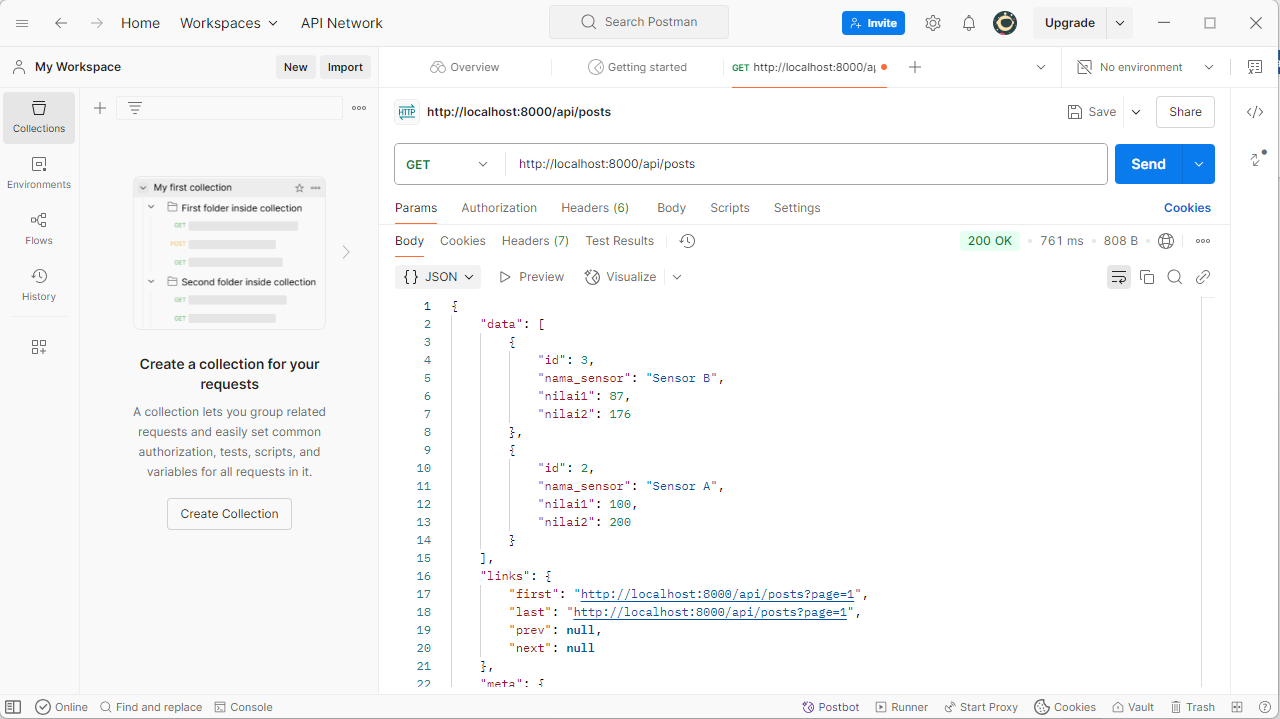
VALUES

(2, 'Sensor A', 100, 200, '2025-02-21 15:21:12', '2025-02-21 15:21:12'),

(3, 'Sensor B', 87, 176, '2025-02-22 23:44:24', '2025-02-22 23:44:24');

****

Untuk mengambil data diatas melalui aplikasi postman, jalankan prosedur berikut :

****

Pada bagian URL masukkan alamat server laravel <http://127.0.0.1:8000/api/posts>

Atau bisa diakses melalui url : <http://localhost:8000/api/posts>

Pilih method **GET** untuk mengambil data dari database , kemudian klik tombol **SEND**

Pastikan data dikembalikan dalam bentuk json seperti tampilan contoh diatas

{

    "data": [

        {

            "id": 3,

            "nama\_sensor": "Sensor B",

            "nilai1": 87,

            "nilai2": 176

        },

        {

            "id": 2,

            "nama\_sensor": "Sensor A",

            "nilai1": 100,

            "nilai2": 200

        }

    ],

    "links": {

        "first": "http://localhost:8000/api/posts?page=1",

        "last": "http://localhost:8000/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://localhost:8000/api/posts?page=1",

                "label": "1",

                "active": true

            },

            {

                "url": null,

                "label": "Next &raquo;",

                "active": false

            }

        ],

        "path": "http://localhost:8000/api/posts",

        "per\_page": 5,

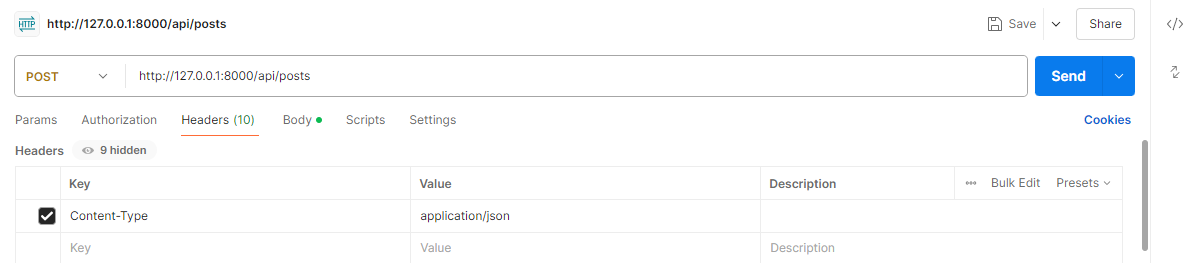
        "to": 2,

        "total": 2

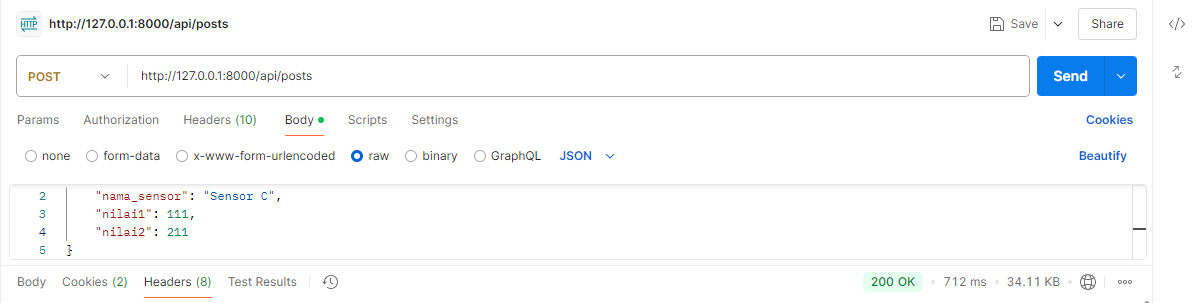
    }

}

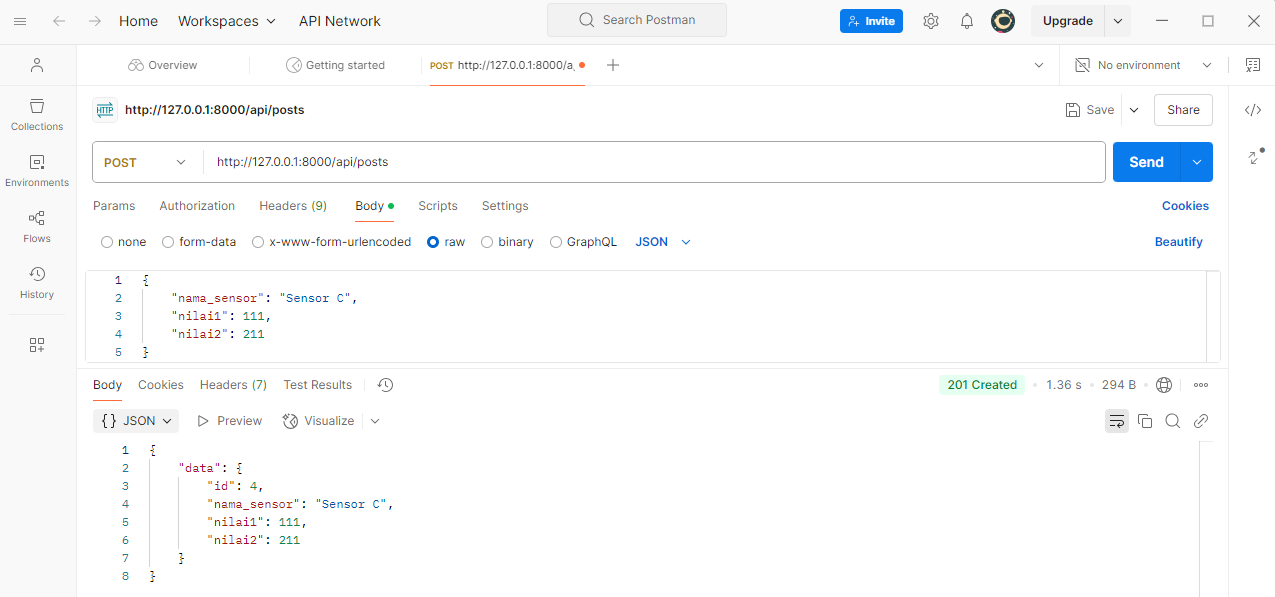
API telah berfungsi untuk mengambil data dari database. Langkah berikutnya adalah melakukan percobaan insert data ke tabel di database menggunakan API. Caranya adalah mengganti method menjadi POST kemudian pada bagian header ubah menjadi sebagai berikut

****

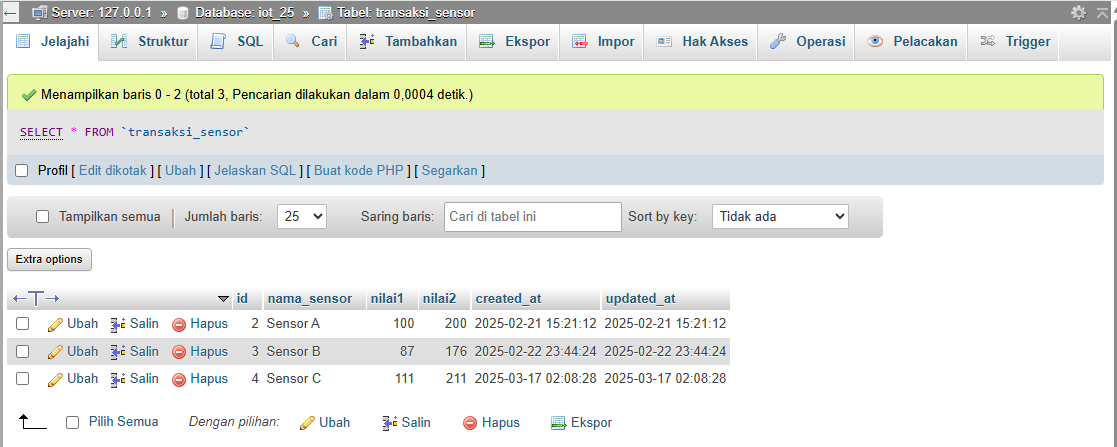
Pada bagian body ubah menjadi sebagai berikut



Kemudian klik send. Pastikan data berhasil di-insert kedatabase seperti tampilan berikut



Check manual di phpmyadmin, pastikan data baru masuk

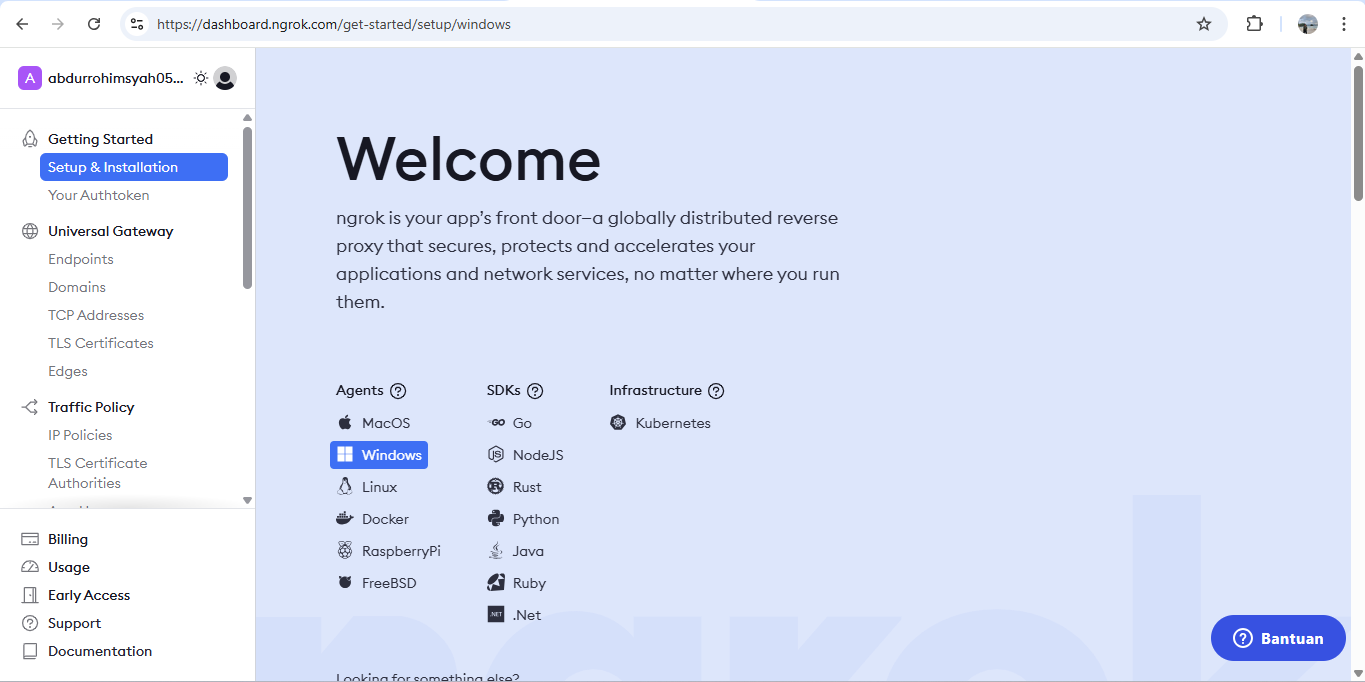


Langkah berikutnya adalah mengonline-kan API menggunakan service ngrok sehingga API dapat diakses melalui device iot atau simulasi wokwi iot.

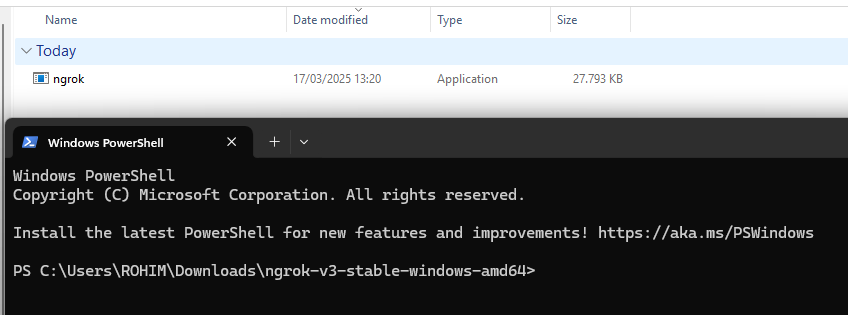
Download dan install aplikasi ngrok pada URL   :  <https://dashboard.ngrok.com/signup>

kemudian lakukan registrasi.

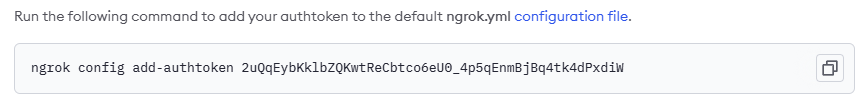
Login ke web ngrok, kemudian download aplikasi ngrok sesuai sistem operasi

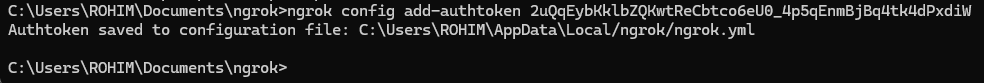


Lakukan ekstraksi dan buka command prompt dari alamat folder ekstraksi seperti berikut :



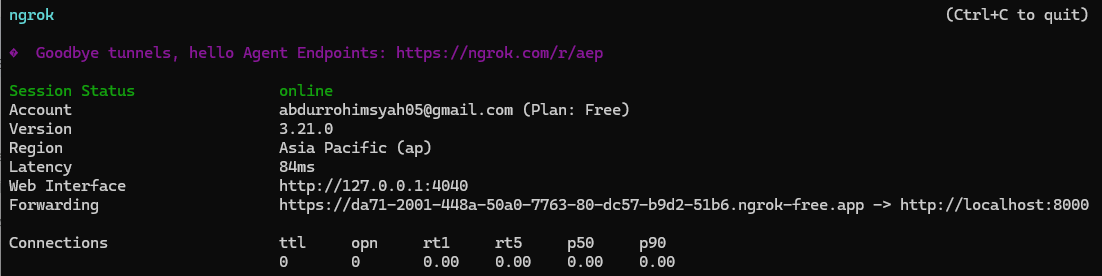
Kemudian jalankan perintah  sesuai yang ada di akun ngrok :





Kemudian jalankan perintah berikut untuk mengonline kan laravel melalui port 8000

**ngrok http** [**http://localhost:8000**](http://localhost:8080)

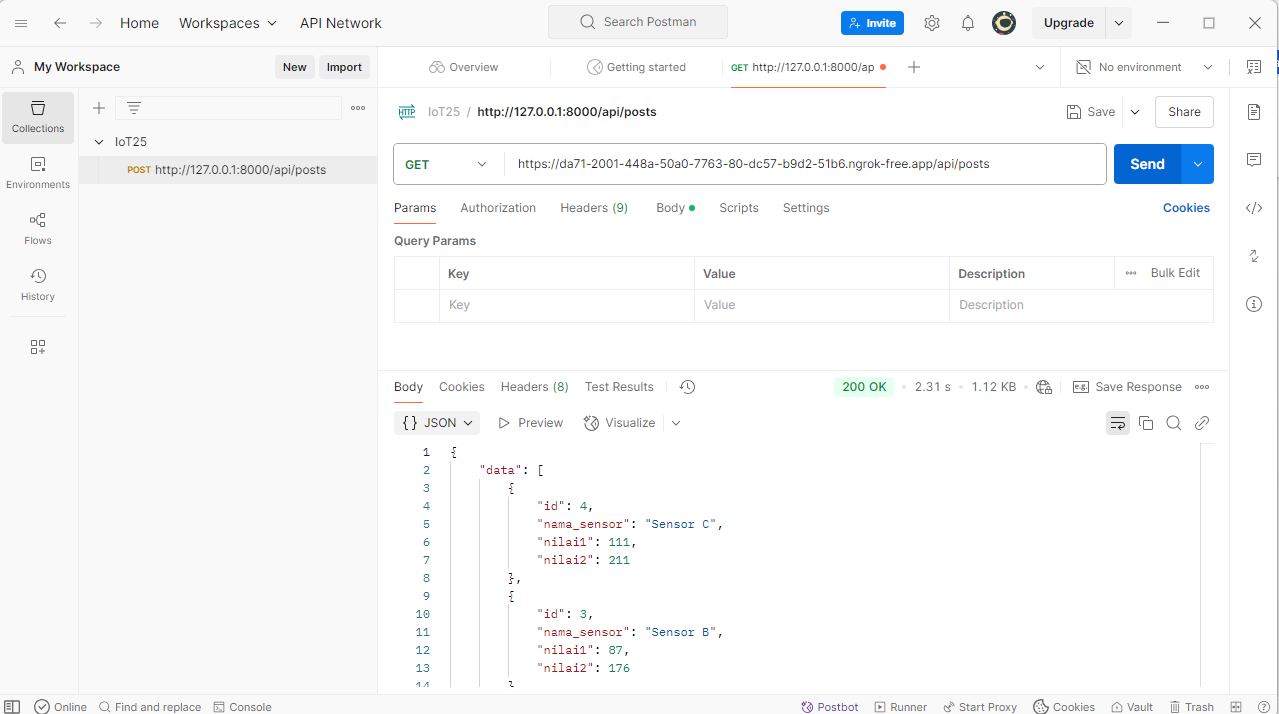


Kemudian lakukan percobaan menggunakan postman menggunakan URL yang diberikan oleh ngrok. Pada contoh ini, ngrok memberikan URL publik yang dapat diakses melalui internet pada alamat , sesuaikan dengan milik Anda

<https://da71-2001-448a-50a0-7763-80-dc57-b9d2-51b6.ngrok-free.app>

Untuk melakukan percobaan GET api , maka URL harus ditambahkan alamat endpoint  menjadi sebagai berikut

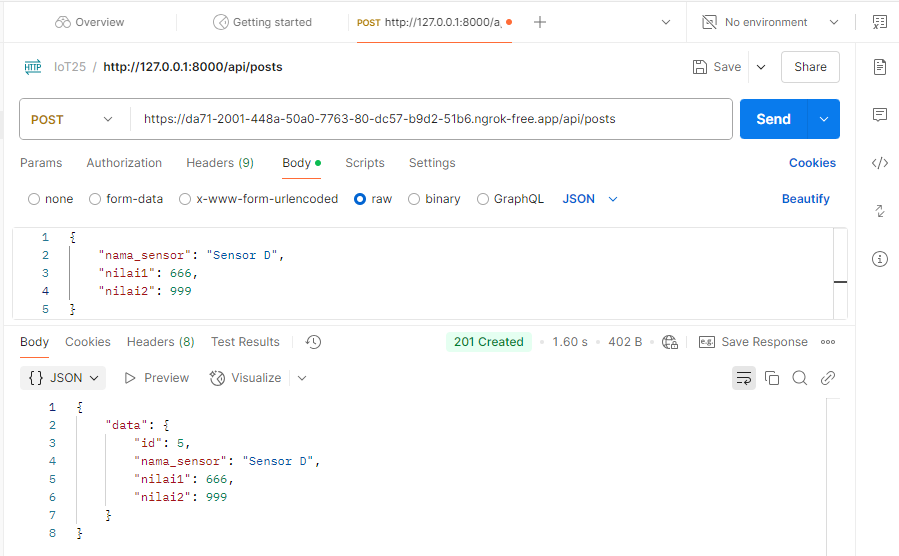
<https://da71-2001-448a-50a0-7763-80-dc57-b9d2-51b6.ngrok-free.app/api/posts>



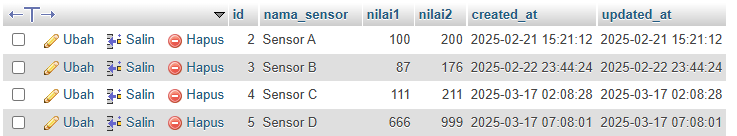
Berikutnya lakukan percobaan melakukan insert data baru melalui API

<https://da71-2001-448a-50a0-7763-80-dc57-b9d2-51b6.ngrok-free.app/api/posts>

Ubah method menjadi POST dan parameter header dan body sesuaikan



Sampai disini API yang dibangun menggunakan laravel sudah dapat berjalan dengan baik dan dapat diakses melalui URL publik.



1. **Result and Discussion (Hasil dan Pembahasan)**
   1. **Experimental Results (Hasil Eksperimen)**

Eksperimen pembuatan API menggunakan Laravel 11 dan Ngrok berhasil dilakukan dengan hasil sebagai berikut:

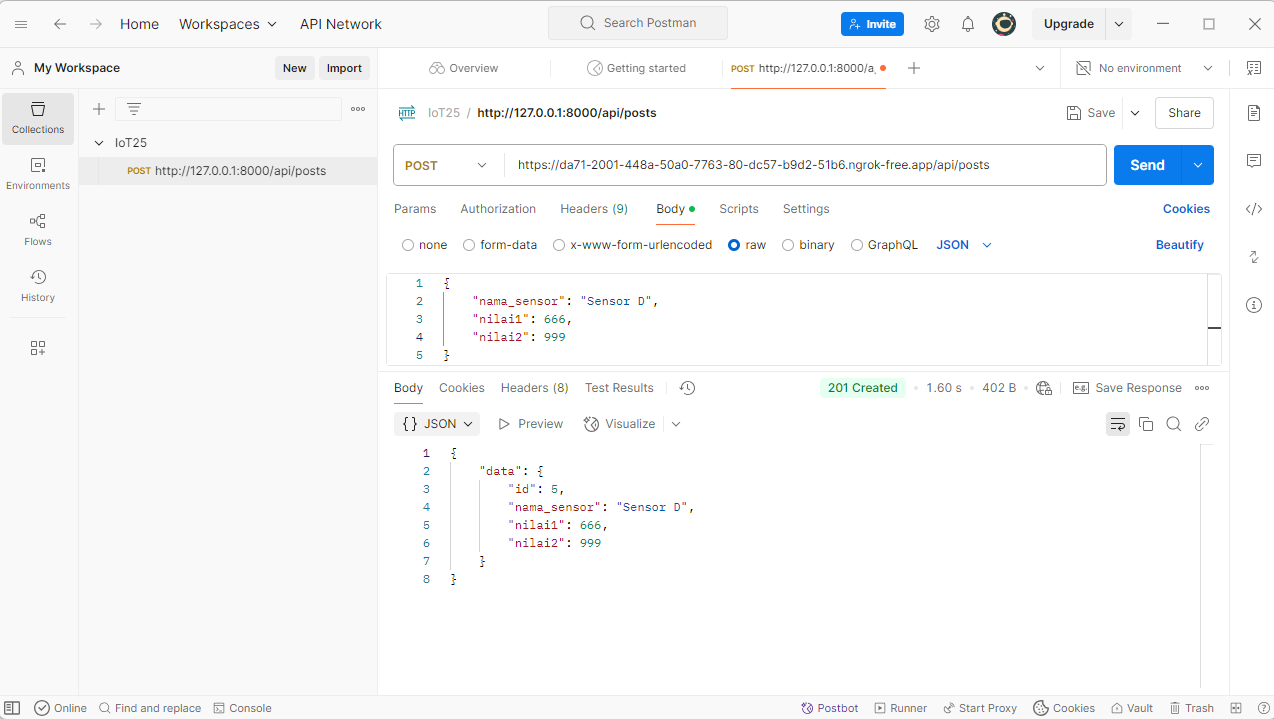
1. **Instalasi dan Konfigurasi Laravel:**
   * Laravel 11 berhasil diinstal dan dijalankan pada localhost menggunakan perintah php artisan serve. Aplikasi berjalan pada port default http://127.0.0.1:8000.
   * Database berhasil dikonfigurasi menggunakan XAMPP dengan nama database iot\_25. Tabel transaksi\_sensor dibuat dengan struktur yang sesuai, termasuk kolom id, nama\_sensor, nilai1, dan nilai2.
2. **Pembuatan API:**
   * Endpoint CRUD (Create, Read, Update, Delete) berhasil dibuat menggunakan Laravel Resource Controller (TransaksiSensorController).
   * Data dari tabel transaksi\_sensor dapat diakses melalui endpoint API menggunakan metode HTTP berikut:
     + **GET** untuk membaca data.
     + **POST** untuk menambahkan data baru.
     + **PUT** untuk memperbarui data.
     + **DELETE** untuk menghapus data.
3. **Pengujian API:**
   * Pengujian dilakukan menggunakan Postman:
     + Endpoint GET (http://127.0.0.1:8000/api/posts) berhasil mengembalikan data dalam format JSON.
     + Endpoint POST berhasil menambahkan data baru ke tabel transaksi\_sensor.
     + Data yang dimasukkan melalui API diverifikasi secara manual di phpMyAdmin.
4. **Integrasi dengan Ngrok:**
   * Ngrok digunakan untuk membuat API dapat diakses secara publik melalui URL yang diberikan oleh Ngrok (contoh: https://da71-2001-448a.ngrok-free.app).
   * Endpoint publik berhasil diuji menggunakan Postman, baik untuk metode GET maupun POST.

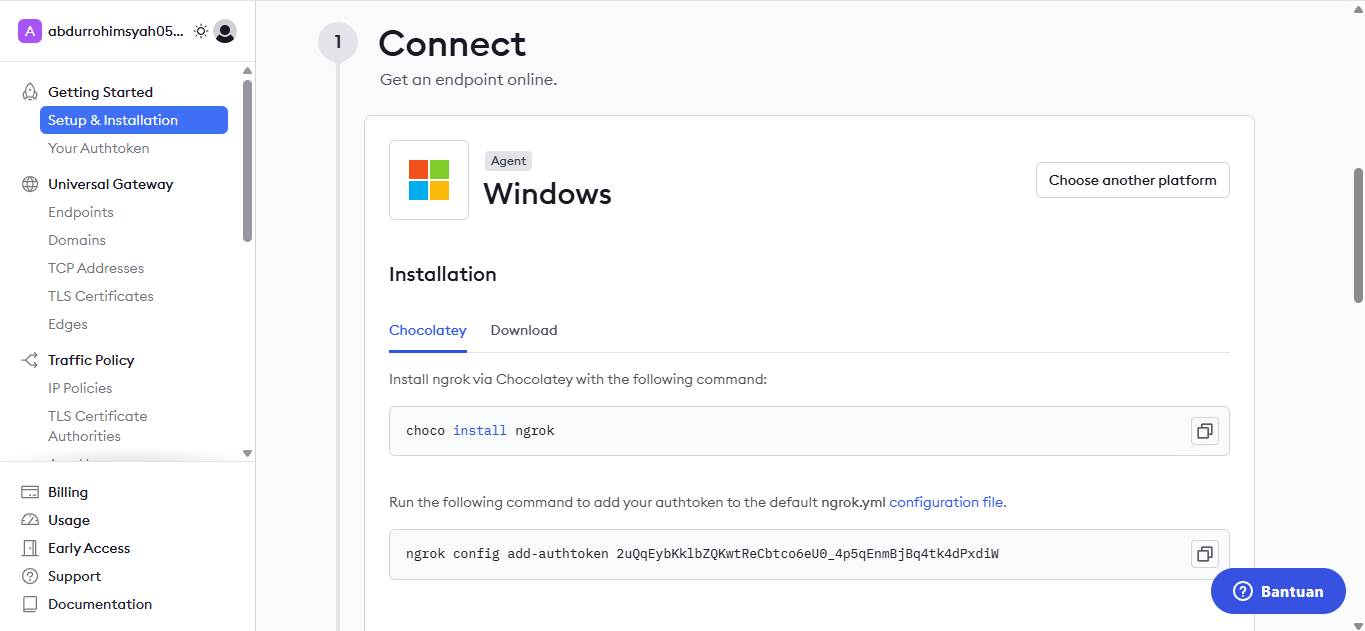
**Pembahasan**

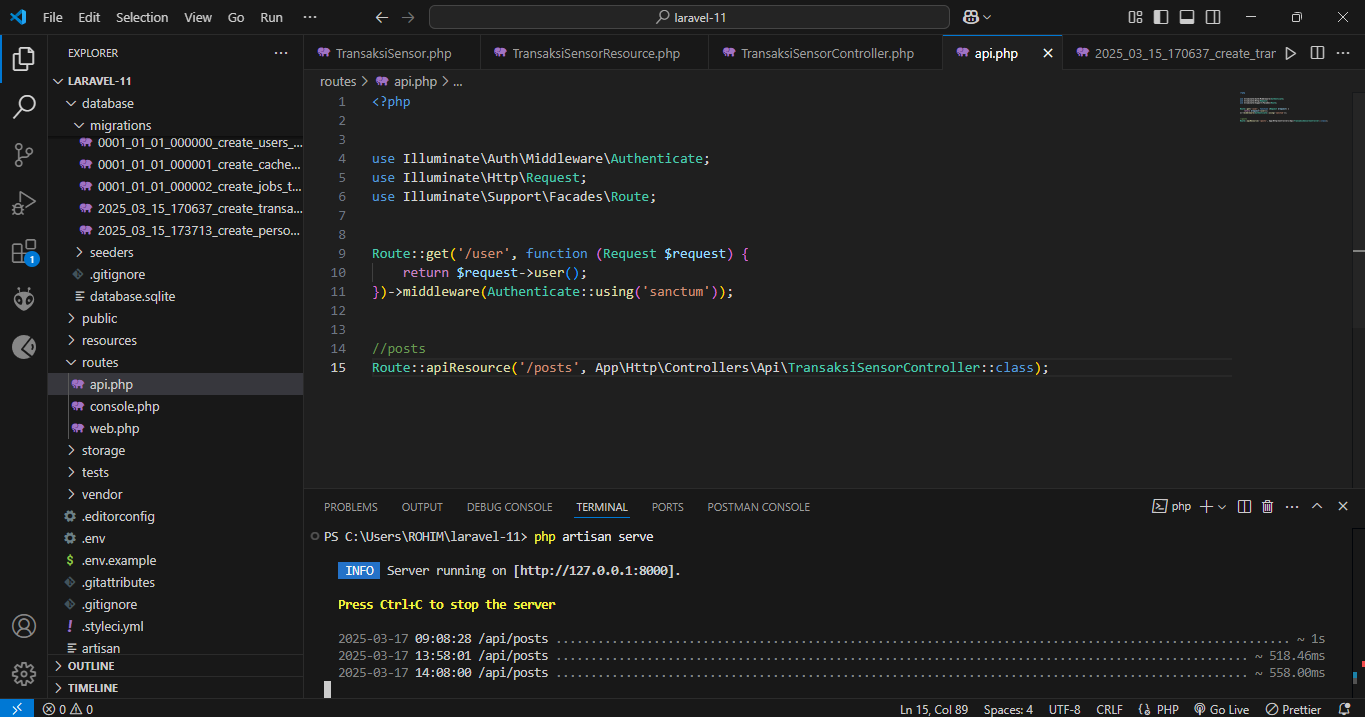
1. **Keberhasilan Implementasi:**
   * Framework Laravel 11 memberikan kemudahan dalam membangun API RESTful dengan fitur bawaan seperti Artisan Command dan Resource Controller.
   * Integrasi dengan Ngrok memungkinkan pengembang untuk mengakses API dari perangkat lain tanpa memerlukan konfigurasi server tambahan, sehingga sangat berguna untuk pengujian dan kolaborasi tim.
2. **Efisiensi Pengembangan:**
   * Dengan memanfaatkan Laravel Resource Controller, proses pembuatan endpoint CRUD menjadi lebih cepat karena otomatisasi sebagian besar kode boilerplate.
   * Penggunaan Postman sebagai alat pengujian mempermudah validasi fungsi API secara real-time.
3. **Tantangan yang Dihadapi:**
   * Sinkronisasi antara konfigurasi database di Laravel (.env) dan phpMyAdmin perlu dilakukan dengan hati-hati untuk menghindari kesalahan koneksi.
   * Saat menggunakan Ngrok, URL publik yang dihasilkan bersifat sementara sehingga perlu diperbarui setiap kali sesi baru dimulai.
4. **Potensi Pengembangan Lebih Lanjut:**
   * Menambahkan autentikasi menggunakan Laravel Sanctum atau Passport untuk meningkatkan keamanan API.
   * Mengintegrasikan API dengan perangkat IoT atau aplikasi frontend untuk simulasi kasus penggunaan nyata.
   * Menggunakan layanan hosting permanen untuk membuat API dapat diakses tanpa batasan waktu seperti pada Ngrok.
   1. **Kesimpulan**

Eksperimen ini menunjukkan bahwa pembuatan API menggunakan Laravel 11 dan pengintegrasian dengan Ngrok dapat dilakukan dengan mudah dan efisien. Hasilnya adalah sebuah API RESTful yang dapat diakses secara lokal maupun publik, mendukung operasi CRUD, serta memungkinkan pengujian real-time melalui Postman.

1. **Appendix**

****

****

****