

**LAPORAN PRAKTIKUM INTERNET OF THINGS (IoT)
PRAKTIK SIMULASI ESP32 & SENSOR SUHU KELEMBABAN**



Nadia Aulia Zahra
Fakultas Vokasi, Universitas Brawijaya
Email: nadiaaulia@student.ub.ac.id

**Fakultas Vokasi
Universitas Brawijaya
Tahun Ajaran 2025**

ABSTRAK

Pada penelitian ini, dilakukan simulasi penggunaan relay, button, dan LED serta implementasi sensor jarak ultrasonik menggunakan platform berbasis mikrokontroler. Selain itu, dikembangkan API menggunakan Laravel 11 yang dihubungkan dengan perangkat keras melalui Ngrok untuk memungkinkan akses dari jaringan eksternal. Relay digunakan untuk mengontrol perangkat listrik, button sebagai input manual, dan LED sebagai indikator. Sensor ultrasonik berfungsi untuk mendeteksi jarak suatu objek dan mengirimkan data ke server. API yang dibuat dengan Laravel 11 memungkinkan komunikasi antara perangkat dan aplikasi berbasis web secara real-time. Ngrok digunakan untuk menyalurkan API agar dapat diakses secara global tanpa perlu konfigurasi jaringan yang kompleks. Hasil simulasi menunjukkan bahwa sistem dapat berfungsi dengan baik dalam membaca input dari button dan sensor, mengendalikan relay dan LED, serta mengirim dan menerima data melalui API.

Kata kunci: *Relay, Button, LED, Sensor Ultrasonik, API, Laravel 11, Ngrok.*

1. Pendahuluan

1.1. Latar Belakang

Dalam dunia Internet of Things (IoT), komunikasi antara perangkat keras dan aplikasi berbasis web menjadi hal yang krusial. Salah satu metode untuk mencapai komunikasi ini adalah dengan menggunakan API (Application Programming Interface). Laravel 11 sebagai framework PHP modern menawarkan kemudahan dalam membangun API yang dapat diakses dari berbagai platform. Untuk memungkinkan akses dari jaringan eksternal, Ngrok digunakan sebagai tunneling service. Selain itu, penggunaan relay, button, LED, dan sensor ultrasonik semakin banyak dimanfaatkan dalam sistem otomatisasi rumah, industri, dan keamanan. Oleh karena itu, eksperimen ini dilakukan untuk mensimulasikan interaksi antara perangkat keras dan API menggunakan Laravel 11 serta Ngrok.

1.2. Tujuan Eksperimen

Tujuan dari eksperimen ini adalah:

- Mempelajari cara kerja relay, button, LED, dan sensor ultrasonik dalam sistem otomatisasi.
- Mengembangkan API menggunakan Laravel 11 untuk berkomunikasi dengan perangkat keras.
- Menggunakan Ngrok untuk memungkinkan akses API dari jaringan eksternal.
- Menguji integrasi antara perangkat keras dan API dalam berbagai skenario.

2. Metodologi

2.1. Tools & Materials (Alat dan Bahan)

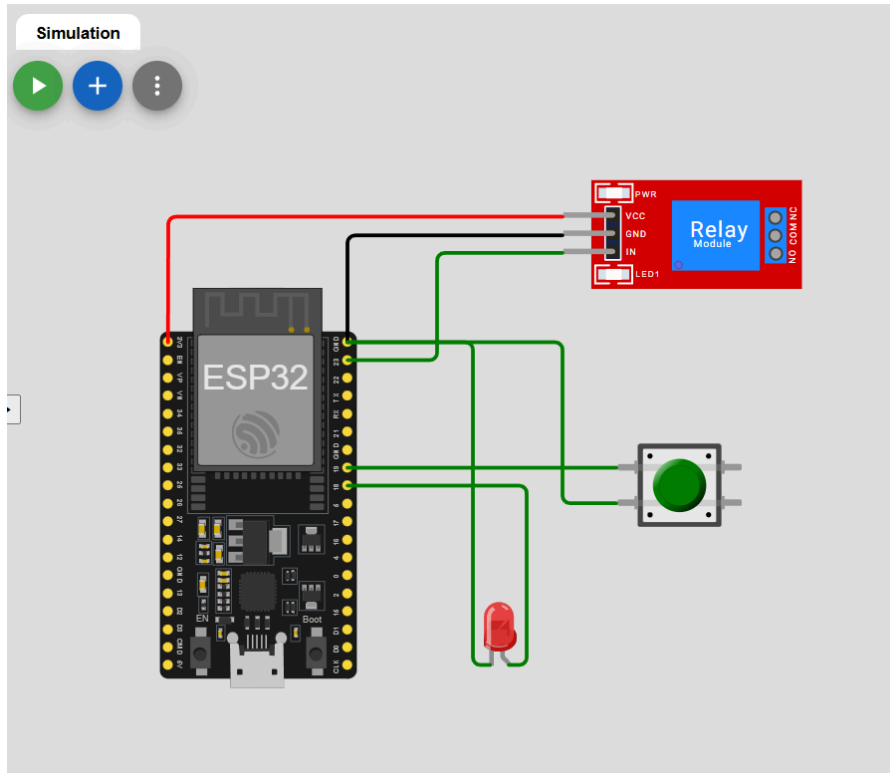
Eksperimen ini dilakukan dengan pendekatan berikut:

- **Perancangan Sistem:** Menentukan kebutuhan perangkat keras dan perangkat lunak.
- **Implementasi Perangkat Keras:** Menghubungkan relay, button, LED, dan sensor ultrasonik dengan mikrokontroler.
- **Pengembangan API:** Membangun API dengan Laravel 11 untuk menangani komunikasi data.
- **Integrasi dengan Ngrok:** Mengaktifkan Ngrok untuk membuat API dapat diakses secara global.
- **Pengujian dan Evaluasi:** Mengamati kinerja sistem dalam membaca input dan mengendalikan output melalui API.

2.2. Langkah Implementasi

2.2.1. Langkah Implementasi Praktik Simulasi Relay Button dan LED

- 1) Membuka web wokwi dan membuat diagram seperti berikut



- 2) Menginput kode ke file main.cpp di project platformio yang telah dibuat

```
File Edit Selection View Go Run Terminal Help ← → LatihanSimulasiRelayButtondanLED
EXPLORER
  OPEN EDITORS
    main.cpp x diagram.json wokwi.toml platformio.ini PIO Home
  LATIHANSIMULASIRELAYBUTTONDANLED
    .pio
    .vscode
    include
    lib
    src
      main.cpp
    test
    .gitignore
    diagram.json
    platformio.ini
    wokwi.toml
  OUTLINE
src > main.cpp > setup()
1 #include <Arduino.h>
2
3 // Define pin numbers
4 const int ButtonPin = 19; // GPIO19 connected to the pushbutton
5 const int LedPin = 18; // GPIO18 connected to the LED
6 const int RelayPin = 23; // GPIO23 connected to the relay module
7
8 void setup() {
9   // Set pin modes
10  pinMode(ButtonPin, INPUT_PULLUP); // Set the button pin as an input with an internal pull-up resistor
11  pinMode(LedPin, OUTPUT); // Set the LED pin as an output
12  pinMode(RelayPin, OUTPUT); // Set the relay pin as an output
13
14  // Initialize the outputs to be OFF
15  digitalWrite(LedPin, LOW);
16  digitalWrite(RelayPin, LOW);
17 }
18
19 void loop() {
20   // Read the state of the button
21   int buttonState = digitalRead(ButtonPin);
22
23   // Check if the button is pressed
24   // Since the button is wired to pull the pin LOW when pressed, we check for LOW
25   if (buttonState == LOW) {
26     digitalWrite(LedPin, HIGH); // Turn on the LED
27     digitalWrite(RelayPin, HIGH); // Turn on the relay
28   } else {
29     digitalWrite(LedPin, LOW); // Turn off the LED
30     digitalWrite(RelayPin, LOW); // Turn off the relay
31   }
32 }
33
```

3) Menginput kode di file wokwi.toml

[wokwi]

version = 1

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

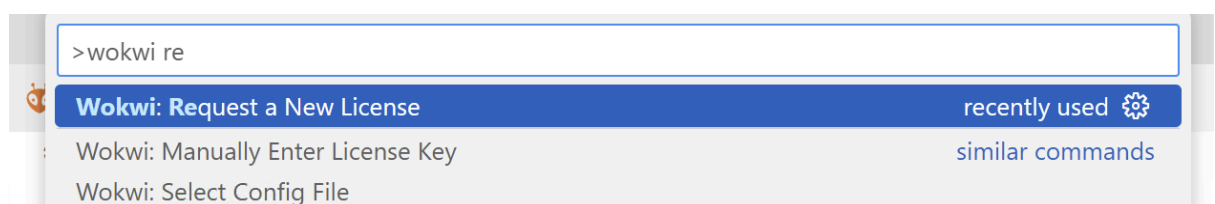
elf = '.pio\build\esp32doit-devkit-v1\firmware.elf'

4) Menginput kode di file diagram.json

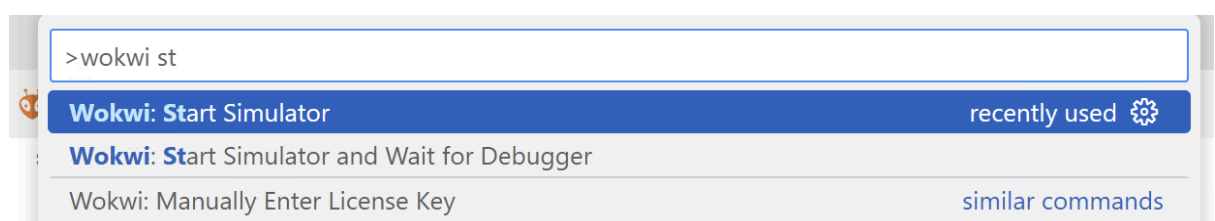
```
1  diagram.json > ...
2  {
3    "version": 1,
4    "author": "nadia",
5    "editor": "wokwi",
6    "parts": [
7      { "type": "board-esp32-devkit-c-v4", "id": "esp", "top": -9.6, "left": -167.96, "attrs": {} },
8      { "type": "wokwi-led", "id": "led1", "top": 150, "left": -5.8, "attrs": { "color": "red" } },
9      { "type": "wokwi-pushbutton",
10        "id": "btn1",
11        "top": 73.4,
12        "left": 76.8,
13        "attrs": { "color": "green", "xray": "1" }
14      },
15      { "type": "wokwi-relay-module", "id": "relay1", "top": -67, "left": 48, "attrs": {} }
16    ],
17    "connections": [
18      [ "esp:TX", "$serialMonitor:RX", "", [ ] ],
19      [ "esp:RX", "$serialMonitor:TX", "", [ ] ],
20      [ "esp:3V3", "relay1:VCC", "red", [ "h0.15", "v-67.2", "h211.2" ] ],
21      [ "esp:GND.2", "relay1:GND", "black", [ "v0" ] ],
22      [ "esp:GND.2", "btn1:2.1", "green", [ "h115.2", "v86.2" ] ],
23      [ "esp:23", "relay1:IN", "green", [ "h48", "v-57.4" ] ],
24      [ "esp:GND.2", "led1:C", "green", [ "h67.2", "v172.8" ] ],
25      [ "esp:19", "btn1:1.1", "green", [ "h0" ] ],
26      [ "esp:18", "led1:A", "green", [ "h96", "v96" ] ]
27    ],
28    "dependencies": {}
29  }
```

5) Langkah berikutnya lakukan compile pada file main.cpp, lalu anda akan mendapatkan 2 file baru yaitu firmware.bin dan firmware.elf

6) Langkah berikutnya lakukan request license ke wokwi.com

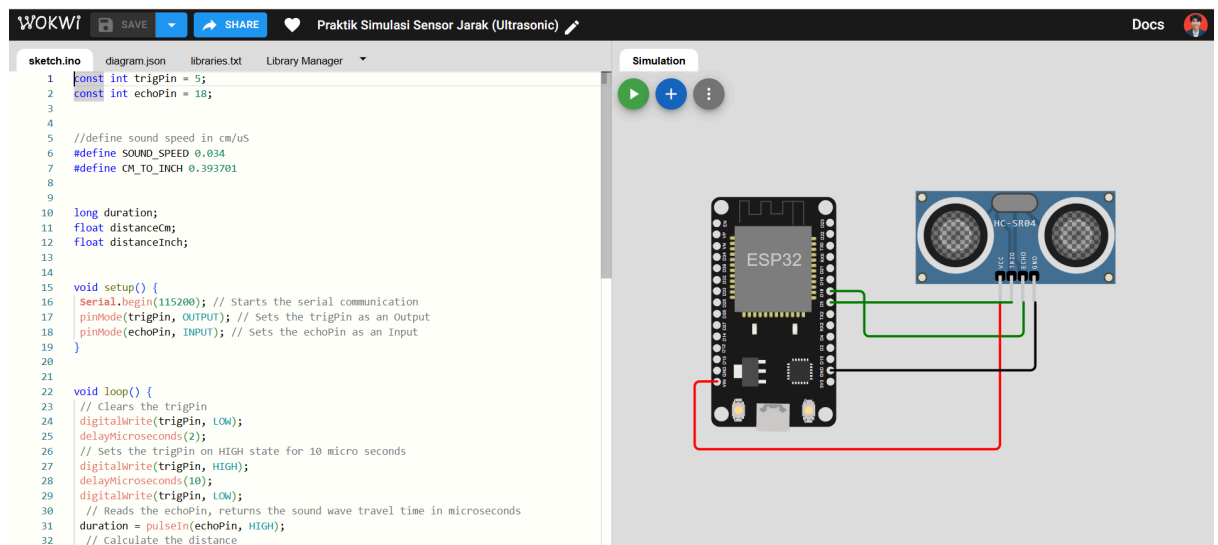


7) Langkah terakhir jalankan simulasi dengan mengetik perintah :

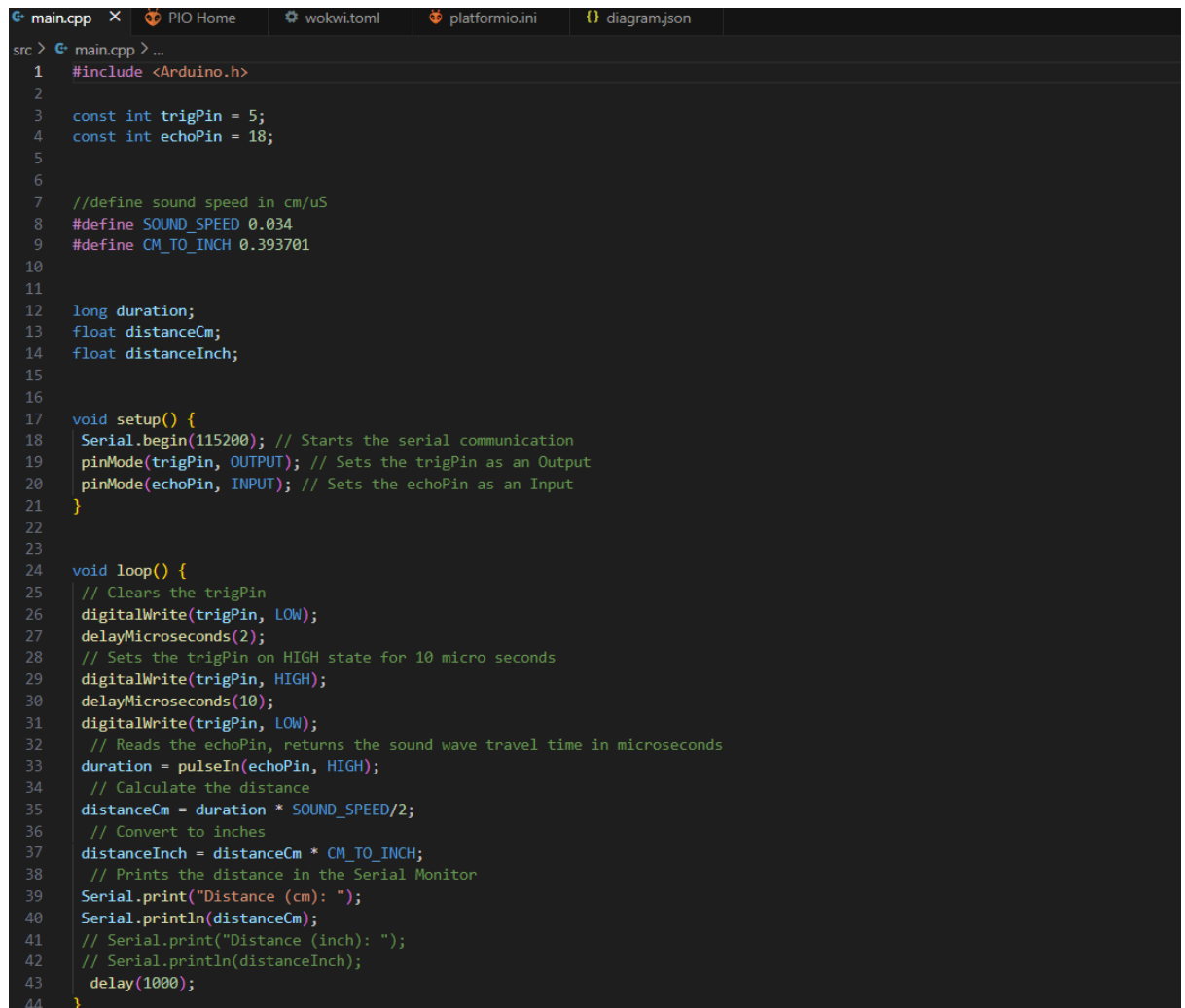


2.2.2. Langkah Implementasi Praktik Simulasi Sensor Jarak (Ultrasonic)

1) Membuka web wokwi dan membuat diagram seperti berikut



2) Menginput kode ke file main.cpp di project platformio yang telah dibuat



3) Menginput kode di file wokwi.toml

```
[wokwi]
```

```
version = 1
```

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

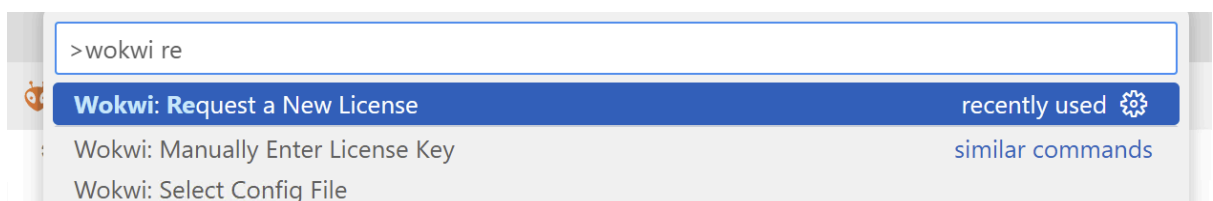
```
elf = '.pio\build\esp32doit-devkit-v1\firmware.elf'
```

4) Menginput kode di file diagram.json

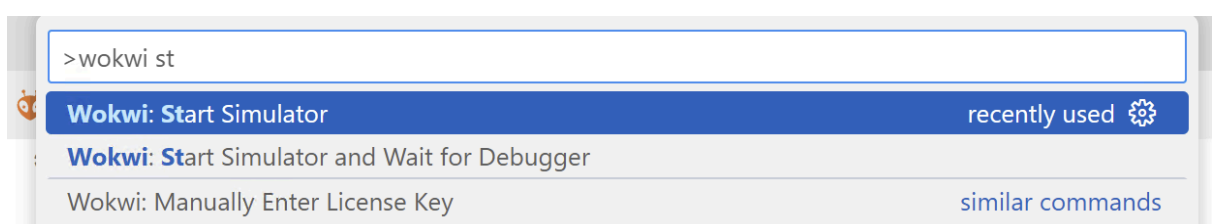
```
{ diagram.json > ...
1  {
2      "version": 1,
3      "author": "nadia",
4      "editor": "wokwi",
5      "parts": [
6          { "type": "wokwi-esp32-devkit-v1", "id": "esp", "top": -91.3, "left": -62.6, "attrs": {} },
7          { "type": "wokwi-hc-sr04", "id": "ultrasonic1", "top": -94.5, "left": 111.1, "attrs": {} }
8      ],
9      "connections": [
10         [ "esp:TX0", "$serialMonitor:RX", "", [ ] ],
11         [ "esp:RX0", "$serialMonitor:TX", "", [ ] ],
12         [ "esp:D18", "ultrasonic1:ECHO", "green", [ "h28.5", "v38.4", "h135.2" ] ],
13         [ "esp:VIN", "ultrasonic1:VCC", "red", [ "h-19.2", "v57.6", "h259.2" ] ],
14         [ "esp:D5", "ultrasonic1:TRIG", "green", [ "h0" ] ],
15         [ "esp:GND.1", "ultrasonic1:GND", "black", [ "h0" ] ]
16     ],
17     "dependencies": {}
18 }
```

5) Langkah berikutnya lakukan compile pada file main.cpp, lalu anda akan mendapatkan 2 file baru yaitu firmware.bin dan firmware.elf

6) Langkah berikutnya lakukan request license ke wokwi.com



7) Langkah terakhir jalankan simulasi dengan mengetik perintah :



2.2.3. Langkah Implementasi Praktik Pembuatan API Menggunakan Laravel 11 dan Ngrok

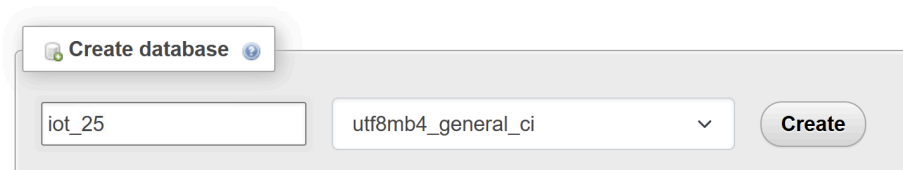
Download paket-paket yang dibutuhkan untuk memulai laravel 11 dengan mengetikkan perintah berikut di terminal/command prompt.

```
composer create-project --prefer-dist laravel/laravel:^11.0 laravel-11
```

```
cd laravel-11
```

Buat database di phpmyadmin dengan nama **iot_25**

Databases



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=caberg2010
```

```
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_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');
    }
};

```

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_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 = [];
}

```

Kemudian jalankan perintah berikut untuk membuat tabel :

php artisan migrate

```
PS E:\htdocs\sivoka2025\laravel-11> php artisan migrate
```

```
INFO Preparing database.
```

```
Creating migration table ..... 12.41ms DONE
```

```
INFO Running migrations.
```

```
0001_01_01_000000_create_users_table ..... 48.48ms DONE
0001_01_01_000001_create_cache_table ..... 15.35ms DONE
0001_01_01_000002_create_jobs_table ..... 44.44ms DONE
2025_02_21_072524_create_personal_access_tokens_table ..... 22.91ms DONE
2025_02_21_074123_create_transaksi_sensors_table ..... 11.14ms DONE
```

Buat Resource dengan menjalankan perintah :

php artisan make:resource TransaksiSensorResource

```
PS E:\htdocs\sivoka2025\laravel-11> php artisan make:resource TransaksiSensorResource
```

```
INFO Resource [E:\htdocs\sivoka2025\laravel-11\app\Http\Resources\TransaksiSensorResource.php] created successfully.
```

The screenshot shows the Visual Studio Code interface. On the left, the 'EXPLORER' sidebar displays the project structure for 'LARAVEL-11'. The 'app' directory is expanded, showing 'Http' > 'Resources'. The file 'TransaksiSensorResource.php' is selected. On the right, the editor window shows the code for 'TransaksiSensorResource.php'. The code defines a class 'TransaksiSensorResource' that extends 'JsonResource'. It includes property definitions for 'status', 'message', and 'resource', and a 'construct' method with parameters for 'status', 'message', and 'resource'.

```
1  <?php
2
3  namespace App\Http\Resources;
4
5  use Illuminate\Http\Request;
6  use Illuminate\Http\Resources\Json\JsonResource;
7
8  class TransaksiSensorResource extends JsonResource
9  {
10     //define properti
11     public $status;
12     public $message;
13     public $resource;
14
15     /**
16      * __construct
17      *
18      * @param mixed $status
19      * @param mixed $message
20      * @param mixed $resource
21      * @return void
22      */
```

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

```

PS E:\htdocs\sivoka2025\laravel-11> php artisan make:controller Api/TransaksiSensorController

  INFO  Controller [E:\htdocs\sivoka2025\laravel-11\app\Http\Controllers\Api\TransaksiSensorController.php] created successfully.

```

Ubah isi 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)
{

```

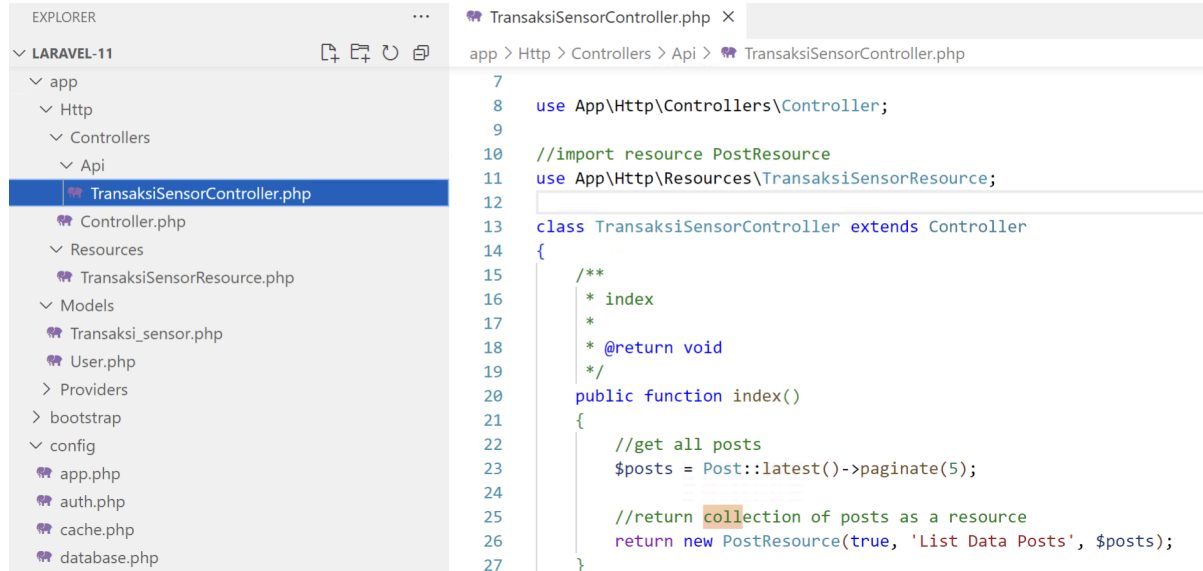
```

$transaksiSensor = TransaksiSensor::findOrFail($id);
$transaksiSensor->delete();

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

}

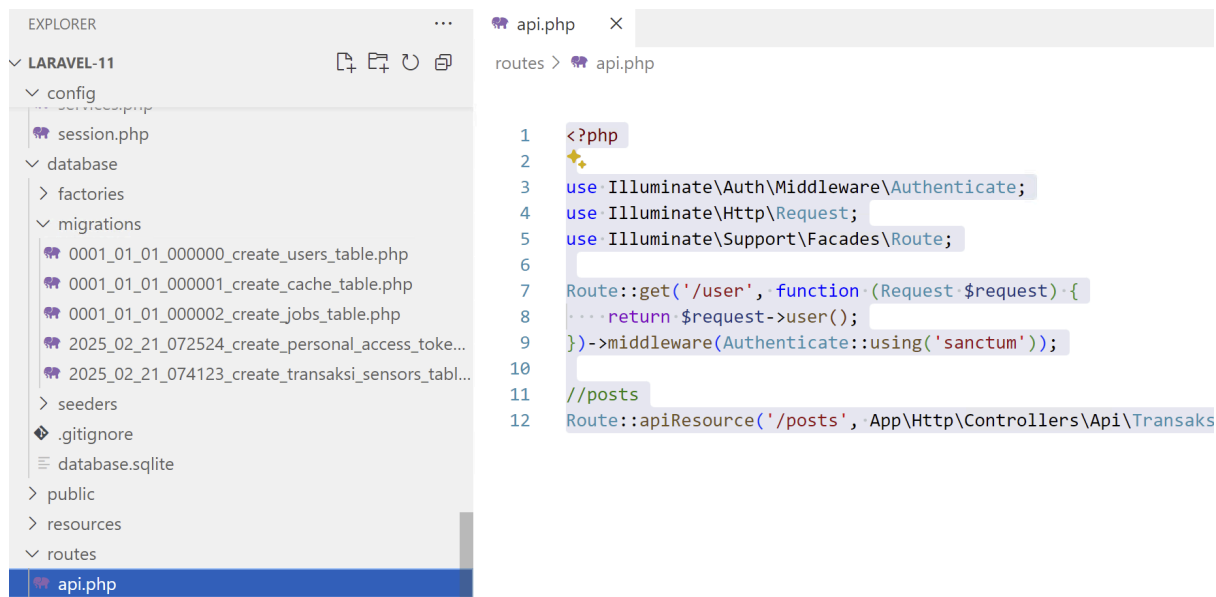
```



Buat route khusus API dengan menjalankan perintah :

php artisan install:api

PS E:\htdocs\sivoka2025\laravel-11> **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 :

```
PS E:\htdocs\sivoka2025\laravel-11> php artisan route:list
```

```
GET|HEAD      / .....
POST          _ignition/execute-solution . ignition.executeSolution > Spatie\LaravelIgnition > ExecuteSolutionController
GET|HEAD      _ignition/health-check ..... ignition.healthCheck > Spatie\LaravelIgnition > HealthCheckController
POST          _ignition/update-config ..... ignition.updateConfig > Spatie\LaravelIgnition > UpdateConfigController
GET|HEAD      api/posts ..... posts.index > Api\TransaksiSensorController@index
POST          api/posts ..... posts.store > Api\TransaksiSensorController@store
GET|HEAD      api/posts/{post} ..... posts.show > Api\TransaksiSensorController@show
PUT|PATCH    api/posts/{post} ..... posts.update > Api\TransaksiSensorController@update
DELETE        api/posts/{post} ..... posts.destroy > Api\TransaksiSensorController@destroy
```

Untuk melakukan testing, gunakan tools postman dengan langkah sebagai berikut :

Download aplikasi postman pada link berikut : <https://www.postman.com/downloads/>



Download Postman

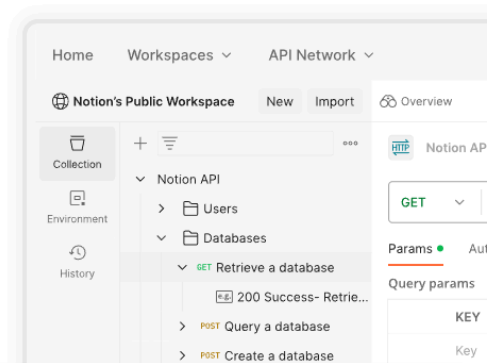
Download the app to get started using the Postman API Platform today. Or, if you prefer a browser experience, you can try the web version of Postman.

The Postman app

Download the app to get started with the Postman API Platform.



By downloading and using Postman, I agree to the [Privacy Policy](#) and [Terms](#).

[Release Notes →](#)

Lakukan prosedur instalasi dan jalankan aplikasi postman.

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

php artisan serve

```
PS E:\htdocs\sivoka2025\laravel-11> php artisan serve
```

INFO Server running on [http://127.0.0.1:8000].

Press Ctrl+C to stop the server

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

Server: 127.0.0.1 » Database: iot_25 » Table: transaksi_sensor

[Browse](#)
[Structure](#)
[SQL](#)
[Search](#)
[Insert](#)
[Export](#)
[Import](#)
[Privileges](#)

✓ Showing rows 0 - 1 (2 total, Query took 0.0003 seconds.)

```
SELECT * FROM `transaksi_sensor`
```

☐ Profiling [[Edit inline](#)] [[Edit](#)] [[Explain SQL](#)] [[Create PHP code](#)] [[Refresh](#)]

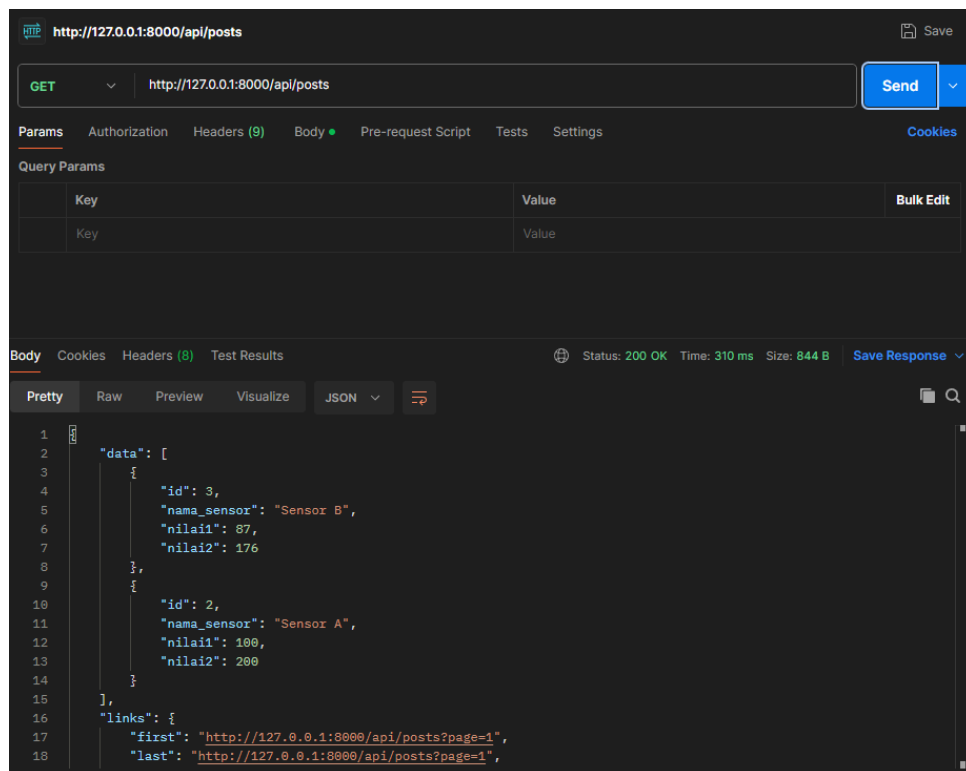
☐ Show all | Number of rows: Filter rows: Sort by key:

Extra options

		id	nama_sensor	nilai1	nilai2	created_at	updated_at
<input type="checkbox"/>	Edit Copy Delete	2	Sensor A	100	200	2025-02-21 15:21:12	2025-02-21 15:21:12
<input type="checkbox"/>	Edit Copy Delete	3	Sensor B	87	176	2025-02-22 23:44:24	2025-02-22 23:44:24

☐ Check all | With selected: [Edit](#) [Copy](#) [Delete](#) [Export](#)

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

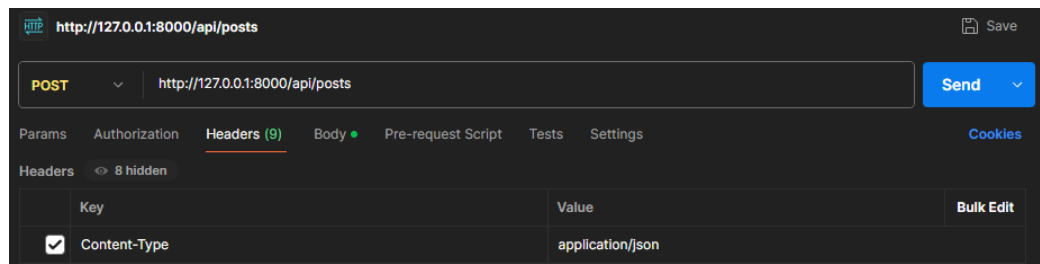
```
{
  "data": [
    {
      "id": 3,
      "nama_sensor": "Sensor B",
      "nilai1": 87,
      "nilai2": 176
    },
    {
      "id": 2,
      "nama_sensor": "Sensor A",
      "nilai1": 100,
      "nilai2": 200
    }
  ],
  "links": {
```

```

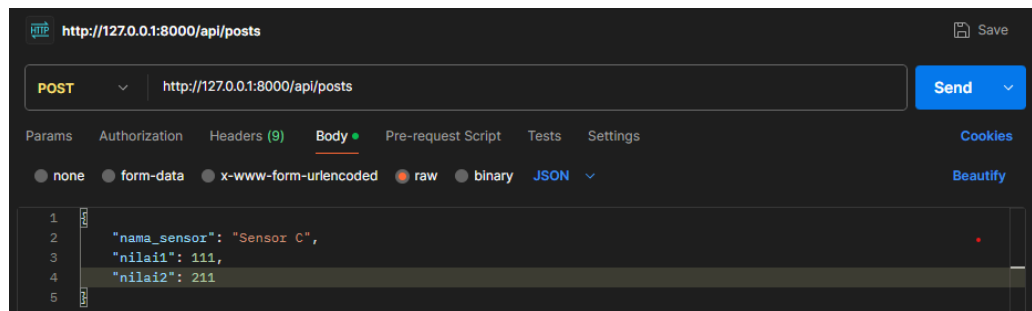
    "first": "http://127.0.0.1:8000/api/posts?page=1",
    "last": "http://127.0.0.1:8000/api/posts?page=1",
    "prev": null,
    "next": null
  },
  "meta": {
    "current_page": 1,
    "from": 1,
    "last_page": 1,
    "links": [
      {
        "url": null,
        "label": "&laquo; Previous",
        "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
  }
}

```

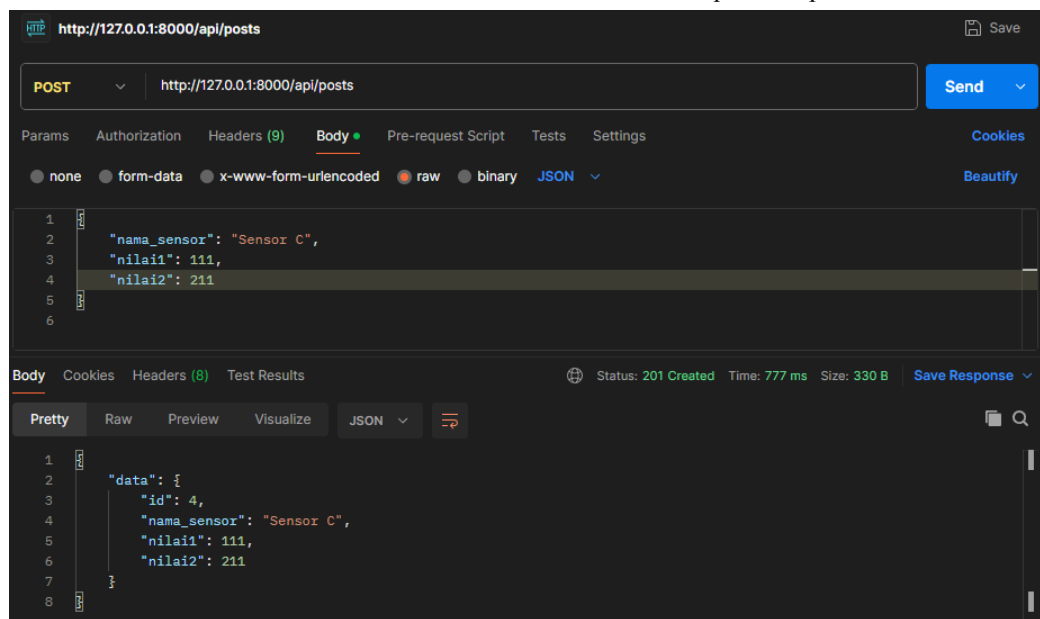
API telah berfungsi untuk mengambil data dari database. Langkah berikutnya adalah melakukan percobaan insert data ke tabel di database menggunakan API. Caranya adalah mengganti method menjadi POST kemudian pada bagian header ubah menjadi sebagai berikut



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

Server: 127.0.0.1 » Database: iot_25 » Table: transaksi_sensor

Browse

Structure

SQL

Search

Insert

Export

Import

More

Showing rows 0 - 2 (3 total, Query took 0.0003 seconds.)

SELECT * FROM `transaksi_sensor`

Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]

Show all

Number of rows: 25

Filter rows: Search this table

Sort by key: None

Extra options

id

nama_sensor

nilai1

nilai2

created_at

updated_at

Edit

Copy

Delete

2

Sensor A

100

200

2025-02-21 15:21:12

2025-02-21 15:21:12

Edit

Copy

Delete

3

Sensor B

87

176

2025-02-22 23:44:24

2025-02-22 23:44:24

Edit

Copy

Delete

4

Sensor C

111

211

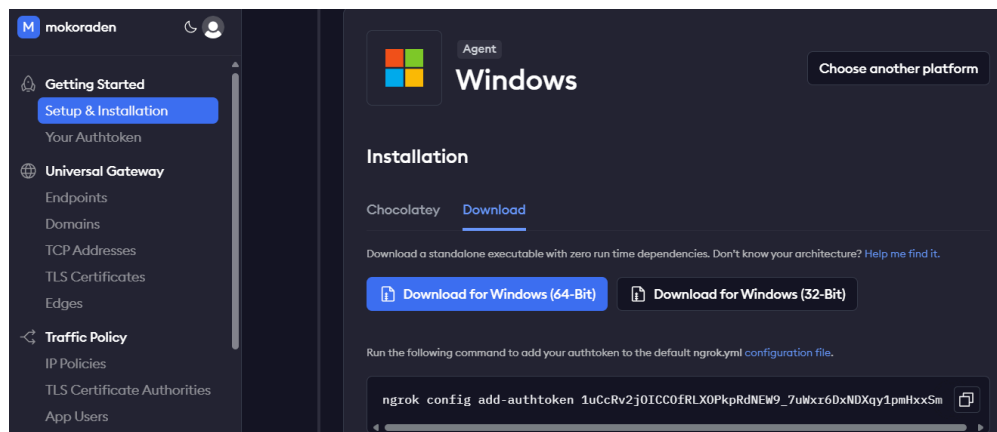
2025-02-24 07:32:22

2025-02-24 07:32:22

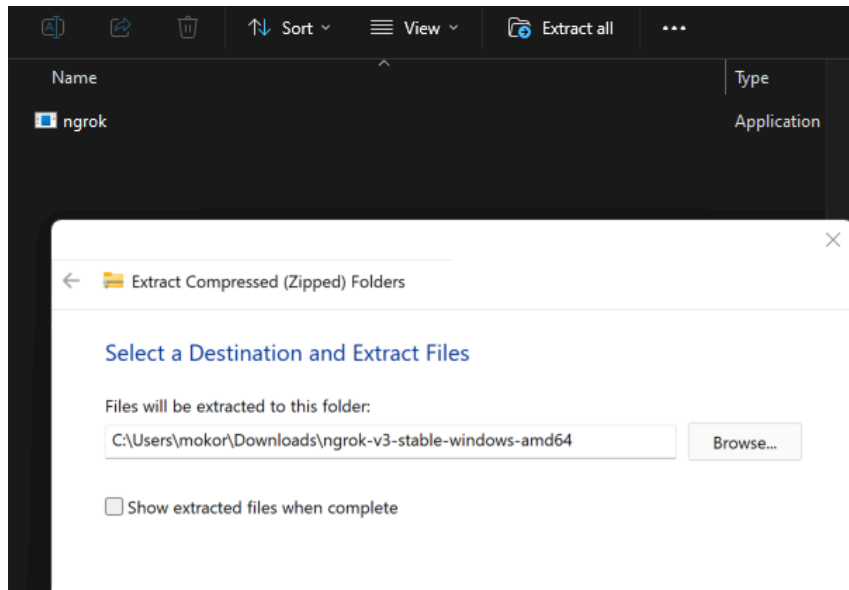
Langkah berikutnya adalah mengonline-kan API menggunakan service ngrok sehingga API dapat diakses melalui device iot atau simulasi wokwi iot.

Download dan install aplikasi ngrok pada URL : <https://dashboard.ngrok.com/signup> kemudian lakukan registrasi.

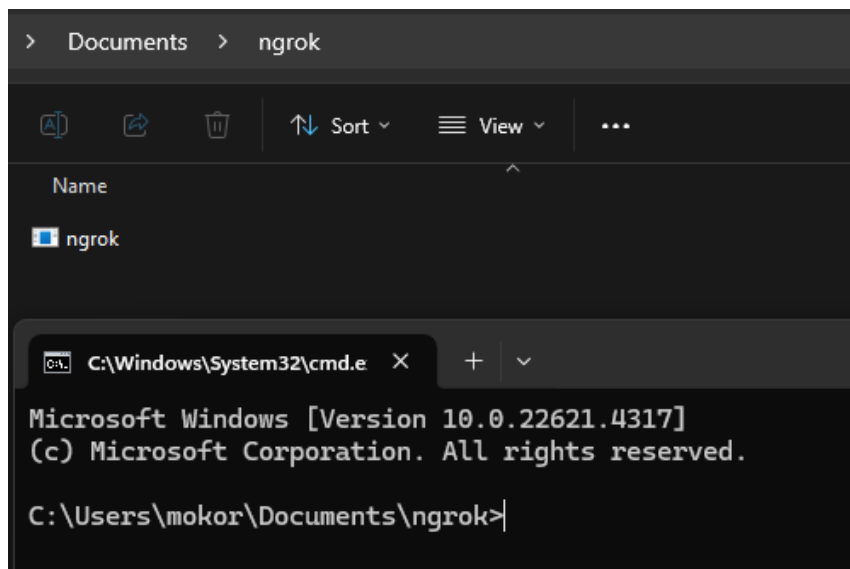
Login ke web ngrok, kemudian download aplikasi ngrok sesuai sistem operasi



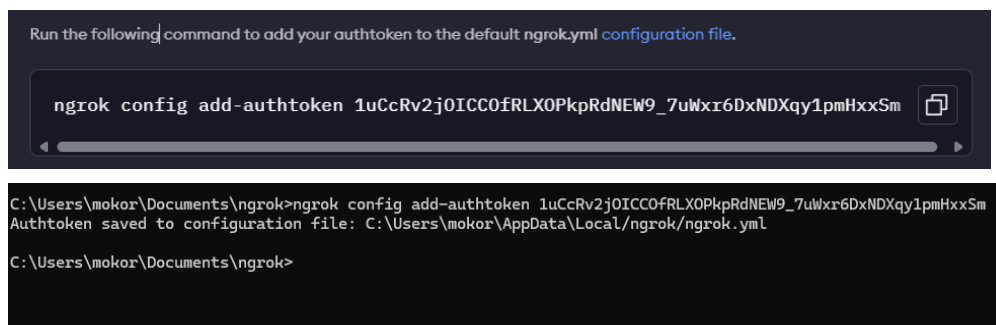
Lakukan ekstraksi



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>

```
C:\Windows\System32\cmd.e x + -
ngrok (Ctrl+C to quit)
> Goodbye tunnels, hello Agent Endpoints: https://ngrok.com/r/aep

Session Status      online
Account             mokoraden (Plan: Free)
Version             3.20.0
Region              Asia Pacific (ap)
Latency              23ms
Web Interface        http://127.0.0.1:4040
Forwarding           https://e521-2405-8740-6315-3520-2099-2415-5077-c12f.ngrok-free.app -> http://localhost:80

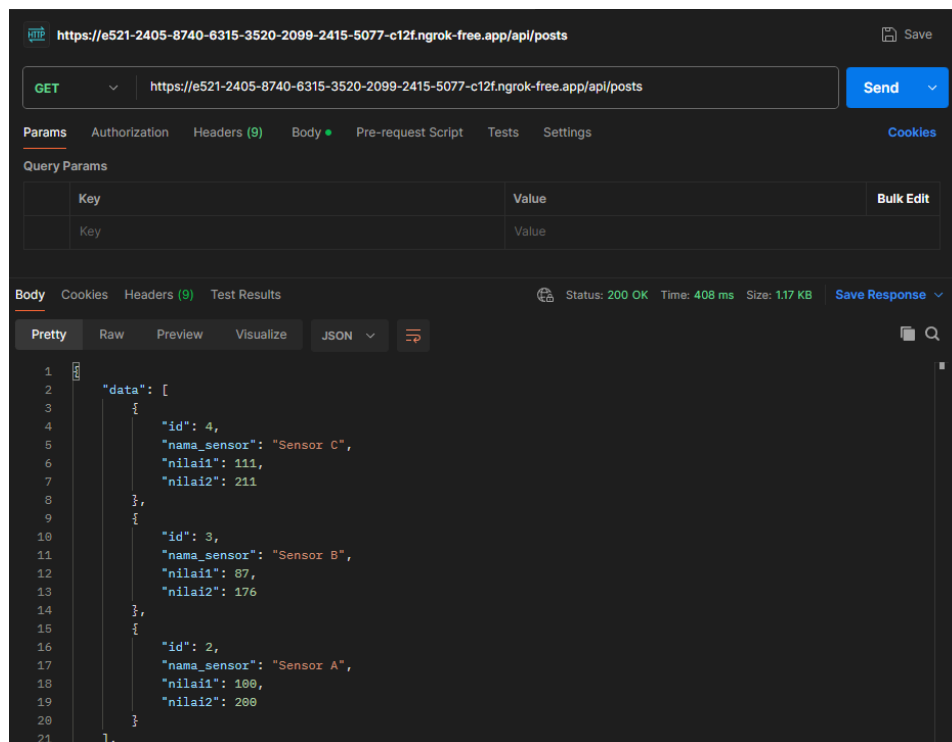
Connections          ttl    opn    rt1    rt5    p50    p90
0                   0      0      0.00   0.00   0.00   0.00
```

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

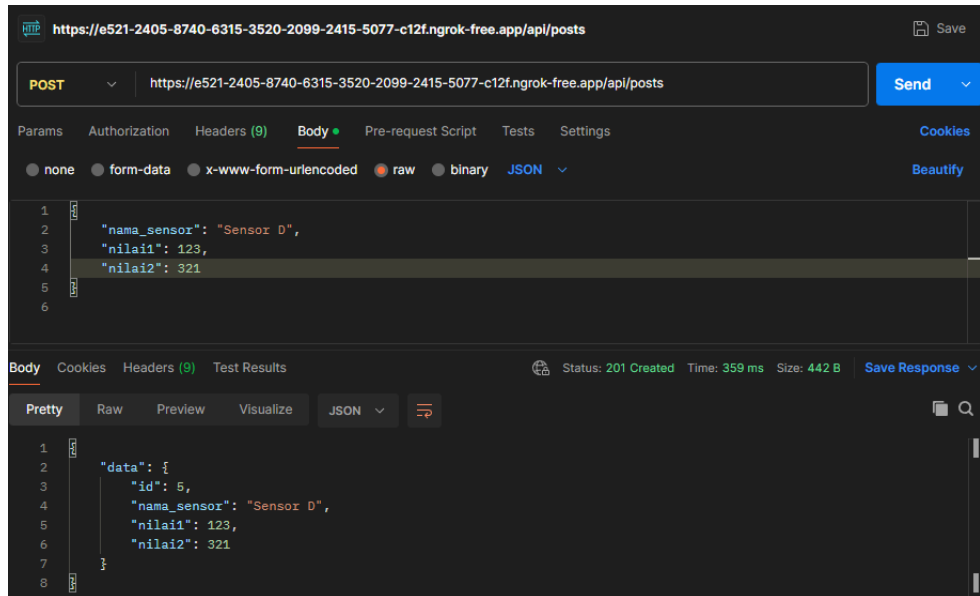
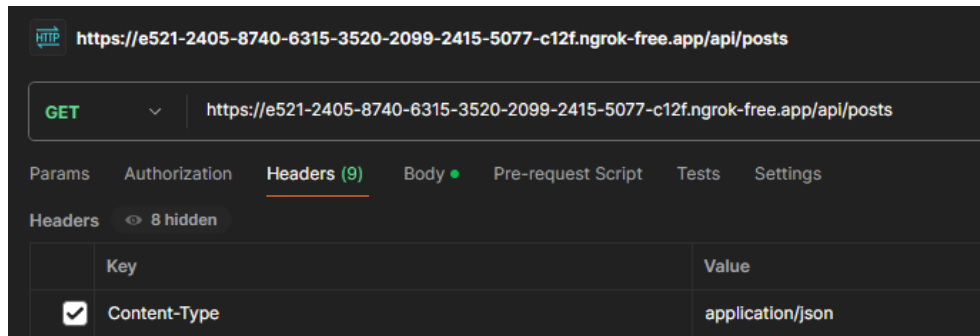
<https://e521-2405-8740-6315-3520-2099-2415-5077-c12f.ngrok-free.app/api/posts>



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.

3. Hasil dan Pembahasan

3.1 Hasil Eksperimen

3.1.1 Simulasi Relay, Button & LED

Hasil eksperimen menunjukkan bahwa relay dapat dikendalikan melalui API, memungkinkan perangkat elektronik dihidupkan dan dimatikan sesuai dengan perintah yang dikirim. Button dapat digunakan sebagai input manual untuk mengubah status relay dan LED. LED berhasil merespon perintah dari API maupun perubahan status yang ditentukan oleh button, menunjukkan integrasi perangkat keras yang berjalan dengan baik.

3.1.2. Sensor Jarak (Ultrasonik)

Sensor ultrasonik dapat membaca jarak objek secara real-time dan mengirimkan datanya ke API. Hasil pengujian menunjukkan bahwa sensor memiliki tingkat akurasi yang baik dalam mengukur jarak dalam rentang tertentu. Namun, terdapat sedikit gangguan pada pengukuran ketika ada objek dengan permukaan tidak rata atau berbentuk kompleks.

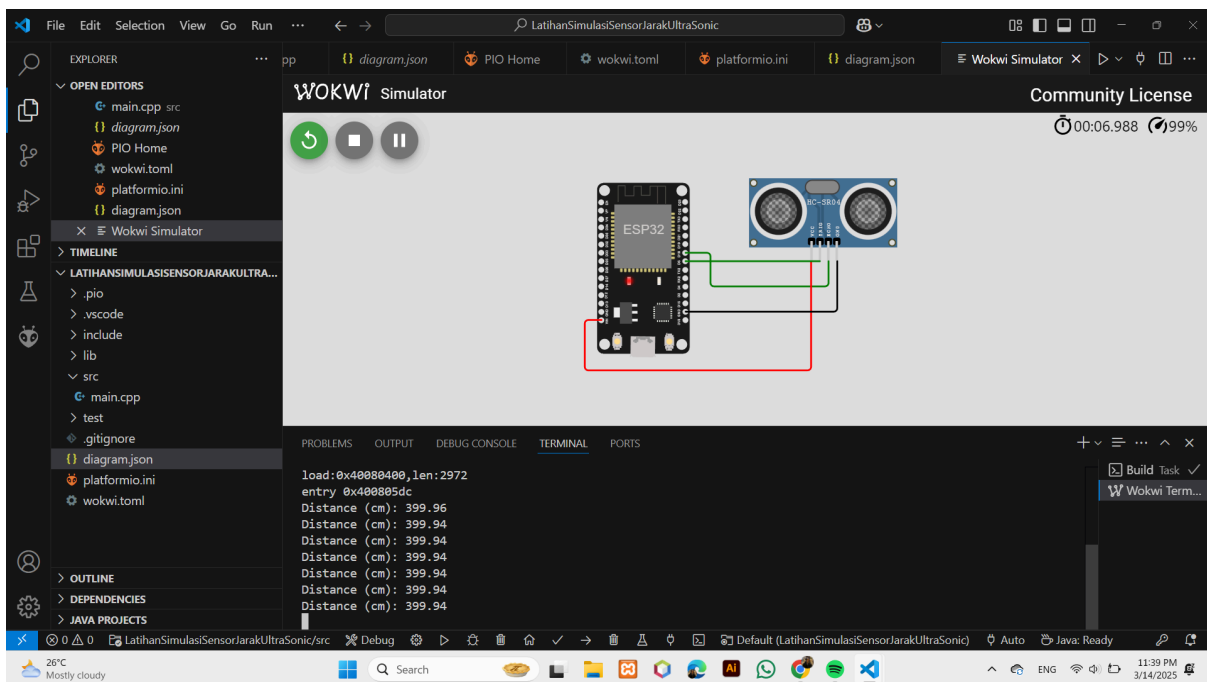
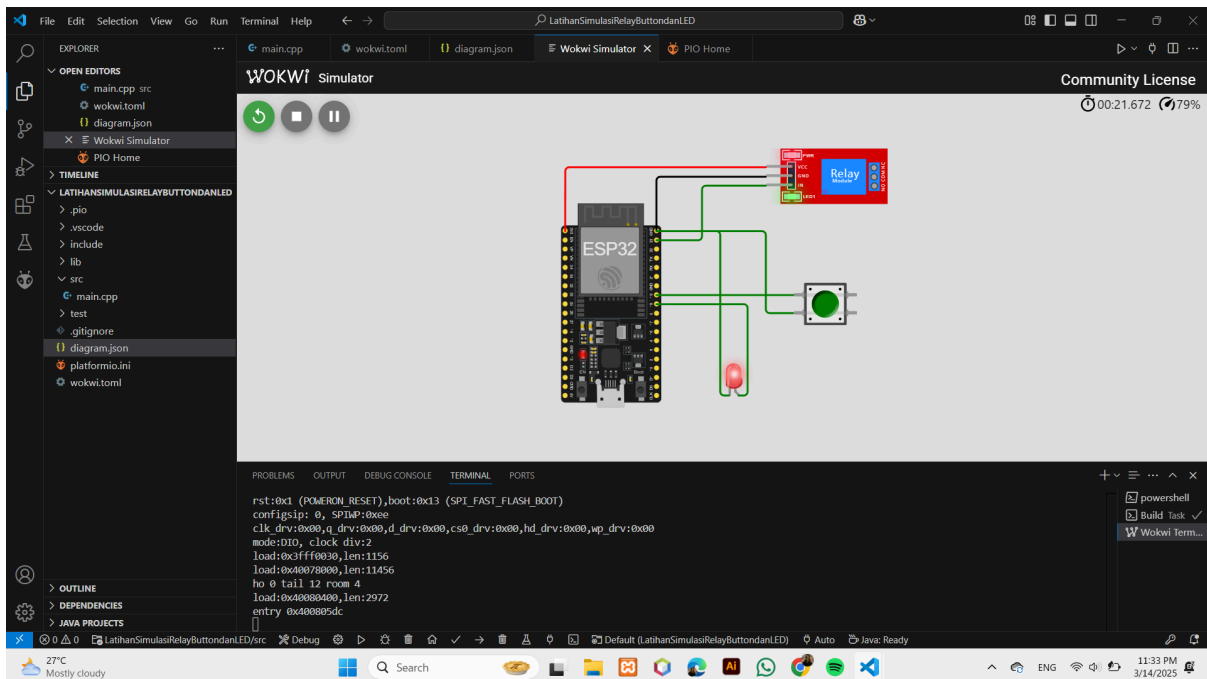
3.1.3. Pembuatan API Menggunakan Laravel 11 dan Ngrok

API yang dikembangkan menggunakan Laravel 11 berhasil mengelola permintaan dari perangkat keras, termasuk menerima data dari sensor ultrasonik dan mengontrol relay serta LED. Integrasi dengan Ngrok memungkinkan API diakses dari luar jaringan lokal tanpa konfigurasi tambahan pada router. Namun, terdapat sedikit latensi dalam pengiriman data saat menggunakan Ngrok, yang perlu diperhitungkan dalam aplikasi yang membutuhkan respons real-time.

3.2 Kesimpulan

Eksperimen ini berhasil membuktikan bahwa integrasi antara perangkat keras dan API menggunakan Laravel 11 serta Ngrok dapat dilakukan dengan baik. Relay, button, LED, dan sensor ultrasonik dapat beroperasi sesuai dengan perintah yang dikirim melalui API. Penggunaan Ngrok mempermudah akses API tanpa perlu konfigurasi jaringan yang kompleks. Implementasi ini berpotensi diterapkan dalam berbagai aplikasi IoT seperti sistem otomatisasi rumah dan pemantauan jarak jauh. Namun, perlu dilakukan optimasi untuk mengurangi latensi pada komunikasi API melalui Ngrok.

4. Lampiran Jika diperlukan



File Edit Selection View Go Run Terminal Help laravel-11

EXPLORER

OPEN EDITORS

database.php config

TransaksiSensorResource.php

TransaksiSensorController.php 4

api.php routes

2025_03_14_080137_create_tr...

TransaksiSensor.php app/Mod...

LARAVEL-11

app

Http

Controllers

Api

TransaksiSensorControll... 4

Controller.php

Resources

Models

Providers

bootstrap

config

database

factories

migrations

0001_01_01_000000_create_user...

0001_01_01_000001_create_cac...

0001_01_01_000002_create_jobs...

2025_03_14_080137_create_tran...

2025_03_14_080535_create_pers...

seeders

.gitignore

database.sqlite

OUTLINE

0 4

28°C Mostly cloudy

```
app > Http > Controllers > Api > TransaksiSensorController.php > PHP Intelephense > TransaksiSensorController > index
1 <?php
2
3
4 namespace App\Http\Controllers\Api;
5 use Illuminate\Http\Request;
6
7
8 use App\Models\TransaksiSensor;
9 use App\Http\Controllers\Controller;
10 use App\Http\Resources\TransaksiSensorResource;
11
12
13 1 reference | 0 implementations
14 class TransaksiSensorController extends Controller
15 {
16
17     /**
18      * Index
19      *
20      * @return \Illuminate\Http\Response
21      */
22     // 0 references | 0 overrides
23     public function index(): AnonymousResourceCollection
24     {
25         // Get all transactions from TransaksiSensor model, paginated
26         $transaksiSensors = TransaksiSensor::latest()->paginate(perPage: 5);
27
28         // Return a collection of transactions as a resource
29         return TransaksiSensorResource::collection(resource: $transaksiSensors);
30     }
31
32     /**
33      * Store a newly created resource in storage.
34      *
35      * @param \Illuminate\Http\Request $request
36      * @return \Illuminate\Http\Response
37      */
38 }
```

localhost / 127.0.0.1 / iot_25 | php: x Laravel x | +

localhost/phmyadmin/index.php?route=/database/structure&db=iot_25

Server: 127.0.0.1 > Database: iot_25

Struktur SQL Cari Kueri Ekspor Impor Operasi Hak Akses Routine Event Trigger Pelacakan Desainer Tengah kolom

Filters

Mengandung kata

Tabel	Tindakan	Baris	Jenis	Penyortiran	Ukuran	Beban
cache	Jelajahi Struktur Cari Tambahkan Kosongkan Hapus	0	InnoDB	utf8mb4_unicode_ci	16.0 KB	-
cache_locks	Jelajahi Struktur Cari Tambahkan Kosongkan Hapus	0	InnoDB	utf8mb4_unicode_ci	16.0 KB	-
failed_jobs	Jelajahi Struktur Cari Tambahkan Kosongkan Hapus	0	InnoDB	utf8mb4_unicode_ci	32.0 KB	-
jobs	Jelajahi Struktur Cari Tambahkan Kosongkan Hapus	0	InnoDB	utf8mb4_unicode_ci	32.0 KB	-
job_batches	Jelajahi Struktur Cari Tambahkan Kosongkan Hapus	0	InnoDB	utf8mb4_unicode_ci	16.0 KB	-
migrations	Jelajahi Struktur Cari Tambahkan Kosongkan Hapus	5	InnoDB	utf8mb4_unicode_ci	16.0 KB	-
password_reset_tokens	Jelajahi Struktur Cari Tambahkan Kosongkan Hapus	0	InnoDB	utf8mb4_unicode_ci	16.0 KB	-
personal_access_tokens	Jelajahi Struktur Cari Tambahkan Kosongkan Hapus	0	InnoDB	utf8mb4_unicode_ci	48.0 KB	-
sessions	Jelajahi Struktur Cari Tambahkan Kosongkan Hapus	2	InnoDB	utf8mb4_unicode_ci	48.0 KB	-
transaksi_sensor	Jelajahi Struktur Cari Tambahkan Kosongkan Hapus	4	InnoDB	utf8mb4_unicode_ci	16.0 KB	-
users	Jelajahi Struktur Cari Tambahkan Kosongkan Hapus	0	InnoDB	utf8mb4_unicode_ci	32.0 KB	-
11 tabel	Jumlah	11	InnoDB	utf8mb4_general_ci	288.0 KB	0 B

Pilih Semua Dengan pilihan:

Cetak Kamus data

Create new table

Nama tabel Jumlah kolom

Konsol

3:43 PM 3/14/2025

GET

⌵

http://127.0.0.1:8000/api/posts

ParamsAuthorizationHeaders (11)Body●ScriptsSettings

☐ none

☐ form-data

☐ x-www-form-urlencoded

☒ raw

☐ binary

☐ GraphQL

JSON ⌵

1 {

2

3 "nama_sensor": "Sensor C",

4 "nilai1": 123,

5 "nilai2": 321

6

7 }

BodyCookies (2)Headers (7)Test Results⌚

PrettyRawPreviewVisualizeJSON ⌵⌵

1 {

2 "data": [

3 {

4 "id": 7,

5 "nama_sensor": "Sensor C",

6 "nilai1": 123,

7 "nilai2": 321

8 },

9 {

10 "id": 6,

11 "nama_sensor": "Sensor D",

12 "nilai1": 123,

```

ngrok
* Goodbye tunnels, hello Agent Endpoints: https://ngrok.com/r/aep

Session Status      online
Account             nadiauliazahra (Plan: Free)
Version             3.21.0
Region              Asia Pacific (ap)
Latency              163ms
Web Interface        http://127.0.0.1:4040
Forwarding           https://8de5-175-45-191-12.ngrok-free.app -> http://localhost:8000

Connections          ttl      opn      rt1      rt5      p50      p90
                    2        0        0.01     0.00     0.76     1.12

HTTP Requests
-----
15:41:53.516 +07 POST /api/posts      201 Created
15:40:57.704 +07 GET  /api/posts      200 OK

```

The screenshot shows the Postman application interface. At the top, there's a header with 'Workspaces' and 'More' dropdowns, a search bar, and an 'Upgrade' button. Below this is a sidebar with icons for Overview, Environment, and Collections. The main area displays a POST request to the URL 'https://8de5-175-45-191-12.ngrok-free.app/api/posts'. The request body is a JSON object: `{ "nama_sensor": "Sensor D", "nilai1": 123, "nilai2": 321 }`. The response is shown in the 'Body' tab, indicating a '201 Created' status with a response time of 603 ms and a size of 354 B. The response body is a JSON object: `{ "data": { "id": 5, "nama_sensor": "Sensor D", "nilai1": 123, "nilai2": 321 } }`. The bottom of the interface shows a 'Console' tab and a 'Postbot' button.

☐ Tampilkan semua

Jumlah baris: 25

Extra options

←T→							id	nama_sensor	nilai1	nilai2	created_at	updated_at
<input type="checkbox"/>		Ubah		Salin		Hapus	2	Sensor A	100	200	2025-03-12 15:18:35	2025-03-12 15:18:35
<input type="checkbox"/>		Ubah		Salin		Hapus	3	Sensor B	87	176	2025-03-12 15:18:35	2025-03-12 15:18:35
<input type="checkbox"/>		Ubah		Salin		Hapus	4	Sensor C	111	211	2025-03-14 08:31:44	2025-03-14 08:31:44
<input type="checkbox"/>		Ubah		Salin		Hapus	5	Sensor D	123	321	2025-03-14 08:41:53	2025-03-14 08:41:53
<input type="checkbox"/>		Ubah		Salin		Hapus	6	Sensor D	123	321	2025-03-14 08:43:34	2025-03-14 08:43:34
<input type="checkbox"/>		Ubah		Salin		Hapus	7	Sensor C	123	321	2025-03-14 08:43:53	2025-03-14 08:43:53