### **Practical API Development Using Laravel 11 and Ngrok with Wokwi Simulation**

*Candra Wahyu Perdana*

Vocational Faculty, Brawijaya University

Email: candraega28@gmail.com

**Abstract**

This practical work aims to develop a RESTful API using Laravel 11, accessed via Ngrok, and simulated with the Wokwi platform using ESP32 and a DHT22 sensor. The API manages sensor transaction data stored in a MySQL database. The implementation includes creating models, database migrations, resource transformations, and API controllers for CRUD operations. Testing was conducted using Postman and the Wokwi simulator, demonstrating successful data transmission and storage in the database.

*Keywords—Laravel 11, API Development, Ngrok, Wokwi, ESP32, RESTful API, CRUD, IoT, MySQL*

**1. Introduction**

**1.1 Background**

The growth of the Internet of Things (IoT) has increased the demand for reliable APIs to manage sensor data in real time. Laravel simplifies API development with built-in features like Eloquent ORM and straightforward routing. With Ngrok, a local API can be accessed publicly, allowing testing with external devices, such as ESP32 simulated through Wokwi.

**1.2 Objectives**

* Develop a RESTful API using Laravel 11 to manage sensor data.
* Connect the API to ESP32 through the Wokwi simulation.
* Test API communication with DHT22 sensor data and store it in MySQL.

**2. Methodology**

**2.1 Tools & Materials**

* **Software:** Laravel 11, Ngrok, Postman, PlatformIO, Wokwi Simulator, MySQL
* **Hardware (simulated):** ESP32, DHT22 sensor

**Environment:**

* **Laravel Server: php artisan serve --host=0.0.0.0 --port=8080**
* **Ngrok: ngrok http --scheme=http 8080**
* **API Testing:** Postman & Wokwi Simulator

#### **2.2 Implementation Steps**

### **1. Database Setup**

CREATE DATABASE iot\_25;

### Create a new database named **iot\_25** in **phpMyAdmin**.

**2. Laravel Model & Migration**

php artisan make:model TransaksiSensor -m

**Modify the migration file:**

| Schema::create('transaksi\_sensors', function (Blueprint $table) {  $table->id();  $table->string('nama\_sensor');  $table->float('nilai1');  $table->float('nilai2');  $table->timestamps();  }); |
| --- |

### **Run the migration:**

php artisan migrate

### **3. Resource & API Controller**

Create a resource and controller:

| php artisan make:resource TransaksiSensorResource  php artisan make:controller Api/TransaksiSensorController |
| --- |

Implement CRUD functionality in the controller:

| public function store(Request $request)  {  $sensor = TransaksiSensor::create($request->all());  return new TransaksiSensorResource($sensor);  } |
| --- |

**4. API Routing**

Add API routes:

| Route::apiResource('posts', TransaksiSensorController::class); |
| --- |

**5. API Testing via Postman**

Run the laravel server:

| php artisan serve --host=0.0.0.0 --port=8080 |
| --- |

Test the API locally:

* URL: http://127.0.0.1:8080/api/posts
* Methods: **GET, POST, PUT, DELETE**

**6. Expose API with Ngrok** Run Ngrok:

| ngrok http --scheme=http 8080 |
| --- |

**7. Wokwi Simulator Setup**

Create a main.cpp file to connect ESP32 to the API:

| #include <WiFi.h>  #include <HTTPClient.h>  #include "DHT.h"  #define DHTPIN 27  #define DHTTYPE DHT22  DHT dht(DHTPIN, DHTTYPE);  const char\* ssid = "Wokwi-GUEST";  const char\* password = "";  const char\* serverUrl = "http://<ngrok-url>/api/posts";  void setup() {  Serial.begin(115200);  WiFi.begin(ssid, password);  while (WiFi.status() != WL\_CONNECTED) delay(500);  dht.begin();  }  void loop() {  if (WiFi.status() == WL\_CONNECTED) {  HTTPClient http;  http.begin(serverUrl);  http.addHeader("Content-Type", "application/json");  float h = round(dht.readHumidity());  float t = round(dht.readTemperature());  String payload = "{\"nama\_sensor\":\"Sensor GD\", \"nilai1\":" + String(h) + ", \"nilai2\":" + String(t) + "}";  int httpResponseCode = http.POST(payload);  Serial.println("HTTP Response Code: " + String(httpResponseCode));  http.end();  delay(5000);  }  } |
| --- |

**8. Run Simulation**

**Start Laravel server**: php artisan serve

**Start Ngrok**: ngrok http --scheme=http 8080

**Run Wokwi Simulator**: Click **Start Simulator**

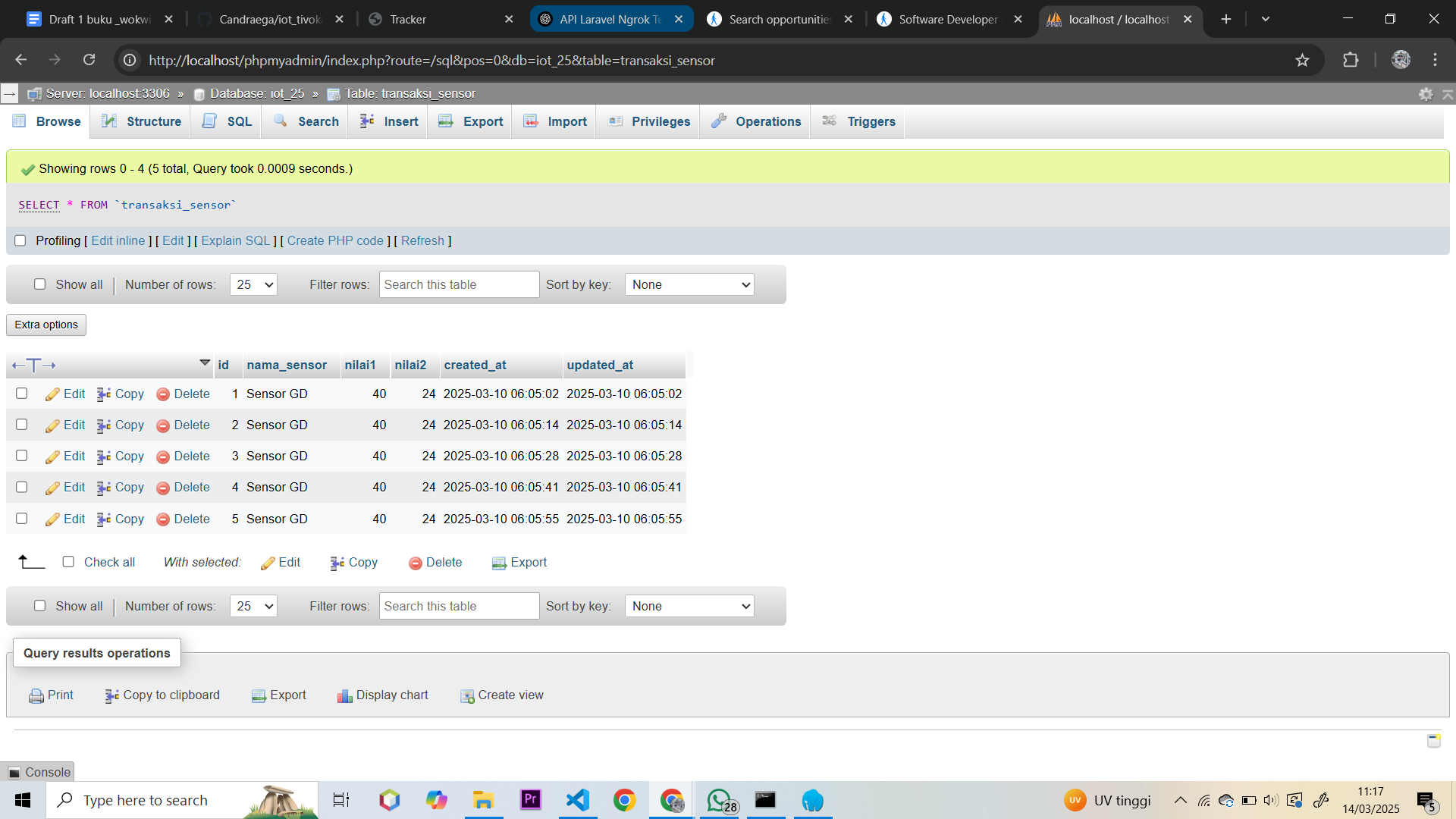
Check the output in the serial monitor:

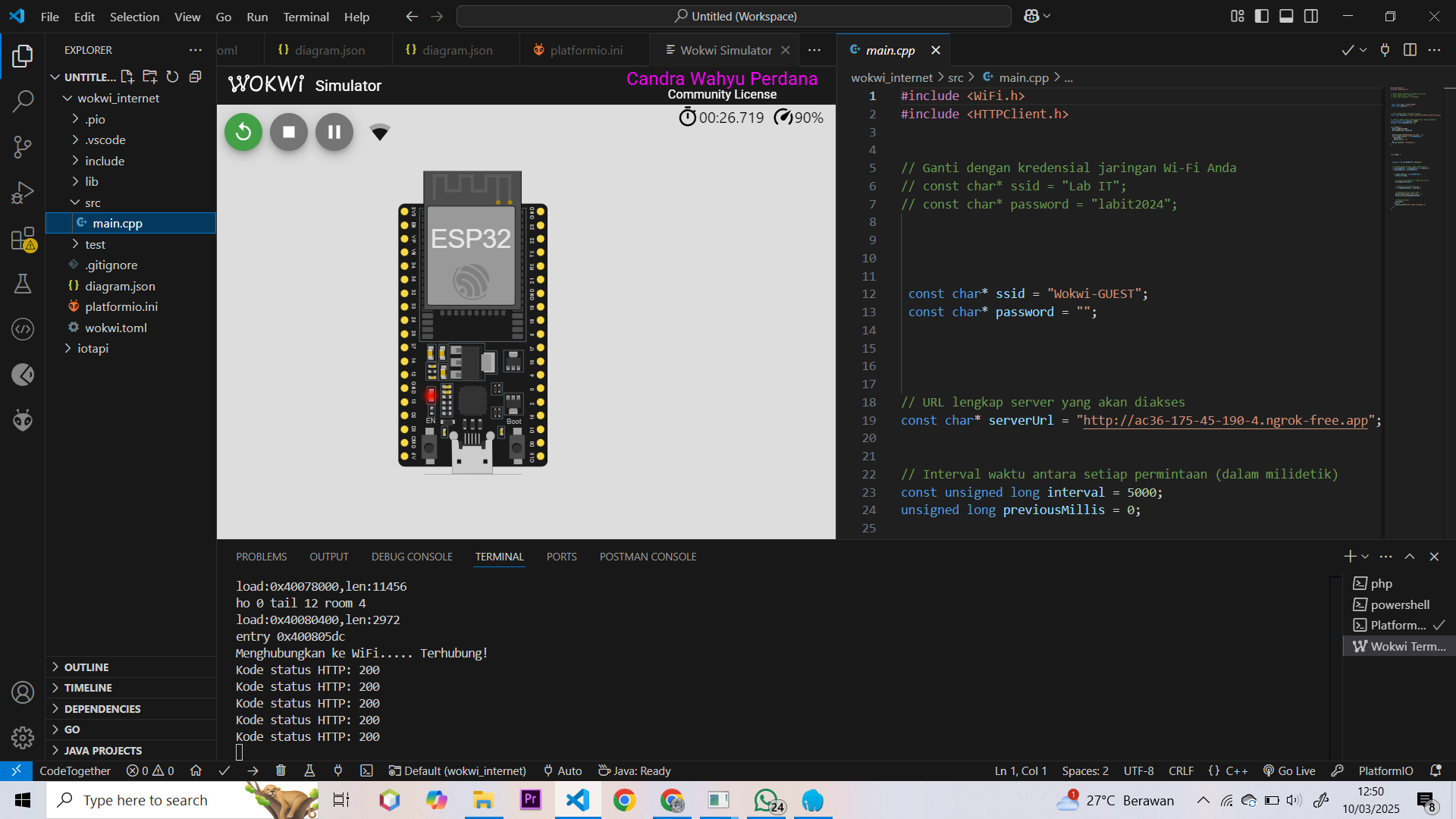
| HTTP Response Code: 201 |
| --- |

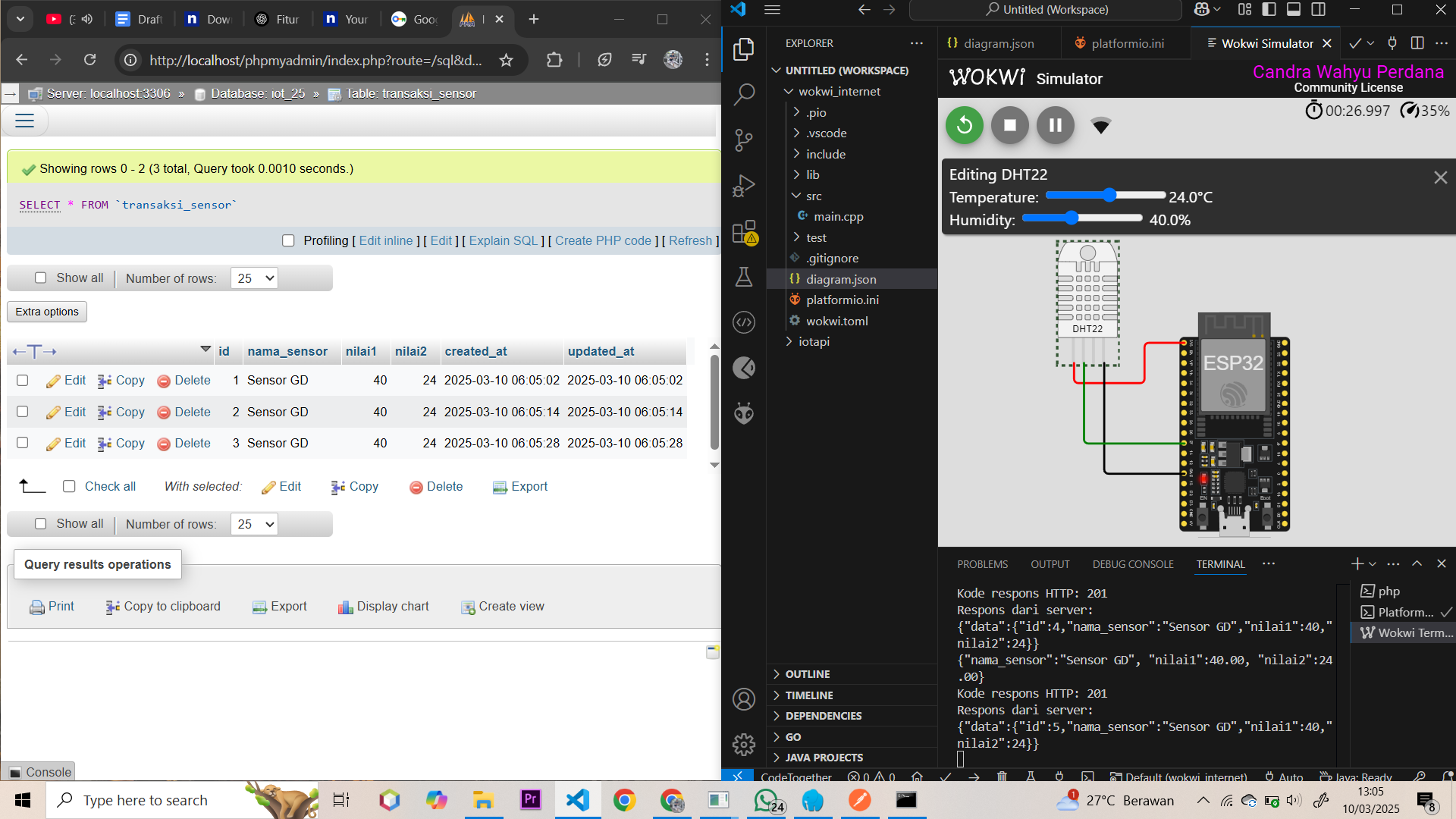
## **3. Results & Discussion**

The ESP32 successfully connected to the Laravel API through Ngrok, sending temperature and humidity data every 5 seconds, and storing it in the MySQL database. The HTTP 201 Created response indicates successful data transmission and storage.

Data stored in MySQL:







## **4. Conclusion**

This practical work successfully developed a RESTful API using Laravel 11, connected it to an ESP32 via Wokwi and Ngrok, and integrated it with a MySQL database. The API effectively received periodic temperature and humidity data, demonstrating great potential for real-time environmental monitoring applications.