# LAPORAN PRAKTIKUM INTERNET OF THINGS (IoT)

# Fakultas Vokasi, Universitas Brawijaya



**Praktik Simulasi Relay, Button & LED**

**Tugas BAB 10**

*Adesti Mayerga*

*Fakultas Vokasi, Universitas Brawijaya   
Email:* [*adestimayerga97@gmail.com*](mailto:adestimayerga97@gmail.com)

**Abstract** (Abstrak)

Praktikum ini bertujuan untuk memahami cara kerja relay, tombol (button), dan LED menggunakan mikrokontroler ESP32 dalam simulasi Wokwi. Relay digunakan sebagai saklar elektronik untuk mengendalikan perangkat, sedangkan tombol berfungsi sebagai input untuk mengontrol status LED dan relay. Simulasi dilakukan dengan menggunakan Visual Studio Code (VSCode) serta PlatformIO sebagai lingkungan pengembangan. Hasil dari simulasi menunjukkan bahwa relay dapat diaktifkan dan dinonaktifkan menggunakan tombol, serta LED dapat menyala saat relay aktif.

**1. Introduction**

* 1. **Latar belakang**

Relay merupakan komponen penting dalam sistem kendali elektronik yang memungkinkan pengendalian perangkat berdaya tinggi dengan sinyal berdaya rendah dari mikrokontroler. Pada simulasi ini, relay dikombinasikan dengan tombol dan LED untuk menunjukkan cara kerja dasar sistem kontrol berbasis ESP32.

* 1. **Tujuan eksperimen**

1.Memahami cara kerja relay dan integrasinya dengan ESP32.

2.Mempelajari penggunaan tombol sebagai input untuk mengendalikan relay.

3.Mengamati perubahan status LED sebagai indikator aktif atau tidaknya relay.

**2. Methodology (Metodologi)**

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

Laptop/Pc, wokwi, dan vscode

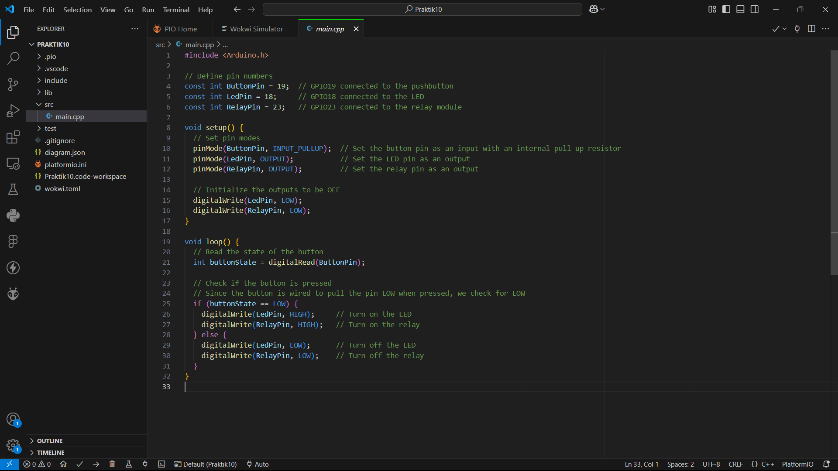
**2.2 Implementation Steps (Langkah Implementasi)**

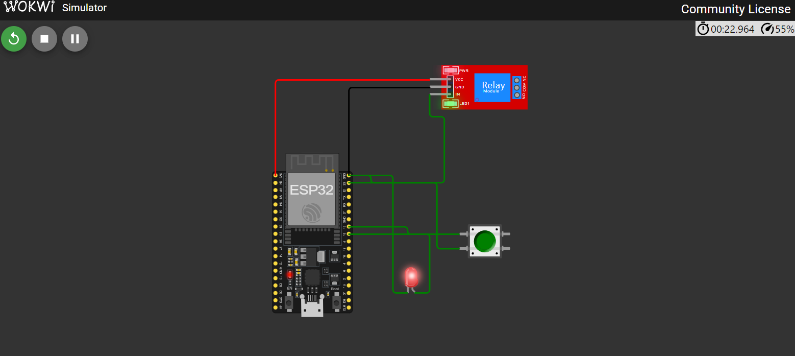
1. **Persiapan Lingkungan Pengembangan**
   * Membuat proyek baru untuk ESP32 di PlatformIO.
   * Menghubungkan ESP32 dengan modul relay, tombol, dan LED di Wokwi.
2. **Rangkaian Elektronik di Wokwi**
   * **Relay Module**
     + VCC → 3.3V ESP32
     + GND → GND ESP32
     + IN → GPIO23 ESP32
   * **Push Button**
     + Satu kaki ke GPIO19 ESP32
     + Satu kaki ke GND
   * **LED**
     + Katoda ke GND
     + Anoda ke GPIO18 ESP32 melalui resistor
3. **Penulisan Kode**
   * Menggunakan bahasa pemrograman C++ di dalam PlatformIO.
   * Menggunakan fungsi digitalRead() untuk membaca status tombol.
   * Menggunakan fungsi digitalWrite() untuk mengontrol LED dan relay.
   * Menggunakan pinMode() untuk mengatur mode pin ESP32.
4. **Simulasi dan Pengujian**
   * Menjalankan simulasi di Wokwi.
   * Menekan tombol untuk mengaktifkan dan menonaktifkan relay serta LED.
   * Mengamati perubahan status LED dan relay dalam simulasi.

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

**3.1 Experimental Results (Hasil Eksperimen)**

Hasil simulasi menunjukkan bahwa relay dapat dikontrol dengan baik menggunakan tombol. Saat tombol ditekan, relay aktif dan LED menyala, menandakan perangkat yang dikontrol oleh relay dalam keadaan ON. Berikut adalah tampilan simulasi:

**** Kode main.cpp



# LAPORAN PRAKTIKUM INTERNET OF THINGS (IoT)

# Fakultas Vokasi, Universitas Brawijaya



**Praktik Simulasi Sensor Jarak  
(Ultrasonic)**

**Tugas BAB 11**

*Adesti Mayerga*

*Fakultas Vokasi, Universitas Brawijaya   
Email* [*adestimayerga97@gmail.com*](mailto:adestimayerga97@gmail.com)

**Abstract** (Abstrak)

Praktikum ini bertujuan untuk memahami cara kerja sensor ultrasonik HC-SR04 dalam mengukur jarak serta mengintegrasikannya dengan mikrokontroler ESP32. Sensor ultrasonik bekerja dengan prinsip pemantulan gelombang suara untuk menghitung jarak suatu objek. Program dikembangkan menggunakan PlatformIO di Visual Studio Code serta diuji melalui simulator Wokwi. Hasil pengujian menunjukkan bahwa sensor dapat mengukur jarak dengan akurasi yang cukup baik dan menampilkan hasilnya di Serial Monitor.

**1. Introduction**

* 1. **Latar belakang**

Sensor ultrasonik HC-SR04 sering digunakan dalam berbagai aplikasi seperti robotika, sistem parkir otomatis, dan perangkat IoT untuk mendeteksi objek serta mengukur jarak secara non-kontak. ESP32 sebagai mikrokontroler dengan fitur komunikasi nirkabel dapat memproses data dari sensor ini dengan cepat dan efisien.

**1.2 Tujuan eksperimen**

1. Memahami prinsip kerja sensor ultrasonik HC-SR04 dalam mengukur jarak.
2. Mengintegrasikan sensor HC-SR04 dengan ESP32.
3. Melakukan simulasi pengukuran jarak menggunakan Wokwi dan menampilkan hasil di Serial Monitor.

**2. Methodology (Metodologi)**

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

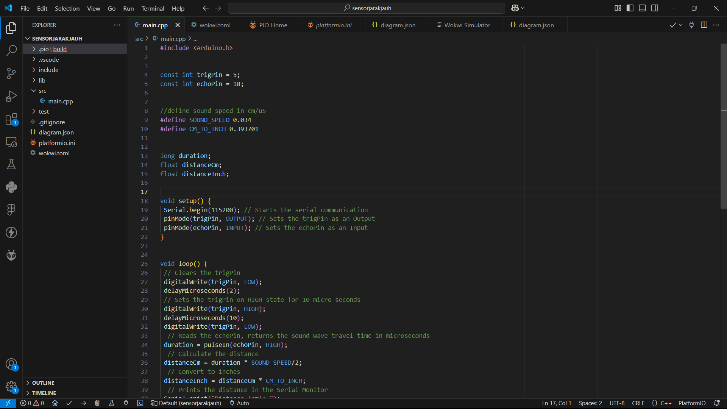
Laptop/Pc, wokwi, dan vscode

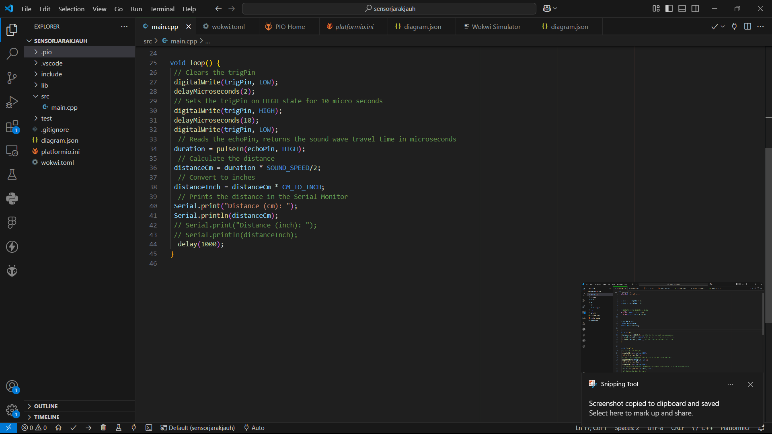
**2.2 Implementation Steps (Langkah Implementasi)**

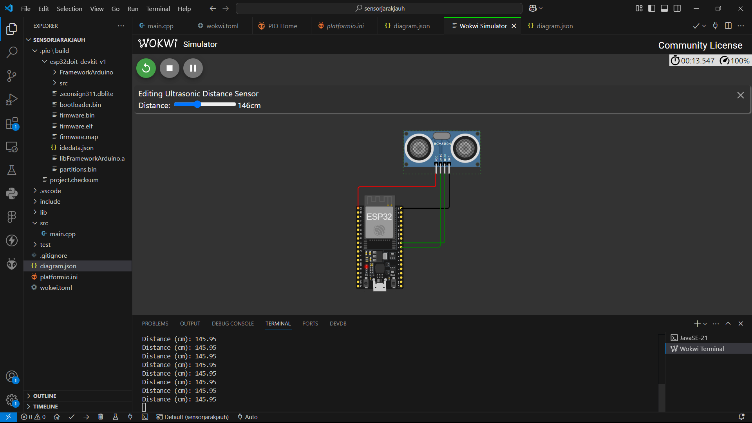
1. **Persiapan Lingkungan Pengembangan**
   * Membuat proyek baru untuk ESP32 di PlatformIO.
   * Menyusun rangkaian ESP32 dengan sensor ultrasonik di Wokwi.
2. **Konfigurasi Rangkaian Elektronik di Wokwi**
   * Sensor HC-SR04
     + VCC → 3.3V ESP32
     + GND → GND ESP32
     + Trig → GPIO 5 ESP32
     + Echo → GPIO 18 ESP32
3. **Penulisan Kode Program**
   * Menggunakan fungsi pulseIn() untuk membaca durasi pantulan gelombang ultrasonik.
   * Menghitung jarak menggunakan rumus:
   * Menampilkan hasil pengukuran pada Serial Monitor**.**
4. **Simulasi dan Pengujian**
   * Menjalankan simulasi di Wokwi.
   * Memantau perubahan jarak pada Serial Monitor.
   * Menyesuaikan nilai jarak dalam simulator untuk melihat perubahan output.

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

**3.1 Experimental Results (Hasil Eksperimen)**

 Kode main.cpp





# LAPORAN PRAKTIKUM INTERNET OF THINGS (IoT)

# Fakultas Vokasi, Universitas Brawijaya

**Praktik Pembuatan API Menggunakan Laravel dan Ngrok**

**Tugas BAB 12**

****

*Adesti Mayerga*

*Fakultas Vokasi, Universitas Brawijaya Email :* [*adestimayerga97@gmail.com*](mailto:adestimayerga97@gmail.com)

**Abstract** (Abstrak)

Laporan praktikum ini membahas proses pembuatan REST API menggunakan Laravel, framework PHP yang populer, serta pemanfaatan ngrok untuk melakukan dalam laporan praktikum ini, kami membahas bagaimana membuat REST API dengan framework PHP yang populer Laravel. Kami juga membahas bagaimana menggunakan ngrok untuk tunneling agar API yang dibuat dapat diakses secara publik meskipun berjalan pada server lokal. Tujuan dari praktikum ini adalah untuk memberikan pemahaman dasar tentang pembuatan API dan bagaimana mereka dapat diintegrasikan ke dalam ekosistem Internet of Things (IoT). Praktikum ini menjelaskan beberapa langkah, mulai dari instalasi Laravel, pembuatan endpoint API, pengujian menggunakan Postman, dan implementasi ngrok untuk memberikan API akses di luar jaringan lokal. Hasil praktikum menunjukkan bahwa kombinasi Laravel 11 dan ngrok dapat memberikan solusi yang efektif untuk mengembangkan dan menguasai API.

*Keyword : Laravel, ngrok, rest API, API*

1. **1. Introduction**
   1. **Latar Belakang**

Perkembangan teknologi Internet of Things (IoT) telah membawa dampak besar pada cara perangkat terhubung dan berkomunikasi melalui jaringan. Salah satu komponen utama dalam implementasi IoT adalah Application Programming Interface (API) yang memungkinkan pertukaran data antar perangkat secara efisien dan terstruktur. Laravel merupakan salah satu framework PHP yang mendukung pembuatan API dengan cepat dan aman. Dengan fitur-fitur bawaan seperti routing, middleware, dan autentikasi, Laravel memudahkan pengembang dalam membangun API yang handal.

Dalam lingkungan pengembangan, sering kali dibutuhkan akses ke API dari luar jaringan lokal, terutama untuk keperluan demonstrasi atau integrasi dengan perangkat IoT lainnya. Ngrok adalah solusi yang memungkinkan pengembang untuk membuat tunnel dari server lokal ke internet, sehingga API yang dikembangkan dapat diakses secara real-time dari perangkat atau lokasi manapun. Dengan menggabungkan Laravel dan ngrok, praktikum ini bertujuan untuk memberikan pemahaman mendalam mengenai pembuatan dan pengujian API dalam konteks IoT.

**1.2 Tujuan eksperimen**

* Memahami konsep REST API: Memahami prinsip dasar pembuatan API yang mendukung komunikasi antar perangkat IoT.
* Mengimplementasikan Laravel 10: Mempelajari cara menginstal, mengkonfigurasi, dan membuat endpoint API menggunakan Laravel 10.
* Integrasi dengan Ngrok: Menggunakan Ngrok untuk membuat tunnel yang memungkinkan API diakses dari luar jaringan lokal.

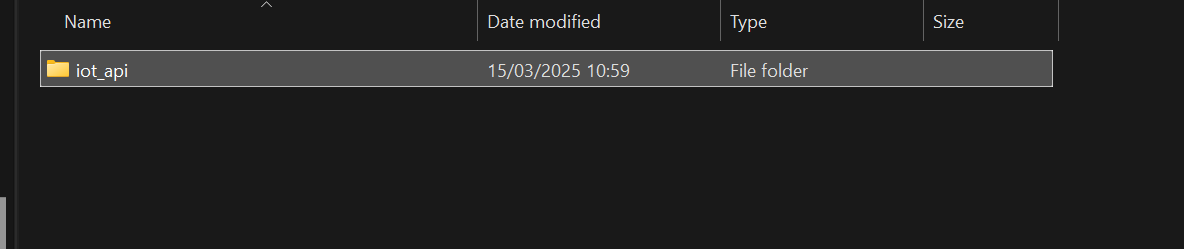
**2. Methodology (Metodologi)**

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

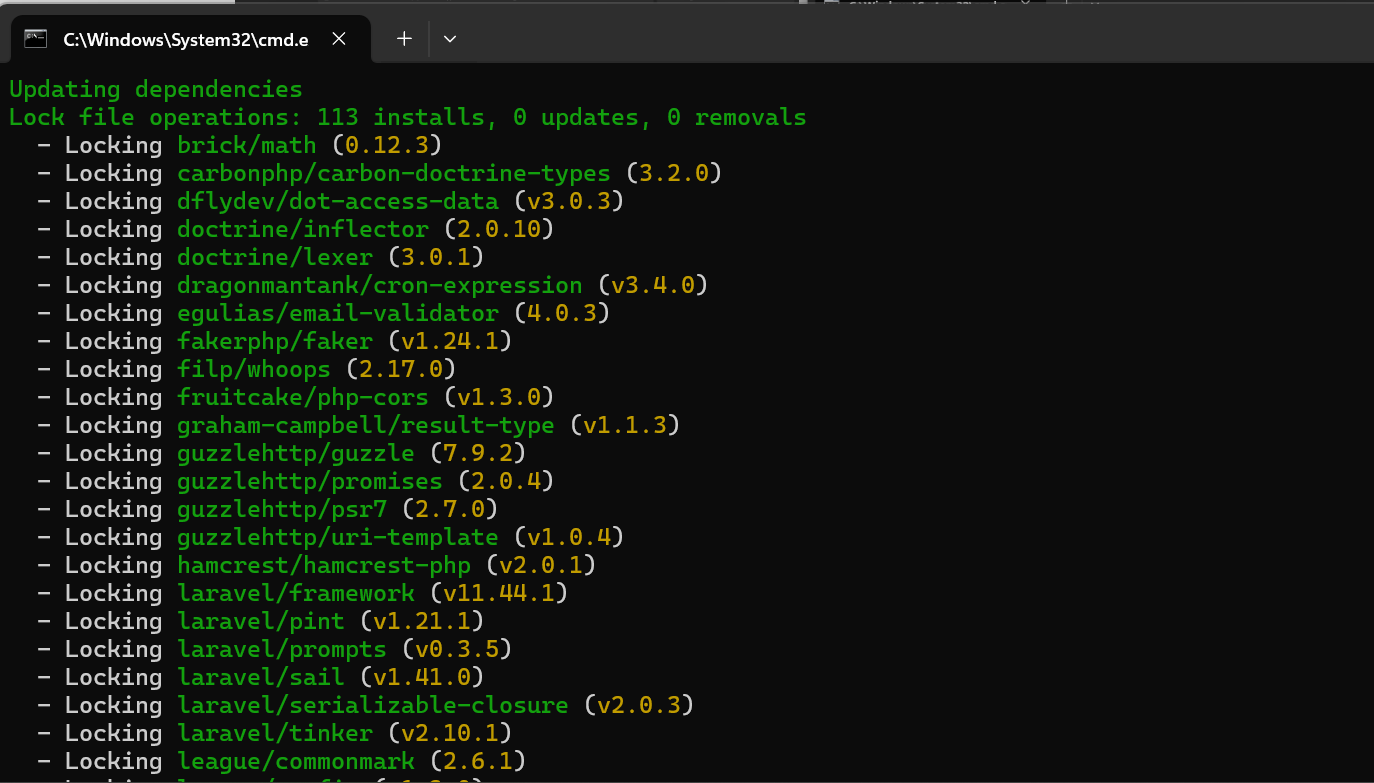
* + 1. Laptop
    2. Internet
    3. Web Server Lokal: XAMPP, WAMP, atau LAMP.
    4. Composer: Manajer paket untuk PHP.
    5. Laravel: Framework PHP yang digunakan untuk membangun API.
    6. Ngrok: Tool untuk membuat tunnel ke server lokal.
    7. Text Editor/IDE: Visual Studio Code, Sublime Text, atau sejenisnya.
    8. Postman: Aplikasi untuk menguji API

**2.2 Implementation Steps (Langkah Implementasi)**

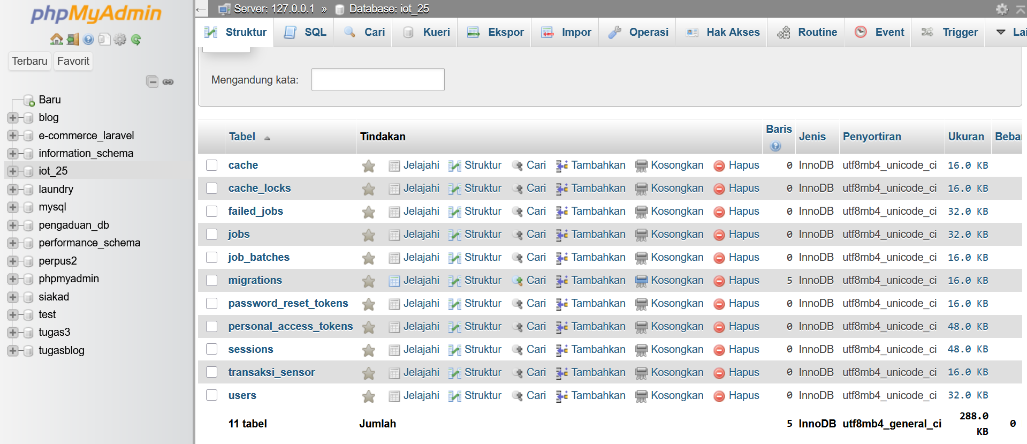
1. Buatlah folder baru di file manager

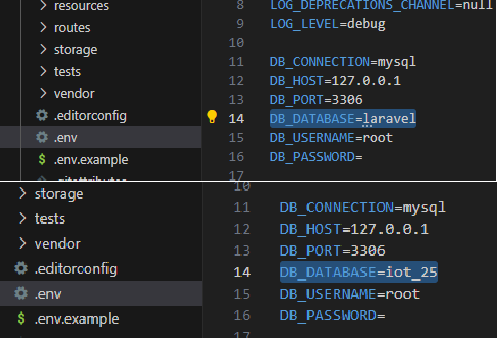


1. Kemudian Open command prompt(CMD) dan Lakukan perintah untuk create composer



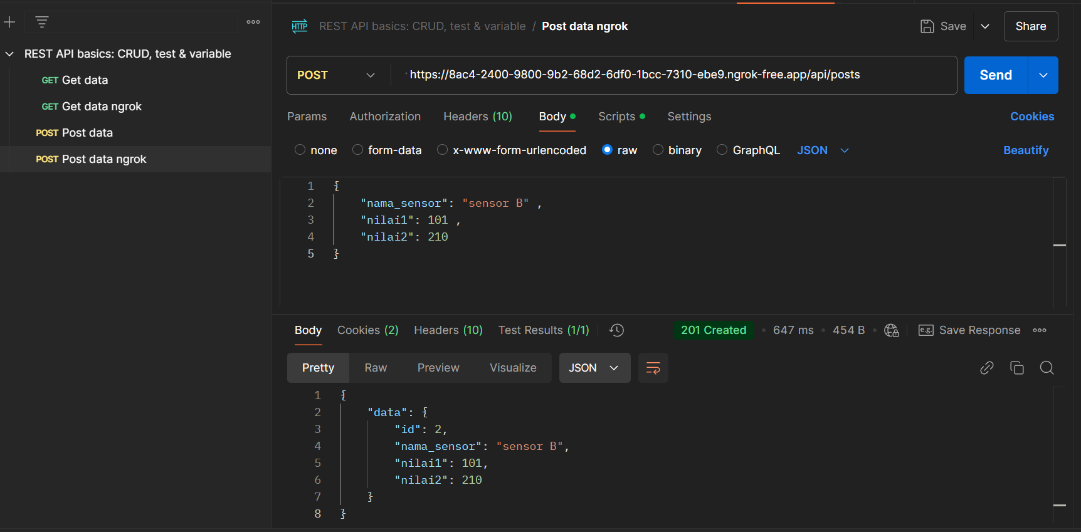
1. Buatlah database MySql



1. Open folder di vscode dan kemudian open file .env untuk melakukan pengeditan nama pada database
2. Buat file model TransaksiSensor.php dengan cara menjalankan perintah di terminal Vscode

### 

1. Lalu masuk ke postman



1. Setelah itu melakun pengeditan juga pada file **2025\_03\_10\_012536\_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.

\*/

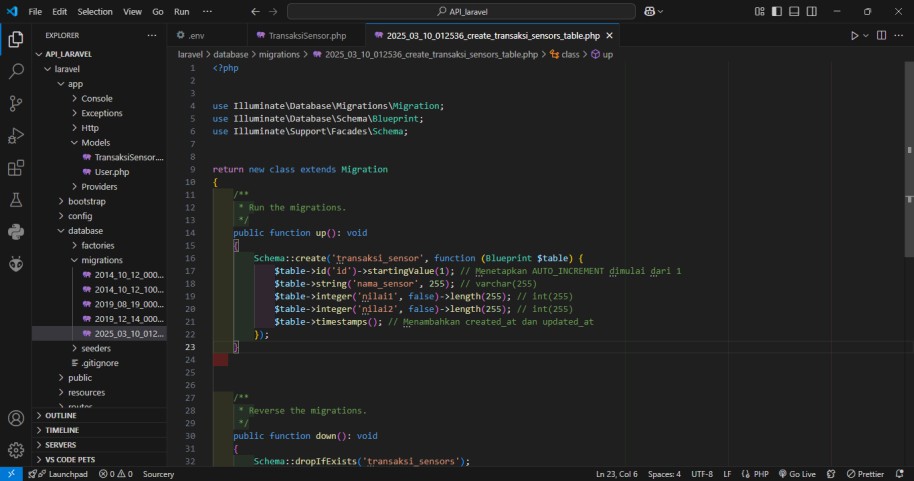
public function down(): void

{

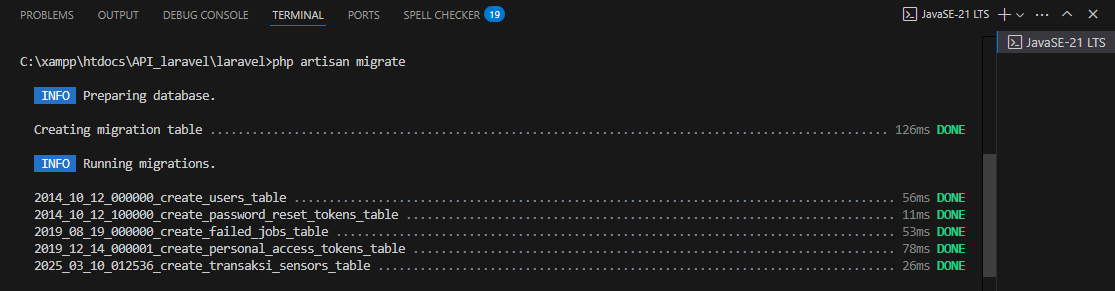
Schema::dropIfExists('transaksi\_sensors');

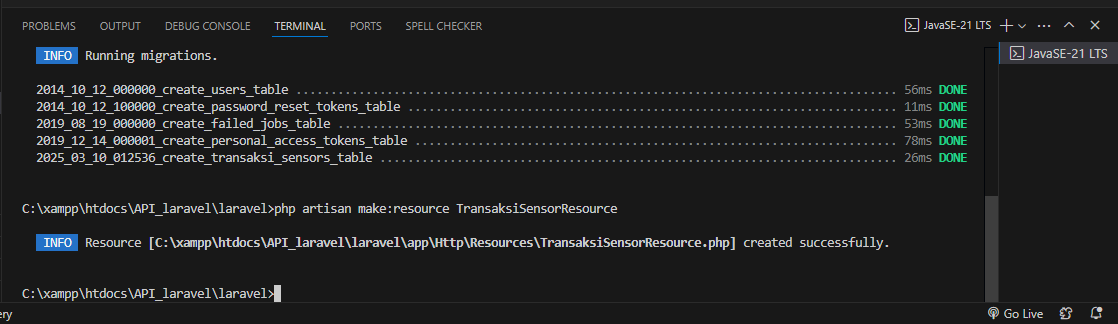
}

};



1. Kemudian lakukan perintah **php artisan migrate** diterminal dan maka pada database akan terinput terdapat table – table data



1. ****Kemudian buatlah resource dengan scema **php artisan make:resource TransaksiSensorResource**
2. Lalu lakukan pada pengeditan pada file **TransaksiSensorResource**

<?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,

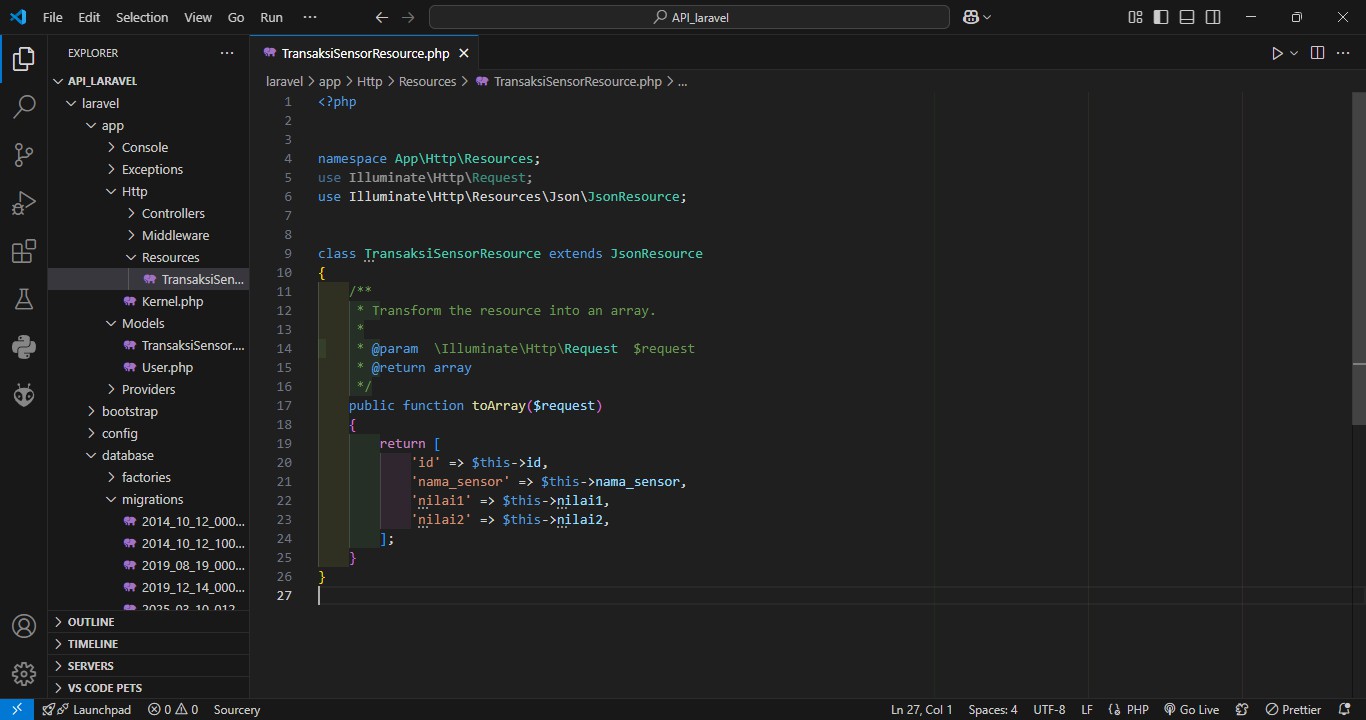
'nama\_sensor' => $this->nama\_sensor, 'nilai1' => $this->nilai1,

'nilai2' => $this->nilai2,

];

}

}



1. Buatlah API controller dengan menjalankan scema

**php artisan make:controller Api/TransaksiSensorController** dan melakukan pengeditan pada 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.

\*

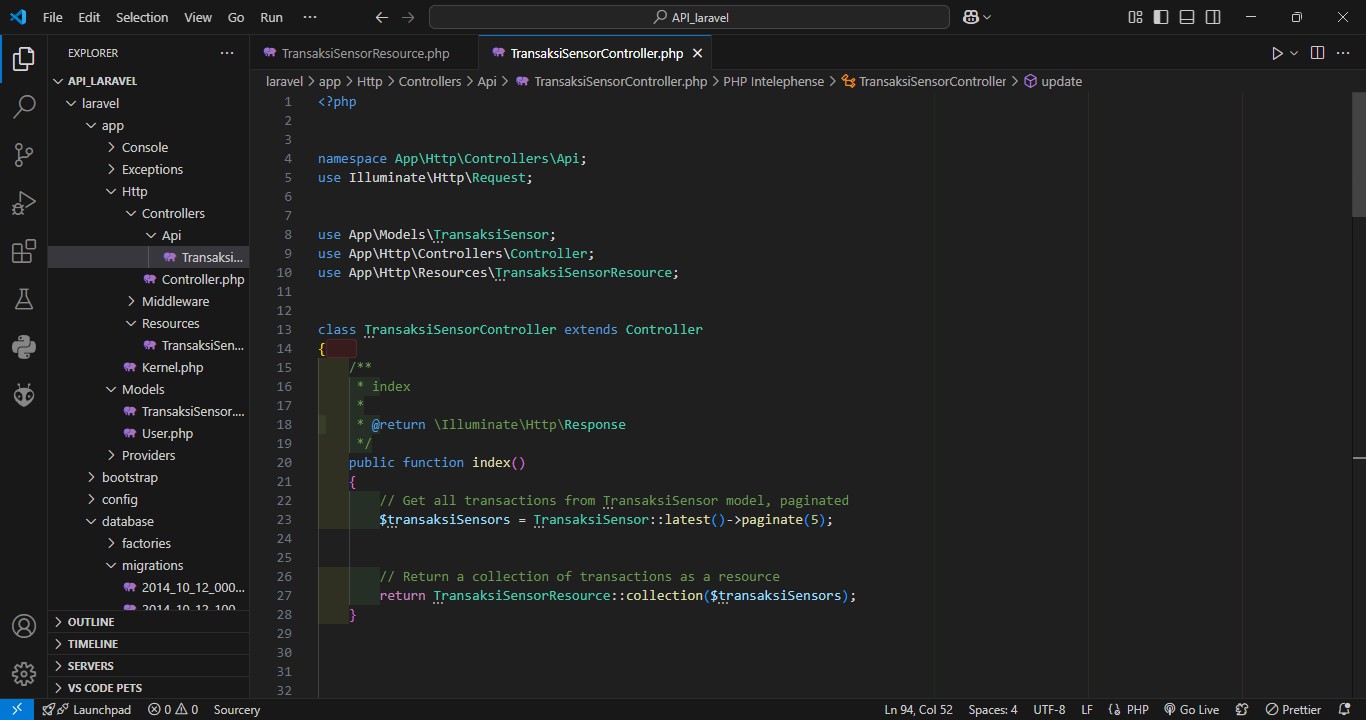
\* @param \Illuminate\Http\Request $request

\* @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);

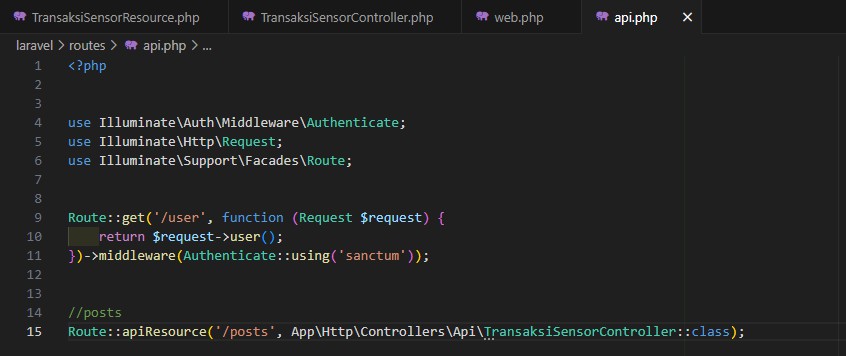
$transaksiSensor->delete();

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

}

}

1. Buat route khusus API dengan menjalankan scema **php artisan make:api** dan melakukan pengeditan file



<?php

use Illuminate\Auth\Middleware\Authenticate; use Illuminate\Http\Request;

use Illuminate\Support\Facades\Route;

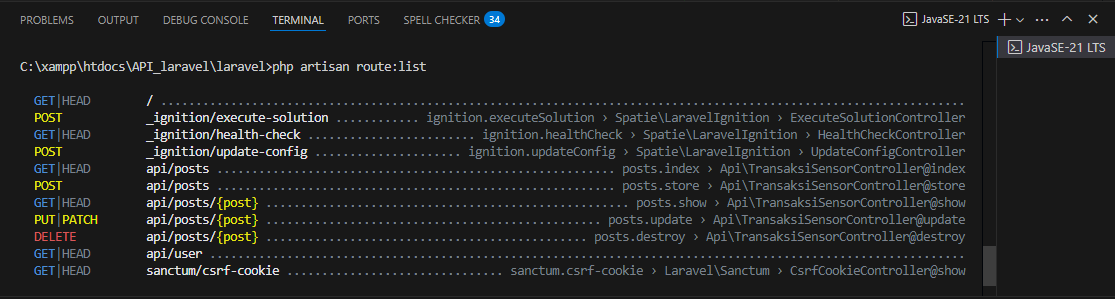
Route::get('/user', function (Request $request) { return $request->user();

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

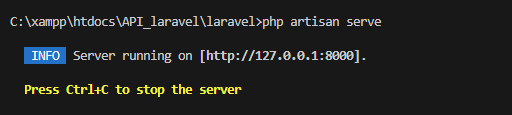
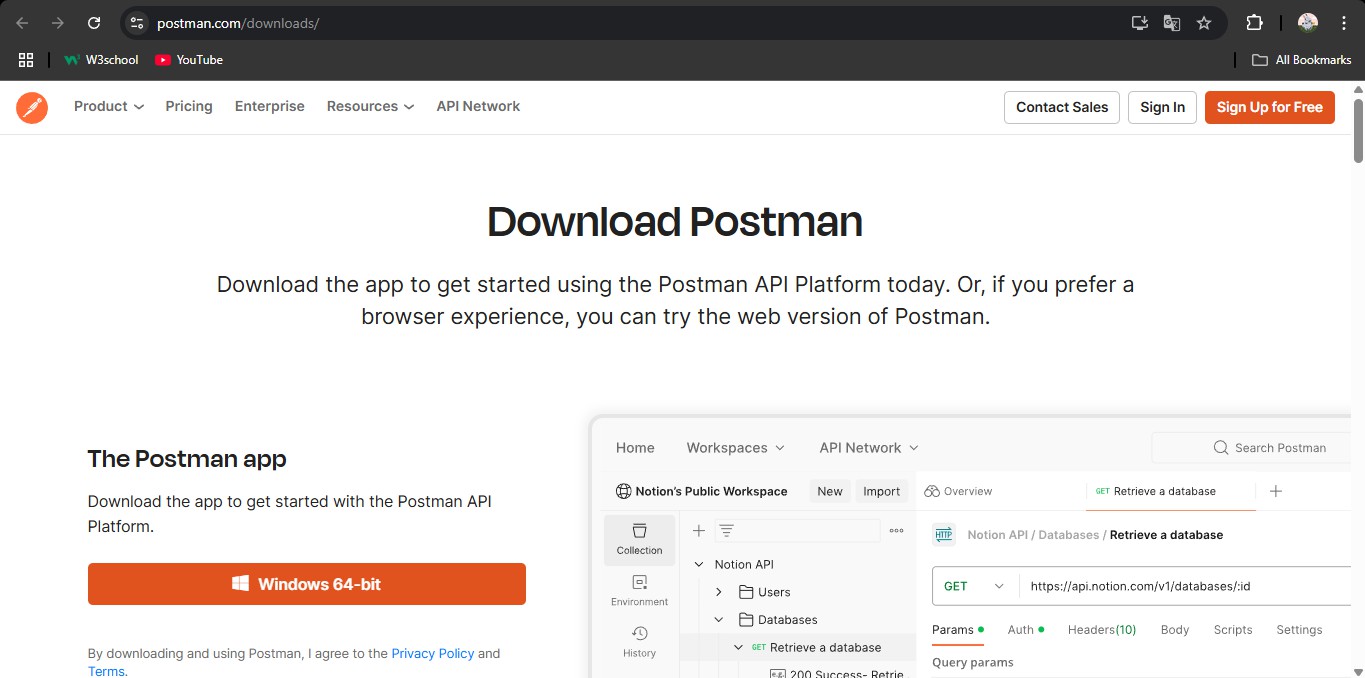
//posts Route::apiResource('/posts',

App\Http\Controllers\Api\TransaksiSensorController::class);

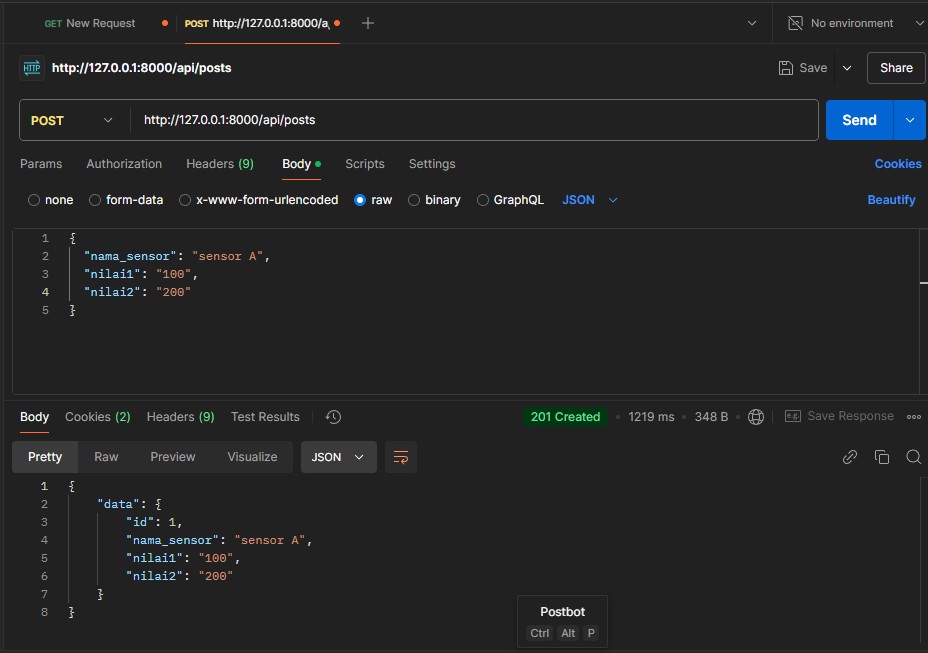
1. Kemudian pastikan routes telah terbentuk dengan menjalankan scema **php artisan route:list**



1. Download POSTMAN dan lakukan scema **php artisan serve** pada terminal



1. Kemudian lakukan pengambilan data melalui postman dengan melakukan scema berikut;

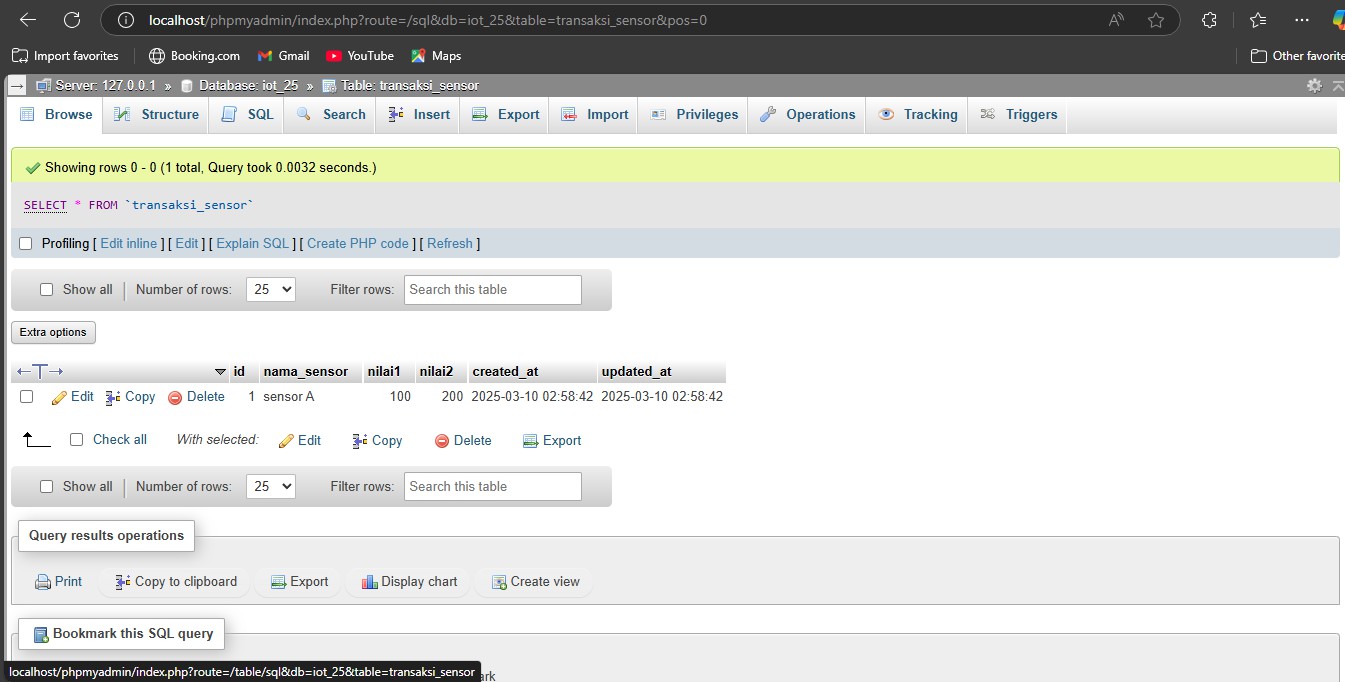
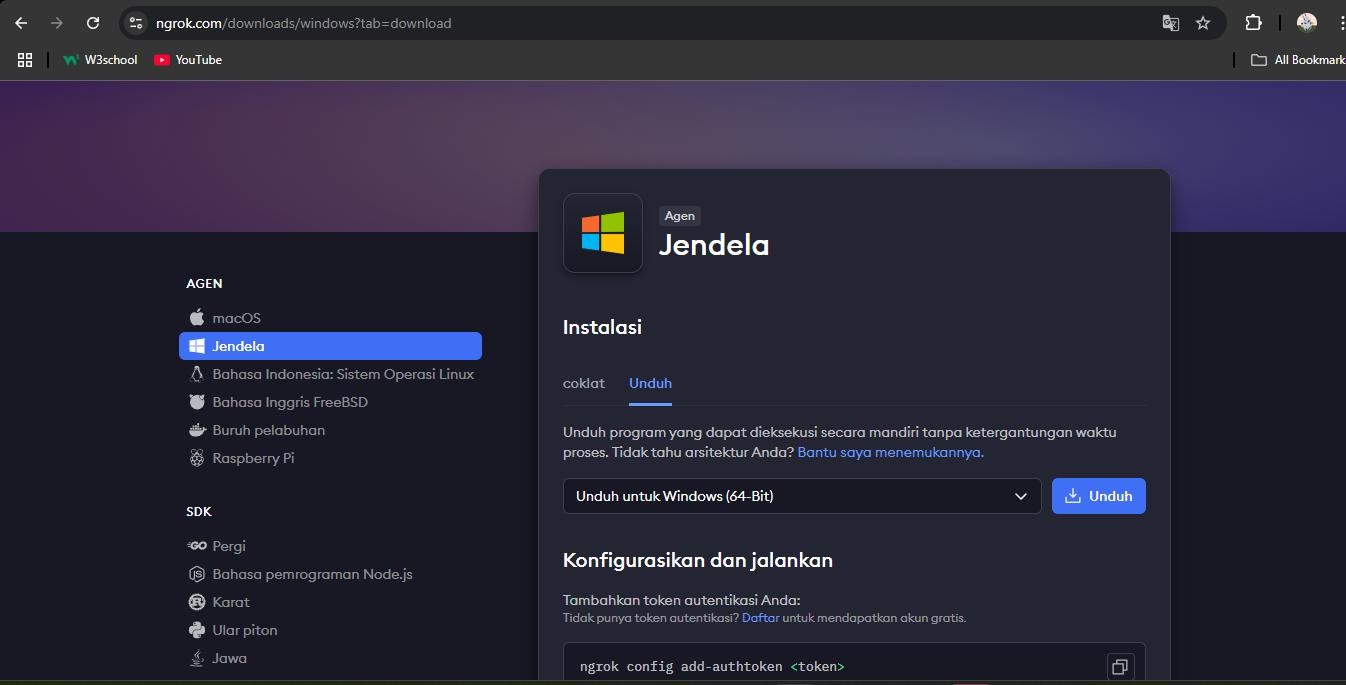


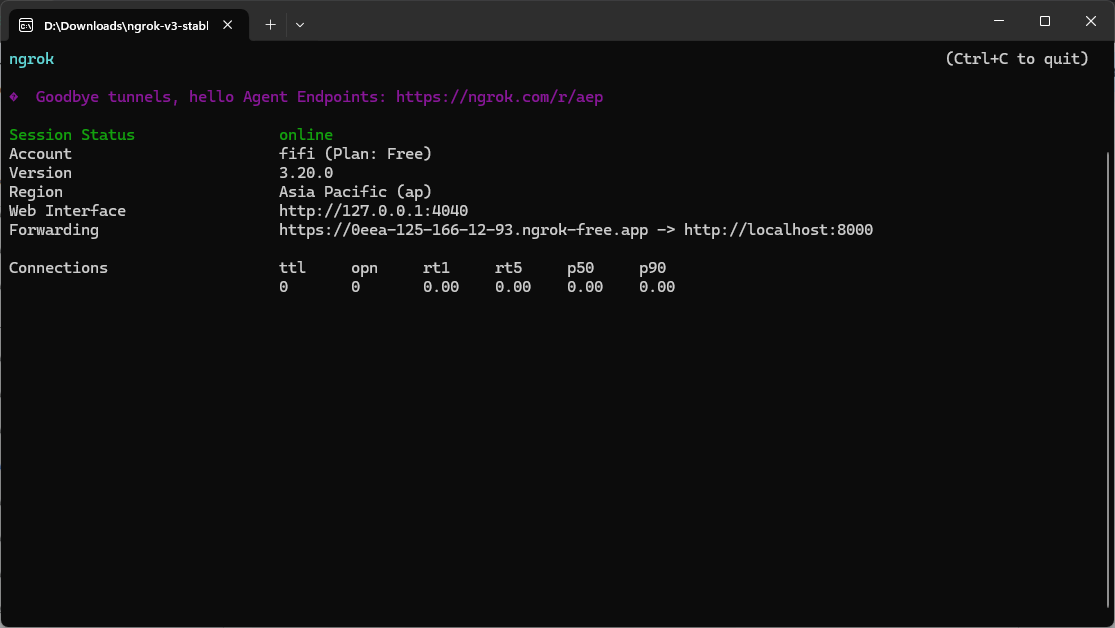
{

"nama\_sensor": "sensor A", "nilai1": "100",

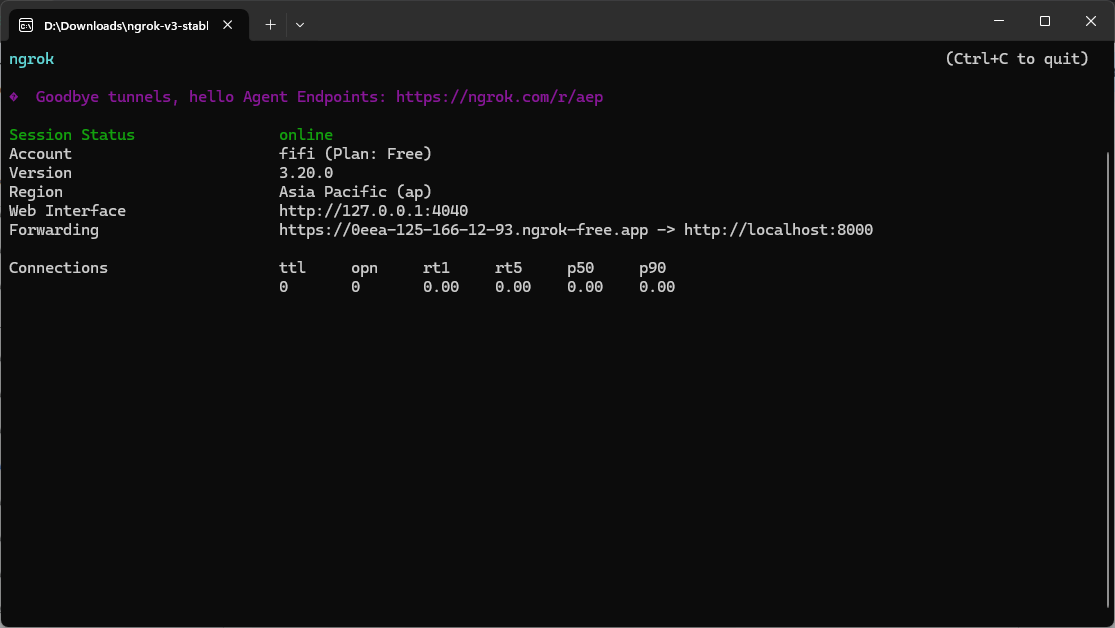
"nilai2": "200"

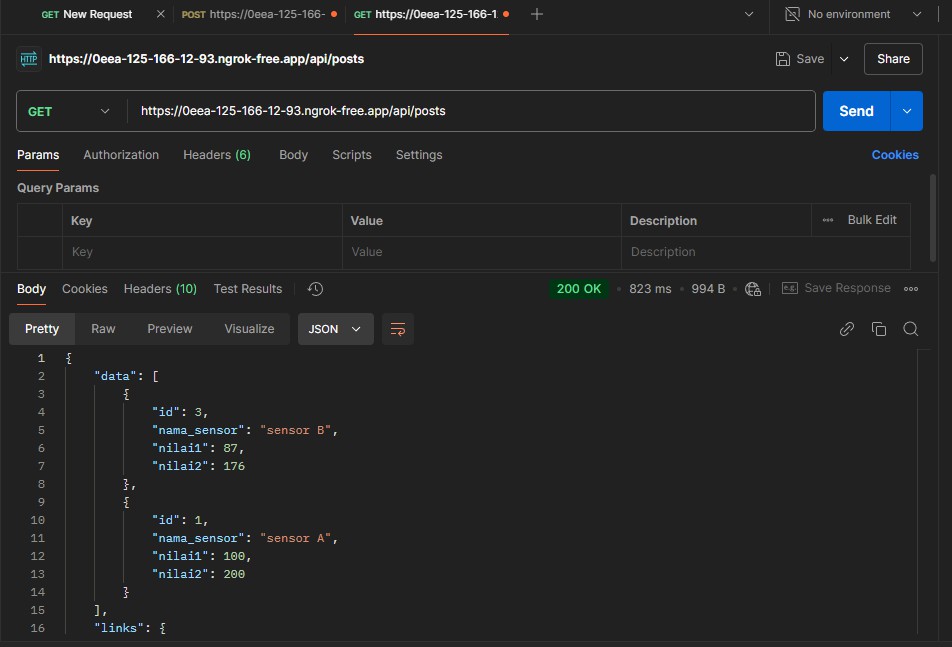
}

1. Setelah itu lakukan pengecekan database pada localhost
2. Langkah berikutnya adalah mengonline-kan API menggunakan service ngrok sehingga API dapat diakses melalui device iot atau simulasi wokwi iot
   1. Download aplikasi ngrok
   2. Kemudian lakukan konfigurasi pada file ngrok yang sudah di ekstrak
   3. Kemudian jalankan perintah berikut untuk mengonline kan laravel melalui port 8000 dengan scema berikut:

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

* 1. Dengan link yang diberikan oleh ngrok kita dapat melakukan percobaan munggunakan postman denan menanbahkan **/api/posts**



****