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

**Praktik Pembuatan API Menggunakan Laravel 11 dan Ngrok**

**& Praktik Akses API Melalui Simulasi WOKWI Dan Penambambahan Dashboard Sensor**

Author : Ahmad Surya Alam Syah

Fakultas Vokasi, Universitas Brawijaya

Email: [alamsyah200527@gmail.com](mailto:alamsyah200527@gmail.com)

**Abstrak**

Eksperimen ini bertujuan untuk belajar membuat API menggunakan Laravel 11 dan menghubungkannya dengan simulasi IoT di platform WOKWI. API yang dibuat akan menerima data dari mikrokontroler ESP32 yang telah disiapkan dalam simulasi WOKWI dan kemudian diproses dan ditampilkan secara real-time oleh API Laravel. Selain itu, saya akan menggunakan Ngrok untuk membuat tunnel ke server lokal Laravel untuk memastikan API dapat diakses oleh publik.

**Keywords***: Internet of Things, API, Laravel 12, Ngrok, Tunnel*

**1. Pendahuluan**

* 1. **Latar Belakang**

API (Application Programming Interface), yang berfungsi untuk menghubungkan aplikasi perangkat lunak (seperti server backend) dengan perangkat keras (seperti ESP32), merupakan komponen penting dari teknologi Internet of Things (IoT). Perangkat fisik dapat berkomunikasi dengan sistem digital melalui jaringan internet.

Karena banyaknya fitur dan kemudahan penggunaannya, Framework Laravel adalah alat pengembangan API yang populer. Eksperimen ini akan menggunakan Laravel 11 untuk membuat API yang dapat menerima data dari platform WOKWI yang disimulasikan ESP32. Data ini akan diproses dan disimpan dalam database. saya juga akan menggunakan Ngrok untuk tunnel ke server lokal Laravel untuk memastikan API dapat diakses secara publik.

* 1. **Tujuan Eksperimen**

Tujuan dari eksperimen ini adalah untuk mempelajari cara membuat API menggunakan framework Laravel 11 dan mengintegrasikannya dengan simulasi IoT di platform WOKWI. Kita juga akan belajar bagaimana API yang dibuat dapat menerima data dari mikrokontroler ESP32 yang telah dikonfigurasi dalam simulasi WOKWI. saya juga akan menggunakan Ngrok untuk membuat API Laravel dapat diakses publik. Dengan melakukan praktik ini, diharapkan kita dapat memperoleh pemahaman yang lebih baik tentang cara menghubungkan perangkat IoT (ESP32) dan server backend (Laravel), serta tentang pengembangan API IoT dan integrasi perangkat lunak dan keras.

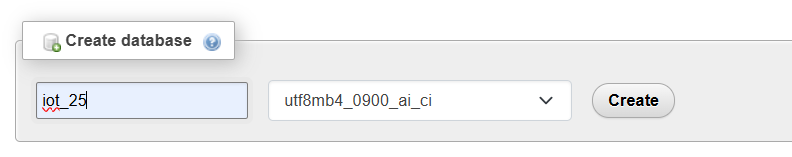
**2. Metodologi**

1. **Tools**
2. Laravel 11
3. Laragon
4. Ngrok
5. Postman
6. PHP 8.2.23
7. VS Code
8. **Langkah Implementasi**
9. Persiapan Perangkat:
10. Laptop
11. internet
12. Download paket-paket yang dibutuhkan untuk memulai laravel 11 dengan mengetikkan perintah berikut di terminal/command prompt.

* composer create-project --prefer-dist laravel/laravel:^11.0 iot\_api
* cd iot\_api
* buka code menggunakan perintah: code .

1. Konfigurasi Database:

* Buka phpmyadmin
* Klik new
* Buat database di phpmyadmin dengan nama **iot\_25**



* Ubah isi konfigurasi file .env

DB\_CONNECTION=mysql

DB\_HOST=127.0.0.1

DB\_PORT=3306

DB\_DATABASE=iot\_25

DB\_USERNAME=root

DB\_PASSWORD=

1. Pembuatan Model dan Migrations

* Buat file model TransaksiSensor.php dengan cara menjalankan perintah berikut di terminal : php artisan make:model TransaksiSensor –m
* Ubah file 2025\_03\_06\_065344\_create\_transaksi\_sensors\_table.php Manjadi seperti ini:

<?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\_sensors', 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\Model;

class TransaksiSensors extends Model

{

    protected $table = 'transaksi\_sensors';

    protected $fillable = [

        'nama\_sensor',

        'nilai1',

        'nilai2',

    ];

}

* Kemudian jalankan perintah berikut untuk membuat tabel : php artisan migrate

1. Pembuatan Resource

* Buat Resource dengan menjalankan perintah : php artisan make:resource TransaksiSensorResource
* Ubah isi file TransaksiSensorResource.php yang ada di folder app-Http-Resources 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,

        ];

    }

}

1. Pembuatan API controller

* 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\TransaksiSensors;

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 = TransaksiSensors::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',

        ]);

        $transaksiSensor = TransaksiSensors::create($validatedData);

        return new TransaksiSensorResource($transaksiSensor);

    }

    /\*\*

     \* Display the specified resource.

     \*

     \* @param  int  $id

     \* @return \Illuminate\Http\Response

     \*/

    public function show($id)

    {

        $transaksiSensor = TransaksiSensors::findOrFail($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',

        ]);

        $transaksiSensor = TransaksiSensors::findOrFail($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)

    {

        $transaksiSensor = TransaksiSensors::findOrFail($id);

        $transaksiSensor->delete();

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

    }

}

1. Pembuatan Route API

* 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) {

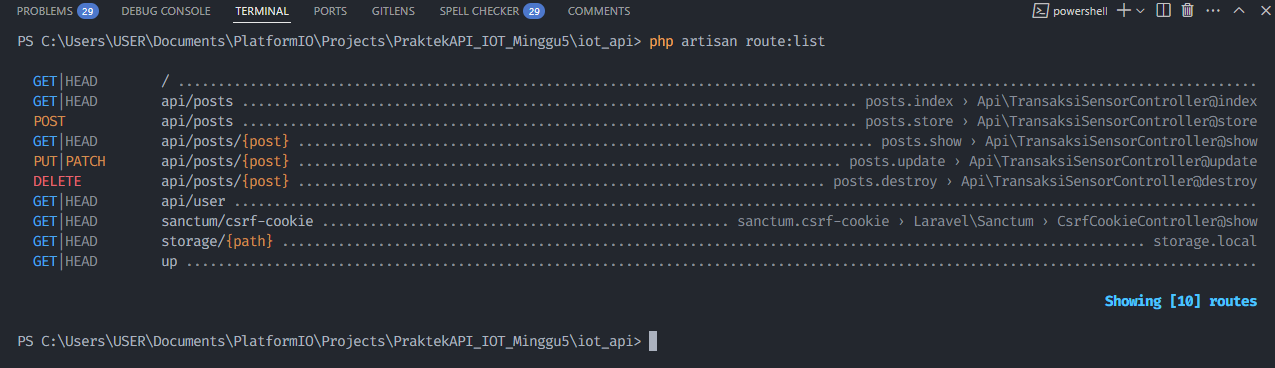
    return $request->user();

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

//posts

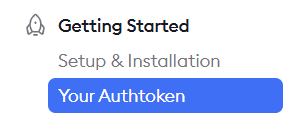
Route::apiResource('/posts', App\Http\Controllers\Api\TransaksiSensorController::class);

* Kemudian pastikan routes telah terbentuk dengan menjalankan perintah : php artisan route:list
* pastikan tampilan sebagai berikut :

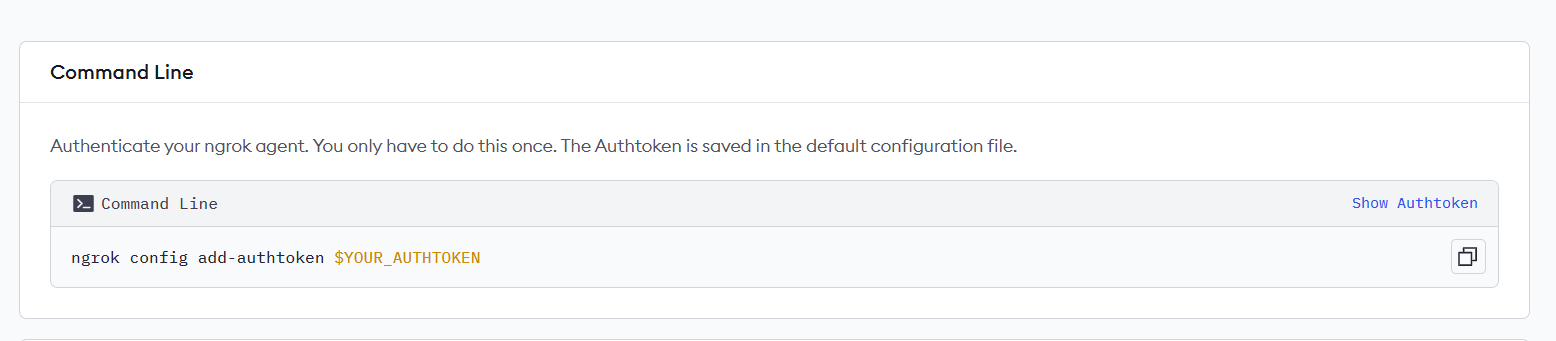


1. Melakukan testing dengan Ngrok dan Postmen

* Untuk melakukan testing, gunakan tools postman dan ngrok dengan langkah sebagai berikut :
* Download aplikasi postman pada link berikut : <https://www.postman.com/downloads/>
* Download dan install aplikasi ngrok pada URL : https://dashboard.ngrok.com/signup kemudian lakukan registrasi.
* Aktifasi Akun Ngrok
* Buka command prompt dari alamat folder ekstraksi Ngrok
* Masuk Ke dashboard web Ngrok
* Lakukan Login
* Masuk ke Halaman Your Auth



* Copy dan paste pada section Command Line ke CMD hasil ekstraksi ngrok

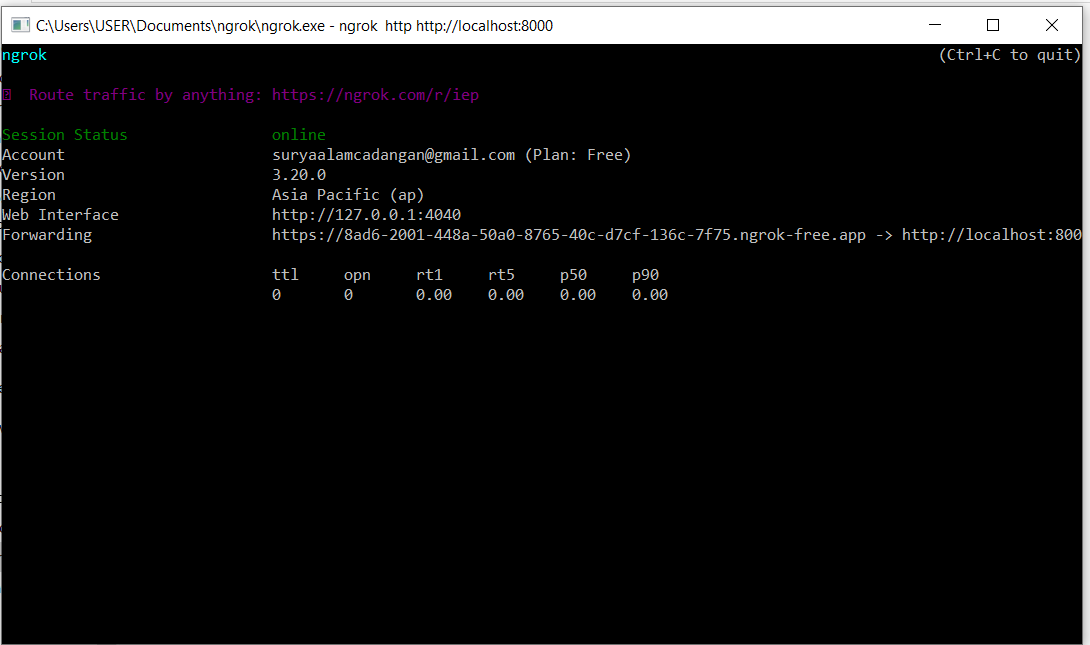


1. Percobaan akses api

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



* Kemudian jalankan perintah berikut untuk mengonline kan laravel melalui port 8000: ngrok http <http://localhost:8000>



* Kemudian lakukan percobaan menggunakan postman menggunakan URL yang diberikan oleh ngrok: https://8ad6-2001-448a-50a0-8765-40c-d7cf-136c-7f75.ngrok-free.app
* Buka Postmen
* Untuk melakukan percobaan POST api, maka URL harus ditambahkan alamat endpoint menjadi sebagai berikut : https://8ad6-2001-448a-50a0-8765-40c-d7cf-136c-7f75.ngrok-free.app/api/posts
* Masuk ke body lalu pilih raw dan pilih ubah format menjadi json
* Isi form seperti ini:

{

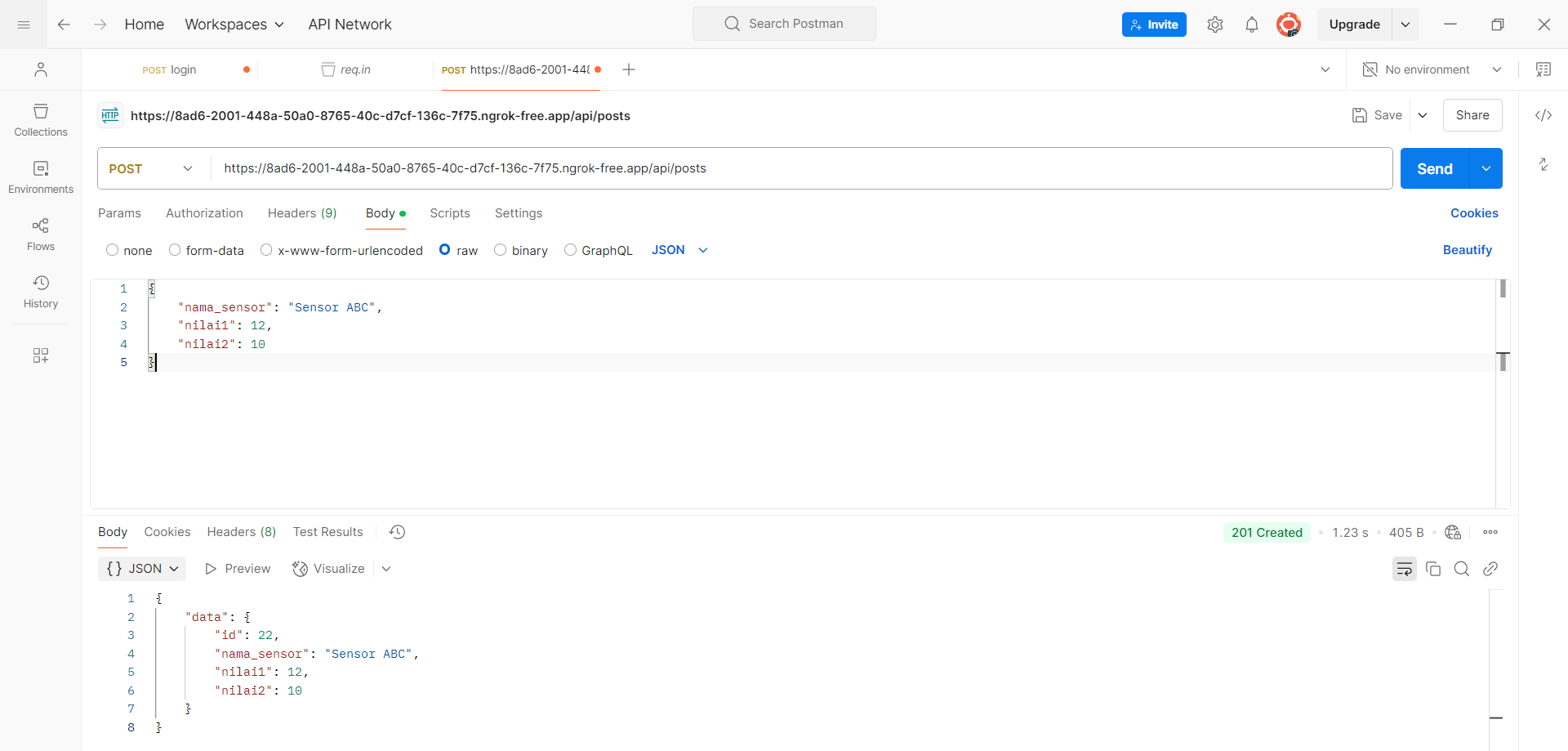
    "nama\_sensor": "Sensor ABC",

    "nilai1": 12,

    "nilai2": 10

}

* Klik Send

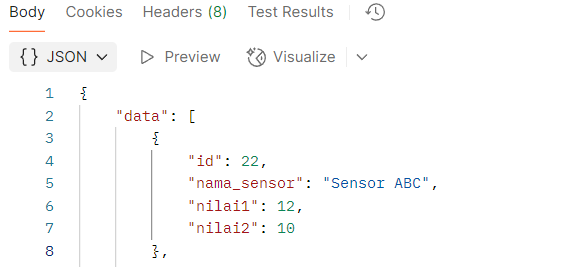


Disini terlihat bahwa mendapat status code 201 yang artinya data berhasil di buat

* Untuk Mamverifikasi ubah metod POST ke Get

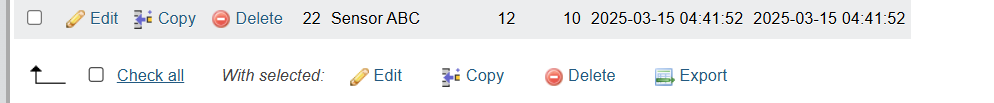


* Lalu klik send



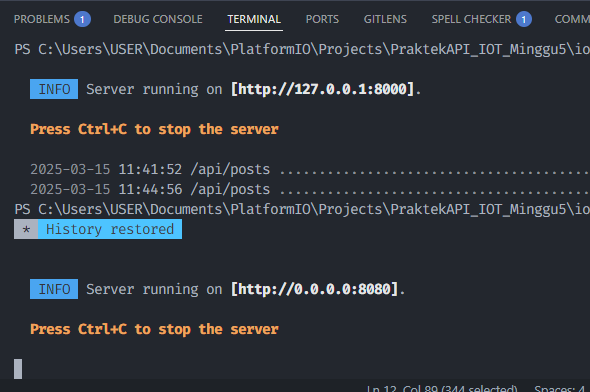
Disini terlihat bahwa data berhasil di ambil

* Masuk ke phpmyadmin
* Klik table transaksi\_sensors
* Cek Apakah Data sudah masuk ke Database



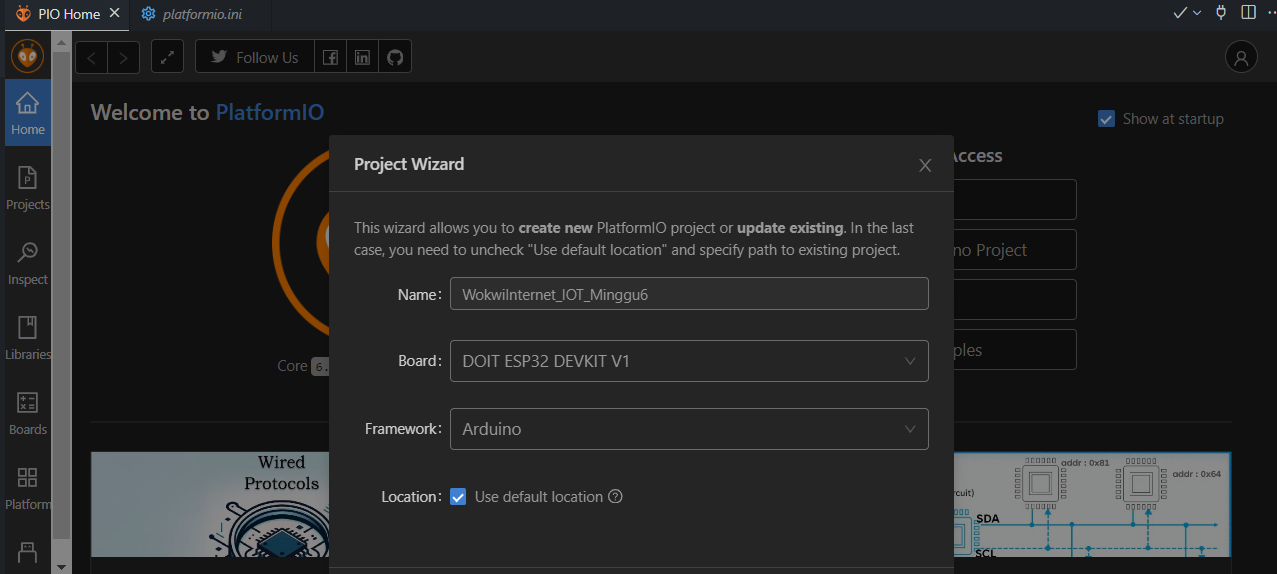
* Disini terlihat bahwa Data yang di POST dari posment sudah berhasil masuk ke Database

1. Sampai disini API yang dibangun menggunakan laravel sudah dapat berjalan dengan baik dan dapat diakses melalui URL publik.
2. Selanjutnya akan dilakukan akses API melalui simulator WOKWI ESP32
3. Jalankan API laravel dengan perintah : php artisan serve --host=0.0.0.0 --port=8080

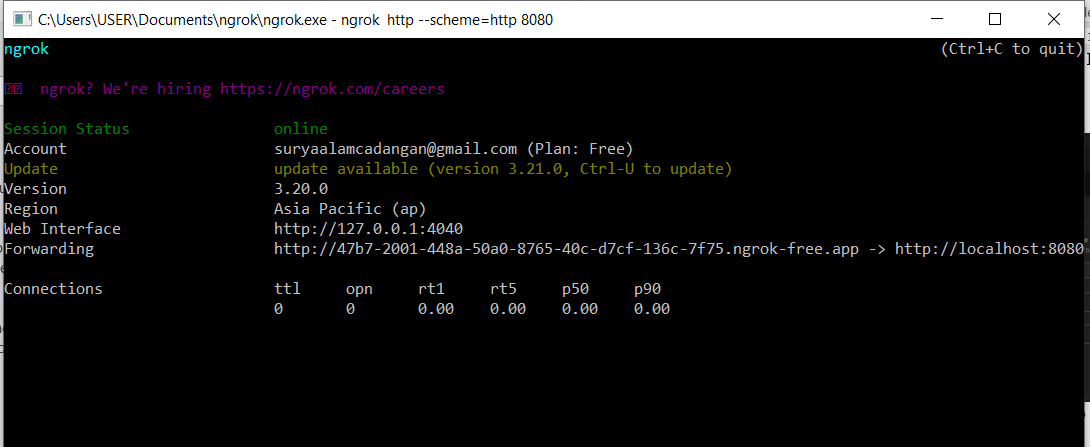


Perintah diatas memastikan API laravel dapat diakses dari IP Address manapun dan memastikan bekerja pada port 8080.

1. Buat file baru wokwi simulator di platform.io



1. Jalankan ngrok dengan perintah: ngrok http --scheme=http 8080



Sesuaikan alamat port 8080 sesuai dengan port berjalannya aplikasi Laravel Anda.

1. Tambahkan file wokwi.toml

[wokwi]

version = 1

firmware = '.pio\build\esp32doit-devkit-v1\firmware.bin'

elf = 'C:\Users\USER\Documents\PlatformIO\Projects\WokwiInternet\_IOT\_Minggu6\.pio\build\esp32doit-devkit-v1\firmware.elf'

1. Tambahkan file diagram.json

{

  "version": 1,

  "author": "KAVITH BUDVIN",

  "editor": "wokwi",

  "parts": [

    { "type": "board-esp32-devkit-c-v4", "id": "esp", "top": 86.4, "left": 24.04, "attrs": {} },

    { "type": "wokwi-dht22", "id": "dht1", "top": 19.5, "left": -91.8, "attrs": {} }

  ],

  "connections": [

    [ "esp:TX", "$serialMonitor:RX", "", [] ],

    [ "esp:RX", "$serialMonitor:TX", "", [] ],

    [ "dht1:VCC", "esp:3V3", "red", [ "v19.2", "h67.2", "v-67.2" ] ],

    [ "esp:GND.1", "dht1:GND", "black", [ "h0" ] ],

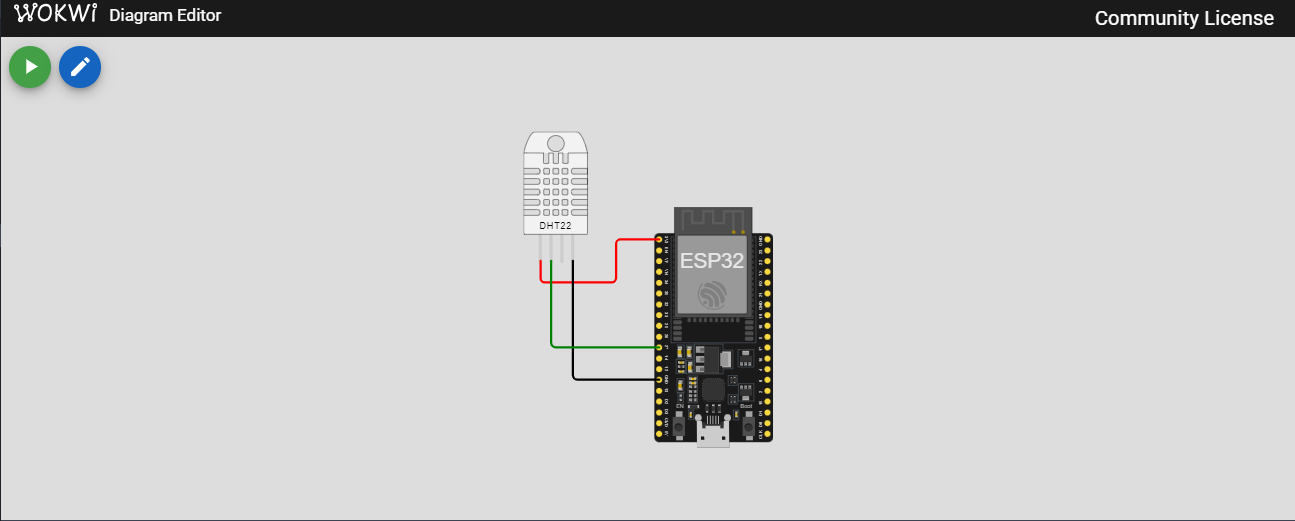
    [ "dht1:SDA", "esp:27", "green", [ "v0" ] ]

  ],

  "dependencies": {}

}

Hasilnya Akan Seperti ini:



1. Ubah file main.cpp

#include <Arduino.h>

#include <WiFi.h>

#include <HTTPClient.h>

#include "DHT.h"

#define DHTPIN 27

#define DHTTYPE DHT22

DHT dht(DHTPIN, DHTTYPE);

// Ganti dengan kredensial WiFi Anda

const char\* ssid = "Wokwi-GUEST";

const char\* password = "";

unsigned long previousMillis = 0;

const long interval = 5000;  // Interval 5 detik (5000 ms)

void setup() {

  Serial.begin(115200);

  // Hubungkan ke WiFi

  WiFi.begin(ssid, password);

  Serial.print("Menghubungkan ke WiFi");

  while (WiFi.status() != WL\_CONNECTED) {

    delay(500);

    Serial.print(".");

  }

  Serial.println(" Terhubung!");

  dht.begin();

  // Tunggu sebentar agar koneksi stabil

  delay(1000);

}

void loop() {

  unsigned long currentMillis = millis();

  // Lakukan POST setiap interval yang telah ditentukan

  if (currentMillis - previousMillis >= interval) {

    previousMillis = currentMillis;

    float h = round(dht.readHumidity());

    // Read temperature as Celsius (the default)

    float t = round(dht.readTemperature());

    // Check if any reads failed and exit early (to try again).

    if (isnan(h) || isnan(t)) {

      Serial.println(F("Failed to read from DHT sensor!"));

      return;

    }

    // Compute heat index in Celsius (isFahreheit = false)

    float hic = dht.computeHeatIndex(t, h, false);

    // Inisialisasi HTTPClient

    HTTPClient http;

    String url = "http://47b7-2001-448a-50a0-8765-40c-d7cf-136c-7f75.ngrok-free.app/api/posts"; // Ganti dengan URL ngrok yang benar

    http.begin(url);  // Menggunakan HTTP, bukan HTTPS

    http.addHeader("Content-Type", "application/json");

    String payload = "{\"nama\_sensor\":\"Sensor GD\", \"nilai1\":" + String(h) + ", \"nilai2\":" + String(t) + "}";

    Serial.println(payload);  // Untuk melihat apakah payload sudah terbentuk dengan benar

    // Kirim POST request

    int httpResponseCode = http.POST(payload);

    // Tampilkan kode respons HTTP

    Serial.print("Kode respons HTTP: ");

    Serial.println(httpResponseCode);

    // Tampilkan respons dari server jika request berhasil

    if (httpResponseCode == 200 || httpResponseCode == 201) {

      String response = http.getString();

      Serial.println("Respons dari server:");

      Serial.println(response);

    } else {

      Serial.println("Gagal mengirim data");

    }

    // Tutup koneksi HTTP

    http.end();

  }

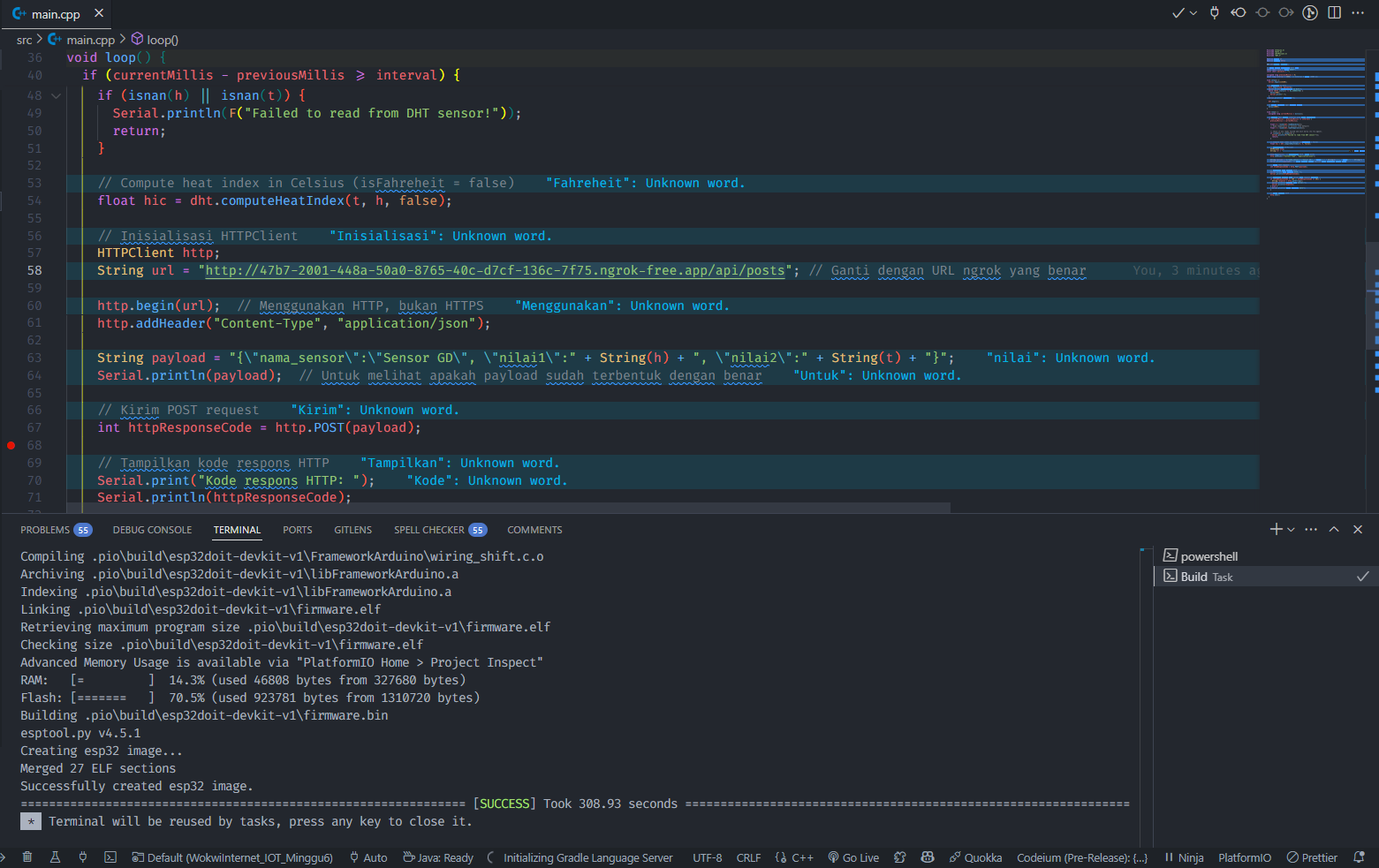
}

Pada bagian berikut sesuaikan dengan URL NGROK anda

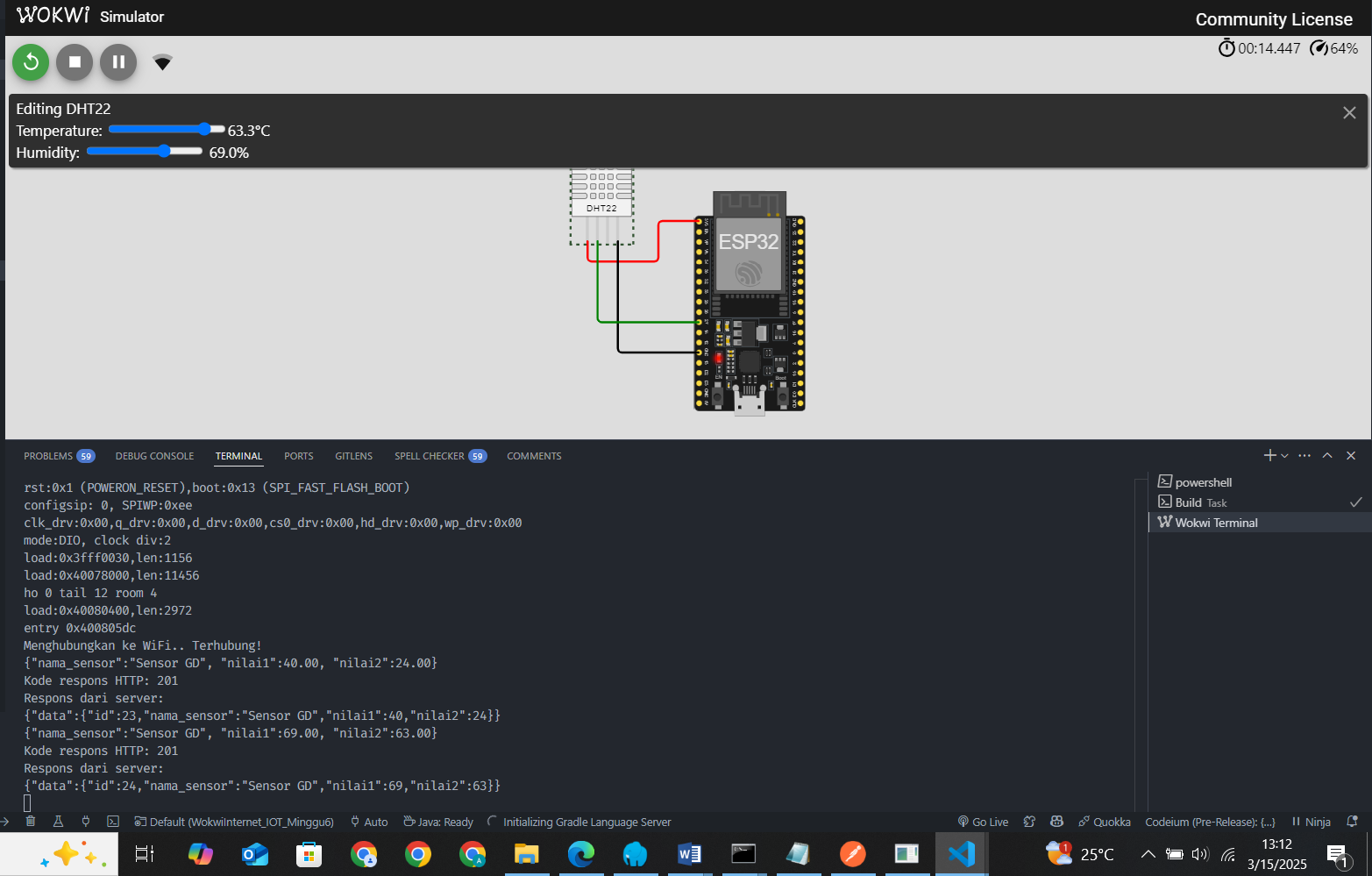
"http://47b7-2001-448a-50a0-8765-40c-d7cf-136c-7f75.ngrok-free.app/api/posts"; //

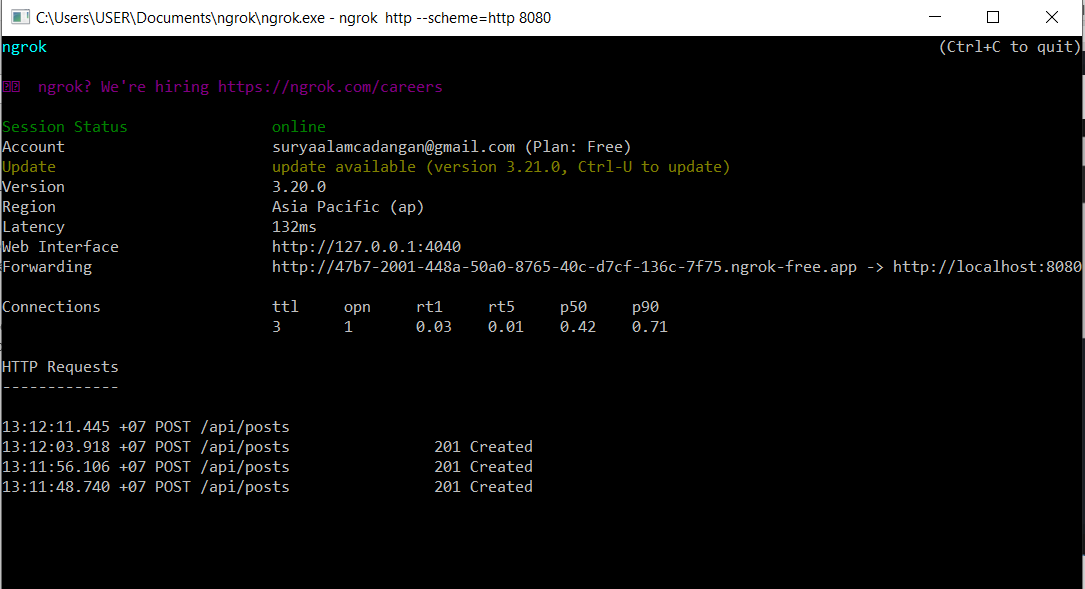
Ganti dengan URL ngrok yang benar

1. Build main.cpp



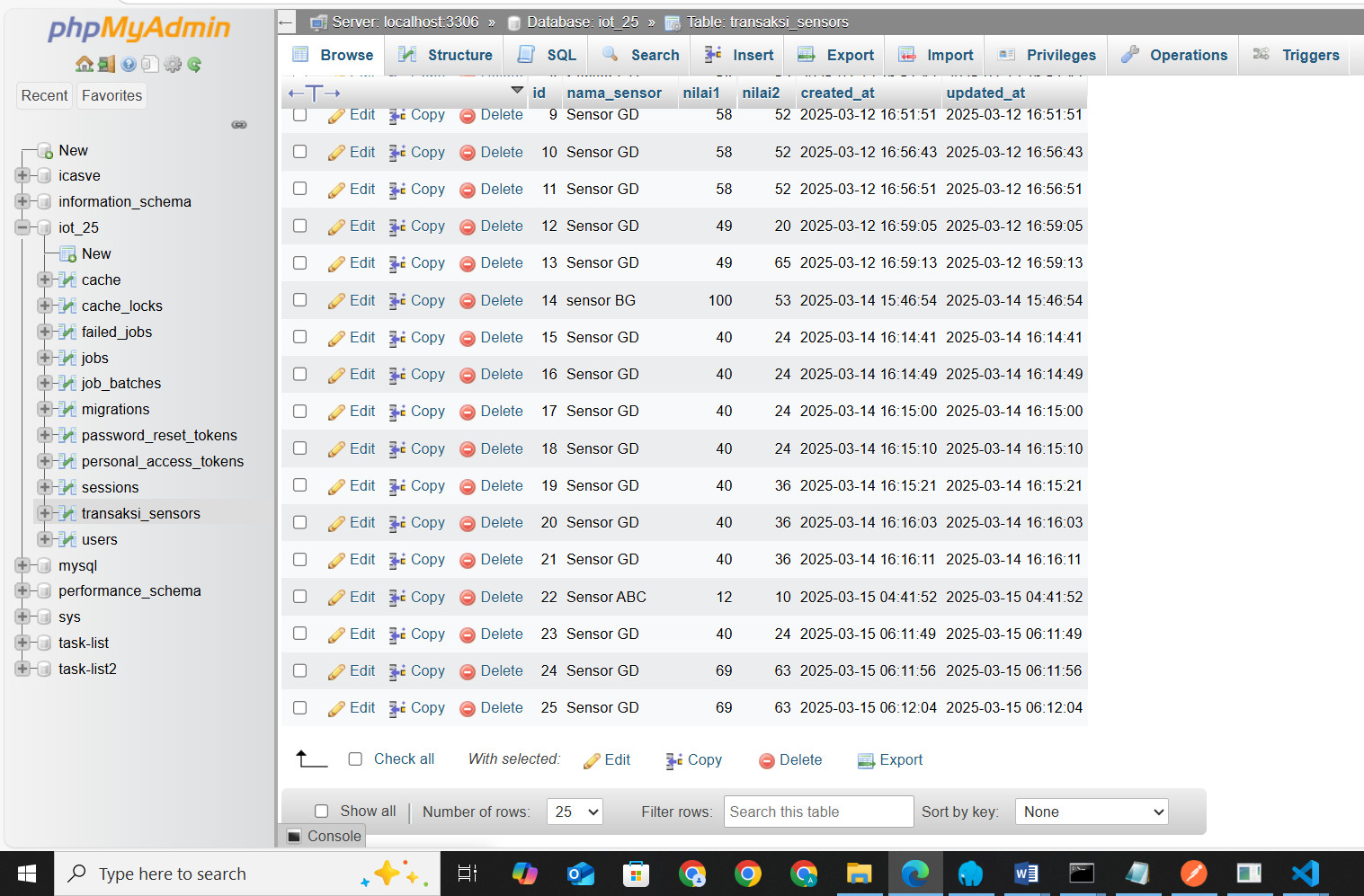
1. Start file diagram.json





Disini kita berhasil menjalankan dan mentapat respon 201 yang mengartikan data berhasil di buat

1. Pastikan di database, data telah muncul dan tersimpan



1. Buka Vs code lagi lalu Buka terminal dan jalankan code berikut:

* composer require maatwebsite/excel
* php artisan make:controller GraphController

1. Setelah itu tambahkan code berikut pada GraphController :

<?php

namespace App\Http\Controllers;

use App\Models\TransaksiSensors;

use App\Exports\TransaksiSensorExport;

use Maatwebsite\Excel\Facades\Excel;

class GraphController extends Controller

{

    /\*\*

     \* Menampilkan grafik transaksi sensor.

     \*

     \* @return \Illuminate\View\View

     \*/

    public function index()

    {

        // Mengambil data transaksi sensor

        $transaksiSensors = TransaksiSensors::latest()->take(10)->get();

        // Mengambil data label

        $labels = $transaksiSensors->pluck('nama\_sensor');

        // Mengambil data nilai1 dan nilai2 untuk grafik

        $dataNilai1 = $transaksiSensors->pluck('nilai1');

        $dataNilai2 = $transaksiSensors->pluck('nilai2');

        return view('graph', compact('labels', 'dataNilai1', 'dataNilai2'));

    }

    /\*\*

     \* Mengunduh data transaksi sensor dalam format Excel

     \*

     \* @return \Symfony\Component\HttpFoundation\BinaryFileResponse

     \*/

    public function exportToExcel()

    {

        return Excel::download(new TransaksiSensorExport, 'transaksi\_sensor.xlsx');

    }

}

1. Setelah itu, jalankan perintah ini pada terminal :

php artisan make:export TransaksiSensorExport --model=TransaksiSensor

1. Tambahkan code berikut pada file TransaksiSensorExport :

<?php

namespace App\Exports;

use App\Models\TransaksiSensors;

use Maatwebsite\Excel\Concerns\FromCollection;

class TransaksiSensorExport implements FromCollection

{

    /\*\*

    \* @return \Illuminate\Support\Collection

    \*/

    public function collection()

    {

        return TransaksiSensors::all();

    }

}

1. Setelah itu, edit file web.php yang berada di folder routes menjadi seperti berikut:

<?php

use Illuminate\Support\Facades\Route;

use App\Http\Controllers\GraphController;

Route::get('/', [GraphController::class, 'index'])->name('graph');

Route::get('/graph/export',     [GraphController::class, 'exportToExcel'])->name('graph.export');

1. Setelah itu, buat file graph.blade.php pada folder resouces/views dan isi
2. Setelah semua sudah, jalankan program tersebut dengan perintah berikut:

* php artisan serve

**3. Hasil dan Pembahasan**

1. **Hasil Eksperimen**

Berikut adalah hasil visual dari percobaan:

