LAPORAN PRAKTIKUM INTERNET OF THINGS (IoT)

Fakultas Vokasi, Universitas Brawijaya

**Praktikum Pembuatan API menggunakan Laravel 11 & Ngrok**

****

*Abdur Rohman Syah Sjadja’ah*

Fakultas Vokasi, Universitas Brawijaya

Email: [abdurrohmansyh02@gmail.com](mailto:abdurrohmansyh02@gmail.com)

**Abstract** (Abstrak)

Praktikum ini bertujuan untuk memberikan pemahaman mendalam kepada mahasiswa tentang cara untuk menghubungkan API menggunakan laravel 11 dan mengintegrasikannya dengan Ngrok. Dalam praktikum ini, mahasiswa akan mempelajari cara membangun API RESTful dengan Laravel 11 hingga melakukan pengujian menggunakan Ngrok untuk akses server lokal secara publik. Ngrok memungkinkan peserta untuk menghubungkan server lokal ke internet sehingga API dapat diuji secara langsung tanpa harus melakukan deploy ke server produksi. 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. Praktikum ini juga mencakup penerapan autentikasi berbasis token menggunakan Laravel Sanctum untuk memastikan keamanan akses API. Dengan pendekatan yang sistematis, praktikum ini diharapkan dapat meningkatkan kemampuan peserta dalam membangun aplikasi modern berbasis web yang scalable dan aman

.***Kata kunc :*** *API, Ngrok, Laravel 11, VSCode, Postman*.

1. **Introduction** (Pendahuluan)
   1. **Latar Belakang**

Dengan pesatnya perkembangan teknologi saat ini keterampilan dalam pemrograman, dan manajemen proyek semakin penting, terutama dalam pengembangan system IoT. Contohnya seperti API, API adalah serangkaian peraturan dan protokol yang memungkinkan aplikasi atau layanan untuk berkomunikasi dan bertukar data. 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, mahasiswa dapat menuliskan progam file laravel.php dan menjalankan server serta melakukan pengujian melalui Postman.

Melalui praktikum ini, pengguna akan mempelajari cara menghubungkan API menggunakan Laravel 11 dan mengujinya menggunakan Postman dan XAMPP sebagai databasenya.

* 1. **Tujuan Eksperimen**

Praktikum ini bertujuan untuk mengetahui langkah-langkah cara membuat API menggunakan Laravel 11 dan menggunakan Ngrok sebagai terowongan aman untuk mengakses server local ke internet. Dengan Postman mahasiswa dapat melakukan pengujian API untuk mengetahui apakah API sudah terhubung dengan server.

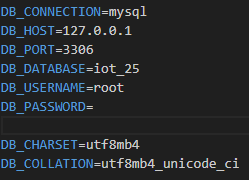
1. **Methodology** (Metodologi)
   1. **Tools & Materials** (Alat & Bahan)

* Laptop
* Internet
* Web Browser (Chrome)
* Postman
* Visual Studio Code
* CMD
* XAMPP
* Composer
* Ngrok
  1. **Implementation Steps** (Langkah Implementasi)
* Instalasi Ngrok (<https://ngrok.com/downloads/windows?tab=download>)
* 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].
* Konfigurasi Database
* Buka XAMPP dan jalankan Apache serta MySQL
* Buat database di phpmyadmin dengan nama **iot\_25**

Sebuah gambar berisi teks, cuplikan layar, Font, garis

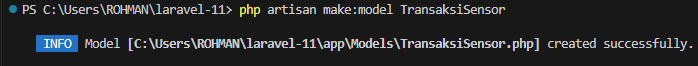
Konten yang dihasilkan AI mungkin salah.

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

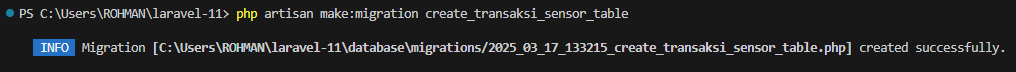


Buat file model  **TransaksiSensor.php** dengan cara menjalankan perintah :

**php artisan make:model TransaksiSensor**

****

Buat migrasinya juga dengan menjalankan perintah **php artisan make:migration create\_transaksi\_sensor\_table**

****

Kemudian ubah isi file **database/migrations/2025\_03\_15\_170637\_create\_transaksi\_sensors\_table.php** dengan kode berikut

<?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.

     \*/

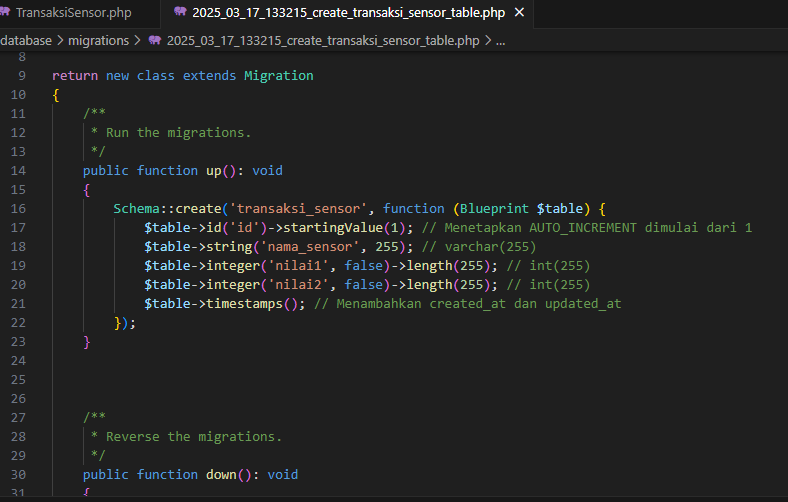
    public function down(): void

    {

        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.

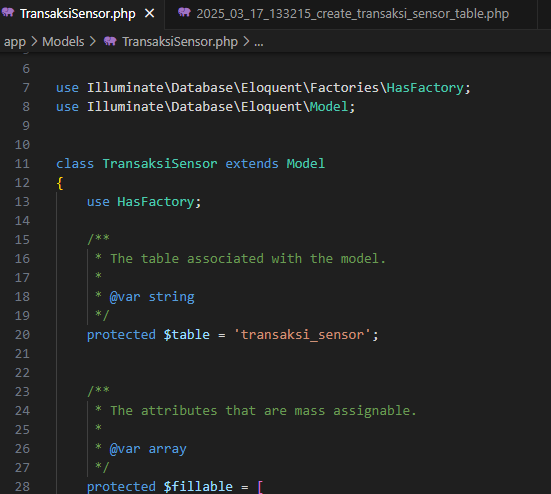
     \*

     \* @var array

     \*/

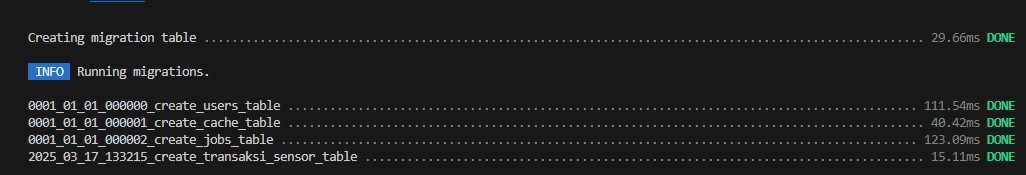
    protected $casts = [];

}



Kemudian jalankan perintah berikut untuk membuat tabel :

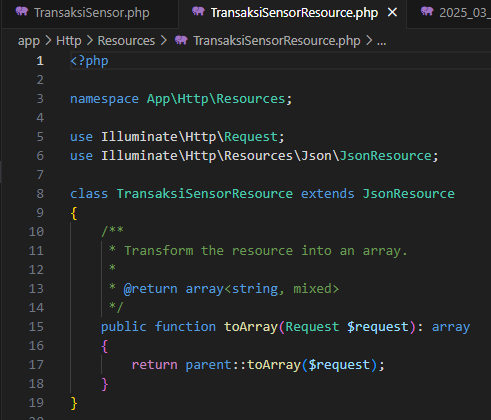
**php artisan migrate**



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,

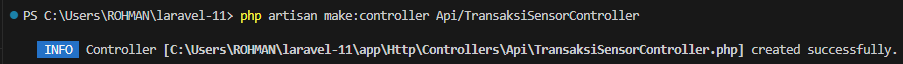
        ];

    }

}

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();

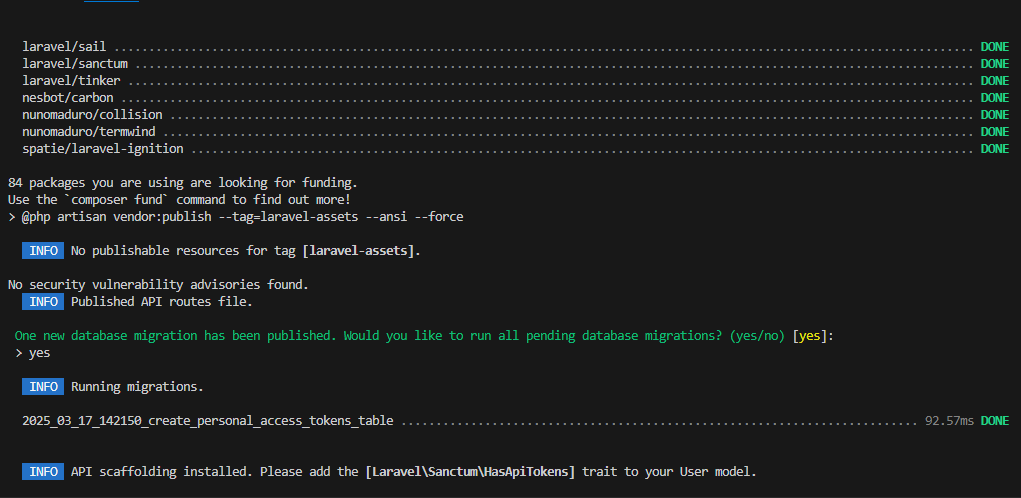
    return response()->json(['message' => 'Deleted successfully'], 204);

}

}

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

**php artisan install:api**



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

<?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'));

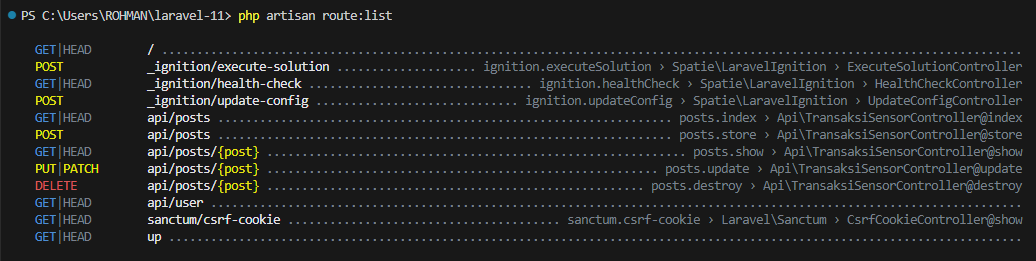
//posts

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

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

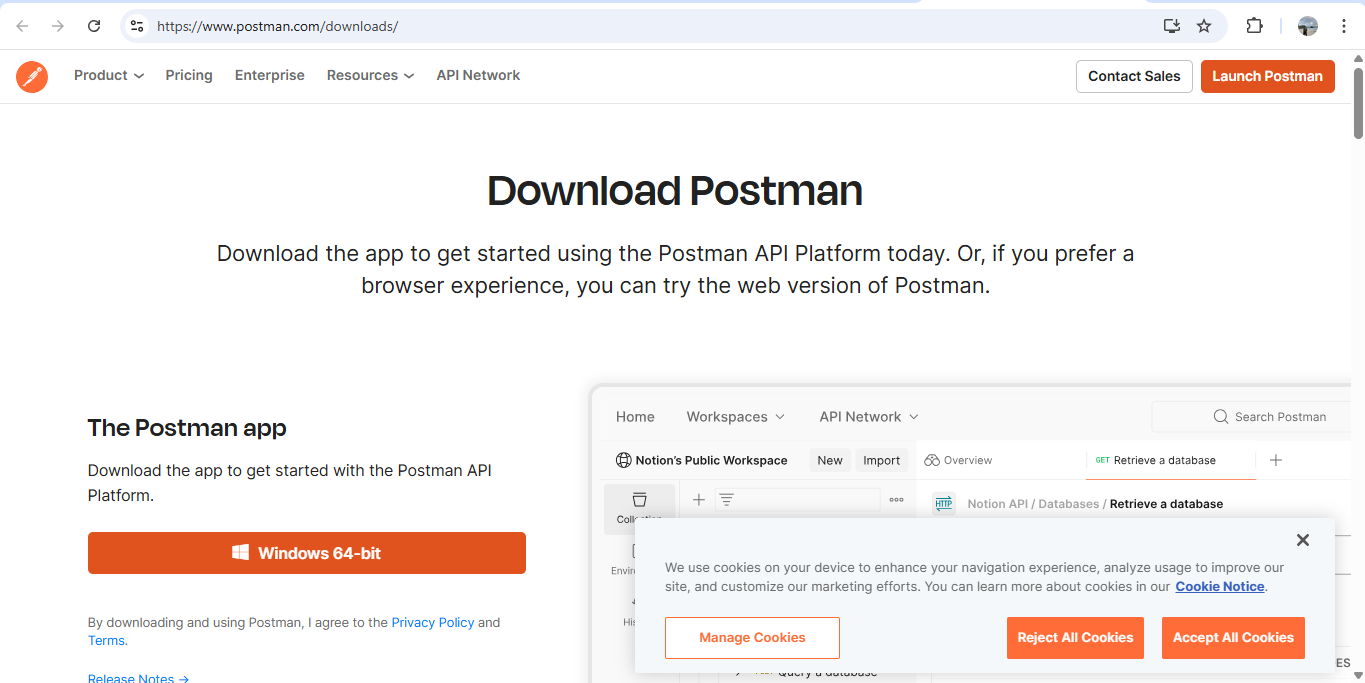
**php artisan route:list**

**Pastikan tampilan seperti di bawah ini!**

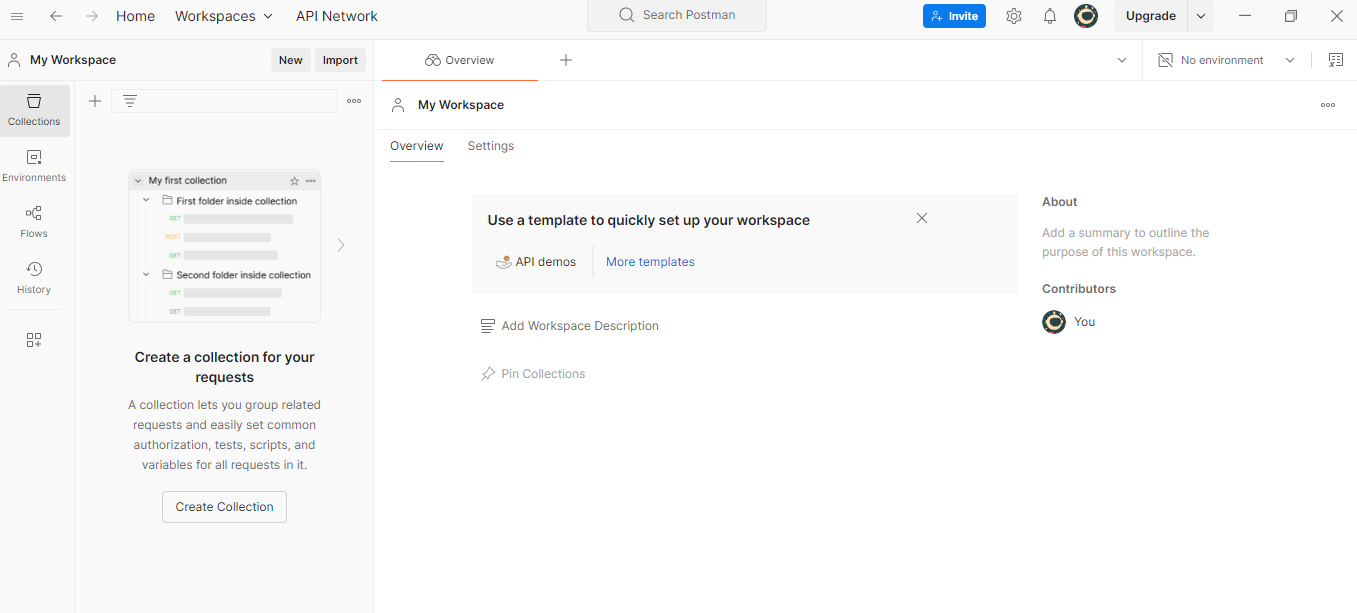
****

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

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

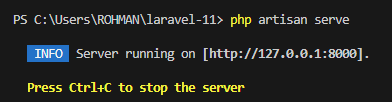
****

**Lakukan prosedur instalasi dan jalankan aplikasi postman**

****

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

**php artisan serve**

****

Pastikan telah ada data yang dimasukkan kedalam tabel di database. Jika belum, buat data terlebih dahulu di table transaksi\_sensor

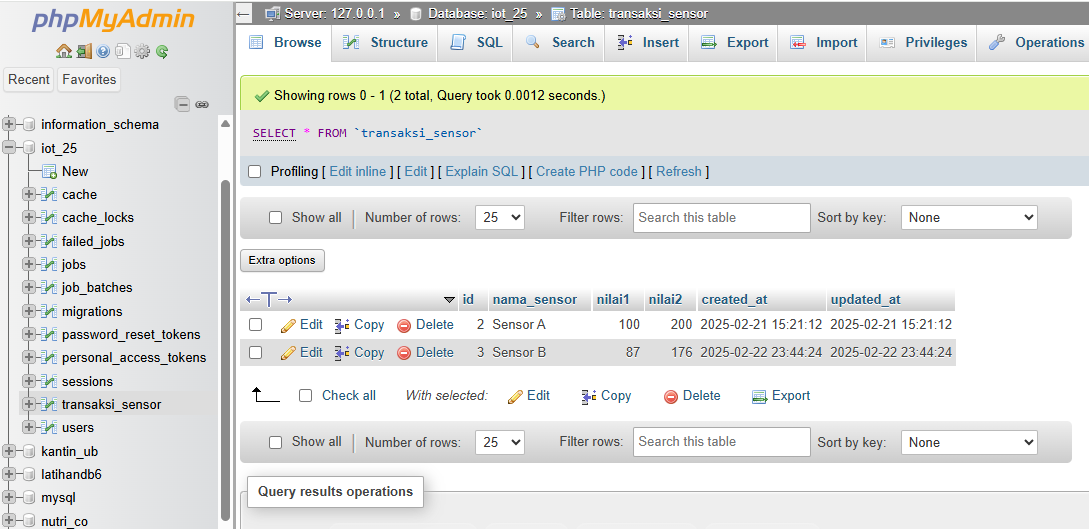
**Tambahkan data**

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

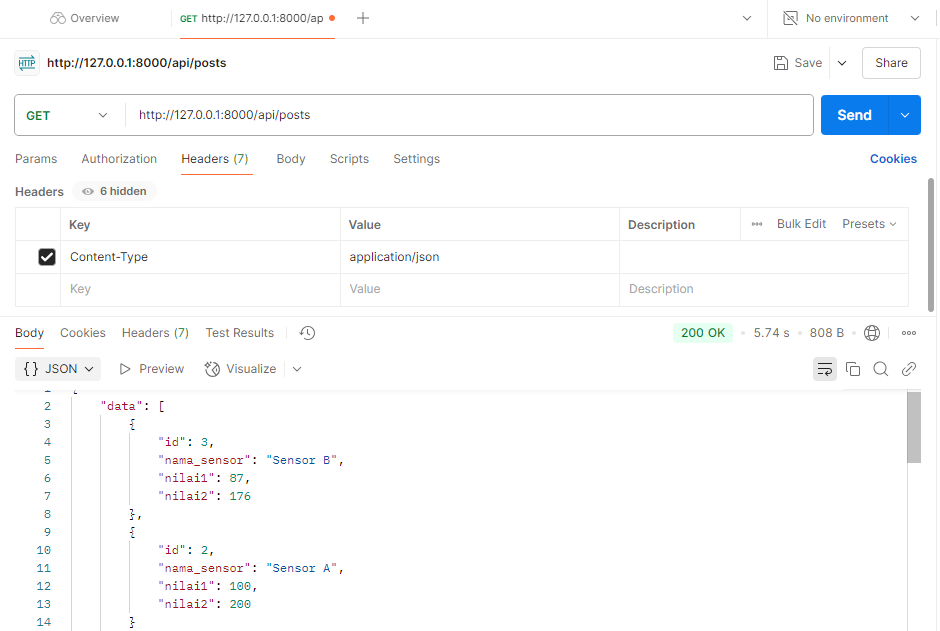
VALUES

(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://127.0.0.1:8000/api/posts?page=1",

        "last": "http://127.0.0.1:8000/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://127.0.0.1:8000/api/posts?page=1",

                "label": "1",

                "active": **true**

            },

            {

                "url": **null**,

                "label": "Next &raquo;",

                "active": **false**

            }

        ],

        "path": "http://127.0.0.1:8000/api/posts",

        "per\_page": 5,

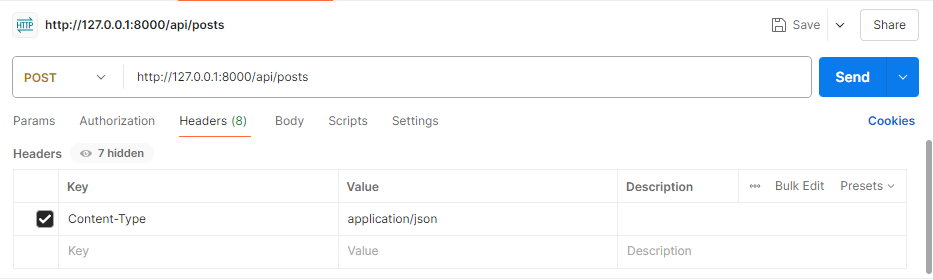
        "to": 2,

        "total": 2

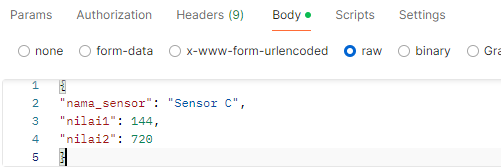
    }

}

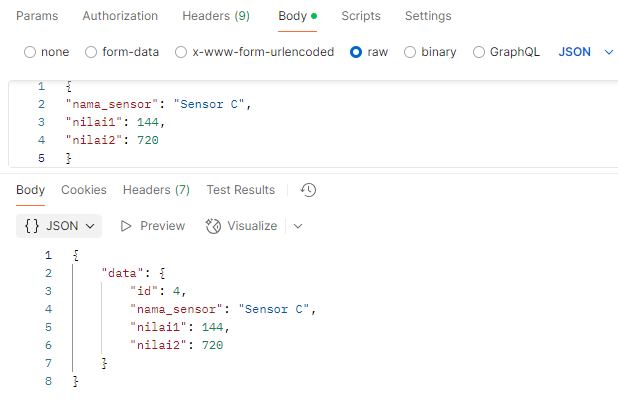
API telah berfungsi untuk mengambil data dari database. Langkah berikutnya adalah melakukan percobaan insert data ke tabel di database menggunakan API. Caranya adalah mengganti method menjadi POST kemudian pada bagian header ubah menjadi sebagai berikut



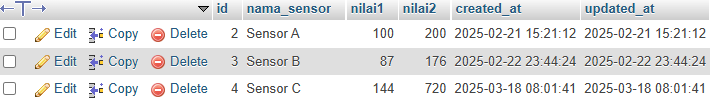
Pada bagian body ubah menjadi sebagai berikut



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



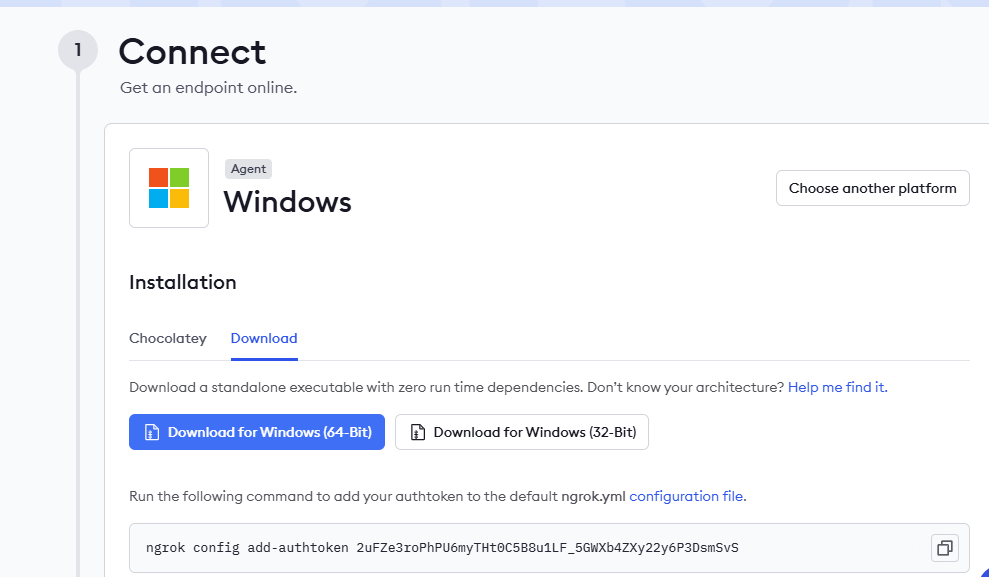
Check manual di phpmyadmin, pastikan data baru masuk



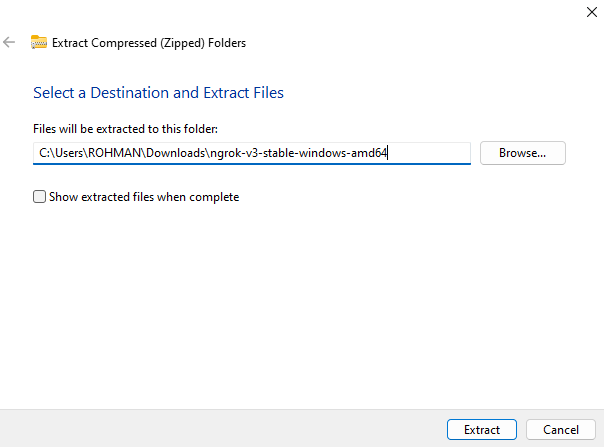
Langkah berikutnya adalah mengonline-kan API menggunakan service ngrok sehingga API dapat diakses melalui device iot atau simulasi wokwi iot (secara Internet).

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

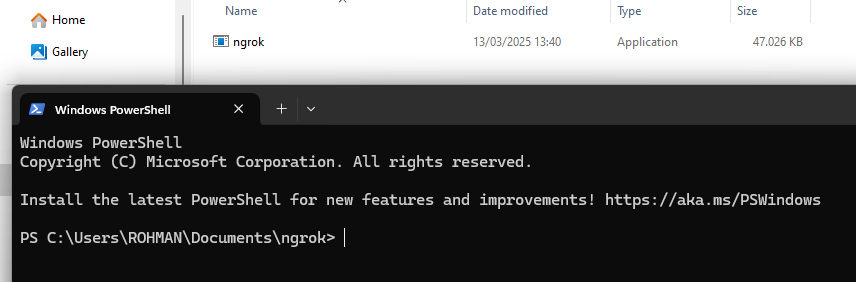
Login ke web ngrok, kemudian download aplikasi ngrok sesuai sistem operasi



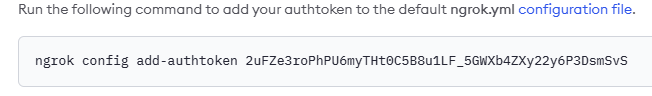
Lakukan ekstraksi

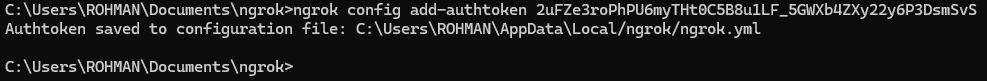


Buka command prompt dari alamat folder ekstraksi seperti berikut



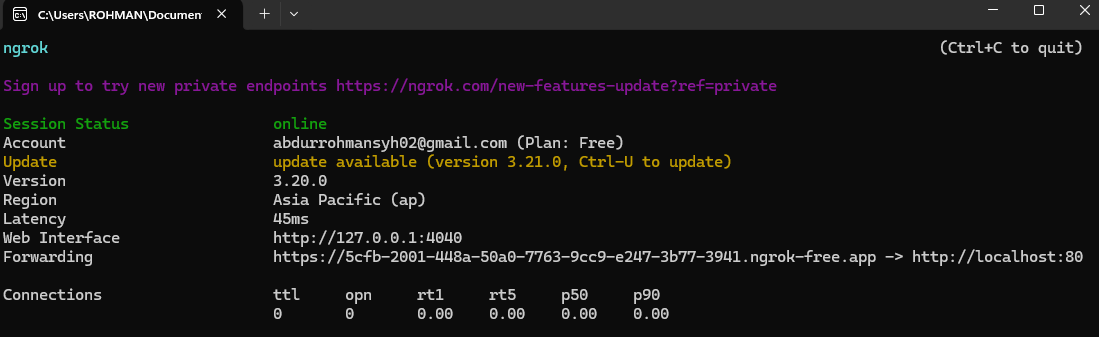
Kemudian jalankan perintah  sesuai yang ada di akun ngrok :





Kemudian jalankan perintah berikut untuk mengonline kan laravel melalui port 8000

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

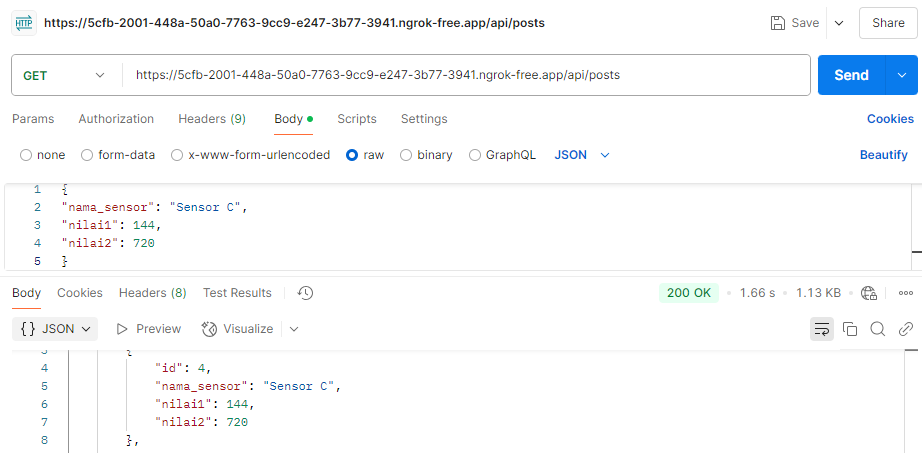


Kemudian lakukan percobaan menggunakan postman menggunakan URL yang diberikan oleh ngrok. Pada contoh ini, ngrok memberikan URL publik yang dapat diakses melalui internet pada alamat , sesuaikan dengan milik Anda .

<https://5cfb-2001-448a-50a0-7763-9cc9-e247-3b77-3941.ngrok-free.app>

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

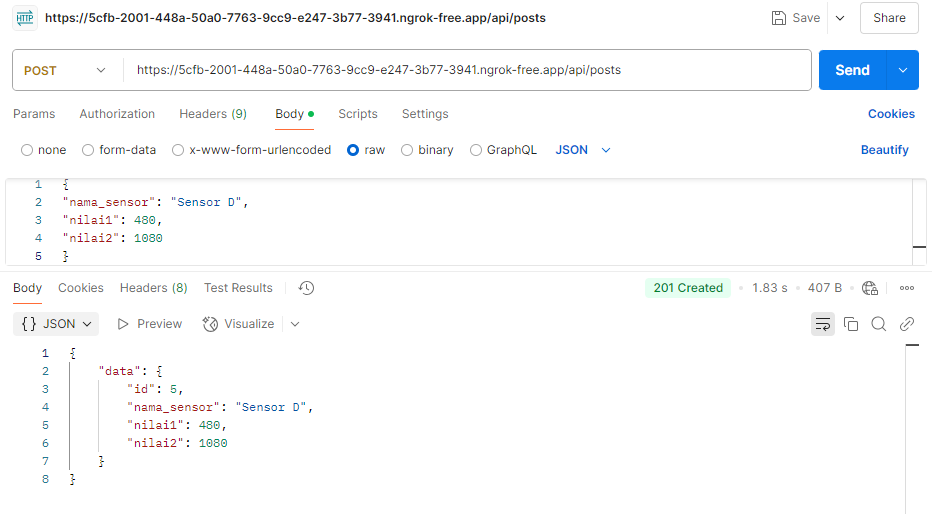
<https://5cfb-2001-448a-50a0-7763-9cc9-e247-3b77-3941.ngrok-free.app>/**api/posts**



Berikutnya lakukan percobaan melakukan insert data baru melalui API

<https://5cfb-2001-448a-50a0-7763-9cc9-e247-3b77-3941.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. Pada bab berikutnya,  akses API akan dilakukan melalui simulator WOKWI ESP32 yang telah dibuat pada bab sebelumnya.

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

**3.1 Experimental Results (Hasil Eksperimen)**  
Praktikum pembuatan API menggunakan Laravel 11 dan Ngrok berhasil dilaksanakan dengan baik, menghasilkan API RESTful yang berfungsi penuh untuk mengelola data produk. Berikut adalah rincian hasil dari setiap tahap praktikum:

* **Pengaturan Lingkungan Pengembangan**: Mahasiswa berhasil menginstal Laravel 11 dan mengkonfigurasi XAMPP sebagai server database. Dengan menggunakan Visual Studio Code (VSCode), mahasiswa dapat menulis kode dan menjalankan server lokal dengan lancar.
* **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.

Endpoint-endpoint ini diuji menggunakan Postman, dan semua permintaan berhasil memberikan respons sesuai yang diharapkan.

* **Implementasi Autentikasi dengan Laravel Sanctum**: Mahasiswa berhasil menerapkan autentikasi berbasis token menggunakan Laravel Sanctum. Endpoint untuk login dan registrasi pengguna berfungsi dengan baik, memungkinkan pengguna untuk mendapatkan token akses yang diperlukan untuk mengakses endpoint yang dilindungi.
* **Pengujian Menggunakan Ngrok**: Dengan menggunakan Ngrok, mahasiswa dapat mengekspos server lokal mereka ke internet. URL publik yang dihasilkan memungkinkan pengujian API dari perangkat lain tanpa perlu melakukan deploy ke server produksi. Pengujian ini menunjukkan bahwa API dapat diakses secara remote, memberikan kemudahan dalam integrasi dengan aplikasi frontend atau mobile.
* **Pengujian Respons API dengan Postman**: Seluruh endpoint diuji menggunakan Postman untuk memastikan bahwa API merespons permintaan dengan benar. Hasil pengujian menunjukkan bahwa semua operasi CRUD berjalan sesuai harapan, dan data dikembalikan dalam format JSON yang sesuai.
* **Dokumentasi API Menggunakan Scribe**: Mahasiswa juga berhasil mengintegrasikan Scribe untuk mendokumentasikan API secara otomatis. Dokumentasi ini mencakup deskripsi endpoint, parameter yang diperlukan, serta contoh respons, sehingga memudahkan pengembang lain dalam menggunakan API yang telah dibuat.

**3.2 Kesimpulan**

Secara keseluruhan, hasil eksperimen menunjukkan bahwa mahasiswa mampu membangun dan menguji API RESTful menggunakan Laravel 11 dengan baik. Praktikum ini memberikan pengalaman praktis yang berharga dalam pengembangan aplikasi berbasis web modern serta pemahaman mendalam tentang pentingnya API dalam arsitektur aplikasi saat ini. Melalui penggunaan Ngrok, mahasiswa juga belajar bagaimana cara mengakses dan menguji aplikasi secara real-time dari lokasi yang berbeda, memperkuat keterampilan mereka dalam pengembangan perangkat lunak.

**Appendix** (Lampiran)

