# LAPORAN PRAKTIKUM INTERNET OF THINGS (IoT)

Fakultas Vokasi, Universitas Brawijaya



**PRAKTIKUM PEMBUATAN API**

**MENGGUNAKAN LARAVEL DAN NGROK**

**Author :**

**Ivan AryaPutra Rachmadhani / 233140700111109**

**Fakultas Vokasi, Universitas Brawijaya**

**Email : ivanarya990@gmail.com**

**Abstract** (Abstrak)

Praktikum ini bertujuan untuk memahami konsep, perancangan, serta implementasi API menggunakan framework Laravel serta menghubungkannya ke internet menggunakan Ngrok. Dalam eksperimen ini, Laravel digunakan sebagai backend API untuk mengelola data sensor yang meliputi operasi Create, Read, Update, dan Delete (CRUD). Ngrok digunakan untuk menyediakan akses publik terhadap API yang dibuat, memungkinkan API dapat diuji dan diakses dari luar jaringan lokal.

Proses praktikum dimulai dengan instalasi Laravel dan konfigurasi database MySQL untuk menyimpan data sensor. Selanjutnya, API dikembangkan menggunakan Laravel dengan pendekatan RESTful API dan diuji menggunakan Postman. Setelah API berhasil dibuat, Ngrok dikonfigurasi untuk menyediakan URL publik yang dapat digunakan untuk mengakses API dari mana saja.

Hasil eksperimen menunjukkan bahwa API yang dikembangkan mampu menangani berbagai request HTTP dengan baik, termasuk pengambilan data (GET), penyimpanan data (POST), pembaruan data (PUT), dan penghapusan data (DELETE). Dengan adanya integrasi antara Laravel dan Ngrok, API dapat diakses secara global tanpa memerlukan konfigurasi server tambahan.

**Keywords : Laravel, API, IoT, Ngrok, PHP**

**1. Introduction** (Pendahuluan)

**1.1 Latar belakang** praktikum IoT yang dilakukan

Dalam era digital saat ini, penggunaan API menjadi elemen krusial dalam pengembangan sistem berbasis IoT. API memungkinkan komunikasi antara perangkat dan server secara efisien, sehingga data dapat dikirim, diproses, dan disimpan dengan optimal. Laravel, sebagai salah satu framework PHP yang populer, menyediakan berbagai fitur yang mempermudah pembuatan API, seperti sistem routing, ORM (Eloquent), dan middleware.

Namun, dalam pengujian API, sering kali terdapat kendala akses jika API hanya berjalan di lingkungan lokal. Untuk mengatasi hal ini, Ngrok digunakan sebagai solusi yang memungkinkan API dapat diakses secara publik tanpa perlu deploy ke server eksternal. Dengan adanya Ngrok, pengembang dapat melakukan pengujian dan debugging API secara langsung dari berbagai perangkat atau jaringan yang berbeda.

**1.2 Tujuan eksperimen**

* Mempelajari dasar-dasar pembuatan API menggunakan Laravel.
* Mengembangkan REST API untuk mengelola data sensor dengan fitur CRUD.
* Menghubungkan API Laravel ke internet menggunakan Ngrok untuk akses publik.
* Menguji fungsionalitas API menggunakan Postman guna memastikan respons API sesuai dengan ekspektasi.

**2. Methodology (Metodologi)**

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

> **Software** (XAMPP, VSCode)

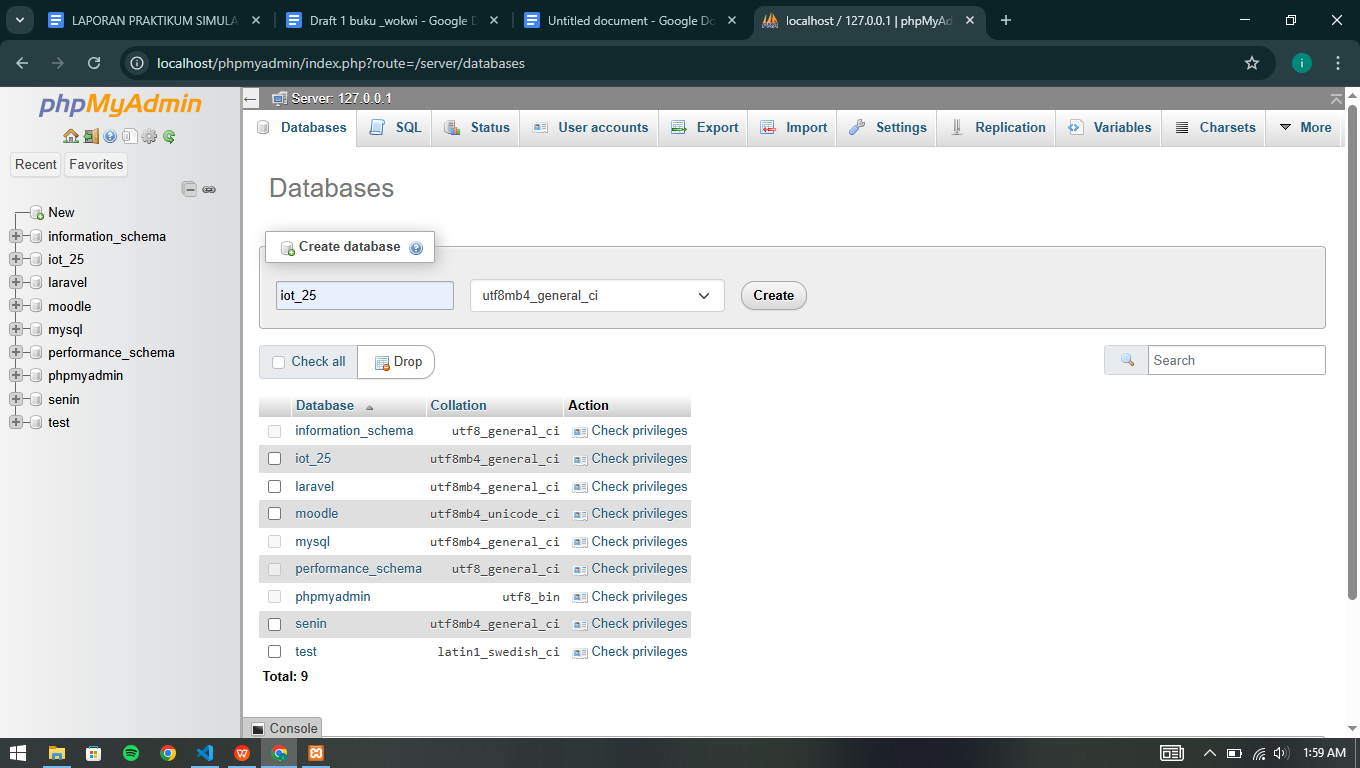
> **Database** (MySql)

> **Server** (Ngrok)

> **Framework** (Laravel 11)

**2.2 Implementation Steps (Langkah Implementasi)**

> Langkah-langkah dalam menyusun praktikum ini adalah diawali dengan membuat database yang diberi nama iot\_25



> Berikutnya buka Command Prompt dan paste perintah berikut untuk mendownload paket dari laravel 11,

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

> Kemudian pindah ke directory laravel 11 dengan perintah berikut,

**cd laravel-11**

> Lalu buka aplikasi Visual Sudio Code dan buka file env, kemudian ubah isinya dengan syntax berikut,

DB\_CONNECTION=mysql

DB\_HOST=127.0.0.1

DB\_PORT=3306

DB\_DATABASE=iot\_25

DB\_USERNAME=root

DB\_PASSWORD=

DB\_CHARSET=utf8mb4

DB\_COLLATION=utf8mb4\_unicode\_ci

> Buat file model **TransaksiSensor.php** dengan cara perintah berikut di command prompt :

**php artisan make:model TransaksiSensor -m**

> Kemudian buka folder database dan menuju ke file migrations, lalu ubah file **2025\_02\_21\_074123\_create\_transaksi\_sensors\_table.php** dengan syntax 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 ke **app/Models/TransaksiSensor.php** danubah isi filenya

**<?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**

Tunggu hingga seluruh running information “Done”

> Lanjut dengan membuat Resource dengan menjalankan perintah berikut, pastikan muncul status “created successfully”

**php artisan make:resource TransaksiSensorResource**

> ubah isi file pada TransaksiSensorResource dengan syntax 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,

];

}

}

> lanjut buat API controller dengan menjalankan perintah seperi pada di bawah ini dan pastikan statusnya “created succesfully”

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

**>** kemudian menuju ke file **app/Http/Controllers/Api/TransaksiSensorController.php,** lalu ubah syntaxnya seperti berikut,

<?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);

}

}

> Selanjutnya buat route khusus API dengan menjalankan perintah seperti berikut,

**php artisan install:api**

> Kemudian menuju 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'));

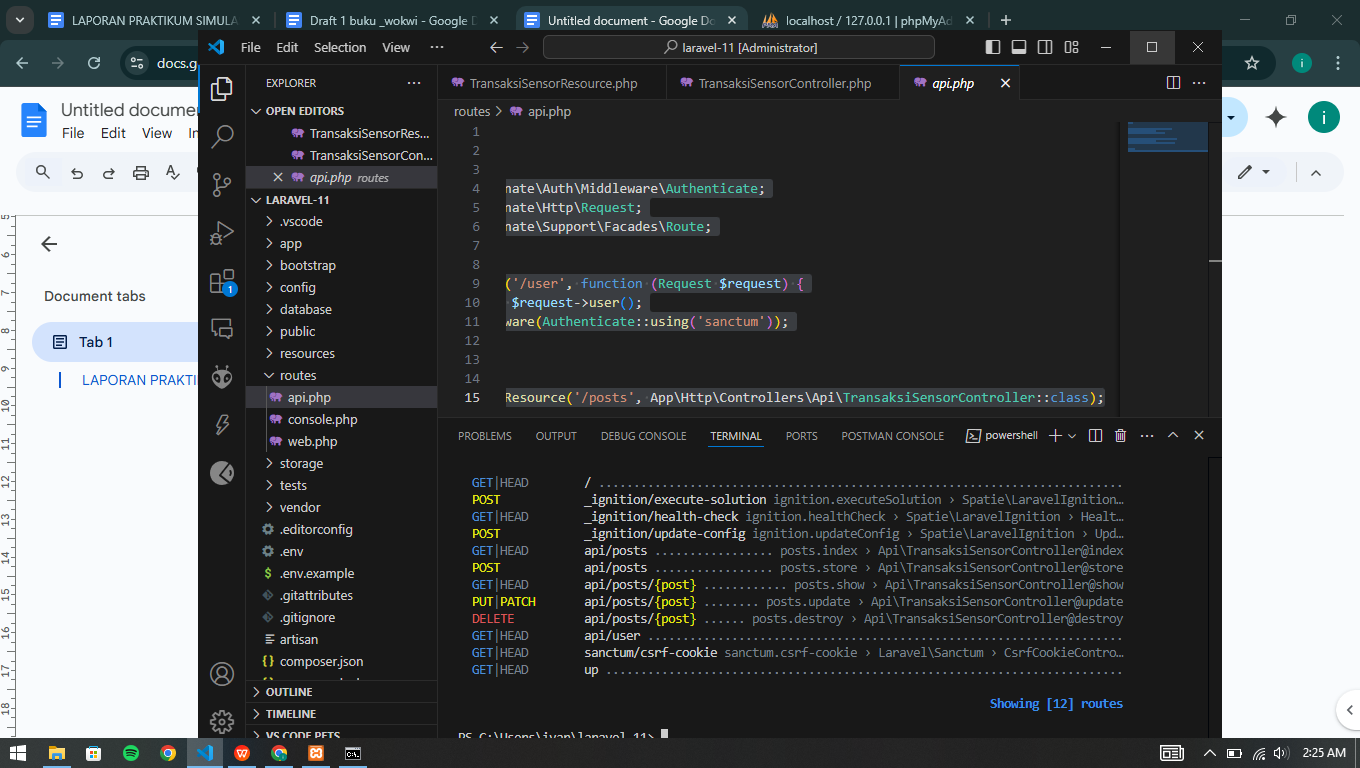
//posts

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

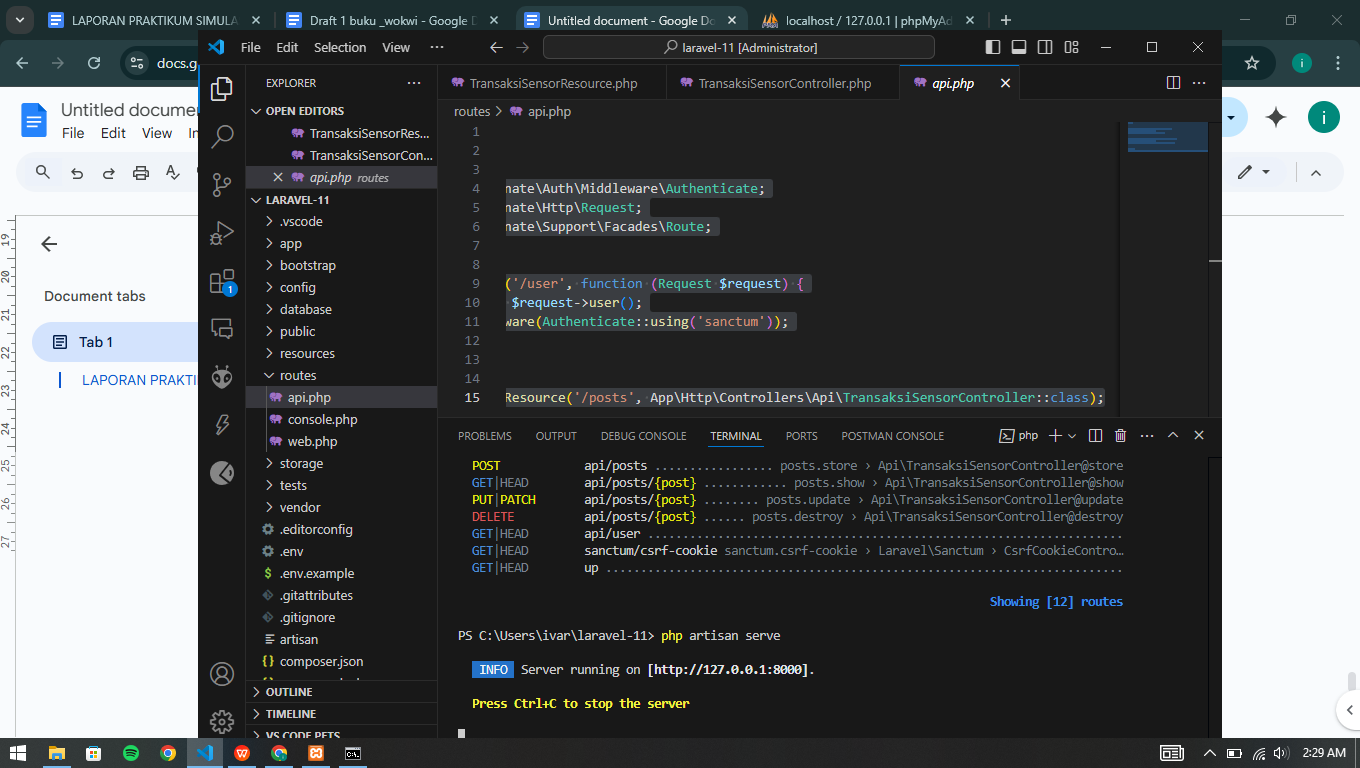
> Kemudian pastikan routes telah muncul dengan menjalankan perintah berikut,

**php artisan route:list**

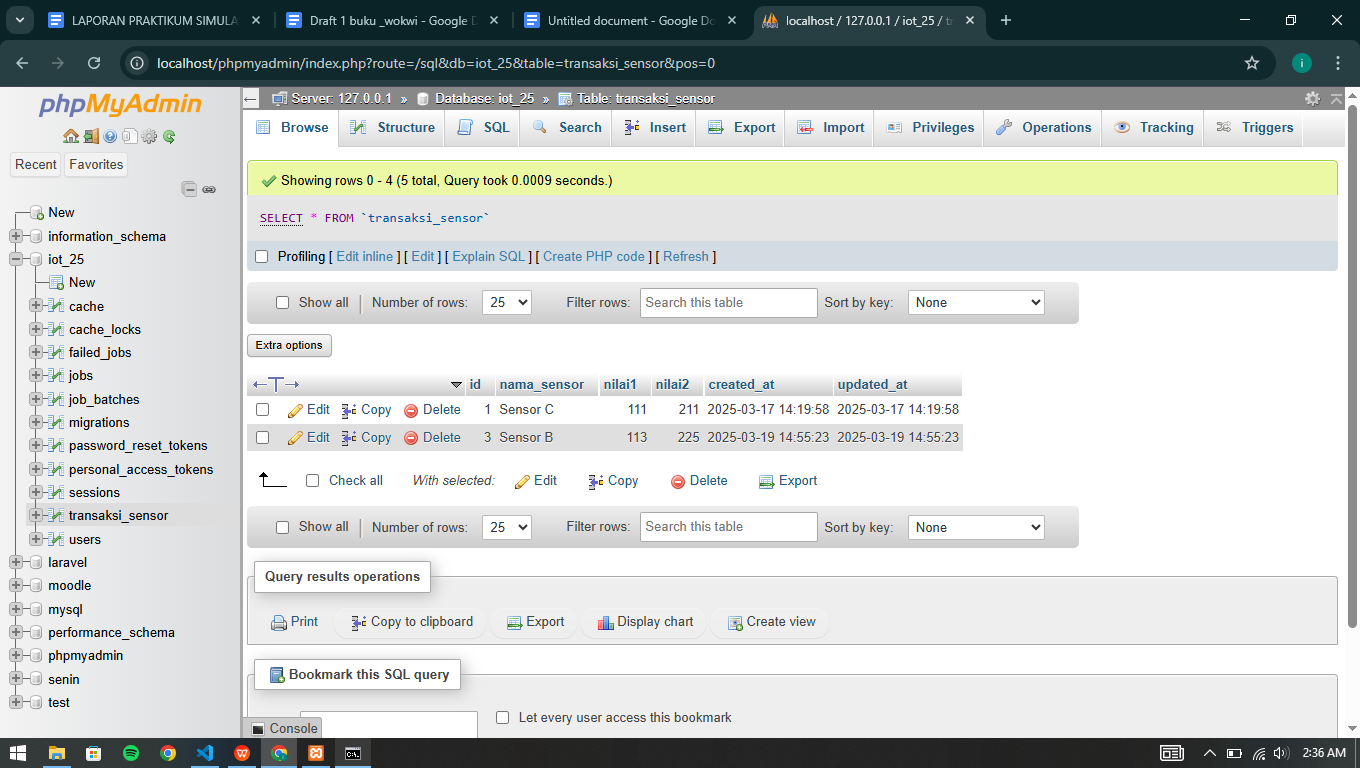
> pastikan tampilan sebagai berikut :



> Selanjutnya download aplikasi Postman dan lakukan Instalasi, untuk menghubungkan laravel dengan postman ketik perintah berikut,   
  
**Php artisan serve**



> Pastikan telah ada data yang masuk ke dalam tabel, seperti pada contoh gambar di bawah,

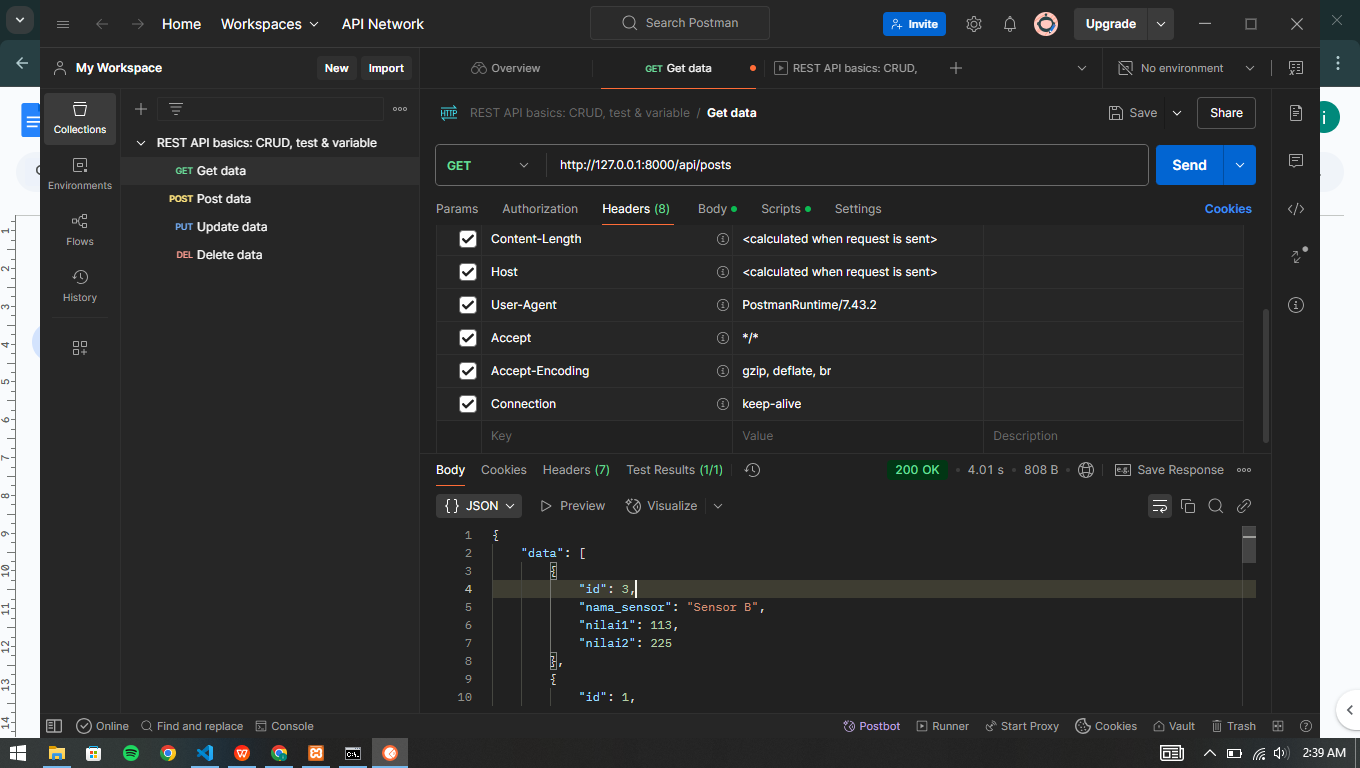


> Untuk mengambil data diatas, buka aplikasi postman dan jalankan prosedur berikut,

bagian URL diisi menggunakan alamat server laravel **http://127.0.0.1:8000/api/posts**

atau akses melalui url : [**http://localhost:8000/api/posts**](http://localhost:8000/api/posts)

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



> Pastikan data yang muncul dalam bentuk json seperti tampilan berikut,

{

"data": [

{

"id": 3,

"nama\_sensor": "Sensor B",

"nilai1": 113,

"nilai2": 225

},

{

"id": 1,

"nama\_sensor": "Sensor C",

"nilai1": 111,

"nilai2": 211

}

],

"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,

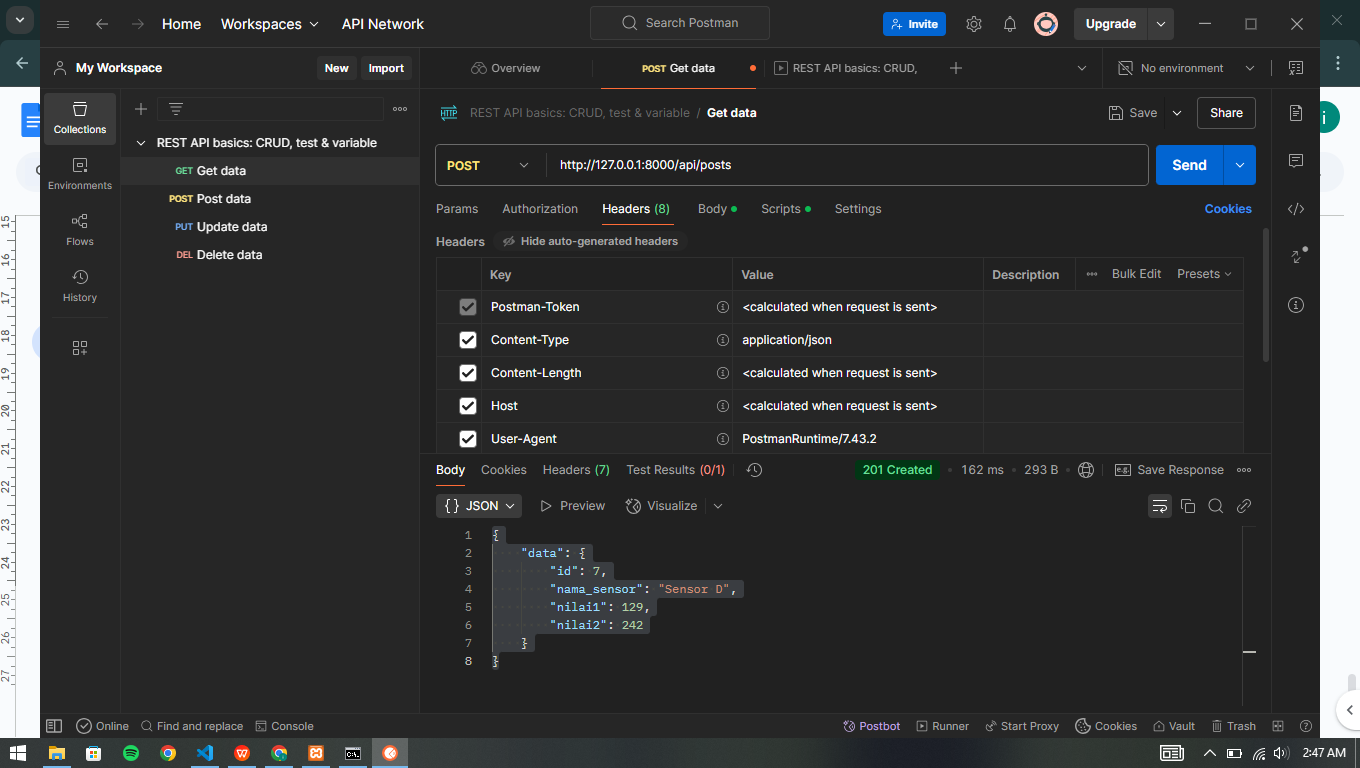
"to": 2,

"total": 2

}

}

> selanjutnya lakukan percobaan untuk memasukkan nilai ke database dengan mengubah method GET menjadi POST, lalu pastikan status pada content-type yang ada di header adalah application/json dan ubah value pada body menjadi seperti berikut,



**raw :**

**{**

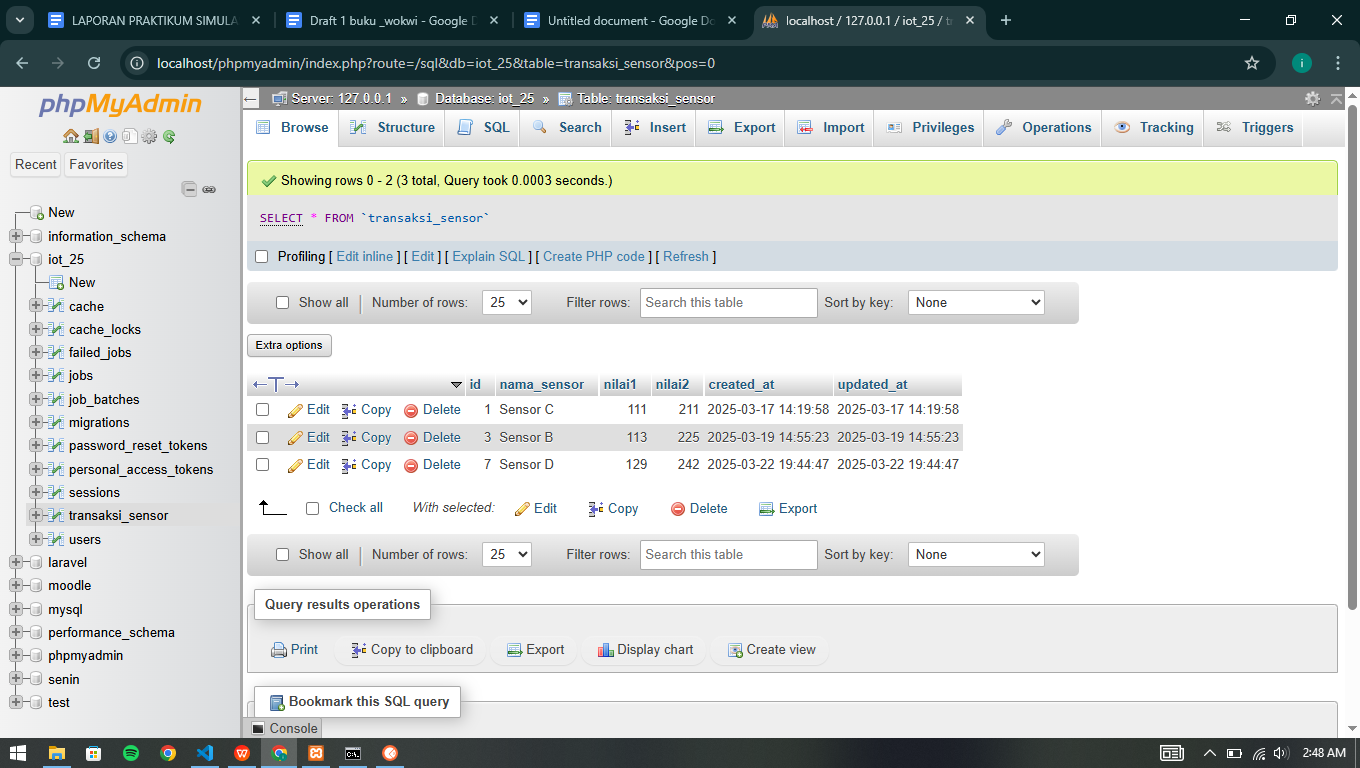
**"nama\_sensor" : "Sensor D",**

**"nilai1" : 129,**

**"nilai2" : 242**

**}**

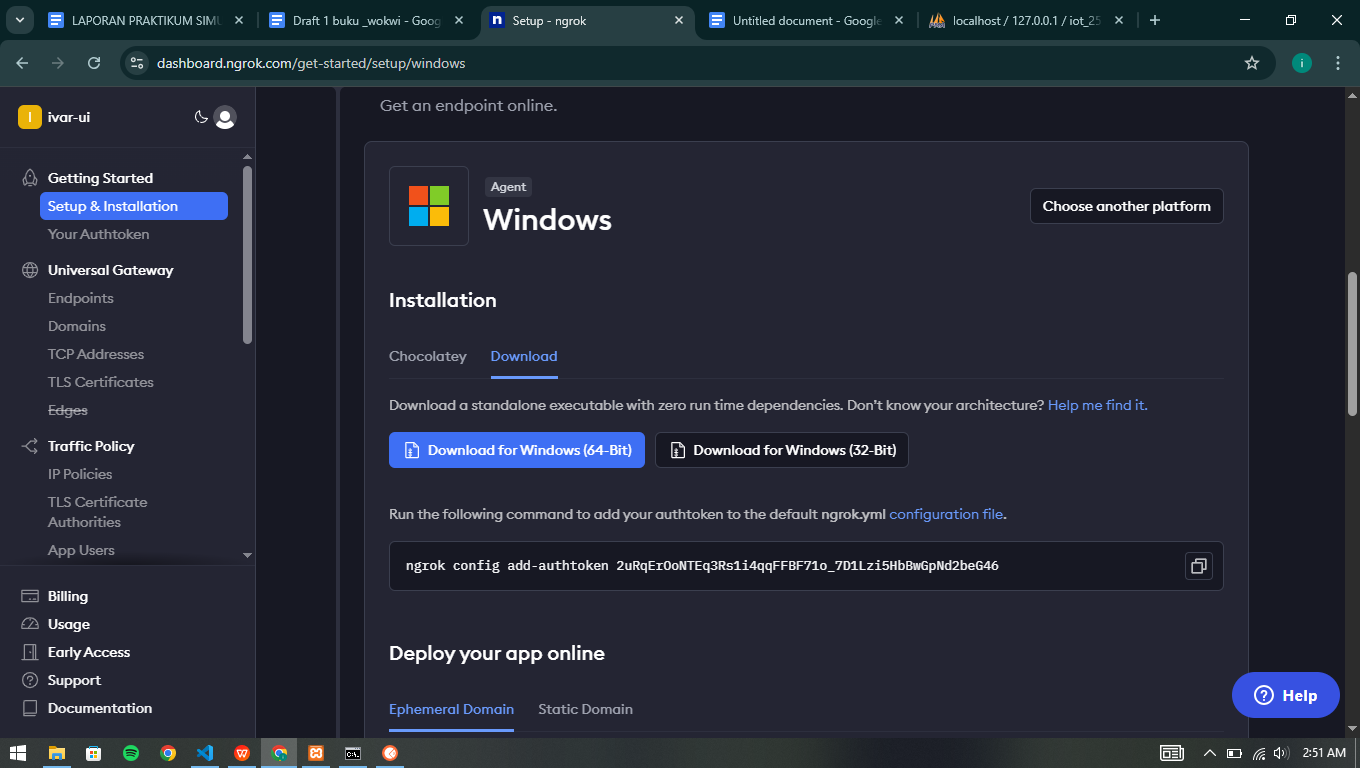
> Maka akan muncul nilai baru pada database,



> Langkah berikutnya, signup pada website Ngrok untuk melakukan installasi,

<https://dashboard.ngrok.com/signup>

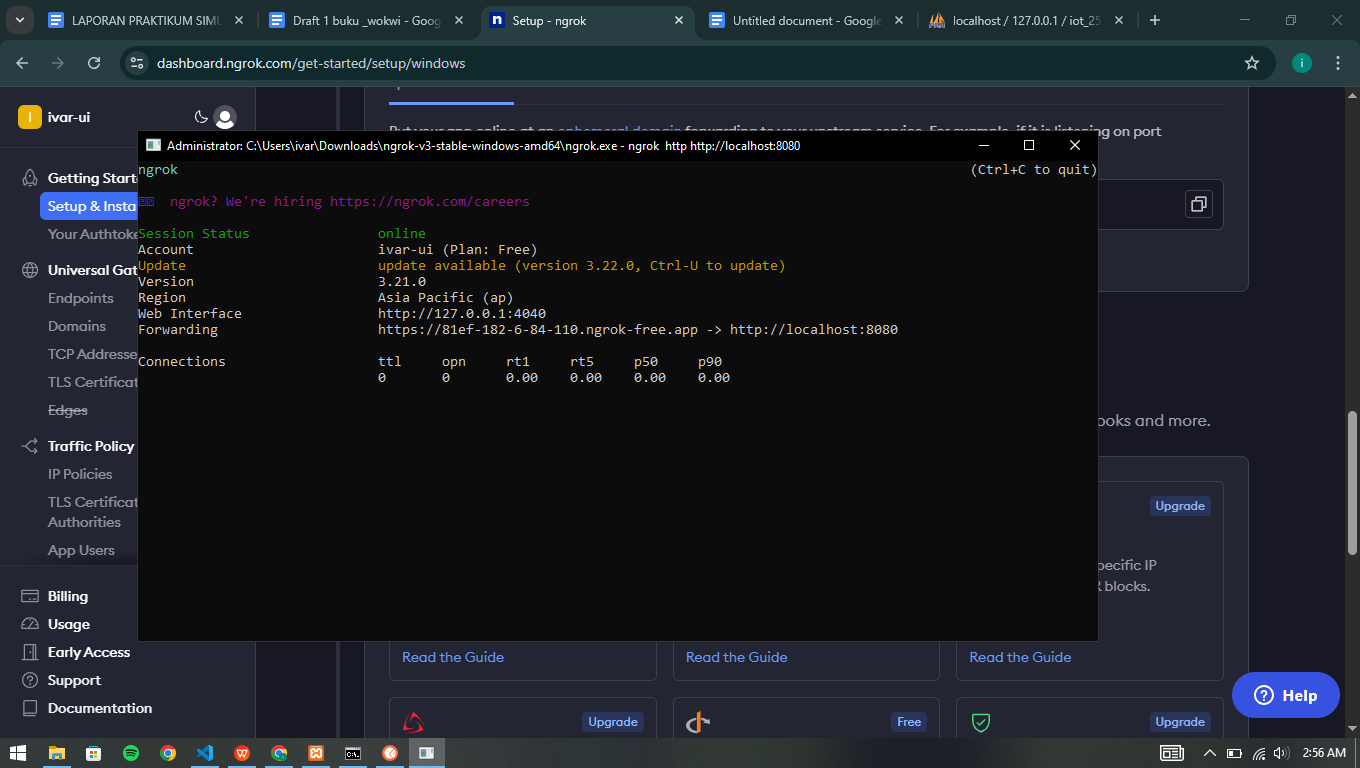
> Install sesuai dengan device masing-masing



> Lakukan extract lalu buka file .exe dan masukkan perintah berikut pada command prompt,  
  
**ngrok config add-authtoken 2uRqErOoNTEq3Rs1i4qqFFBF71o\_7D1Lzi5HbBwGpNd2beG46**

> Masukkan perintah berikut dan pastikan status Ngrok “Online” dan muncul URl localhost

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



> copy pada bagian URL hingga “free.app” lalu tambahkan endpoint untuk melakukan percobaan GET API

[**https://81ef-182-6-84-110.ngrok-free.app/api/posts**](https://81ef-182-6-84-110.ngrok-free.app/api/posts)

> Berikutnya lakukan percobaan melakukan insert data baru melalui API

[**https://81ef-182-6-84-110.ngrok-free.app/api/posts**](https://81ef-182-6-84-110.ngrok-free.app/api/posts)

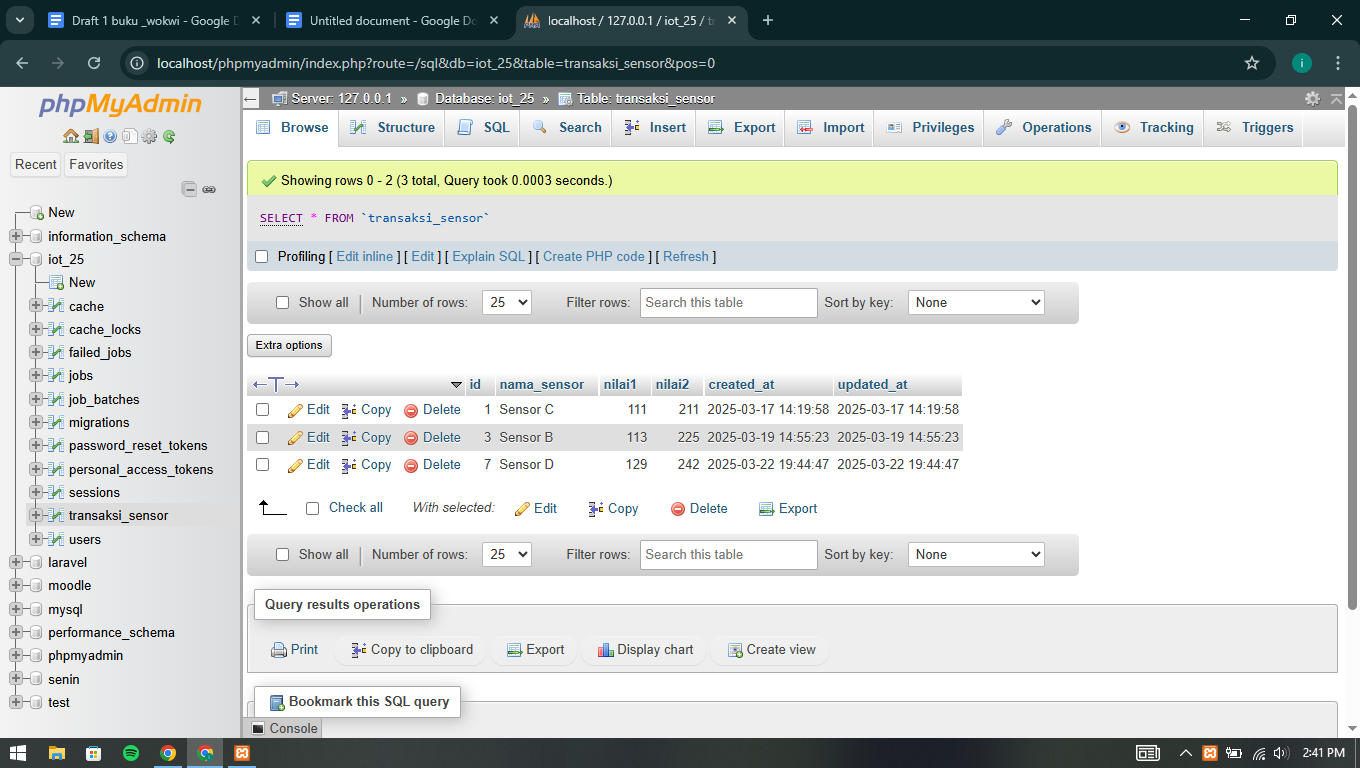
> Ubah method menjadi POST dan parameter header dan body sesuaikan

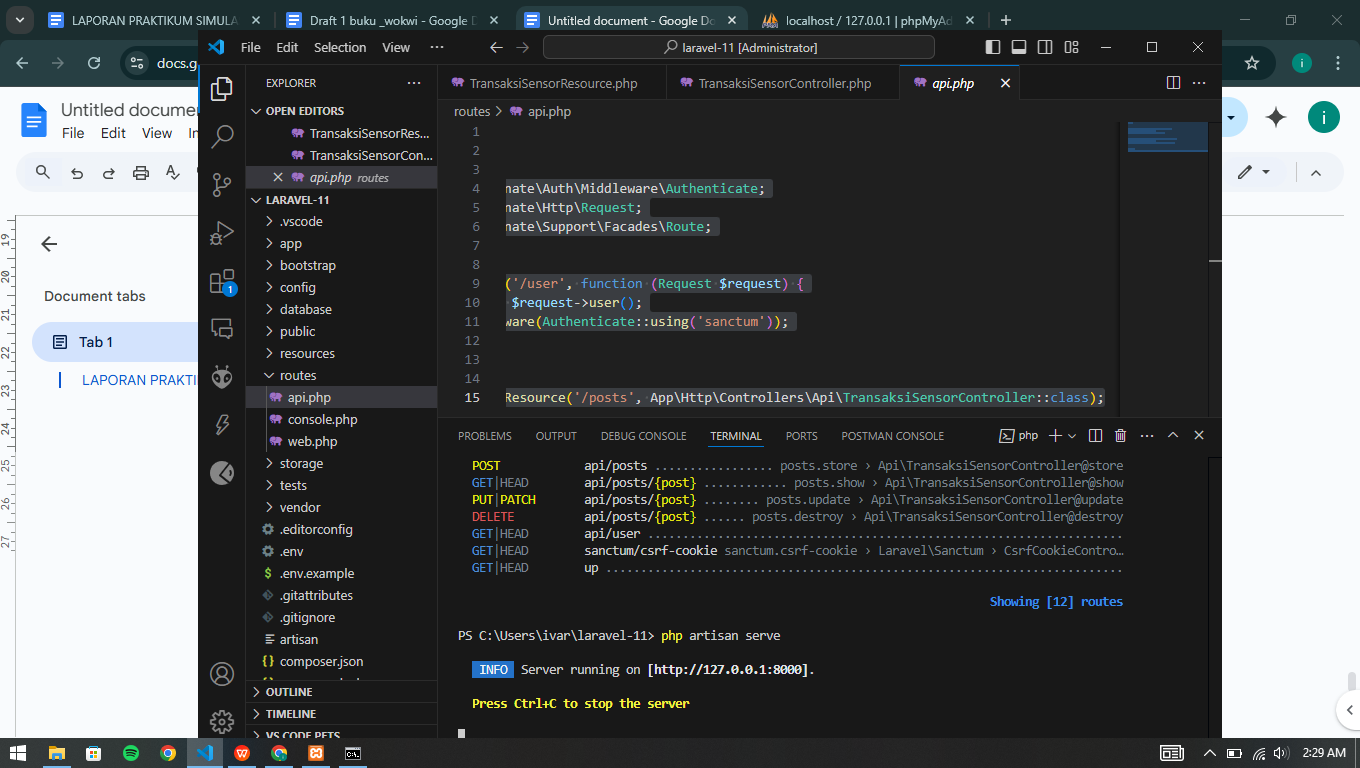
> API telah dibuat menggunakan laravel dan dapat berjalan dengan baik untuk diakses menggunakan URL Publik

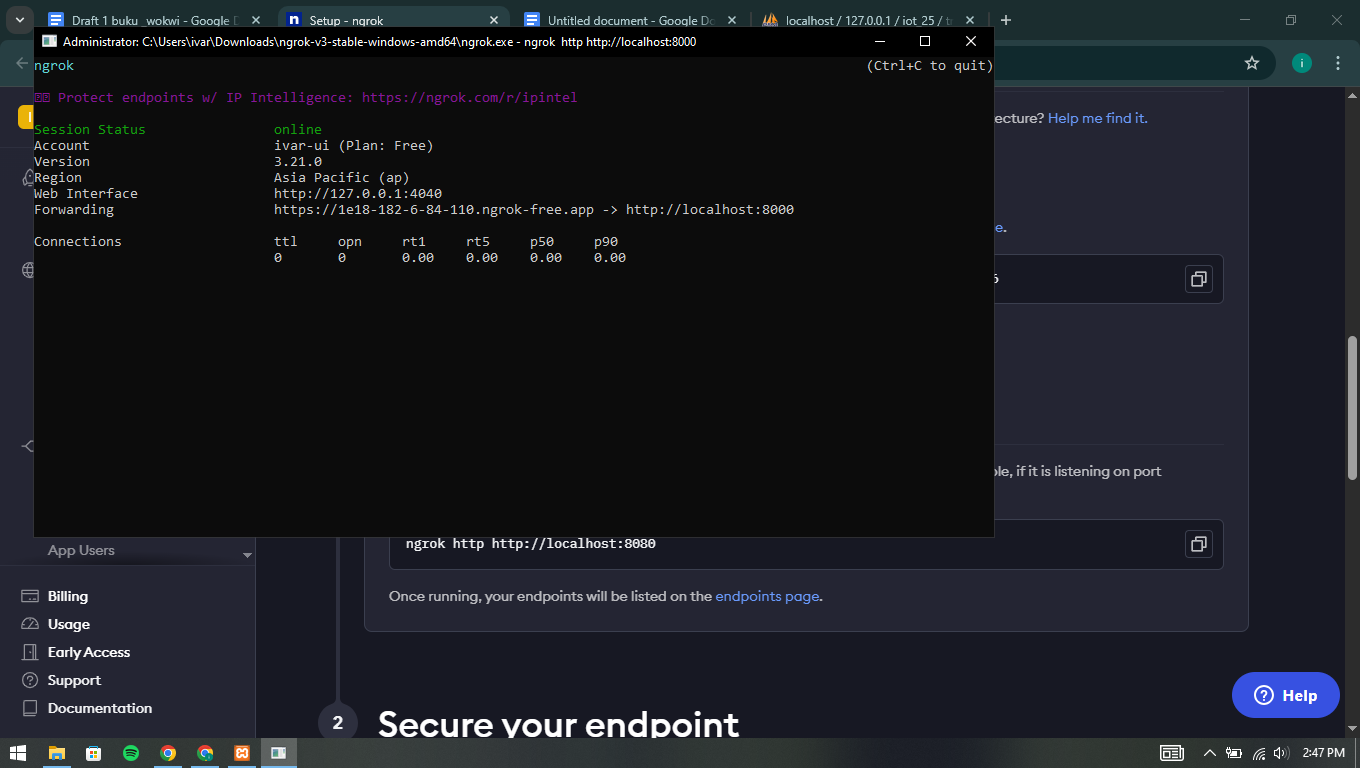
**3. Results and Discussion (Hasil dan Pembahasan)**

**3.1 Experimental Results (Hasil Eksperimen)**

> Menyajikan data hasil praktikum seperti berikut,







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

> Lampiran berupa dokumentasi tambahan saat kegiatan praktikum

