# LAPORAN PRAKTIKUM INTERNET OF THINGS (IoT)

Fakultas Vokasi, Universitas Brawijaya

**API Creation Practice Using Laravel 11 and Ngrok**

**& API Access Practice Through WOKWI Simulation (Continued)**



*Fawwaz Mufid Wardaya*

*Fakultas Vokasi, Universitas Brawijaya*

*Email : mahesfawwaz79@gmail.com*

**Abstract**

Tujuan dari eksperimen ini adalah untuk mempelajari cara membuat API menggunakan framework Laravel 11 dan mengintegrasikannya dengan simulasi IoT di platform WOKWI. API yang dibuat akan menerima data dari mikrokontroler ESP32 yang dikonfigurasi dalam simulasi WOKWI, dan data ini akan diproses dan ditampilkan secara real-time oleh API Laravel. Selain itu, Ngrok digunakan untuk membuat API Laravel publik. Tujuan dari praktik ini adalah untuk mendapatkan pemahaman dasar tentang pengembangan API IoT dan integrasi perangkat keras (ESP32) dan perangkat lunak (Laravel).

**1. Introduction**

**1.1 Latar belakang**

Salah satu komponen penting dari teknologi Internet of Things (IoT) adalah API (Application Programming Interface), yang berfungsi sebagai penghubung antara aplikasi perangkat lunak (seperti server backend) dan perangkat keras (seperti ESP32). IoT memungkinkan perangkat fisik berkomunikasi dengan sistem digital melalui jaringan internet.

Karena fiturnya yang kaya dan kemudahan penggunaannya, Framework Laravel adalah alat pengembangan API yang populer. Kami akan menggunakan Laravel 11 untuk membangun API dasar untuk menerima data dari platform WOKWI yang disimulasikan ESP32. Data ini akan diproses dan disimpan dalam database. Untuk memastikan API dapat diakses secara publik, kami juga akan menggunakan Ngrok untuk membuat tunnel ke server lokal Laravel.

**1.2 Tujuan eksperimen**

Tujuan dari tes ini adalah:

1. Mempelajari cara membuat API menggunakan Laravel 11.
2. Mengintegrasikan API Laravel dengan simulasi ESP32 di platform WOKWI.
3. Menggunakan Ngrok untuk membuat API Laravel dapat diakses secara publik.
4. Memahami alur komunikasi antara perangkat IoT (ESP32) dan server backend (Laravel).

**2. Methodology (Metodologi)**

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

Software :

* Laravel 11
* Ngrok
* WOKWI (Platform Simulasi IoT)
* Visual Studio Code (VSCode)
* XAMPP (untuk menjalankan server lokal Laravel)

Hardware :

* ESP32 (disimulasikan di WOKWI)

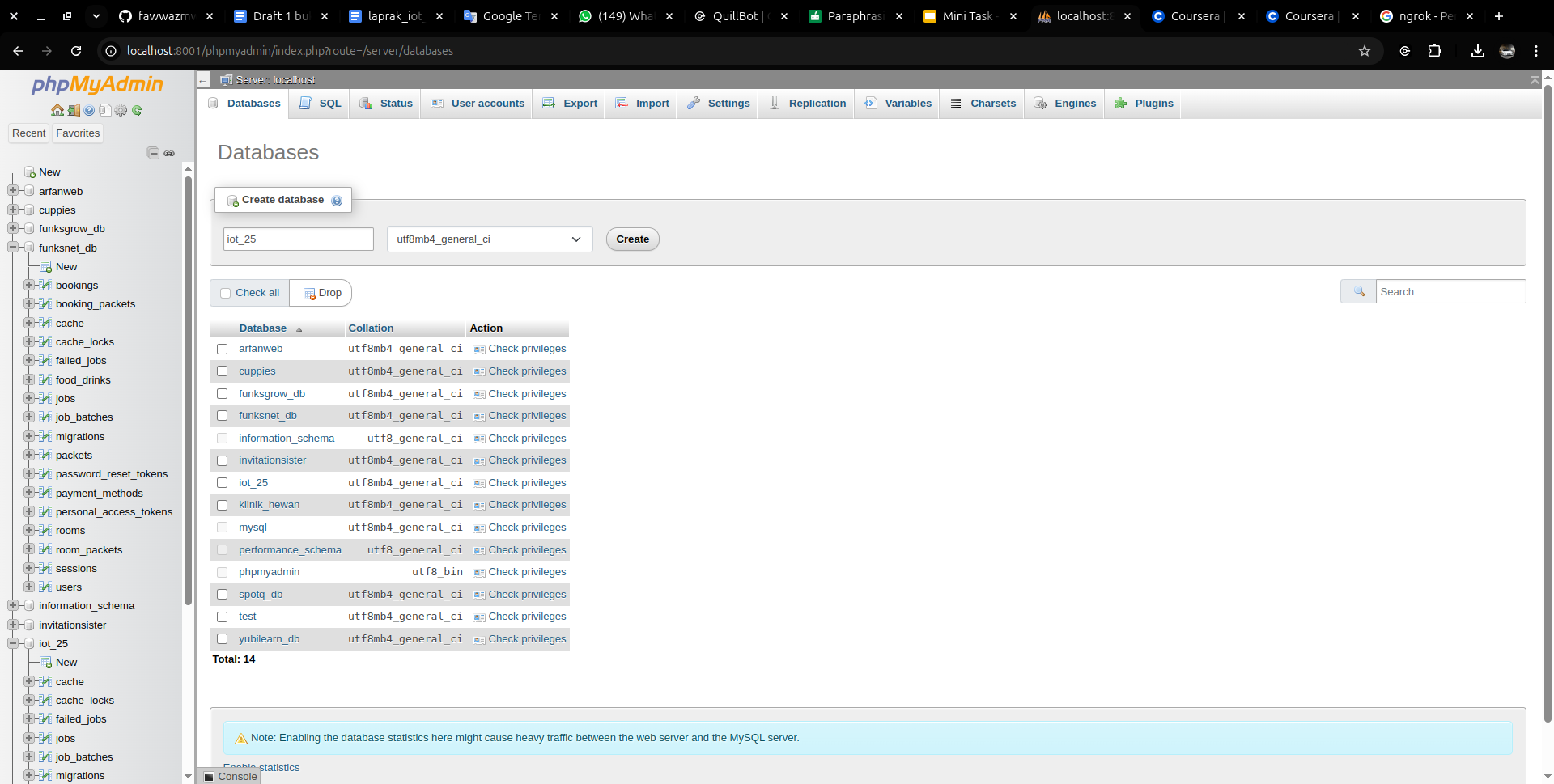
**2.2 Implementation Steps (Langkah Implementasi)**

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 laravel-11**

**cd laravel-11**

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=**

**DB\_CHARSET=utf8mb4**

**DB\_COLLATION=utf8mb4\_unicode\_ci**

Perhatikan bagian DB\_USERNAME dan DB\_PASSWORD sesuaikan dengan setting yang ada di laptop/komputer.

Buat file model TransaksiSensor.php dengan cara menjalankan perintah berikut di terminal :

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

Kemudian ubah file **2025\_02\_21\_074123\_create\_transaksi\_sensors\_table.php**

**Yang ada di dalam folder databases-migrations**

**<?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()->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\_sensors';**

**/\*\***

**\* 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**

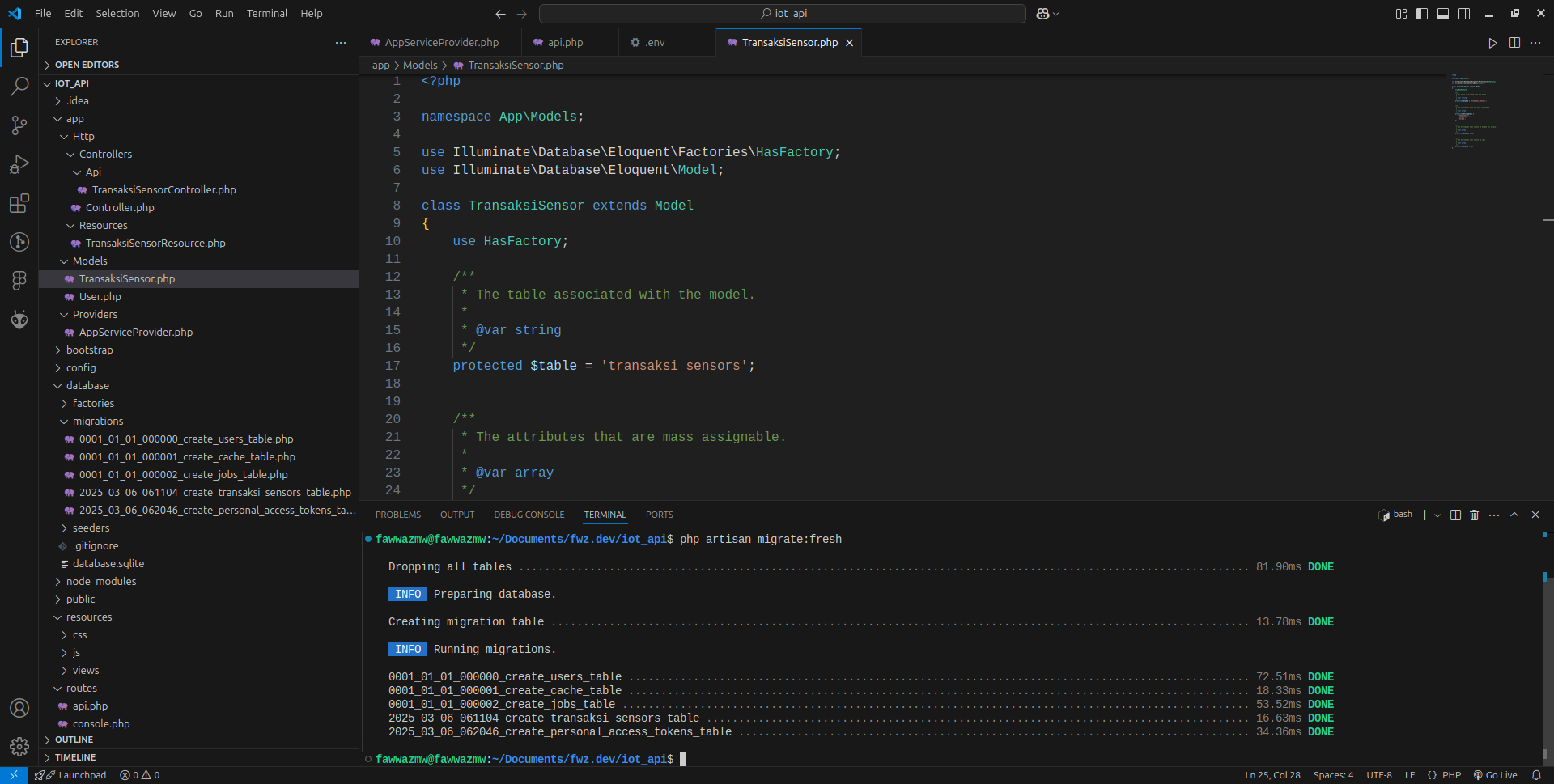
**\*/**

**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 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,**

**];**

**}**

**}**

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 App\Http\Controllers\Controller;**

**use App\Http\Resources\TransaksiSensorResource;**

**use App\Models\TransaksiSensor;**

**use Illuminate\Http\Request;**

**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 :

**<?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'));**

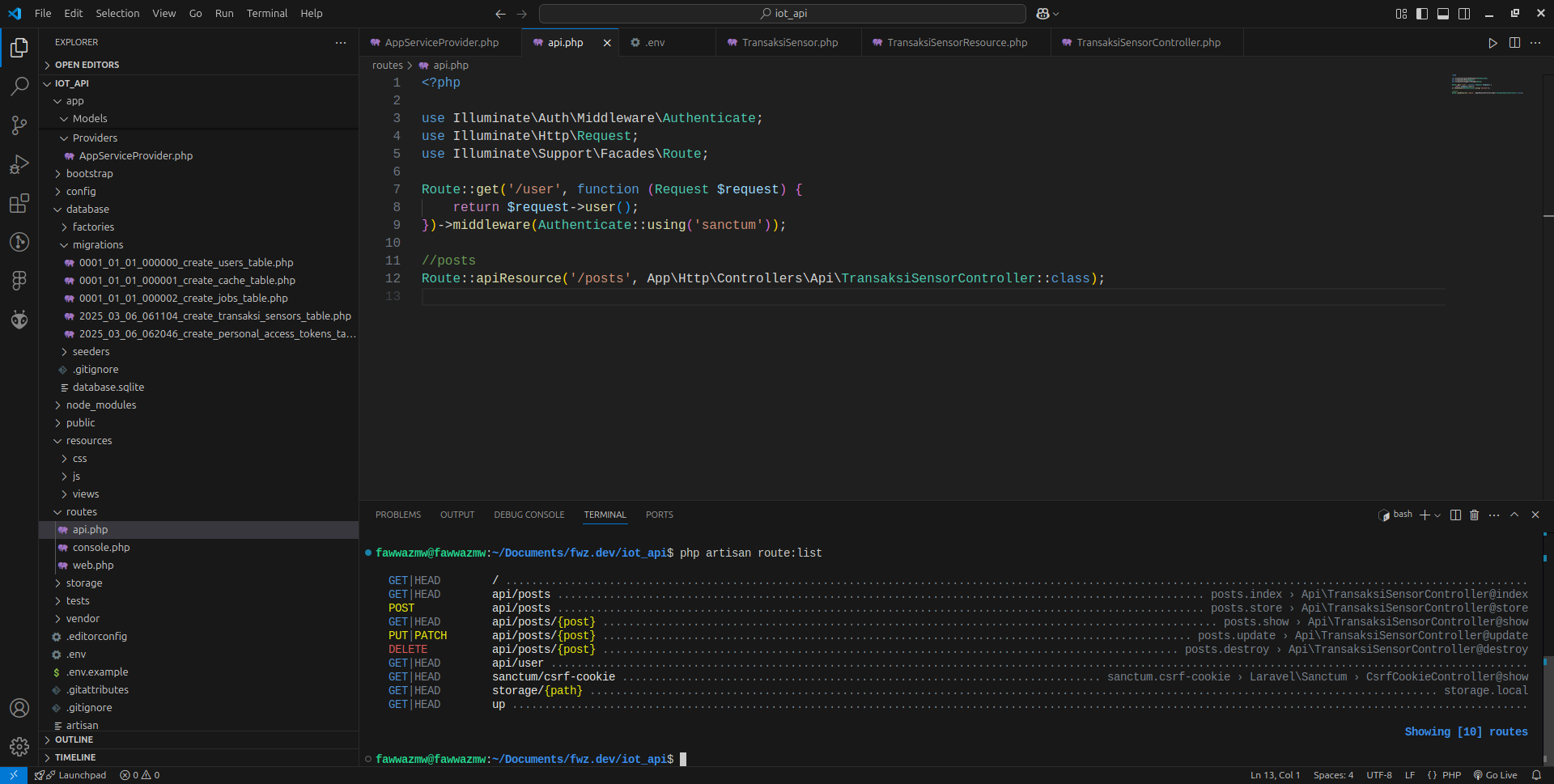
**//posts**

**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 :

****

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.

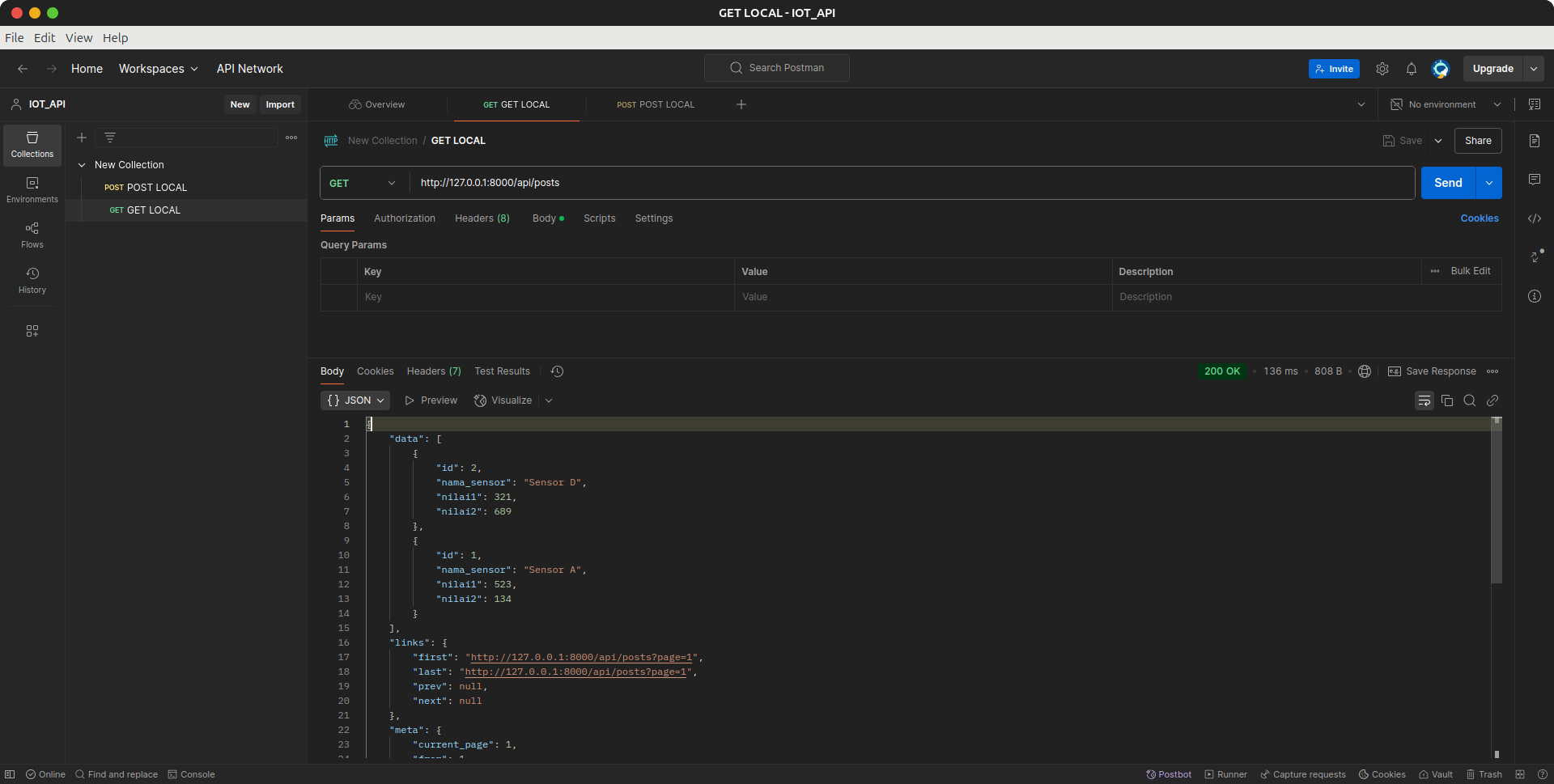
Untuk melakukan percobaan akses api, pastikan aplikasi laravel dijalankan dengan perintah :

**php artisan serve**

****

Pastikan telah data yang dimasukkan kedalam tabel di database. Pada contoh berikut, telah ada 2 baris data pada tabel transaksi\_sensor pada database iot\_25

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": 2,**

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

**"nilai1": 321,**

**"nilai2": 689**

**},**

**{**

**"id": 1,**

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

**"nilai1": 523,**

**"nilai2": 134**

**}**

**],**

**"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**

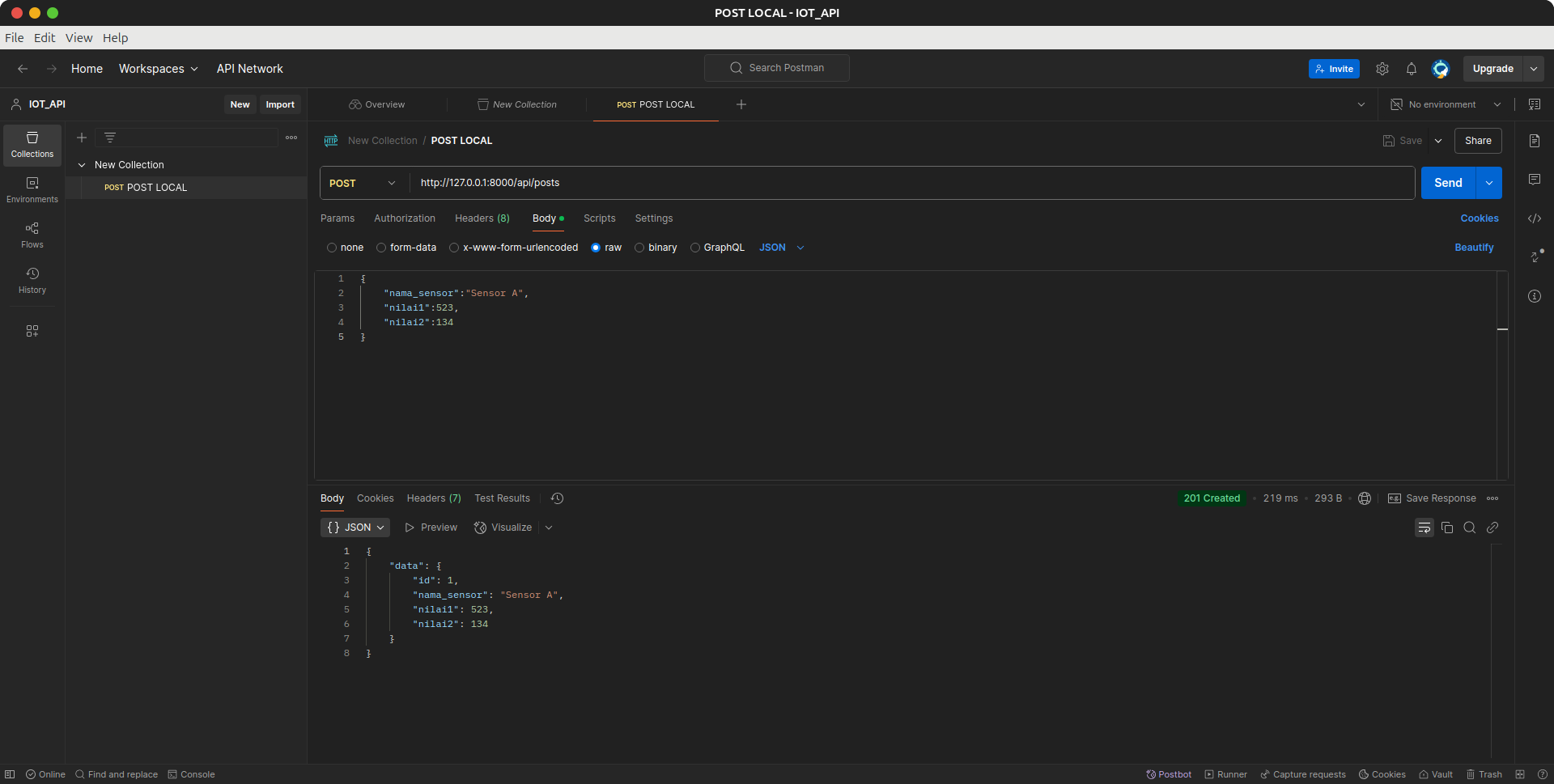
**}**

**}**

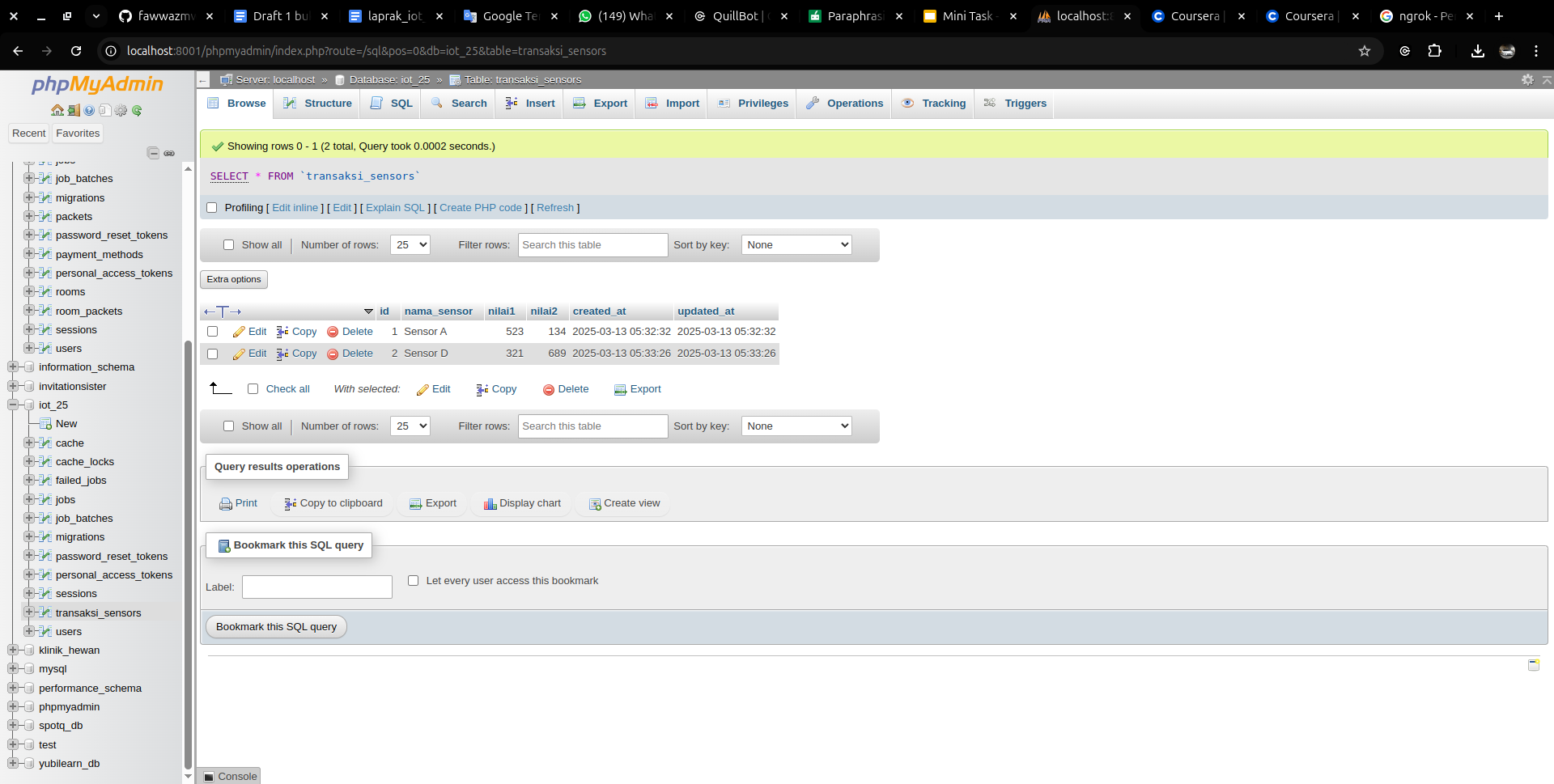
API telah berfungsi untuk mengambil data dari database. Langkah berikutnya adalah melakukan percobaan insert data ke tabel di database menggunakan API. Caranya adalah mengganti method menjadi POST kemudian pada bagian header ubah menjadi sebagai berikut

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.

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

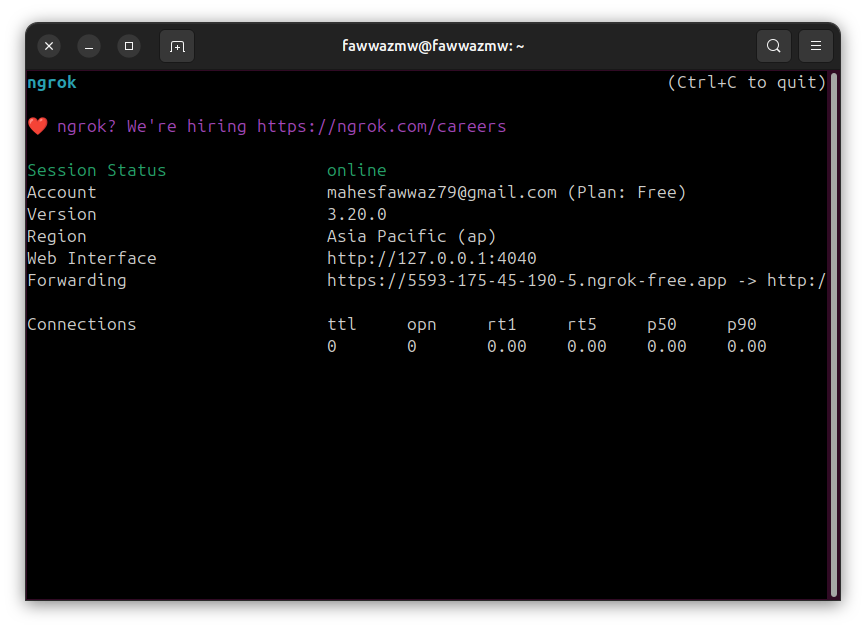
Lakukan ekstraksi

Buka command prompt dari alamat folder ekstraksi seperti berikut :

**Kemudian jalankan perintah sesuai yang ada di akun ngrok :**

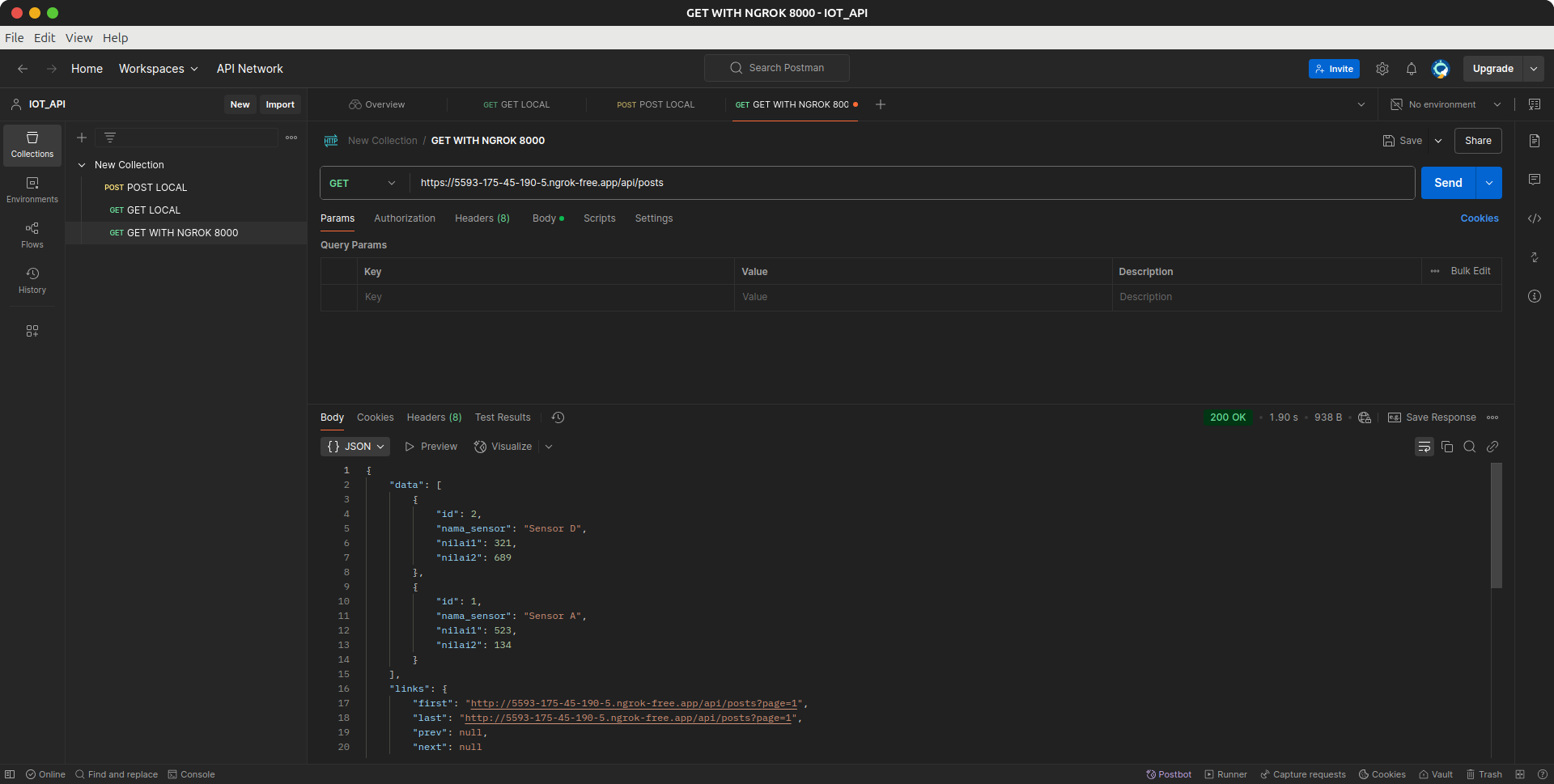
**Kemudian jalankan perintah berikut untuk mengonline kan laravel melalui port 8000**

**ngrok http http://localhost:8000**

****

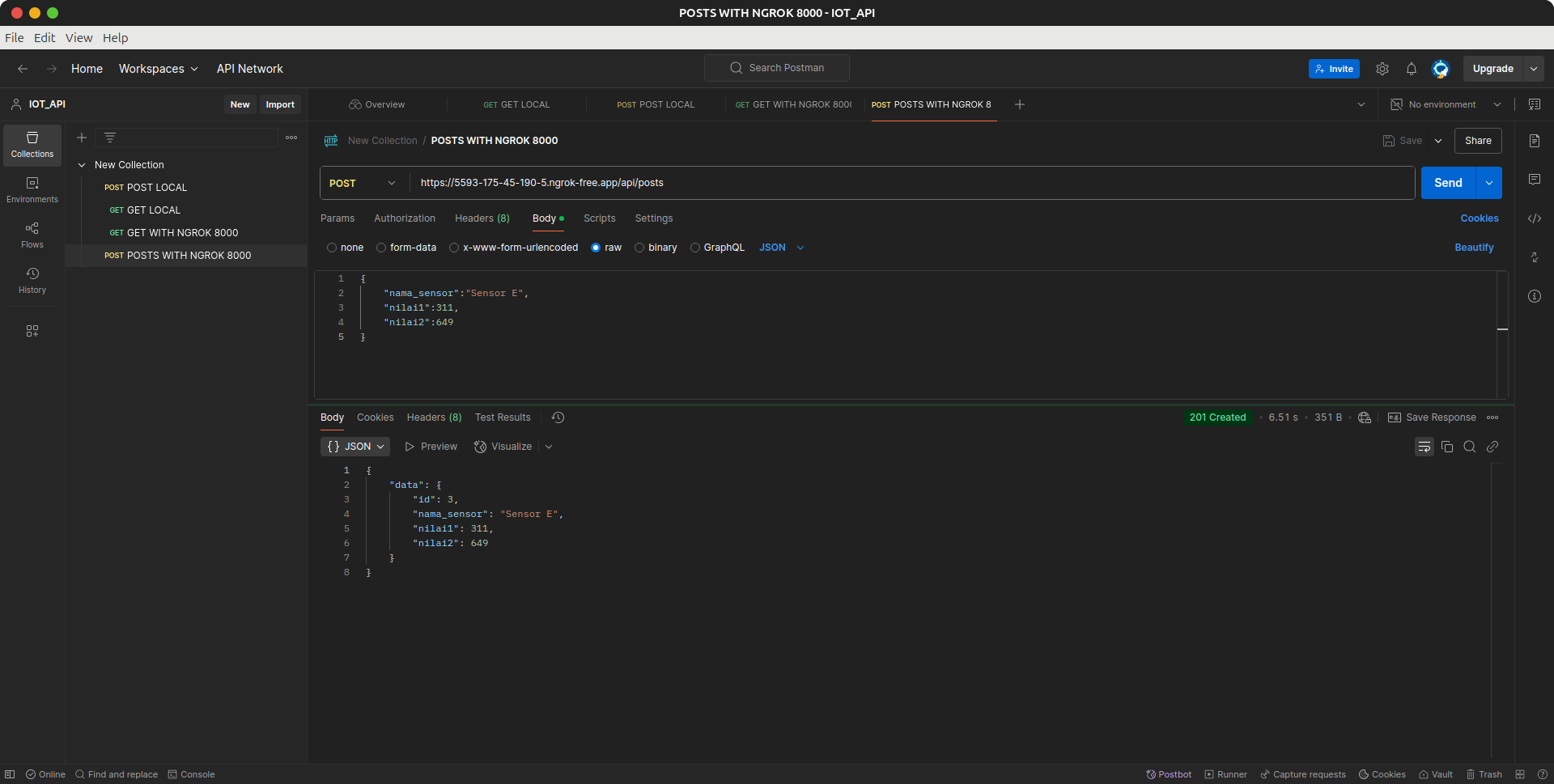
**Kemudian lakukan percobaan menggunakan postman menggunakan URL yang diberikan oleh ngrok. Pada contoh ini, ngrok memberikan URL publik yang dapat diakses melalui internet pada alamat , sesuaikan dengan milik Anda .**

**https://e521-2405-8740-6315-3520-2099-2415-5077-c12f.ngrok-free.app**

****

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

**https://e521-2405-8740-6315-3520-2099-2415-5077-c12f.ngrok-free.app/api/posts**

****

**Berikutnya lakukan percobaan melakukan insert data baru melalui API**

**https://e521-2405-8740-6315-3520-2099-2415-5077-c12f.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.**

**Praktik Akses API Melalui**

**Simulasi WOKWI**

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.

Buat file baru wokwi simulator di platform.io

Berikut adalah script 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(9600);**

**// 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://d87c-175-45-190-4.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();**

**}**

**}**

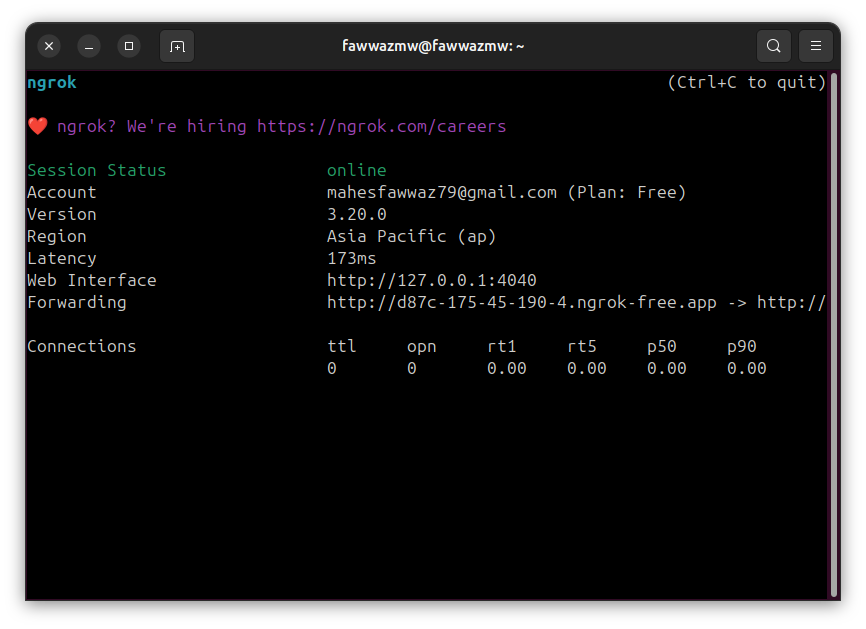
**Perhatikan pada bagian**

**// URL lengkap server yang akan diakses**

**const char\* serverUrl = "http://e6d3-2405-8740-6315-3520-5928-26b-7835-cd79.ngrok-free.app/api/posts";**

**URL diatas adalah URL hasil dari generate perintah NGROK di komputer Anda. Sesuaikan dengan alamat URL yang diberikan oleh NGROK. Cara menjalankan perintah NGROK berbeda dengan Bab sebelumnya, perintah berikut memastikan NGROK memberikan alamat URL dalam bentuk http bukan https. Pada saat ini ESP32 yang digunakan hanya support http sehingga pastikan NGROK memberikan URL dalam bentuk http bukan https.**

**ngrok http --scheme=http 8080**

****

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

**Tambahkan file wokwi.toml**

**[wokwi]**

**version = 1**

**firmware = '.pio/build/esp32doit-devkit-v1/firmware.bin'**

**elf = '.pio/build/esp32doit-devkit-v1/firmware.elf'**

Tambahkan file **diagram.json**

**{**

**"version": 1,**

**"author": "Uri Shaked",**

**"editor": "wokwi",**

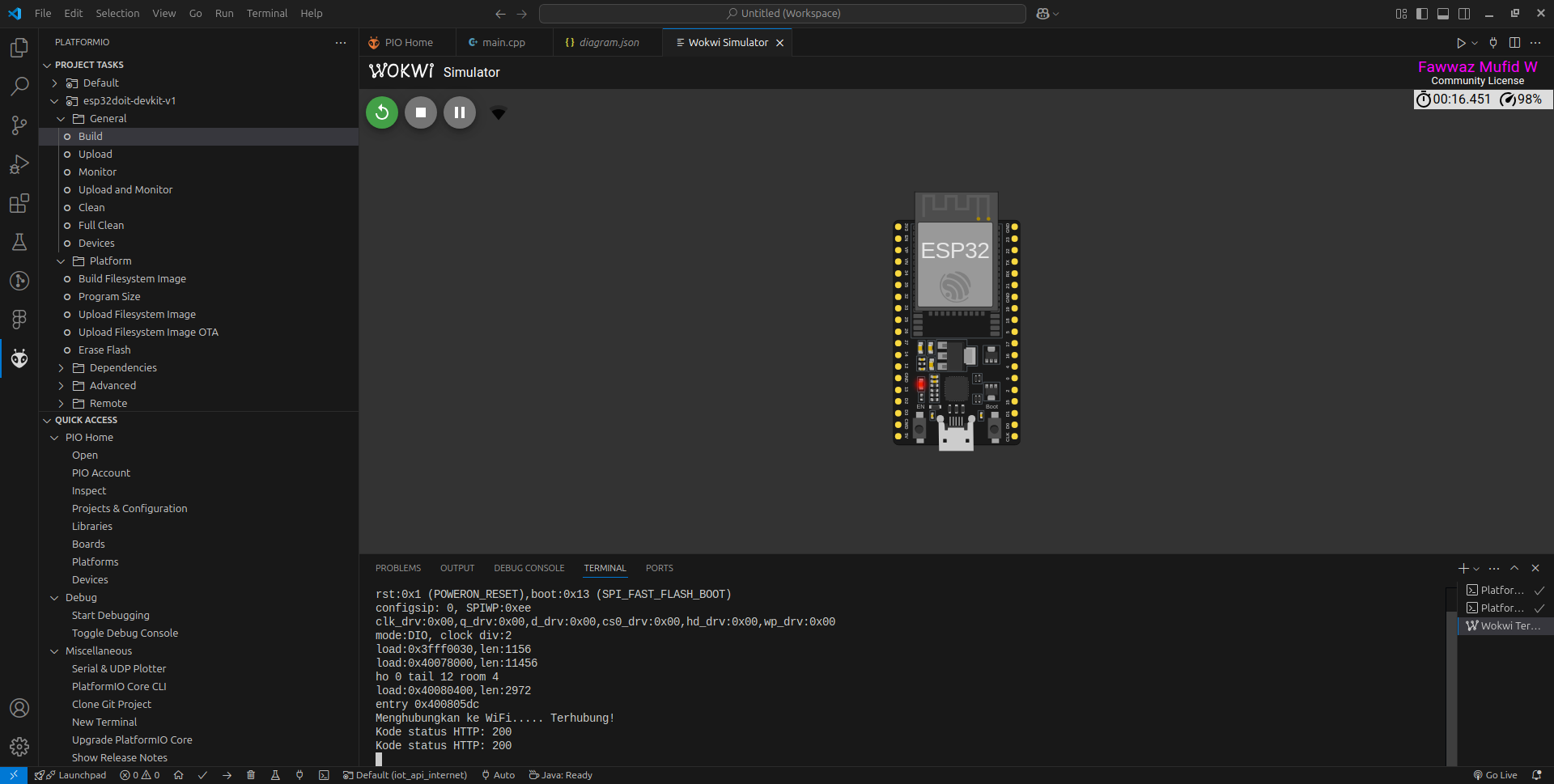
**"parts": [ { "type": "board-esp32-devkit-c-v4", "id": "esp", "top": 0, "left": 0, "attrs": {} } ],**

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

**}**

**Langkah berikutnya adalah melakukan simulasi. Build file main.cpp dan jalankan simulasi dengan perintah**

**> Wokwi Start Simulator**

****

Simulasi diatas menunjukkan, ESP32 berhasil terhubung ke WIFI Wokwi-GUEST dan berhasil mengakses API laravel yang sudah dibuat pada bab sebelumnya.

**Kode Status HTTP:200**

HTTP status code 200 artinya adalah "OK". Ini berarti bahwa permintaan (request) yang dikirim oleh klien (misalnya browser web atau aplikasi IoT) telah berhasil diproses oleh server. Dengan kata lain, halaman web atau data yang diminta telah berhasil dikirim kembali oleh server dan ditampilkan dengan benar kepada pengguna.

Berikutnya adalah melakukan modifikasi simulasi dengan menambahkan sensor suhu dan kelembaban. Skenarionya adalah, wokwi simulator akan mengirimkan data suhu dan kelembaban ke API dan menyimpannya ke database mysql seperti yang telah dibuat pada bab sebelumnya.

Rangkai sensor DHT22 dengan ESP32 seperti contoh diatas. Kemudian salin kode diagram.json ke file **diagram.json** yang ada di vscode.

**{**

**"version": 1,**

**"author": "Fawwaz Mufid W",**

**"editor": "wokwi",**

**"parts": [**

**{ "type": "board-esp32-devkit-c-v4", "id": "esp", "top": 0, "left": 0, "attrs": {} },**

**{ "type": "wokwi-dht22", "id": "dht1", "top": -95.7, "left": -130.2, "attrs": {} }**

**],**

**"connections": [**

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

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

**[ "esp:3V3", "dht1:VCC", "green", [ "h0" ] ],**

**[ "esp:GND.1", "dht1:GND", "black", [ "h-14.21", "v-115.2", "h-76.8" ] ],**

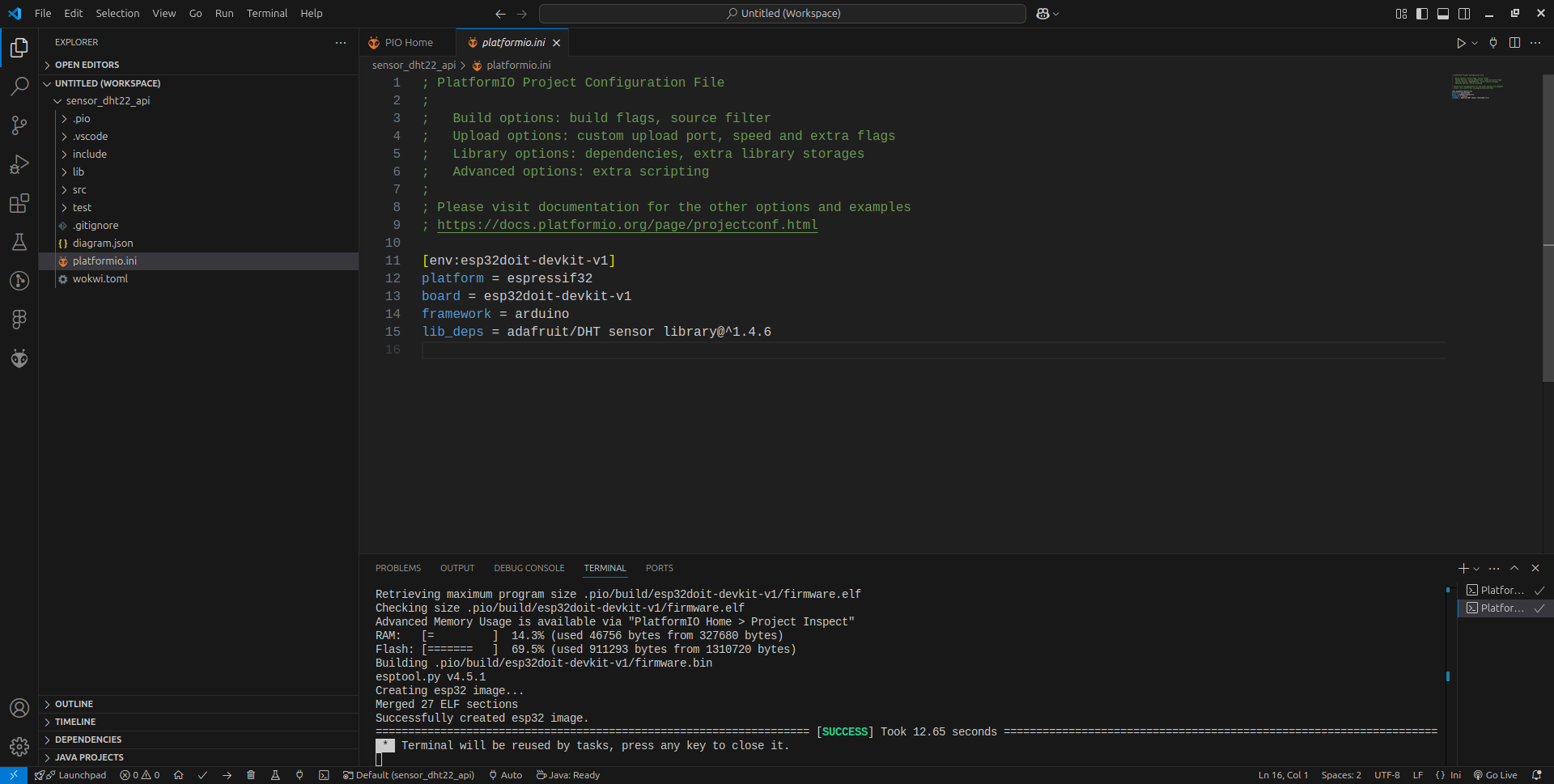
**[ "dht1:SDA", "esp:27", "green", [ "v28.8", "h86.5", "v76.8" ] ]**

**],**

**"dependencies": {}**

**}**

Kemudian ubah setting file **platformio.ini** sebagai berikut :

****

Pada perubahan diatas, ada tambahan 2 setting yaitu monitor speed dan lib\_deps

Monitor speed digunakan untuk memonitor status pengiriman data dari wokwi simulator ke server api laravel yang telah dibuat. Sedangkan lib\_deps adalah library yang digunakan sensor DHT (sensor suhu dan kelembaban).

Modifikasi 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(9600);**

**// 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://d87c-175-45-190-4.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();**

**}**

**}**

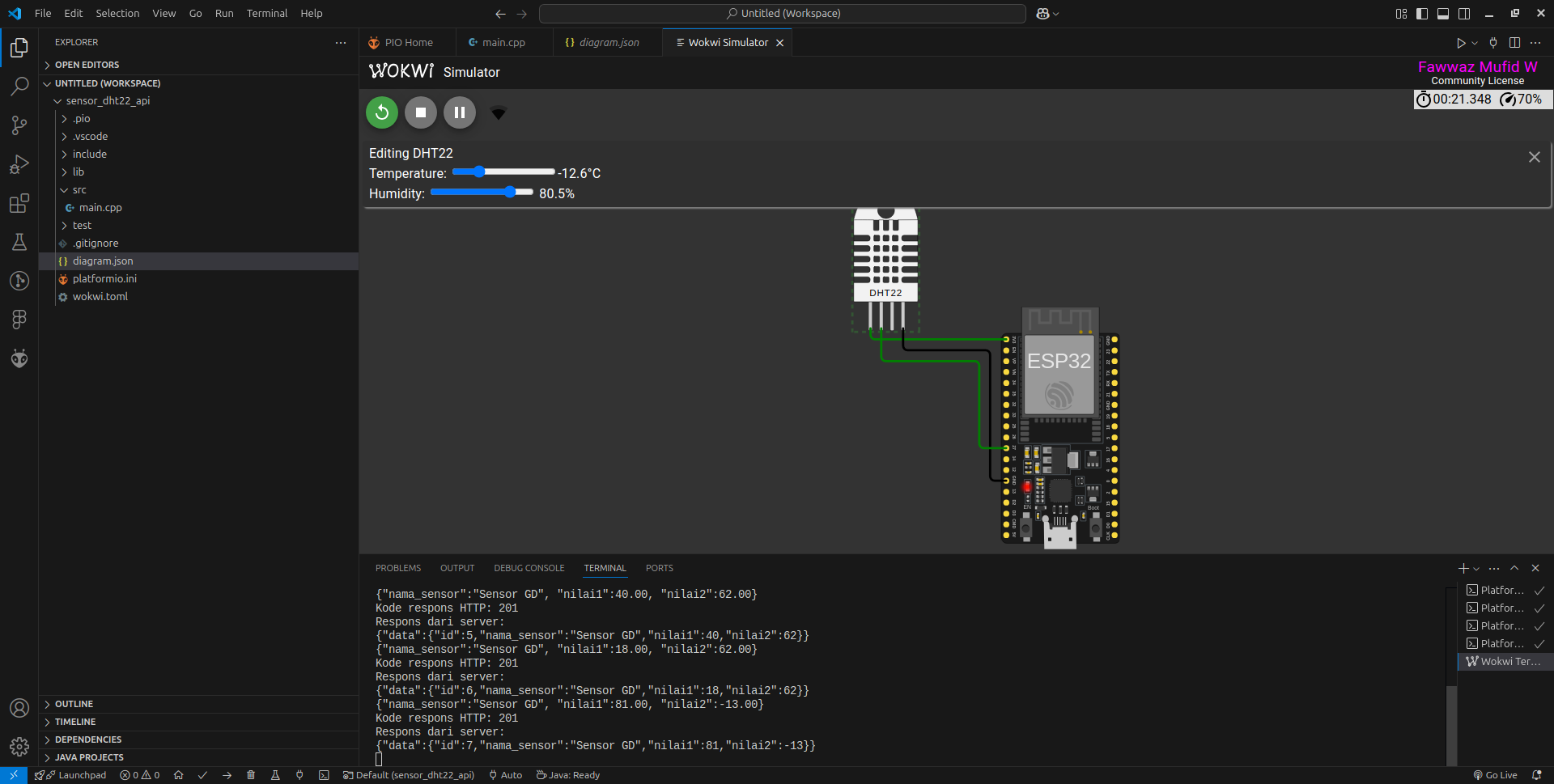
**Pada bagian berikut sesuaikan dengan URL NGROK anda**

**"http://e6d3-2405-8740-6315-3520-5928-26b-7835-cd79.ngrok-free.app/api/posts"; // Ganti dengan URL ngrok yang benar**

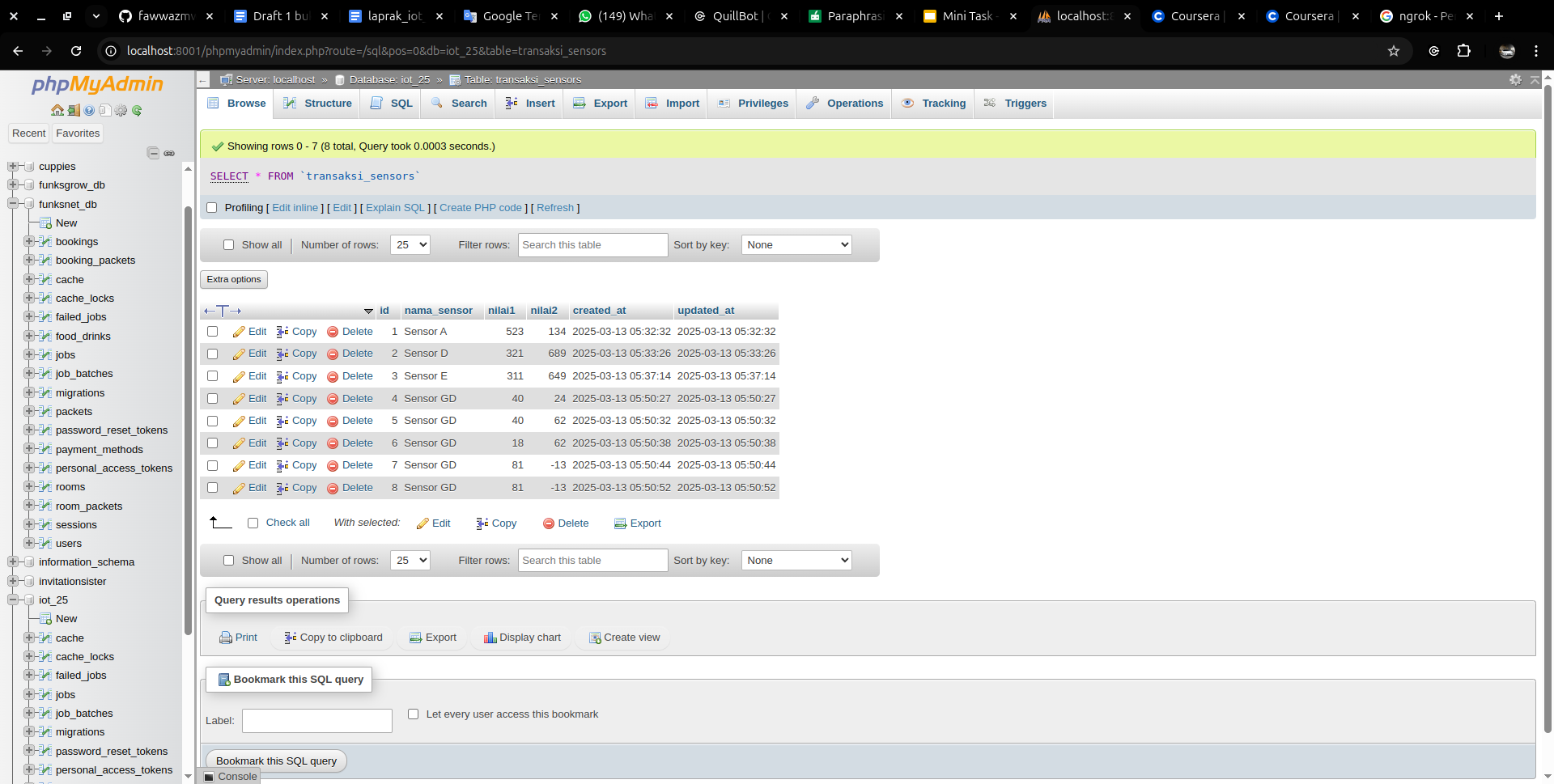
**Jalankan simulasi**

**> Wokwi Start Simulator**

Berikut adalah tampilan pada simulator dan serial monitor.



Pastikan di database, data telah muncul dan tersimpan



**(Continued)  
  
composer require maatwebsite/excel**

**php artisan make:controller GraphController**

Setelah itu tambahkan code berikut pada GraphController :

**<?php**

**namespace App\Http\Controllers;**

**use App\Exports\TransaksiSensorExport;**

**use Maatwebsite\Excel\Facades\Excel;**

**use App\Models\TransaksiSensor;**

**class GraphController extends Controller**

**{**

**/\*\***

**\* Menampilkan grafik transaksi sensor.**

**\***

**\* @return \Illuminate\View\View**

**\*/**

**public function index()**

**{**

**// Mengambil data transaksi sensor**

**$transaksiSensors = TransaksiSensor::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');**

**}**

**}**

Setelah itu, jalankan perintah ini pada terminal :

**php artisan make:export TransaksiSensorExport --model=TransaksiSensor**

Tambahkan code berikut pada file TransaksiSensorExport :

<?php

namespace App\Exports;

use App\Models\TransaksiSensor;

use Maatwebsite\Excel\Concerns\FromCollection;

class TransaksiSensorExport implements FromCollection

{

/\*\*

\* @return \Illuminate\Support\Collection

\*/

public function collection()

{

return TransaksiSensor::all();

}

}

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');

Setelah itu, buat file graph.blade.php pada folder resouces/views dan tambahkan code berikut:

<!DOCTYPE html>

<html lang="id">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Dashboard Monitoring Sensor | Sistem IoT</title>

<link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/6.4.0/css/all.min.css">

<link href="https://fonts.googleapis.com/css2?family=Poppins:wght@300;400;500;600;700&display=swap" rel="stylesheet">

<script src="https://cdn.jsdelivr.net/npm/chart.js"></script>

<link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/animate.css/4.1.1/animate.min.css">

<style>

:root {

--primary-color: #4361ee;

--primary-light: #e0e7ff;

--secondary-color: #3f37c9;

--accent-color: #4cc9f0;

--accent-light: #e0fbfc;

--success-color: #4bb543;

--warning-color: #f8961e;

--danger-color: #f94144;

--light-color: #f8f9fa;

--dark-color: #212529;

--gray-color: #6c757d;

}

\* {

margin: 0;

padding: 0;

box-sizing: border-box;

}

body {

font-family: 'Poppins', sans-serif;

background: linear-gradient(135deg, #f5f7fa 0%, #e2e8f0 100%);

min-height: 100vh;

padding: 2rem 1rem;

color: var(--dark-color);

line-height: 1.6;

}

.dashboard-container {

max-width: 1200px;

margin: 0 auto;

}

.header {

display: flex;

justify-content: space-between;

align-items: center;

margin-bottom: 2rem;

flex-wrap: wrap;

gap: 1rem;

}

.header-title {

font-size: 1.8rem;

font-weight: 600;

color: var(--primary-color);

display: flex;

align-items: center;

gap: 0.75rem;

}

.header-title i {

color: var(--accent-color);

}

.card {

background-color: white;

border-radius: 12px;

box-shadow: 0 4px 20px rgba(0, 0, 0, 0.08);

padding: 1.75rem;

margin-bottom: 2rem;

transition: transform 0.3s ease, box-shadow 0.3s ease;

}

.card:hover {

transform: translateY(-5px);

box-shadow: 0 8px 30px rgba(0, 0, 0, 0.12);

}

.card-header {

display: flex;

justify-content: space-between;

align-items: center;

margin-bottom: 1.5rem;

padding-bottom: 1rem;

border-bottom: 1px solid rgba(0, 0, 0, 0.05);

}

.card-title {

font-size: 1.25rem;

font-weight: 600;

color: var(--primary-color);

display: flex;

align-items: center;

gap: 0.75rem;

}

.card-title i {

font-size: 1.1em;

}

.card-actions {

display: flex;

gap: 0.75rem;

}

.btn {

padding: 0.5rem 1rem;

border-radius: 8px;

border: none;

font-weight: 500;

font-size: 0.9rem;

cursor: pointer;

transition: all 0.3s ease;

display: inline-flex;

align-items: center;

gap: 0.5rem;

}

.btn-primary {

background-color: var(--primary-color);

color: white;

}

.btn-primary:hover {

background-color: var(--secondary-color);

}

.btn-outline {

background-color: transparent;

border: 1px solid var(--primary-color);

color: var(--primary-color);

}

.btn-outline:hover {

background-color: var(--primary-color);

color: white;

}

.btn-success {

background-color: var(--success-color);

color: white;

}

.btn-success:hover {

opacity: 0.9;

}

.chart-container {

position: relative;

height: 400px;

width: 100%;

margin-bottom: 1.5rem;

}

.data-summary {

display: grid;

grid-template-columns: repeat(auto-fit, minmax(250px, 1fr));

gap: 1.25rem;

margin-top: 1.5rem;

}

.summary-card {

background-color: white;

border-radius: 10px;

padding: 1.25rem;

box-shadow: 0 2px 10px rgba(0, 0, 0, 0.05);

transition: transform 0.2s ease;

}

.summary-card:hover {

transform: translateY(-3px);

}

.summary-header {

display: flex;

justify-content: space-between;

align-items: center;

margin-bottom: 0.75rem;

}

.summary-title {

font-size: 0.9rem;

font-weight: 500;

color: var(--gray-color);

}

.summary-icon {

width: 36px;

height: 36px;

border-radius: 8px;

display: flex;

align-items: center;

justify-content: center;

font-size: 1rem;

}

.sensor-1 {

background-color: var(--primary-light);

color: var(--primary-color);

}

.sensor-2 {

background-color: var(--accent-light);

color: var(--accent-color);

}

.summary-value {

font-size: 1.5rem;

font-weight: 600;

margin-bottom: 0.25rem;

}

.summary-change {

font-size: 0.85rem;

display: flex;

align-items: center;

gap: 0.25rem;

}

.positive {

color: var(--success-color);

}

.negative {

color: var(--danger-color);

}

.neutral {

color: var(--gray-color);

}

.time-selector {

display: flex;

justify-content: flex-end;

gap: 0.5rem;

margin-bottom: 1rem;

}

.time-btn {

padding: 0.35rem 0.75rem;

border-radius: 6px;

background-color: var(--light-color);

border: none;

font-size: 0.85rem;

cursor: pointer;

transition: all 0.2s ease;

}

.time-btn.active {

background-color: var(--primary-color);

color: white;

}

.time-btn:hover:not(.active) {

background-color: #e9ecef;

}

@media (max-width: 768px) {

.header {

flex-direction: column;

align-items: flex-start;

}

.chart-container {

height: 300px;

}

.data-summary {

grid-template-columns: 1fr;

}

.card-actions {

width: 100%;

justify-content: space-between;

}

}

.fade-in {

animation: fadeIn 0.6s ease-in-out;

}

@keyframes fadeIn {

from { opacity: 0; transform: translateY(10px); }

to { opacity: 1; transform: translateY(0); }

}

</style>

</head>

<body>

<div class="dashboard-container">

<div class="header animate\_\_animated animate\_\_fadeIn">

<h1 class="header-title">

<i class="fas fa-chart-network"></i>

Dashboard Monitoring Sensor

</h1>

<div class="time-selector">

<button class="time-btn active">24 Jam</button>

<button class="time-btn">7 Hari</button>

<button class="time-btn">30 Hari</button>

<button class="time-btn">Custom</button>

</div>

</div>

<div class="card animate\_\_animated animate\_\_fadeIn animate\_\_delay-1s">

<div class="card-header">

<h2 class="card-title">

<i class="fas fa-wave-square"></i>

Grafik Perbandingan Sensor

</h2>

<div class="card-actions">

<button class="btn btn-outline" onclick="window.location.href='{{ route('graph.export') }}'">

<i class="fas fa-download"></i> Export

</button>

</div>

</div>

<div class="chart-container">

<canvas id="sensorChart"></canvas>

</div>

<div class="data-summary">

<div class="summary-card fade-in">

<div class="summary-header">

<span class="summary-title">Sensor 1 (Rata-rata)</span>

<div class="summary-icon sensor-1">

<i class="fas fa-thermometer-half"></i>

</div>

</div>

<div class="summary-value" id="avg-sensor1">0</div>

<div class="summary-change positive">

<i class="fas fa-arrow-up"></i> <span id="change-sensor1">0%</span> dari periode sebelumnya

</div>

</div>

<div class="summary-card fade-in">

<div class="summary-header">

<span class="summary-title">Sensor 2 (Rata-rata)</span>

<div class="summary-icon sensor-2">

<i class="fas fa-thermometer-quarter"></i>

</div>

</div>

<div class="summary-value" id="avg-sensor2">0</div>

<div class="summary-change negative">

<i class="fas fa-arrow-down"></i> <span id="change-sensor2">0%</span> dari periode sebelumnya

</div>

</div>

<div class="summary-card fade-in">

<div class="summary-header">

<span class="summary-title">Korelasi</span>

<div class="summary-icon">

<i class="fas fa-link"></i>

</div>

</div>

<div class="summary-value" id="correlation-value">0.00</div>

<div class="summary-change neutral">

<i class="fas fa-info-circle"></i> <span id="correlation-strength">Tidak berkorelasi</span>

</div>

</div>

</div>

</div>

</div>

<script>

const labels = @json($labels);

const dataNilai1 = @json($dataNilai1);

const dataNilai2 = @json($dataNilai2);

function calculateStats(data) {

const sum = data.reduce((a, b) => a + b, 0);

const avg = sum / data.length;

const max = Math.max(...data);

const min = Math.min(...data);

return { sum, avg, max, min };

}

function calculateCorrelation(x, y) {

const n = x.length;

let sumX = 0, sumY = 0, sumXY = 0, sumX2 = 0, sumY2 = 0;

for (let i = 0; i < n; i++) {

sumX += x[i];

sumY += y[i];

sumXY += x[i] \* y[i];

sumX2 += x[i] \* x[i];

sumY2 += y[i] \* y[i];

}

const numerator = sumXY - (sumX \* sumY) / n;

const denominator = Math.sqrt((sumX2 - (sumX \* sumX) / n) \* (sumY2 - (sumY \* sumY) / n));

return denominator === 0 ? 0 : numerator / denominator;

}

const stats1 = calculateStats(dataNilai1);

const stats2 = calculateStats(dataNilai2);

const correlation = calculateCorrelation(dataNilai1, dataNilai2);

document.getElementById('avg-sensor1').textContent = stats1.avg.toFixed(2);

document.getElementById('avg-sensor2').textContent = stats2.avg.toFixed(2);

document.getElementById('change-sensor1').textContent = (Math.random() \* 5).toFixed(1) + '%';

document.getElementById('change-sensor2').textContent = (Math.random() \* 3).toFixed(1) + '%';

document.getElementById('correlation-value').textContent = correlation.toFixed(2);

const correlationStrength = document.getElementById('correlation-strength');

if (Math.abs(correlation) > 0.7) {

correlationStrength.textContent = 'Korelasi kuat';

correlationStrength.className = 'positive';

} else if (Math.abs(correlation) > 0.3) {

correlationStrength.textContent = 'Korelasi sedang';

correlationStrength.className = 'neutral';

} else {

correlationStrength.textContent = 'Korelasi lemah';

correlationStrength.className = 'negative';

}

const ctx = document.getElementById('sensorChart').getContext('2d');

const chart = new Chart(ctx, {

type: 'line',

data: {

labels: labels,

datasets: [

{

label: 'Sensor 1',

data: dataNilai1,

borderColor: '#4361ee',

backgroundColor: 'rgba(67, 97, 238, 0.1)',

borderWidth: 2,

tension: 0.3,

fill: true,

pointBackgroundColor: 'white',

pointBorderColor: '#4361ee',

pointBorderWidth: 2,

pointRadius: 4,

pointHoverRadius: 6,

yAxisID: 'y'

},

{

label: 'Sensor 2',

data: dataNilai2,

borderColor: '#4cc9f0',

backgroundColor: 'rgba(76, 201, 240, 0.1)',

borderWidth: 2,

tension: 0.3,

fill: true,

pointBackgroundColor: 'white',

pointBorderColor: '#4cc9f0',

pointBorderWidth: 2,

pointRadius: 4,

pointHoverRadius: 6,

yAxisID: 'y'

}

]

},

options: {

responsive: true,

maintainAspectRatio: false,

interaction: {

mode: 'index',

intersect: false

},

plugins: {

legend: {

position: 'top',

labels: {

usePointStyle: true,

padding: 20,

font: {

size: 13,

weight: '500'

}

}

},

tooltip: {

backgroundColor: 'rgba(0, 0, 0, 0.85)',

titleFont: {

size: 14,

weight: '600'

},

bodyFont: {

size: 13

},

padding: 12,

cornerRadius: 8,

usePointStyle: true,

callbacks: {

label: function(context) {

let label = context.dataset.label || '';

if (label) {

label += ': ';

}

if (context.parsed.y !== null) {

label += context.parsed.y.toFixed(2);

}

return label;

}

}

},

annotation: {

annotations: {

line1: {

type: 'line',

yMin: stats1.avg,

yMax: stats1.avg,

borderColor: '#4361ee',

borderWidth: 1,

borderDash: [5, 5],

label: {

content: 'Rata-rata S1: ' + stats1.avg.toFixed(2),

enabled: true,

position: 'right',

backgroundColor: 'rgba(67, 97, 238, 0.7)'

}

},

line2: {

type: 'line',

yMin: stats2.avg,

yMax: stats2.avg,

borderColor: '#4cc9f0',

borderWidth: 1,

borderDash: [5, 5],

label: {

content: 'Rata-rata S2: ' + stats2.avg.toFixed(2),

enabled: true,

position: 'right',

backgroundColor: 'rgba(76, 201, 240, 0.7)'

}

}

}

}

},

scales: {

y: {

beginAtZero: false,

grid: {

color: 'rgba(0, 0, 0, 0.05)'

},

ticks: {

font: {

size: 12

}

}

},

x: {

grid: {

display: false

},

ticks: {

font: {

size: 12

}

}

}

},

animation: {

duration: 1000,

easing: 'easeOutQuart'

}

}

});

// Time selector functionality

document.querySelectorAll('.time-btn').forEach(btn => {

btn.addEventListener('click', function() {

document.querySelectorAll('.time-btn').forEach(b => b.classList.remove('active'));

this.classList.add('active');

chart.data.datasets.forEach(dataset => {

dataset.data = dataset.data.map(() => Math.random() \* 100);

});

chart.update();

});

});

window.addEventListener('resize', function() {

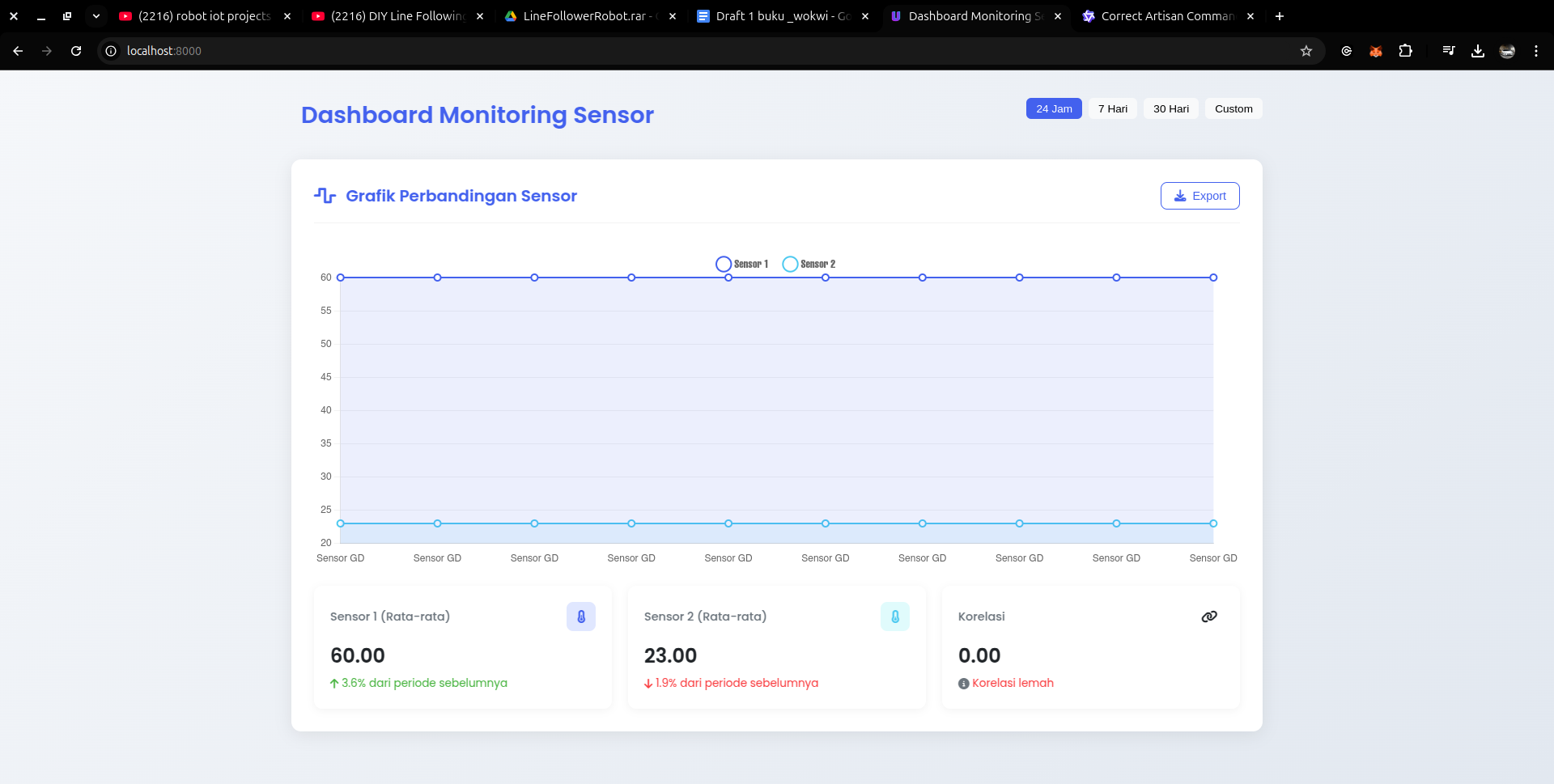
chart.resize();

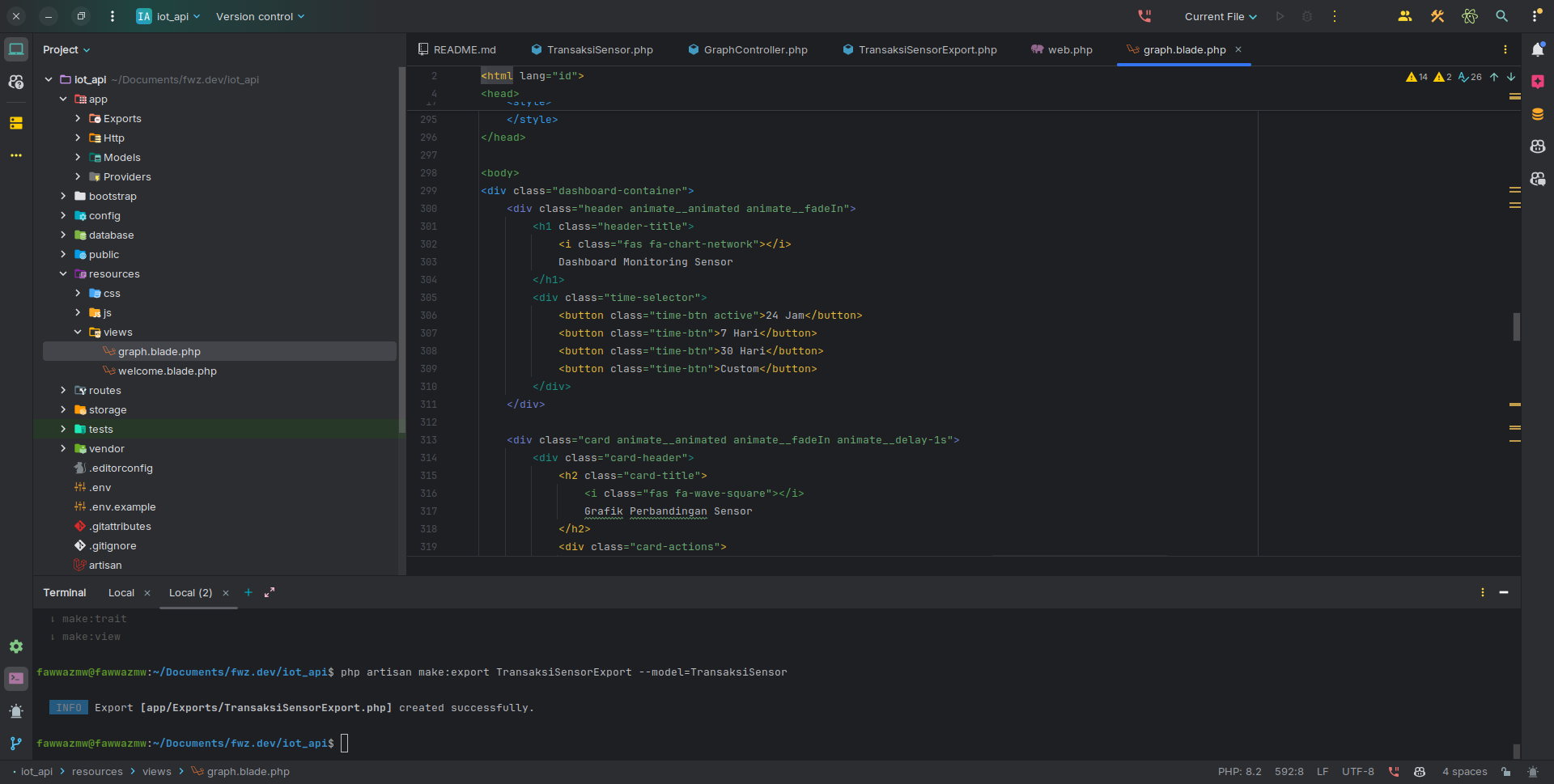
});

</script>

</body>

</html>

****

****

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

**3.1 Experimental Results (Hasil Eksperimen)**

Eksperimen ini berhasil menunjukkan bagaimana API Laravel dapat digunakan untuk menerima data dari perangkat IoT (ESP32). Integrasi antara perangkat keras (ESP32) dan perangkat lunak (Laravel) berjalan lancar, dengan Ngrok memainkan peran penting dalam memungkinkan akses API secara publik.

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