Tugas 3

Laporan Praktikum Simulasi Relay, Button & LED, Simulasi Jarak dan Pembuatan API Menggunakan Laravel 11 dan Ngrok

Nama : Mochamad Iftichor Al Ashief
Kelas : T4C
NIM : 233140700111082

Fakultas Vokasi
Universitas Brawijaya
Email : mhmdashief@gmail.com

Abstrak

Laporan ini membahas praktik simulasi dalam penerapan Internet of Things (IoT), yang mencakup tiga aspek utama: simulasi relay, button, dan LED; simulasi jarak; serta pembuatan API menggunakan Laravel 11 dan Ngrok. Dalam era digital saat ini, teknologi IoT semakin berkembang dan memberikan manfaat yang signifikan di berbagai sektor, seperti industri, kesehatan, dan otomasi rumah. Salah satu faktor utama dalam penerapan IoT adalah kemampuan untuk mengontrol perangkat elektronik secara otomatis dan menghubungkan sistem dengan jaringan berbasis cloud. Oleh karena itu, laporan ini bertujuan untuk memahami prinsip kerja relay dalam mengontrol perangkat elektronik melalui button dan LED, mengukur jarak menggunakan sensor ultrasonik, serta membangun API berbasis Laravel yang dapat diakses secara global melalui Ngrok. Metode yang digunakan dalam penelitian ini adalah eksperimen langsung dengan melakukan simulasi terhadap masing-masing komponen dan menguji fungsionalitasnya. Hasil dari eksperimen menunjukkan bahwa penggunaan relay, button, dan LED dapat meningkatkan efisiensi dalam pengendalian perangkat elektronik, sensor ultrasonik memiliki akurasi yang cukup tinggi dalam pengukuran jarak, serta API yang dikembangkan dengan Laravel dan Ngrok memungkinkan sistem IoT untuk berkomunikasi secara lebih fleksibel dan luas. Dengan adanya laporan ini, diharapkan dapat memberikan wawasan yang lebih mendalam mengenai konsep dan implementasi IoT bagi pengembang serta mahasiswa yang tertarik dalam bidang ini.

Abstract

This report discusses simulation practices in the implementation of the Internet of Things (IoT), which includes three main aspects: relay, button, and LED simulation; distance simulation; and API creation using Laravel 11 and Ngrok. In today's digital era, IoT technology is growing and providing significant benefits in various sectors, such as industry, health, and home automation. One of the main factors in the implementation of IoT is the ability to control electronic devices automatically and connect the system to a cloud-based network. Therefore, this report aims to understand the working principle of relays in controlling electronic devices through buttons and LEDs, measuring distance using ultrasonic sensors, and building a Laravel-based API that can be accessed globally through Ngrok.
The method used in this study is a direct experiment by simulating each component and testing its functionality. The results of the experiment show that the use of relays, buttons, and LEDs can increase efficiency in controlling electronic devices, ultrasonic sensors have quite high accuracy in measuring distance, and the API developed with Laravel and Ngrok allows IoT systems to communicate more flexibly and widely. With this report, it is hoped that it can provide deeper insight into the concept and implementation of IoT for developers and students who are interested in this field.

Keywords: *Mikrokontroler ESP32*

Bab 1

Pendahuluan

1.1 Latar Belakang

Perkembangan teknologi di era digital telah membawa perubahan signifikan dalam berbagai aspek kehidupan manusia, salah satunya adalah teknologi Internet of Things (IoT). IoT merupakan konsep yang memungkinkan perangkat elektronik saling terhubung dan berkomunikasi melalui jaringan internet, sehingga dapat dikendalikan dan dimonitor secara real-time. Penerapan IoT saat ini telah banyak digunakan dalam bidang otomasi rumah, industri, kesehatan, dan transportasi untuk meningkatkan efisiensi serta mempermudah pekerjaan manusia.

Dalam praktik implementasi IoT, terdapat beberapa aspek penting yang perlu diperhatikan, seperti pengendalian perangkat keras menggunakan relay, pemanfaatan sensor untuk mendapatkan data lingkungan, serta komunikasi antara perangkat dengan server berbasis cloud melalui API. Oleh karena itu, laporan ini membahas tiga praktik utama dalam implementasi IoT, yaitu simulasi relay, button, dan LED untuk mengontrol perangkat elektronik; simulasi jarak menggunakan sensor ultrasonik untuk pengukuran data lingkungan; serta pembuatan API menggunakan Laravel 11 dan Ngrok untuk mendukung komunikasi antar perangkat.

Pemahaman terhadap prinsip kerja relay dan button sangat penting karena kedua komponen ini banyak digunakan dalam sistem otomasi, seperti pengendalian lampu, kipas, dan perangkat lainnya. Selain itu, pengukuran jarak dengan sensor ultrasonik dapat diterapkan dalam berbagai aplikasi, seperti kendaraan pintar, sistem parkir otomatis, dan robotika. Sedangkan API berperan sebagai jembatan komunikasi antara perangkat IoT dengan aplikasi berbasis web, sehingga memungkinkan pengolahan dan pengelolaan data secara lebih efisien.

Dengan adanya penelitian ini, diharapkan mahasiswa dan pengembang teknologi dapat memahami konsep dasar serta penerapan IoT dalam kehidupan sehari-hari. Melalui eksperimen yang dilakukan, laporan ini juga memberikan wawasan praktis mengenai cara kerja dan integrasi perangkat IoT untuk mendukung inovasi teknologi di masa depan.

Bab 2

Tujuan

2.1 Tujuan Simulasi Relay, Button & LED

- Memahami prinsip kerja relay dalam mengontrol perangkat listrik.
- Mengetahui cara kerja button sebagai pemicu aksi.
- Menggunakan LED sebagai indikator status relay.

2.2 Tujuan Simulasi Jarak

- Memahami cara kerja sensor jarak ultrasonik.
- Menampilkan hasil pengukuran jarak dalam satuan cm atau m.
- Mengintegrasikan sensor dengan microcontroller.

2.3 Tujuan Pembuatan API Menggunakan Laravel 11 dan Ngrok

- Mempelajari dasar pembuatan API menggunakan Laravel 11.
- Menggunakan Ngrok untuk membuat API dapat diakses secara publik.
- Menghubungkan API dengan perangkat IoT.

Bab 3

Hasil dan Pembahasan

3.1 Hasil dan Pembahasan Simulasi Relay, Button & LED

Simulasi ini menggunakan microcontroller untuk mengontrol relay yang terhubung dengan LED dan button. Saat button ditekan, relay akan aktif dan menyalakan LED. Implementasi ini menunjukkan bagaimana perangkat elektronik dapat dikontrol melalui sistem digital secara efisien.

3.2 Hasil dan Pembahasan Simulasi Jarak

Sensor ultrasonik digunakan untuk mengukur jarak dengan prinsip pemantulan gelombang suara. Hasil pengukuran ditampilkan dalam layar serial monitor. Dari hasil uji coba, ditemukan bahwa sensor memiliki akurasi yang cukup tinggi dalam jarak tertentu, meskipun terdapat deviasi kecil akibat faktor lingkungan.

3.3 Hasil dan Pembahasan Pembuatan API Menggunakan Laravel 11 dan Ngrok

Dalam praktik ini, Laravel 11 digunakan untuk membuat API yang dapat menerima dan mengirim data ke perangkat IoT. Ngrok digunakan untuk menjadikan API dapat diakses dari luar jaringan lokal. Implementasi ini memungkinkan komunikasi dua arah antara perangkat IoT dan server berbasis cloud, membuka peluang integrasi lebih lanjut dalam sistem IoT berbasis web.

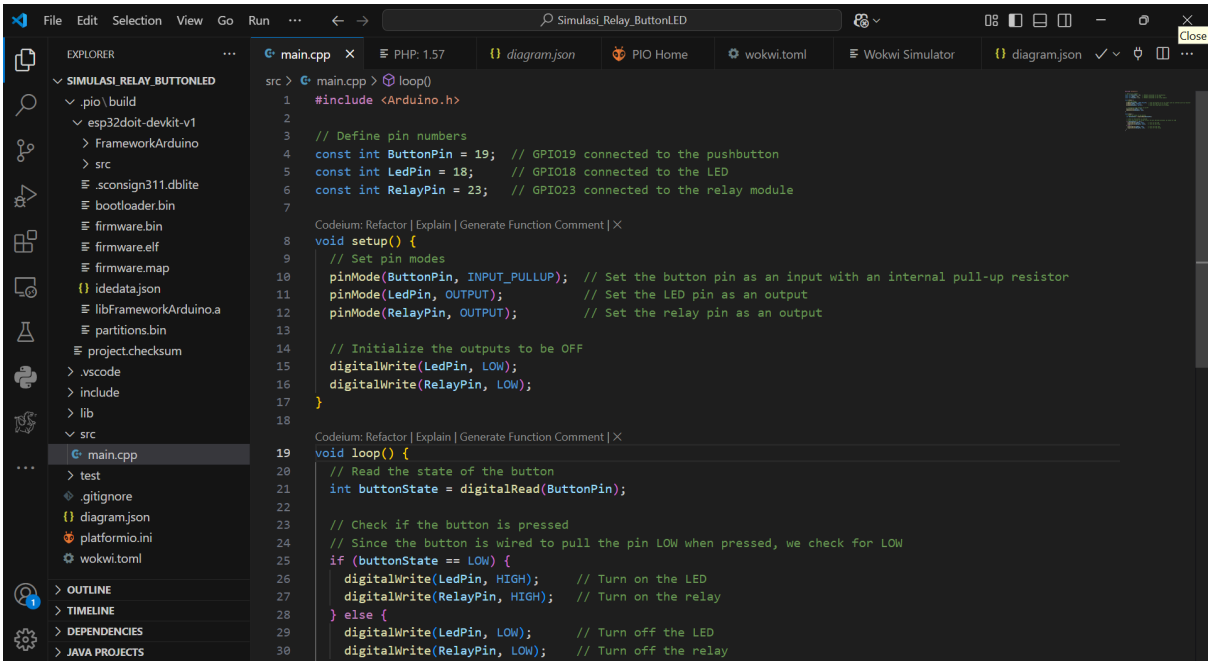
KESIMPULAN

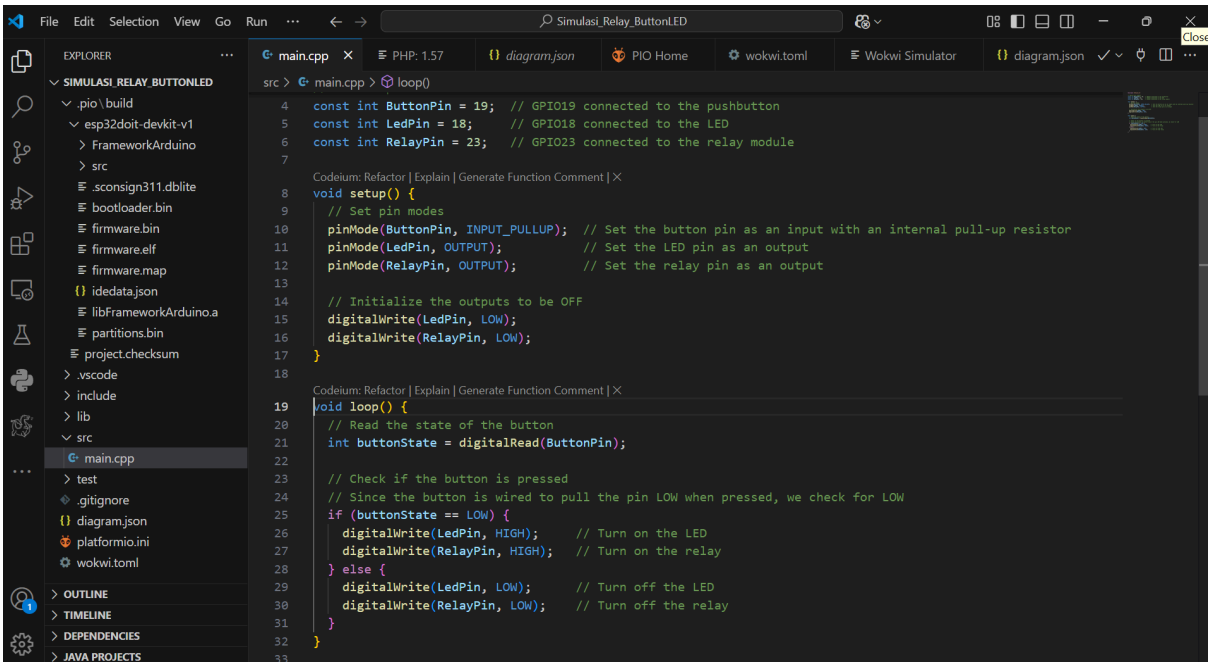
Laporan ini membahas praktik IoT dari berbagai aspek, yaitu simulasi relay dan button, pengukuran jarak menggunakan sensor ultrasonik, serta pembuatan API dengan Laravel 11 dan Ngrok. Hasil praktik menunjukkan bahwa setiap komponen berperan penting dalam membangun ekosistem IoT yang lebih efisien dan dapat diakses dari mana saja. Dengan memahami dasar-dasar ini, pengembangan IoT dapat lebih dioptimalkan untuk berbagai kebutuhan teknologi masa depan.

BAB 4 LAMPIRAN & DOKUMENTASI

1. Proyek Simulasi Relay, Relay & Button

a. main.cpp

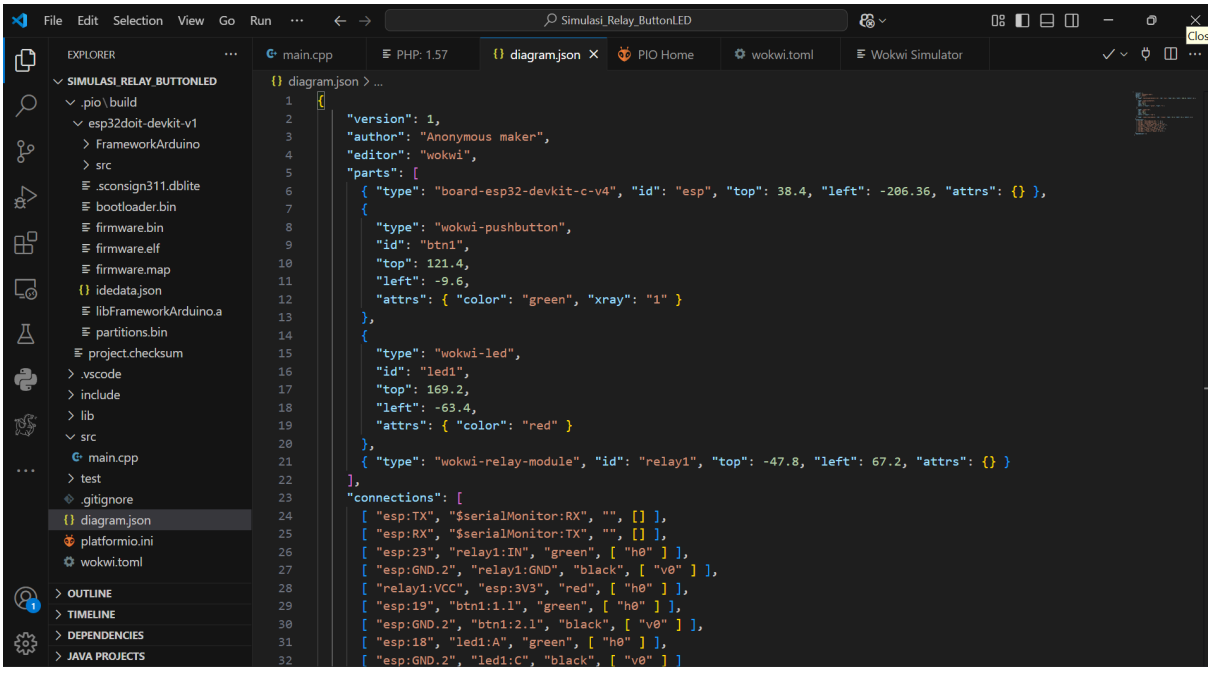





```
#include <Arduino.h>

// Define pin numbers
const int ButtonPin = 19;  // GPIO19 connected to the pushbutton
const int LedPin = 18;     // GPIO18 connected to the LED
const int RelayPin = 23;   // GPIO23 connected to the relay module

void setup() {
```

```
  // Set pin modes
  pinMode(ButtonPin, INPUT_PULLUP);  // Set the button pin as an input with an internal
pull-up resistor
  pinMode(LedPin, OUTPUT);          // Set the LED pin as an output
  pinMode(RelayPin, OUTPUT);        // Set the relay pin as an output

  // Initialize the outputs to be OFF
  digitalWrite(LedPin, LOW);
  digitalWrite(RelayPin, LOW);
}

void loop() {
  // Read the state of the button
  int buttonState = digitalRead(ButtonPin);

  // Check if the button is pressed
  // Since the button is wired to pull the pin LOW when pressed, we check for LOW
  if (buttonState == LOW) {
    digitalWrite(LedPin, HIGH);    // Turn on the LED
    digitalWrite(RelayPin, HIGH);  // Turn on the relay
  } else {
    digitalWrite(LedPin, LOW);     // Turn off the LED
    digitalWrite(RelayPin, LOW);   // Turn off the relay
  }
}
```

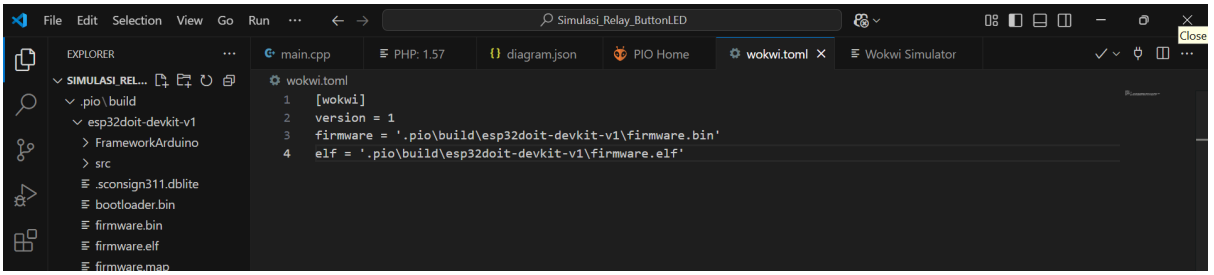
b. diagram.json




```
{
  "version": 1,
  "author": "Anonymous maker",
  "editor": "wokwi",
  "parts": [
```

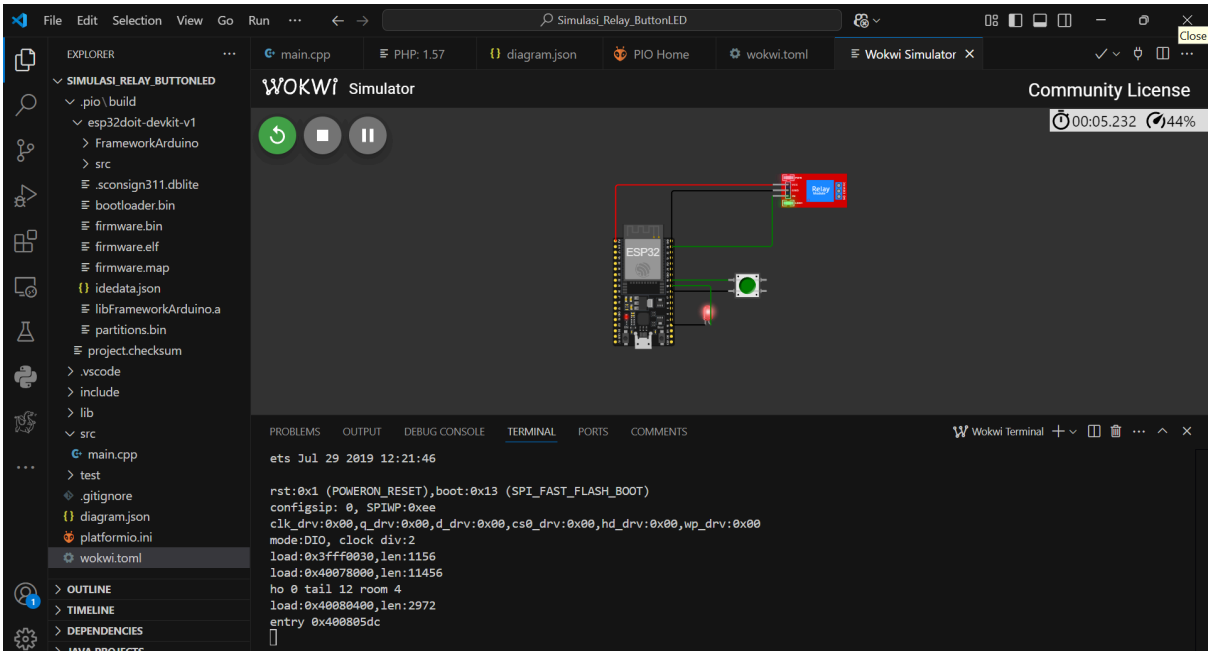
```
    { "type": "board-esp32-devkit-c-v4", "id": "esp", "top": 38.4, "left": -206.36, "attrs": {} },
    {
      "type": "wokwi-pushbutton",
      "id": "btn1",
      "top": 121.4,
      "left": -9.6,
      "attrs": { "color": "green", "xray": "1" }
    },
    {
      "type": "wokwi-led",
      "id": "led1",
      "top": 169.2,
      "left": -63.4,
      "attrs": { "color": "red" }
    },
    { "type": "wokwi-relay-module", "id": "relay1", "top": -47.8, "left": 67.2, "attrs": {} }
  ],
  "connections": [
    [ "esp:TX", "$serialMonitor:RX", "", [] ],
    [ "esp:RX", "$serialMonitor:TX", "", [] ],
    [ "esp:23", "relay1:IN", "green", [ "h0" ] ],
    [ "esp:GND.2", "relay1:GND", "black", [ "v0" ] ],
    [ "relay1:VCC", "esp:3V3", "red", [ "h0" ] ],
    [ "esp:19", "btn1:1.l", "green", [ "h0" ] ],
    [ "esp:GND.2", "btn1:2.l", "black", [ "v0" ] ],
    [ "esp:18", "led1:A", "green", [ "h0" ] ],
    [ "esp:GND.2", "led1:C", "black", [ "v0" ] ]
  ],
  "dependencies": {}
}
```

c. **wokwi.toml**



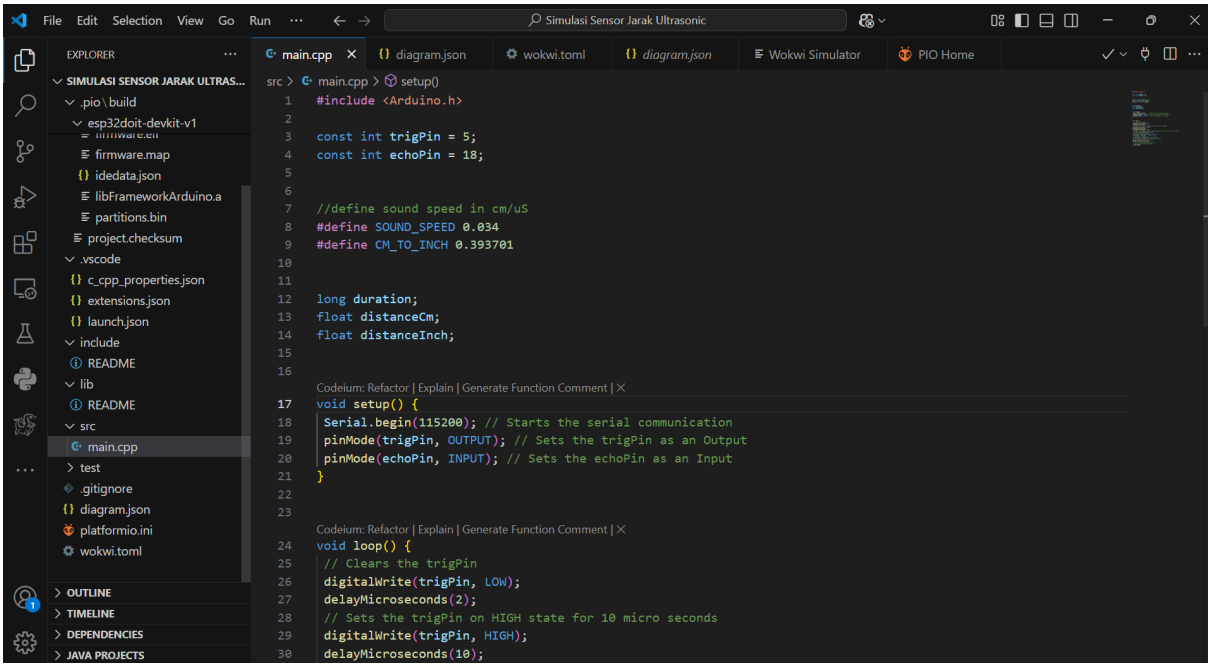

```
[wokwi]
version = 1
firmware = '.pio\build\esp32doit-devkit-v1\firmware.bin'
elf = '.pio\build\esp32doit-devkit-v1\firmware.elf'
```

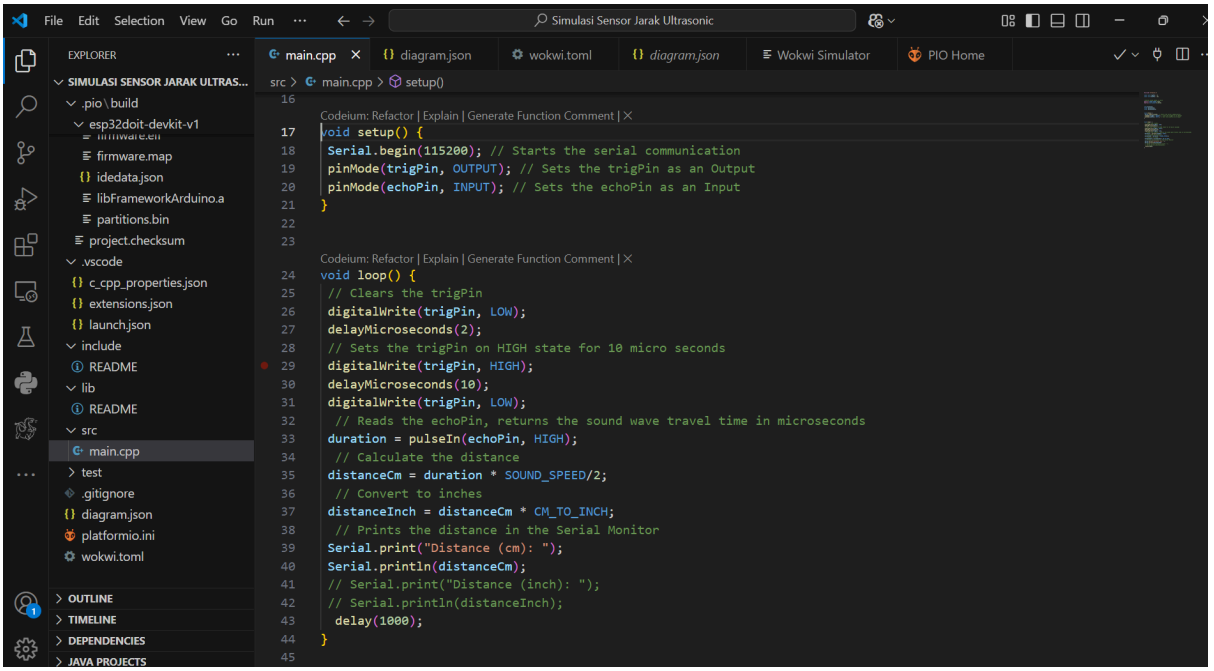
d. **Hasil**



2. Proyek Simulasi Jarak

a. main.cpp

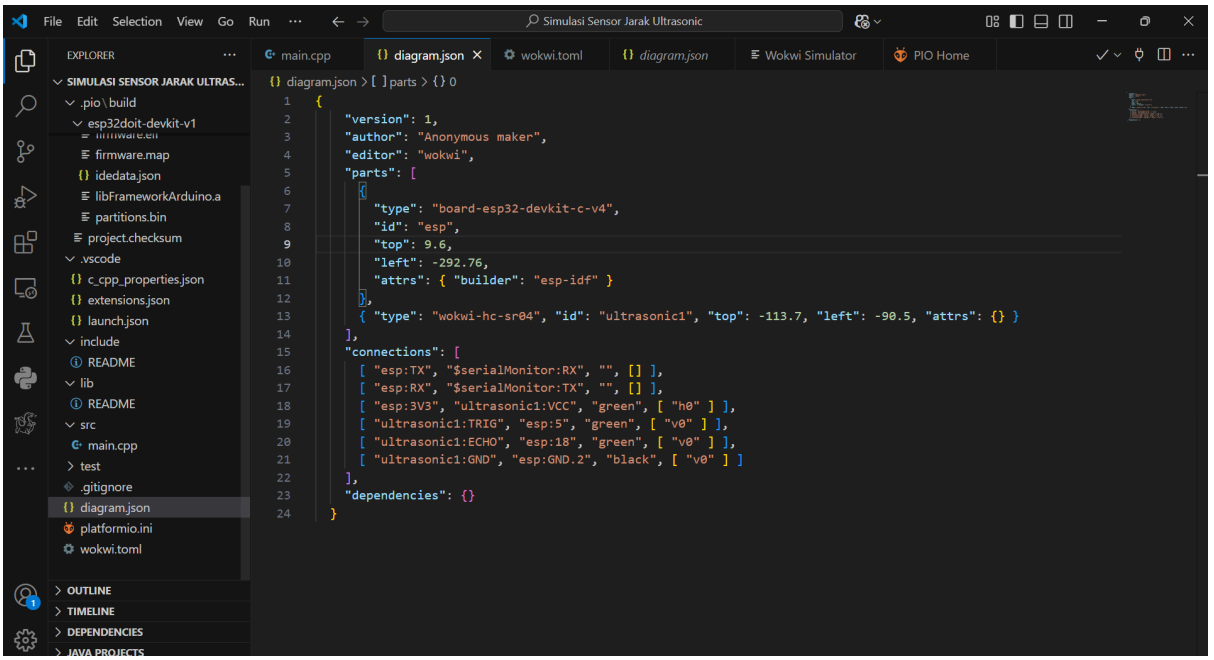





```
#include <Arduino.h>

const int trigPin = 5;
const int echoPin = 18;


//define sound speed in cm/uS
#define SOUND_SPEED 0.034
#define CM_TO_INCH 0.393701


long duration;
float distanceCm;
float distanceInch;


void setup() {
 Serial.begin(115200); // Starts the serial communication
 pinMode(trigPin, OUTPUT); // Sets the trigPin as an Output
 pinMode(echoPin, INPUT); // Sets the echoPin as an Input
}


void loop() {
 // Clears the trigPin
 digitalWrite(trigPin, LOW);
 delayMicroseconds(2);
 // Sets the trigPin on HIGH state for 10 micro seconds
 digitalWrite(trigPin, HIGH);
 delayMicroseconds(10);
 digitalWrite(trigPin, LOW);
  // Reads the echoPin, returns the sound wave travel time in microseconds
 duration = pulseIn(echoPin, HIGH);
```

```
 // Calculate the distance
distanceCm = duration * SOUND_SPEED/2;
 // Convert to inches
distanceInch = distanceCm * CM_TO_INCH;
 // Prints the distance in the Serial Monitor
Serial.print("Distance (cm): ");
Serial.println(distanceCm);
// Serial.print("Distance (inch): ");
// Serial.println(distanceInch);
 delay(1000);
}
```

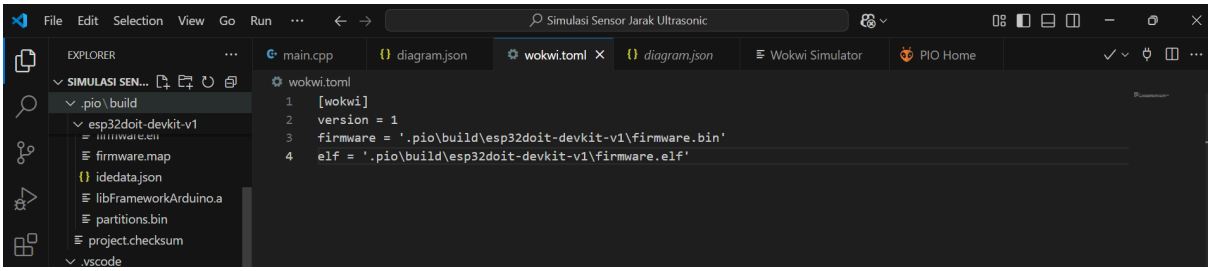
b. diagram.json




```
{
  "version": 1,
  "author": "Anonymous maker",
  "editor": "wokwi",
  "parts": [
   {
    "type": "board-esp32-devkit-c-v4",
    "id": "esp",
    "top": 9.6,
    "left": -292.76,
    "attrs": { "builder": "esp-idf" }
   },
   { "type": "wokwi-hc-sr04", "id": "ultrasonic1", "top": -113.7, "left": -90.5, "attrs": {} }
  ],
  "connections": [
   [ "esp:TX", "$serialMonitor:RX", "", [] ],
   [ "esp:RX", "$serialMonitor:TX", "", [] ],
   [ "esp:3V3", "ultrasonic1:VCC", "green", [ "h0" ] ],
   [ "ultrasonic1:TRIG", "esp:5", "green", [ "v0" ] ],
   [ "ultrasonic1:ECHO", "esp:18", "green", [ "v0" ] ],
   [ "ultrasonic1:GND", "esp:GND.2", "black", [ "v0" ] ]
```

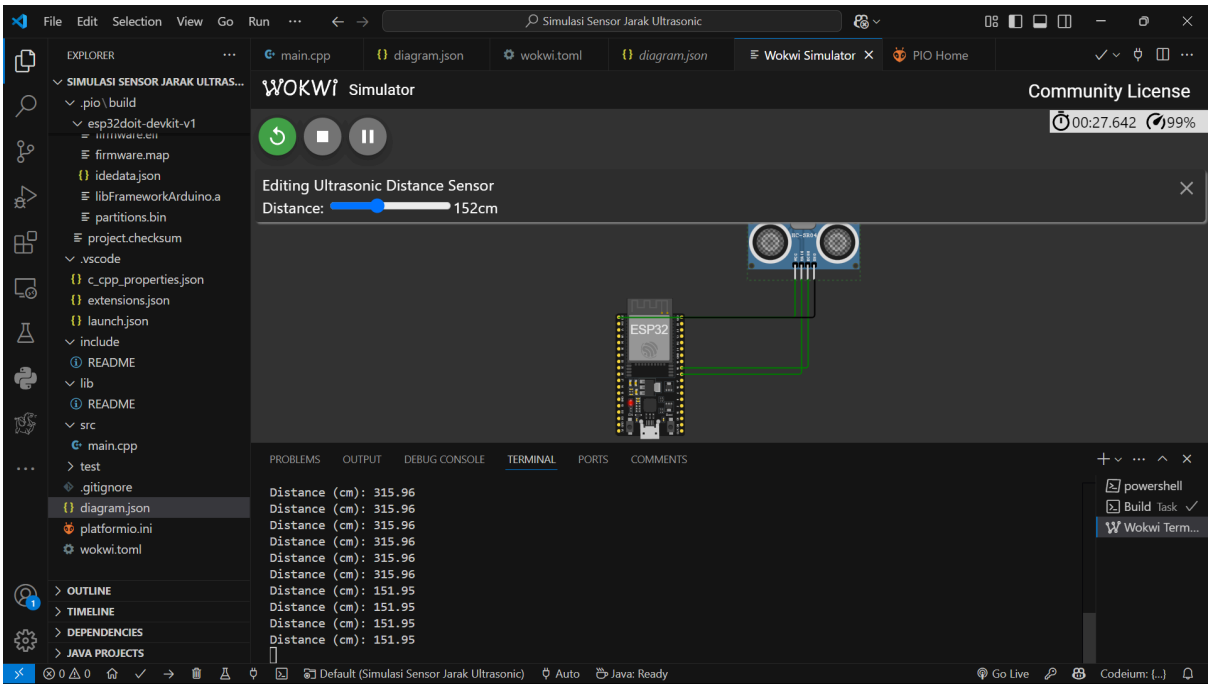
```
  ],
  "dependencies": {}
}
```

c. **wokwi.toml**



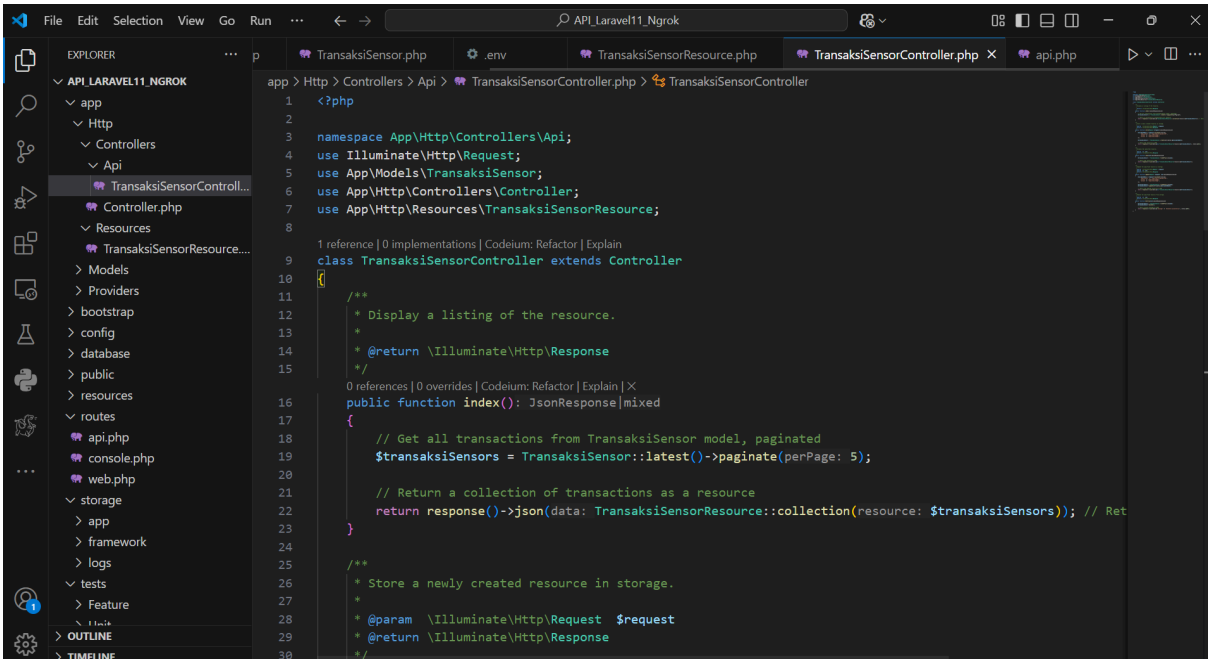

```
[wokwi]
version = 1
firmware = '.pio\build\esp32doit-devkit-v1\firmware.bin'
elf = '.pio\build\esp32doit-devkit-v1\firmware.elf'
```

d. **Hasil**



3. Proyek Pembuatan API Menggunakan Laravel 11 dan Ngrok

a. **TransaksiSensorController.php**

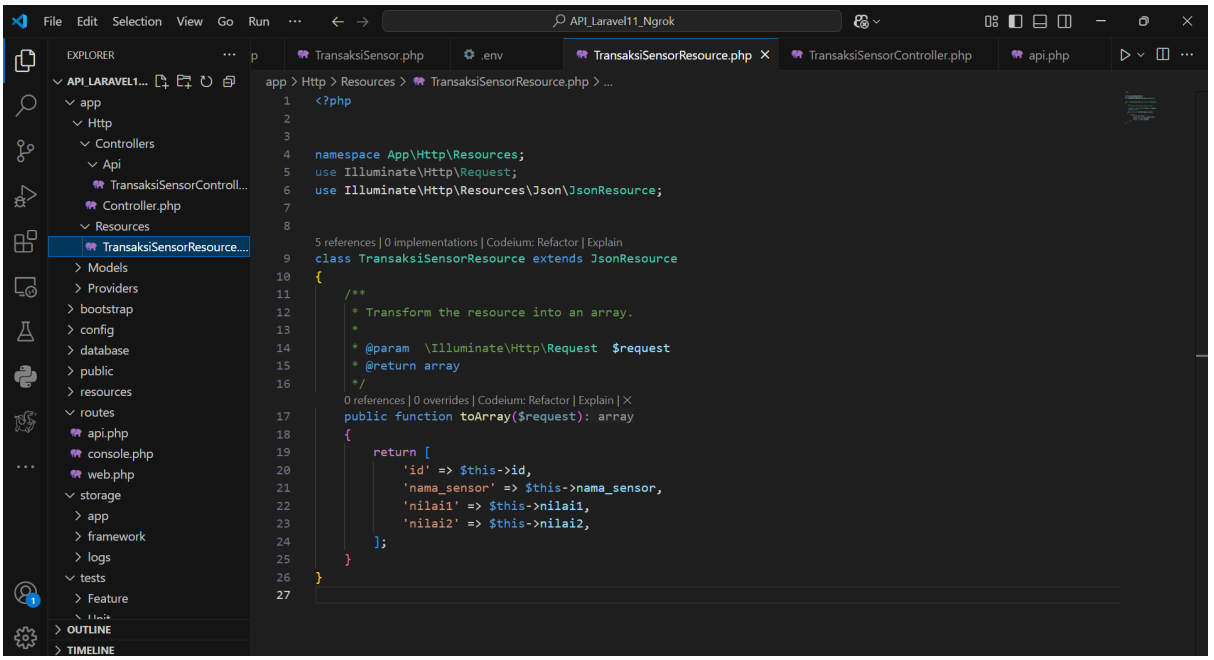



```
<?php

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
{
   /**
    * Display a listing of the resource.
    *
    * @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 response()->json(TransaksiSensorResource::collection($transaksiSensors)); // Return
wrapped in JSON response
   }

   /**
    * 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 the resource as JSON
    return response()->json(new TransaksiSensorResource($transaksiSensor), 201);
}

/**
 * Display the specified resource.
 *
 * @param  int  $id
 * @return \Illuminate\Http\Response
 */
public function show($id)
{
    $transaksiSensor = TransaksiSensor::findOrFail($id);

    // Return the resource as JSON
    return response()->json(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 the updated resource as JSON
    return response()->json(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 success message as JSON
        return response()->json(['message' => 'Deleted successfully'], 204);
    }
}
```

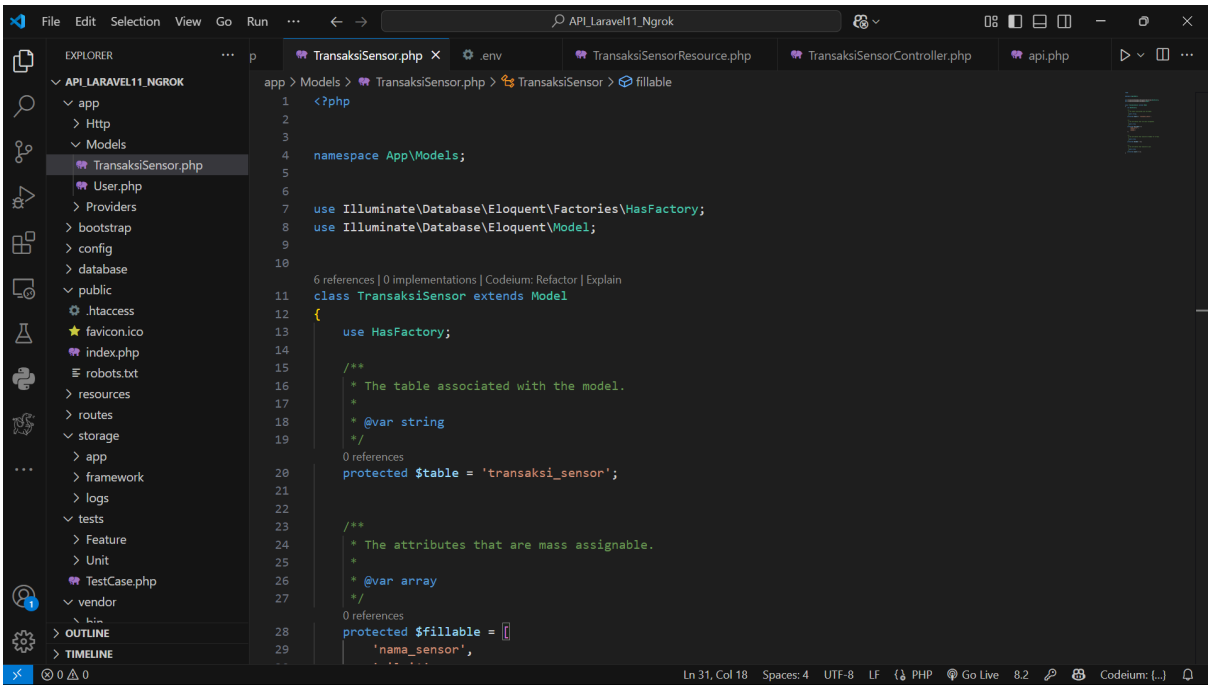
b. TransaksiSensorResource.php




```
<?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,
        ];
    }
}
```

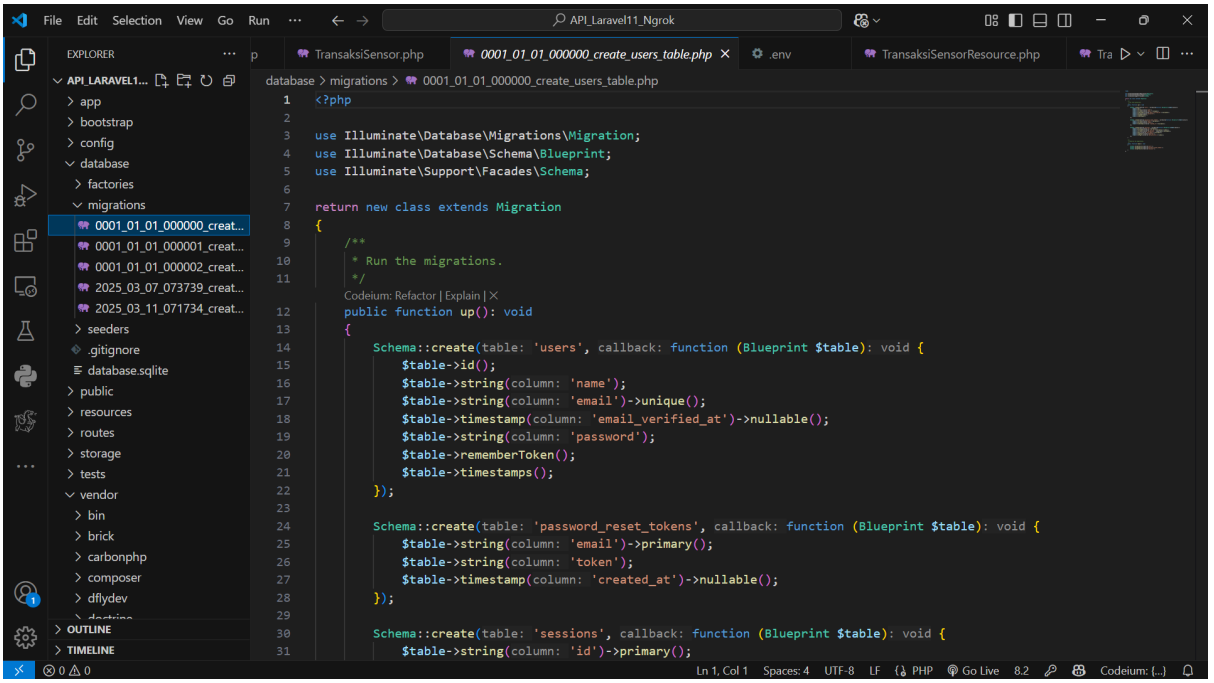
c. **Transaksisensor.php**




```
<?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 = [];
}
```

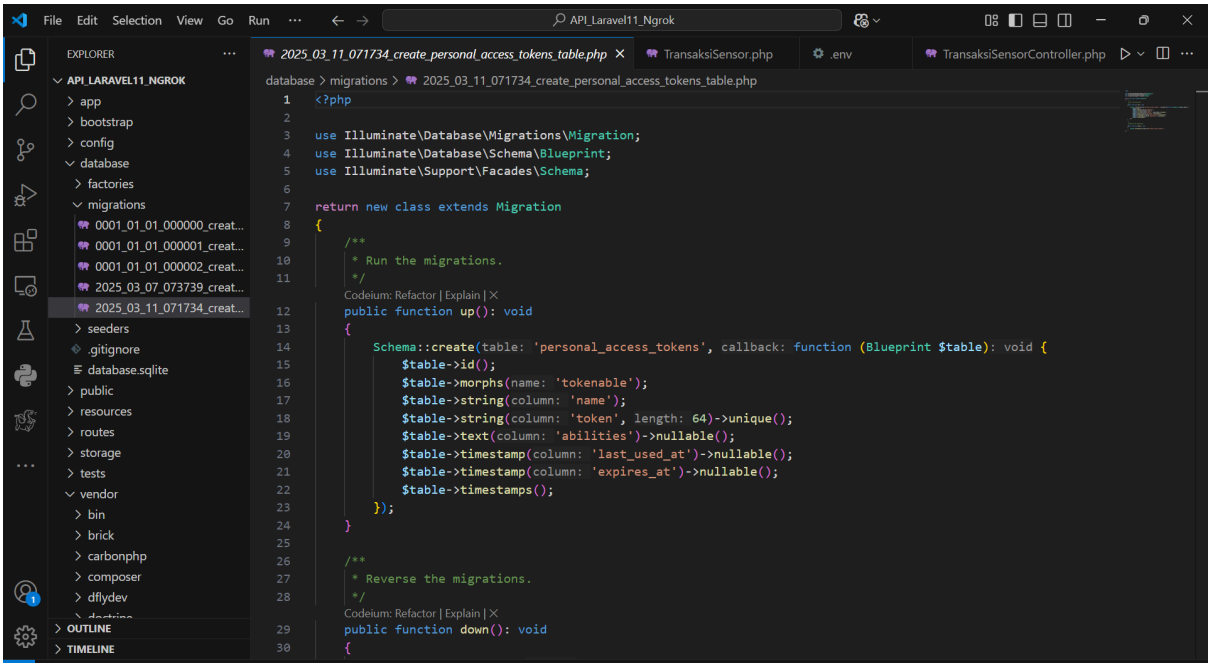
d. 2025_03_09_102944_create_transaksi_sensors_table.php




```
<?php

use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

return new class extends Migration
```

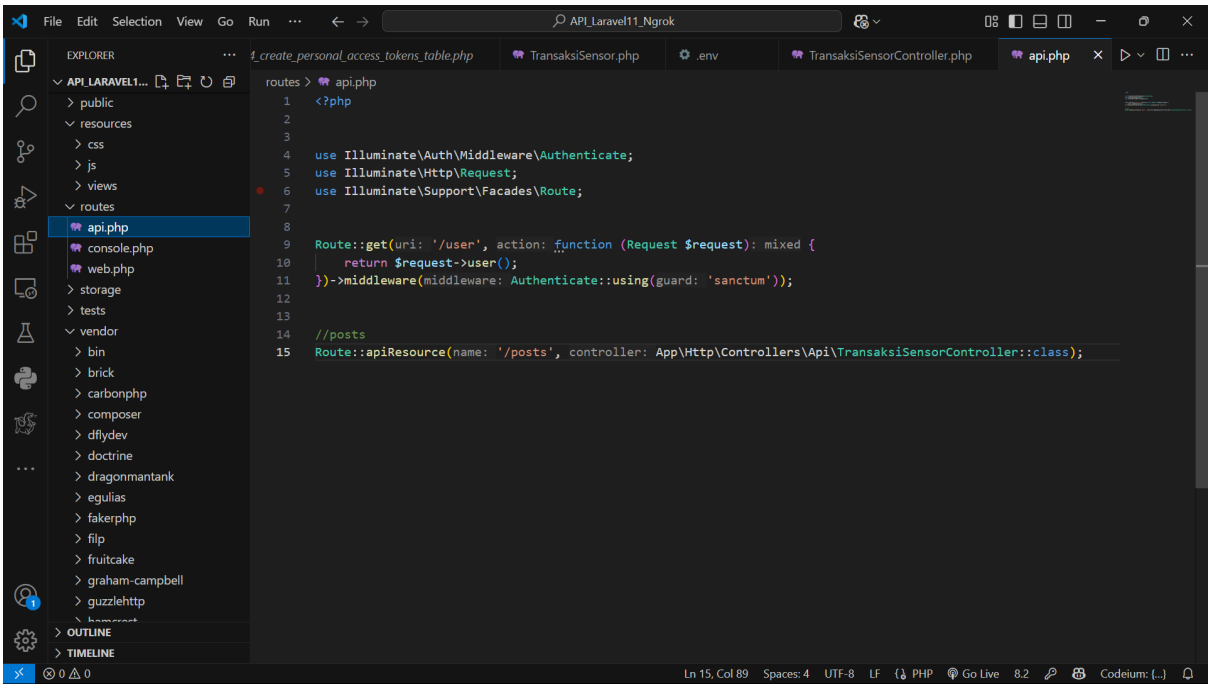
```
{
    /**
     * Run the migrations.
     */
    public function up(): void
    {
        Schema::create('users', function (Blueprint $table) {
            $table->id();
            $table->string('name');
            $table->string('email')->unique();
            $table->timestamp('email_verified_at')->nullable();
            $table->string('password');
            $table->rememberToken();
            $table->timestamps();
        });

        Schema::create('password_reset_tokens', function (Blueprint $table) {
            $table->string('email')->primary();
            $table->string('token');
            $table->timestamp('created_at')->nullable();
        });

        Schema::create('sessions', function (Blueprint $table) {
            $table->string('id')->primary();
            $table->foreignId('user_id')->nullable()->index();
            $table->string('ip_address', 45)->nullable();
            $table->text('user_agent')->nullable();
            $table->longText('payload');
            $table->integer('last_activity')->index();
        });
    }

    /**
     * Reverse the migrations.
     */
    public function down(): void
    {
        Schema::dropIfExists('users');
        Schema::dropIfExists('password_reset_tokens');
        Schema::dropIfExists('sessions');
    }
};
```

e. **2025_03_09_120026_create_personal_access_tokens_table.php**



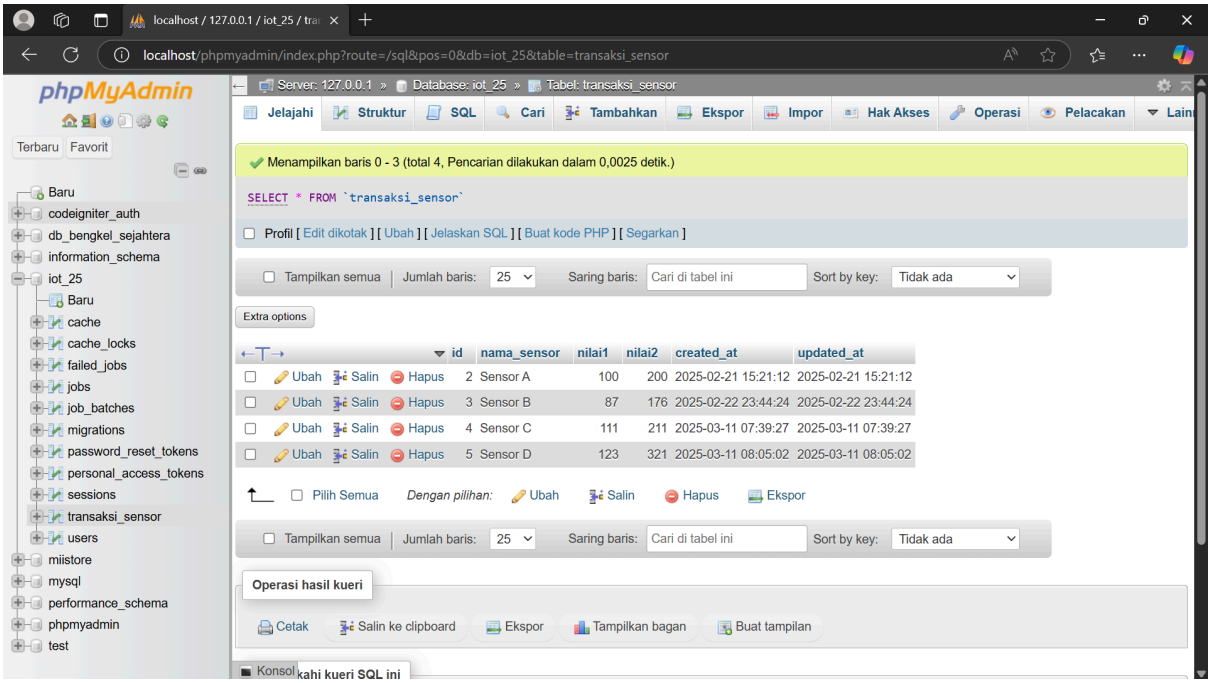

```php
<?php

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('personal_access_tokens', function (Blueprint $table) {
            $table->id();
            $table->morphs('tokenable');
            $table->string('name');
            $table->string('token', 64)->unique();
            $table->text('abilities')->nullable();
            $table->timestamp('last_used_at')->nullable();
            $table->timestamp('expires_at')->nullable();
            $table->timestamps();
        });
    }

    /**
     * Reverse the migrations.
     */
    public function down(): void
    {
        Schema::dropIfExists('personal_access_tokens');
    }
};
```

f. **api.php**

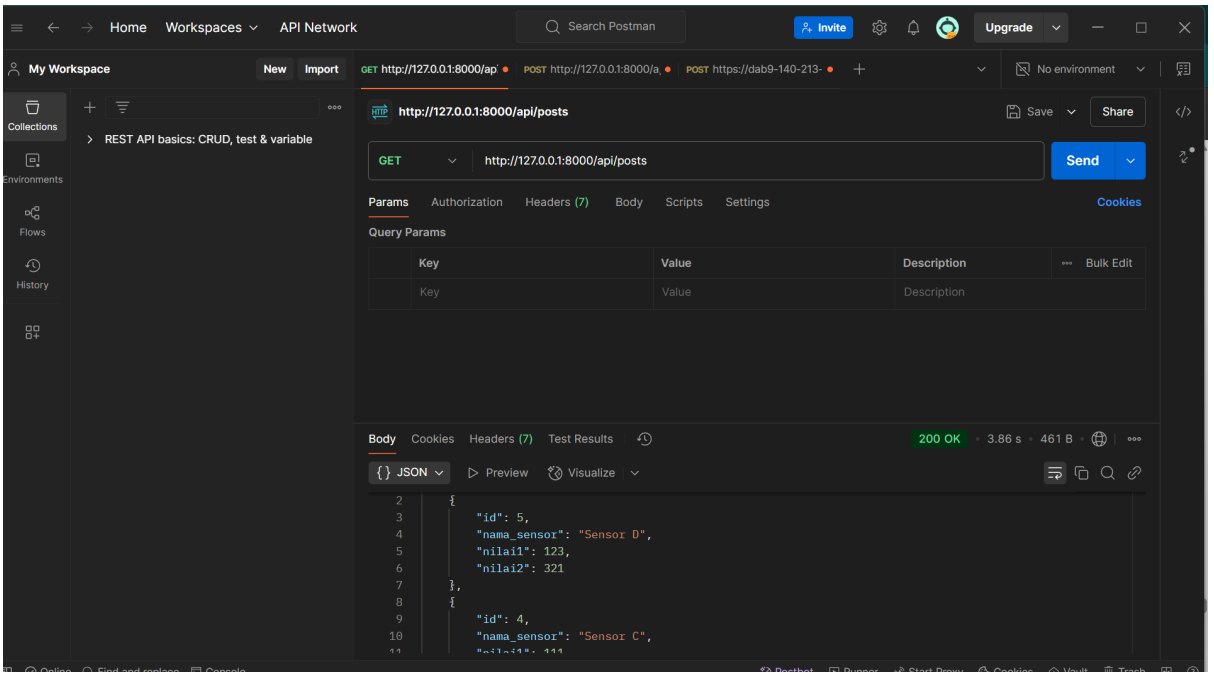



```
<?php


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);
```

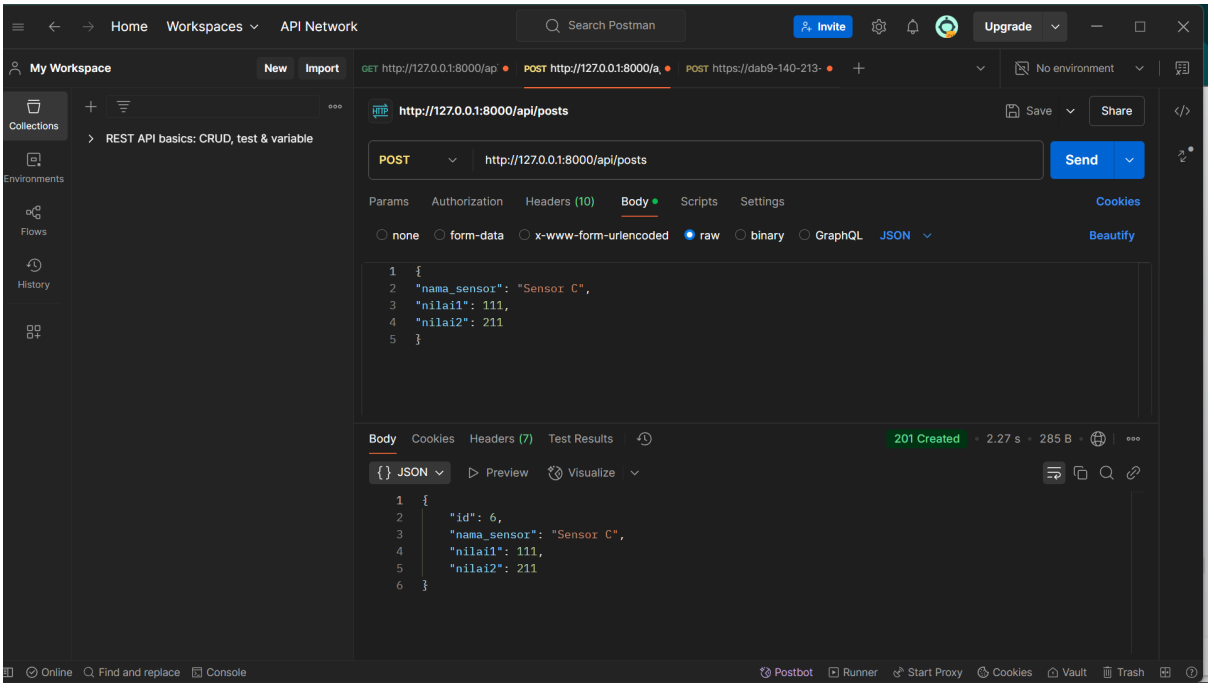
g. **Database**



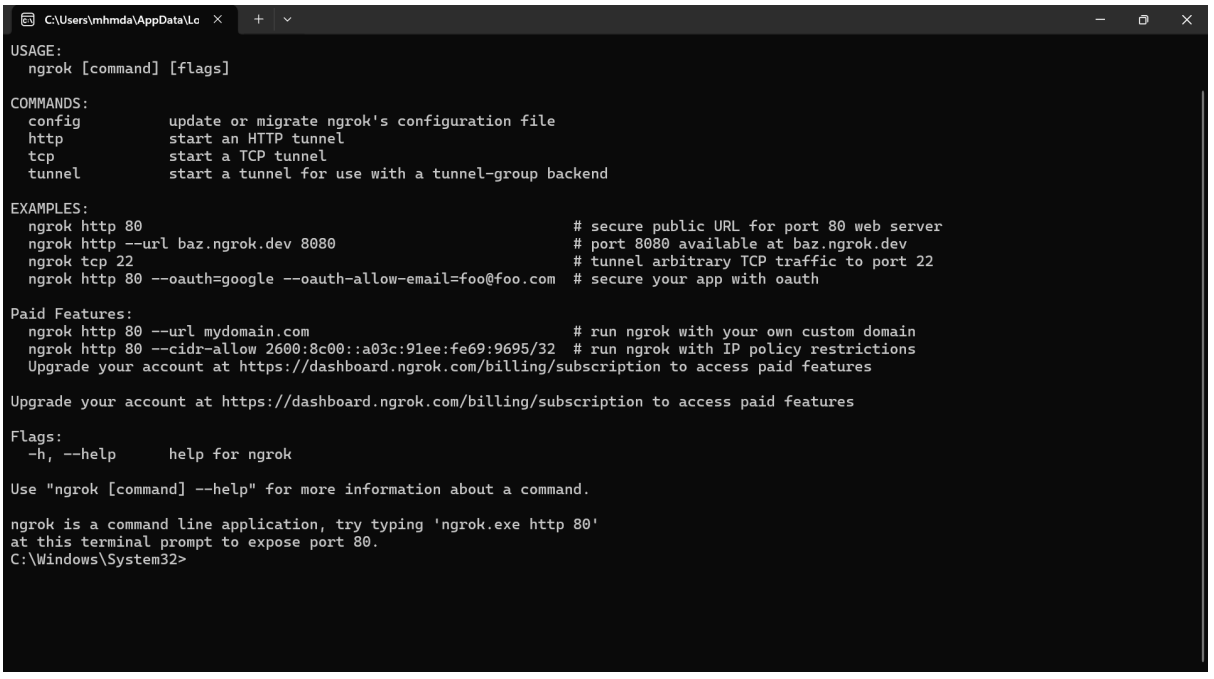
H. Laravel API (Get Postman)



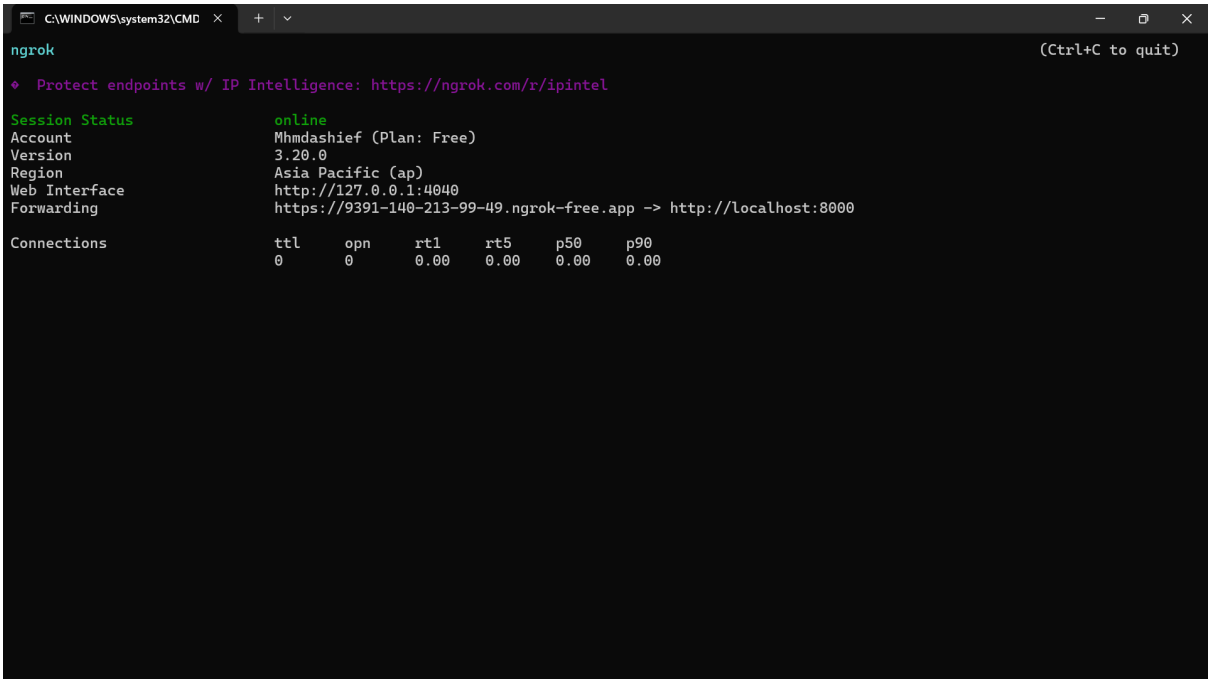
I. Laravel API (Post Postman)



J. ngrok.exe



K. Mengonlinekan ngrok.exe



L. Menambahkan Database (ngrok.exe)