LAPORAN PRAKTIKUM INTERNET OF THINGS (IoT)

Fakultas Vokasi, Universitas Brawijaya

**Praktik Pembuatan API Menggunakan Laravel dan Ngrok**

**Tugas BAB 12**

****

*Davi Ilham Hasyim*

*233140701111048*

*Fakultas Vokasi, Universitas Brawijaya davihasyim1207@gmail.com*

# Abstrak

Laporan praktikum ini membahas proses pembuatan REST API menggunakan Laravel, framework PHP yang populer, serta pemanfaatan ngrok untuk melakukan tunneling agar API yang dibuat dapat diakses secara publik meskipun berjalan pada server lokal. Tujuan praktikum ini adalah untuk memahami konsep dasar pembuatan API dan mengintegrasikannya ke dalam ekosistem Internet of Things (IoT). Dalam praktikum ini, dijelaskan langkah-langkah mulai dari instalasi Laravel, pembuatan endpoint API, pengujian menggunakan Postman, hingga implementasi ngrok untuk mengakses API dari luar jaringan lokal. Hasil praktikum menunjukkan bahwa kombinasi Laravel 11 dan ngrok dapat memberikan solusi yang efisien untuk mengembangkan dan menguji aplikasi IoT berbasis API.

*Keyword : Laravel, ngrok, rest API, API*

# Pendahuluan

* 1. **Latar Belakang**

Perkembangan teknologi Internet of Things (IoT) telah membawa dampak besar pada cara perangkat terhubung dan berkomunikasi melalui jaringan. Salah satu komponen utama dalam implementasi IoT adalah Application Programming Interface (API) yang memungkinkan pertukaran data antar perangkat secara efisien dan terstruktur. Laravel merupakan salah satu framework PHP yang mendukung pembuatan API dengan cepat dan aman. Dengan fitur-fitur bawaan seperti routing, middleware, dan autentikasi, Laravel memudahkan pengembang dalam membangun API yang handal.

Dalam lingkungan pengembangan, sering kali dibutuhkan akses ke API dari luar jaringan lokal, terutama untuk keperluan demonstrasi atau integrasi dengan perangkat IoT lainnya. Ngrok adalah solusi yang memungkinkan pengembang untuk membuat tunnel dari server lokal ke internet, sehingga API yang dikembangkan dapat diakses secara real-time dari perangkat atau lokasi manapun. Dengan menggabungkan Laravel dan ngrok, praktikum ini bertujuan untuk memberikan pemahaman mendalam mengenai pembuatan dan pengujian API dalam konteks IoT.

# Tujuan Eksperimen

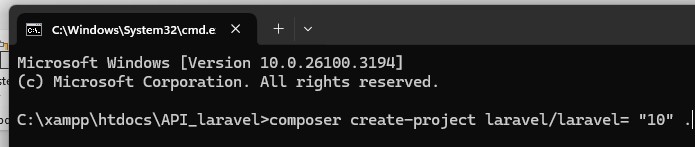
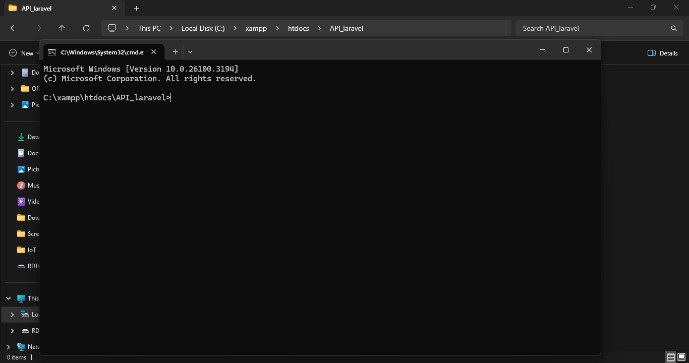
* **Memahami Konsep REST API**: Mengerti prinsip dasar pembuatan API yang mendukung komunikasi antar perangkat IoT.
* **Implementasi Laravel 10**: Mempelajari cara instalasi, konfigurasi, dan pembuatan endpoint API menggunakan Laravel 10.
* **Integrasi dengan Ngrok**: Menggunakan ngrok untuk membuat tunnel yang memungkinkan API diakses dari luar jaringan lokal.

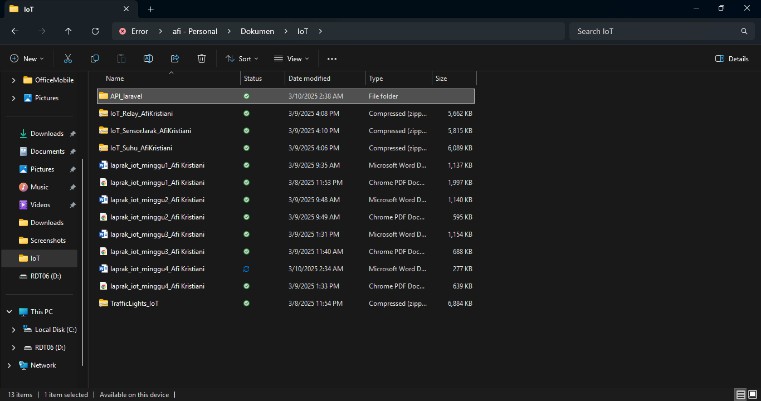
# Metodologi

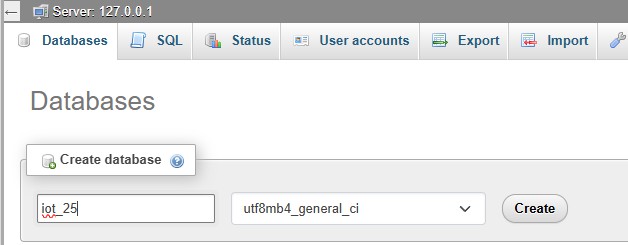
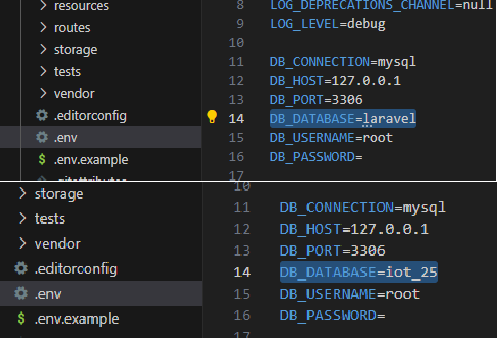
* 1. **Alat dan Bahan**
     1. Laptop
     2. Internet
     3. Web Server Lokal: XAMPP, WAMP, atau LAMP.
     4. Composer: Manajer paket untuk PHP.
     5. Laravel: Framework PHP yang digunakan untuk membangun API.
     6. Ngrok: Tool untuk membuat tunnel ke server lokal.
     7. Text Editor/IDE: Visual Studio Code, Sublime Text, atau sejenisnya.
     8. Postman: Aplikasi untuk menguji API

# Langkah Implementasi

1. Buatlah folder baru di file manager





1. Kemudian Open command prompt(CMD) dan Lakukan perintah untuk create composer
2. Buatlah database MySql
3. Open folder di vscode dan kemudian open file .env untuk melakukan pengeditan nama pada database
4. Buat file model TransaksiSensor.php dengan cara menjalankan perintah di terminal Vscode

### php artisan make:model TransaksiSensor -m

1. Kemudian melakukan pengeditan pada 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;

/\*\*

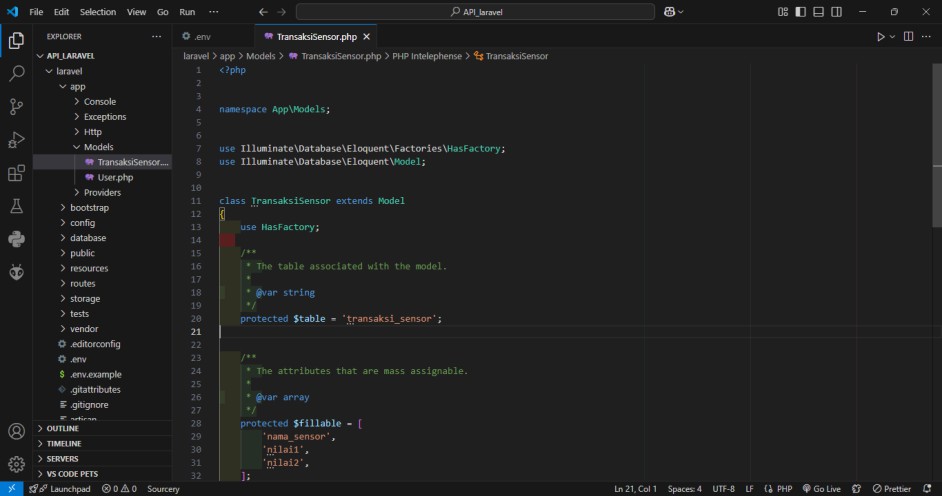
* The table associated with the model.

\*

* @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 = [];

}

1. Setelah itu melakun pengeditan juga pada file **2025\_03\_10\_012536\_create\_transaksi\_sensors\_table.php**

<?php

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

use Illuminate\Support\Facades\Schema;

return new class extends Migration

{

/\*\*

* Run the migrations.

\*/

public function up(): void

{

Schema::create('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.

\*/

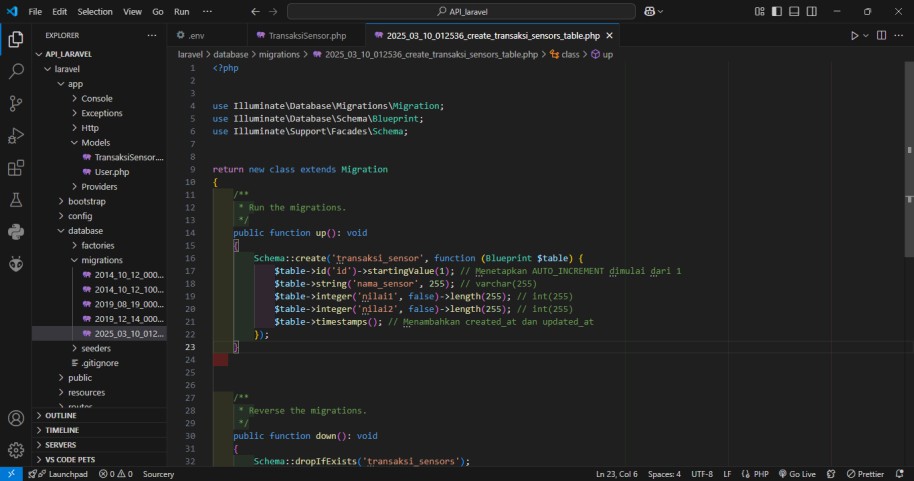
public function down(): void

{

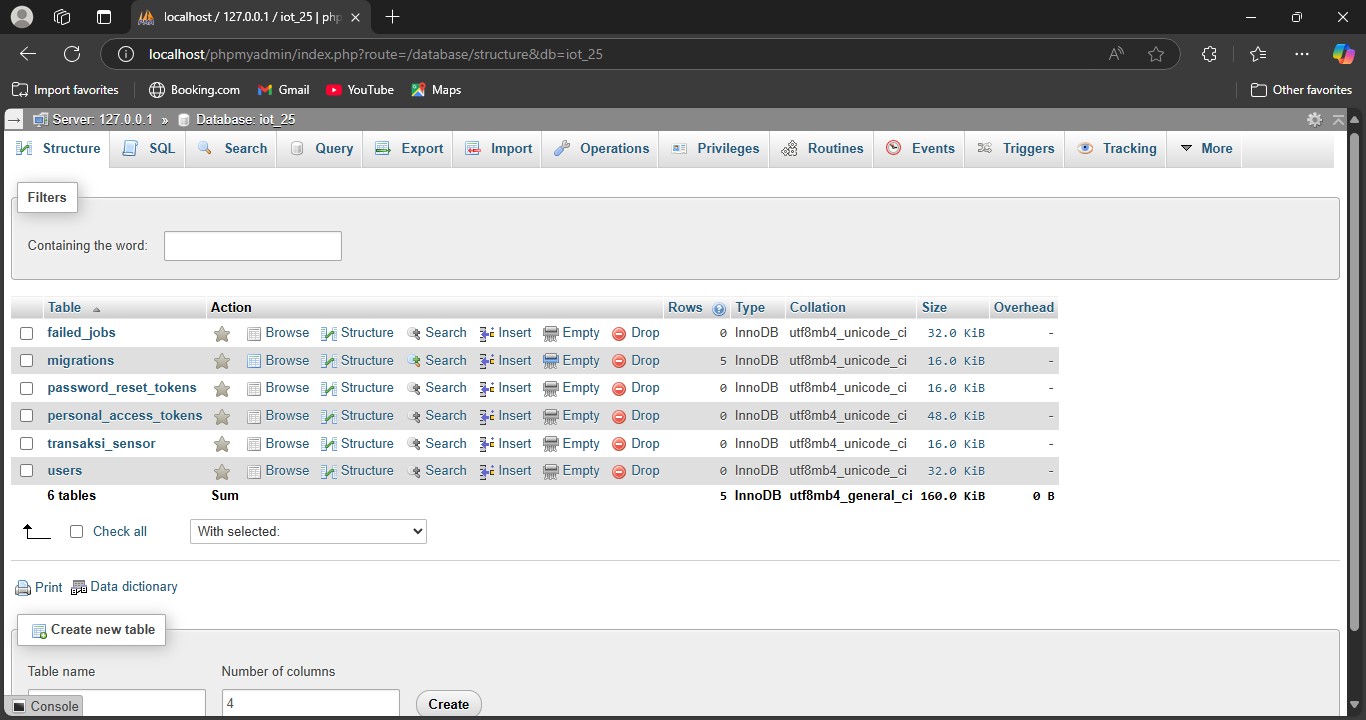
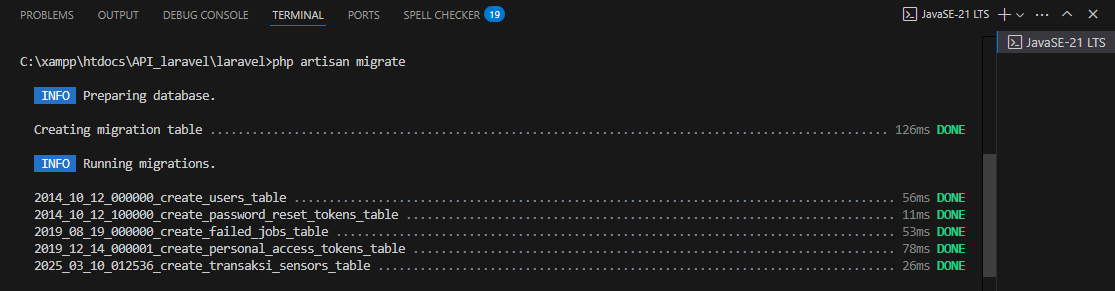
Schema::dropIfExists('transaksi\_sensors');

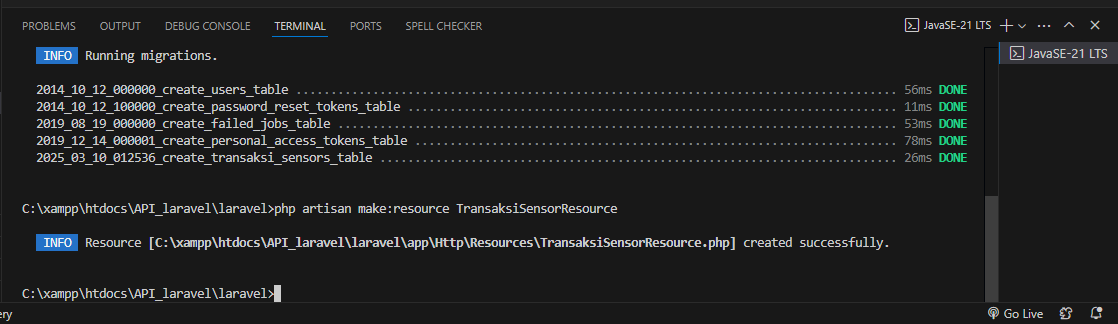
}

};



1. Kemudian lakukan perintah **php artisan migrate** diterminal dan maka pada database akan terinput terdapat table – table data



1. ****Kemudian buatlah resource dengan scema **php artisan make:resource TransaksiSensorResource**
2. Lalu lakukan pada pengeditan pada file **TransaksiSensorResource**

<?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,

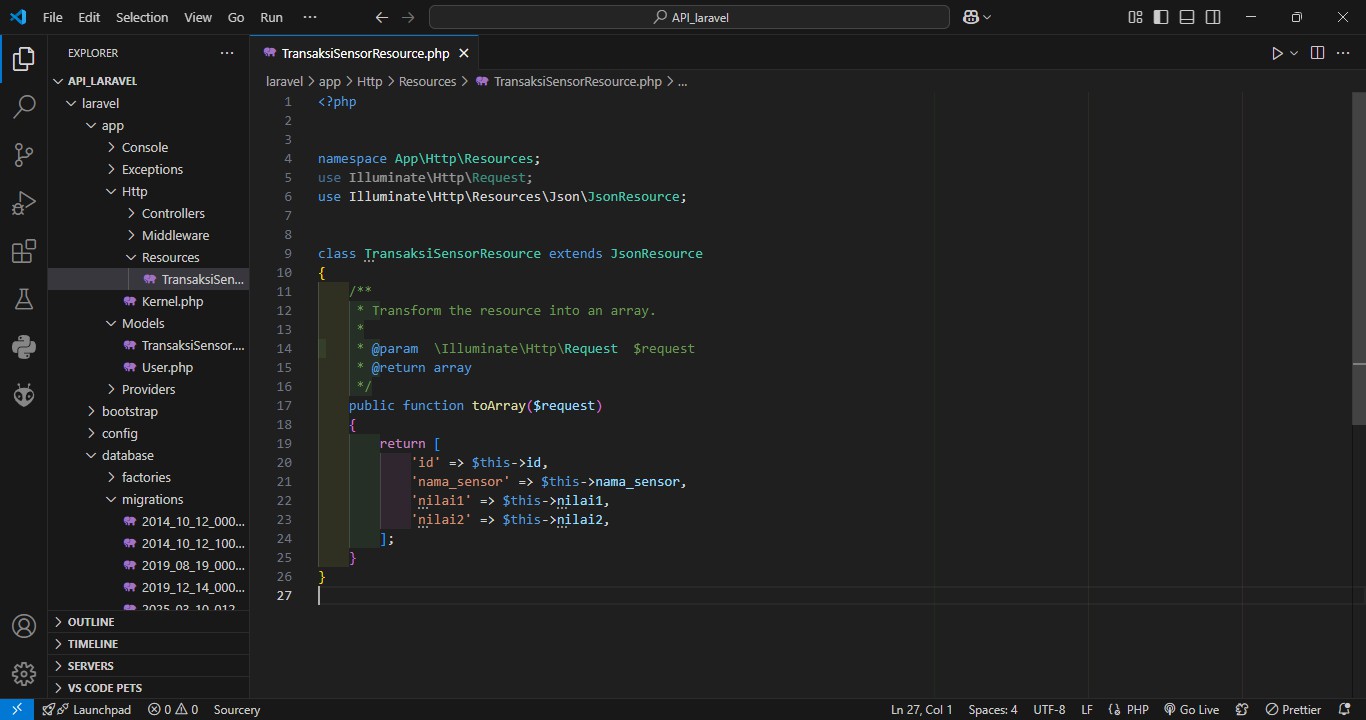
'nama\_sensor' => $this->nama\_sensor, 'nilai1' => $this->nilai1,

'nilai2' => $this->nilai2,

];

}

}



1. Buatlah API controller dengan menjalankan scema

**php artisan make:controller Api/TransaksiSensorController** dan melakukan pengeditan pada 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);

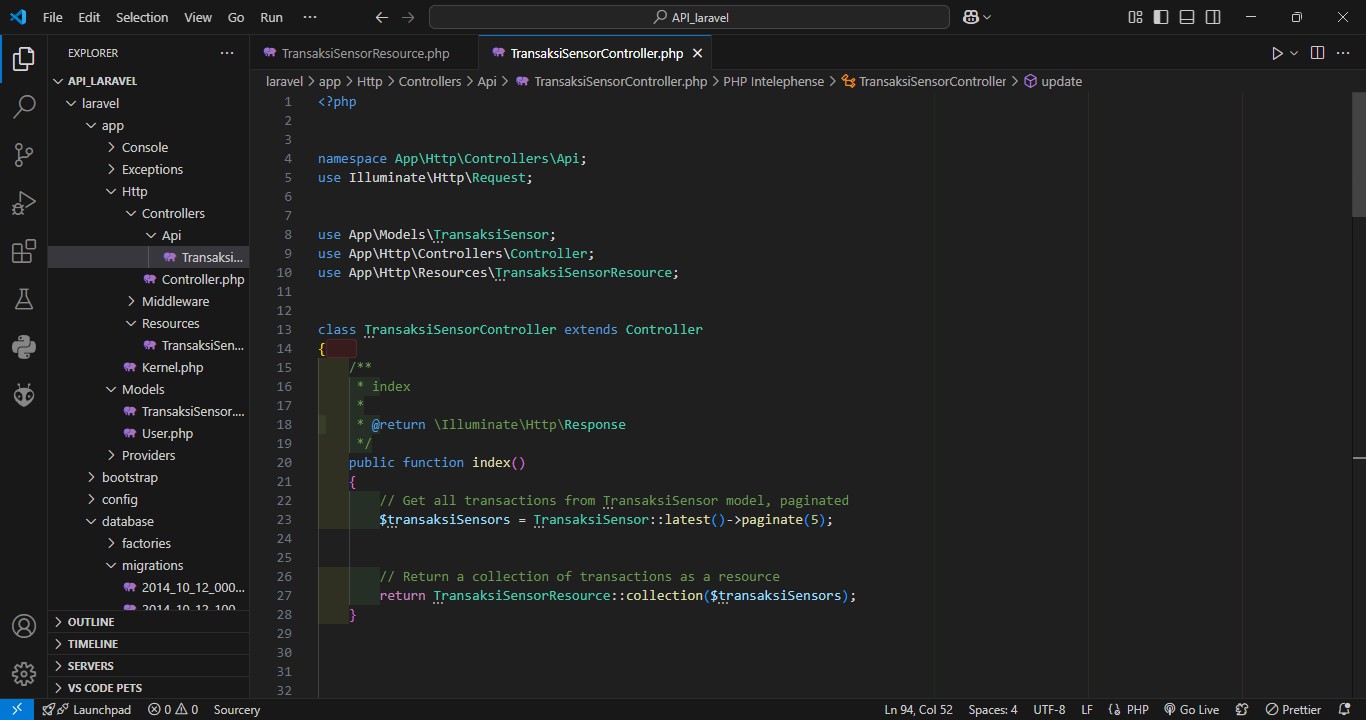
}

/\*\*

* 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);

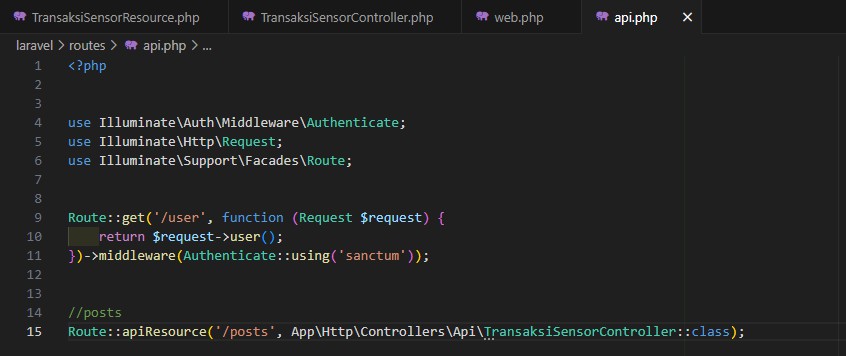
$transaksiSensor->delete();

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

}

}

1. Buat route khusus API dengan menjalankan scema **php artisan make:api** dan melakukan pengeditan file



<?php

use Illuminate\Auth\Middleware\Authenticate; use Illuminate\Http\Request;

use Illuminate\Support\Facades\Route;

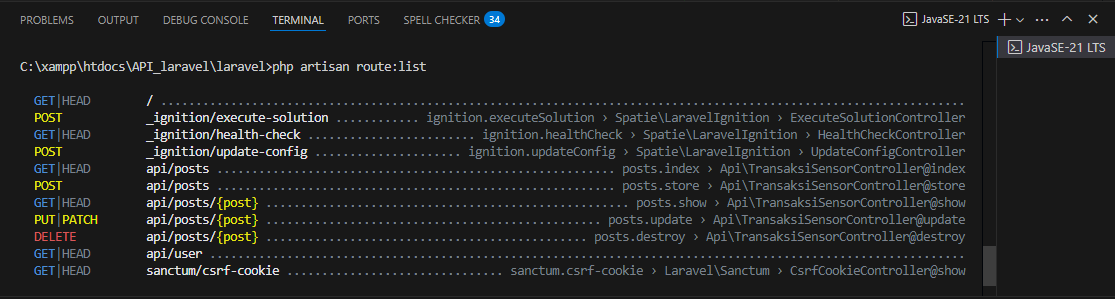
Route::get('/user', function (Request $request) { return $request->user();

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

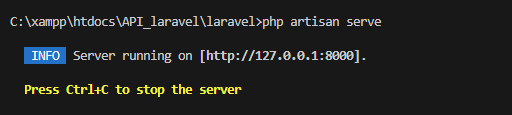
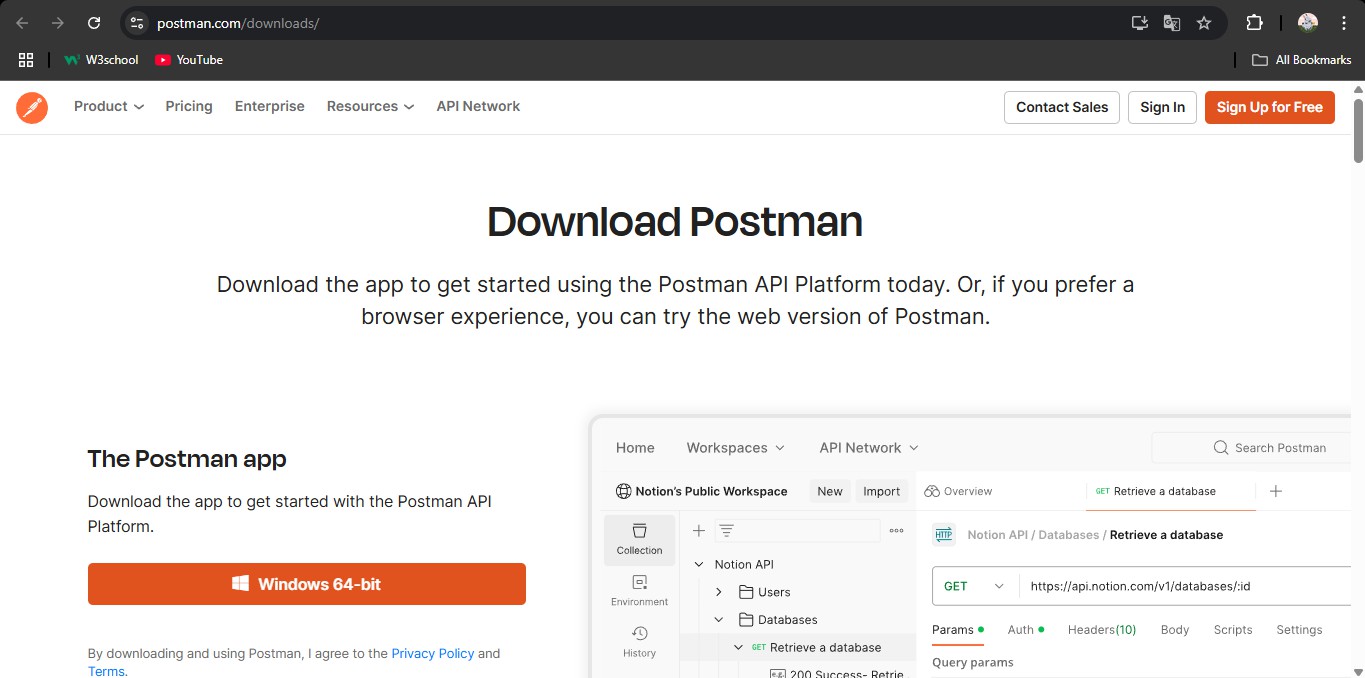
//posts Route::apiResource('/posts',

App\Http\Controllers\Api\TransaksiSensorController::class);

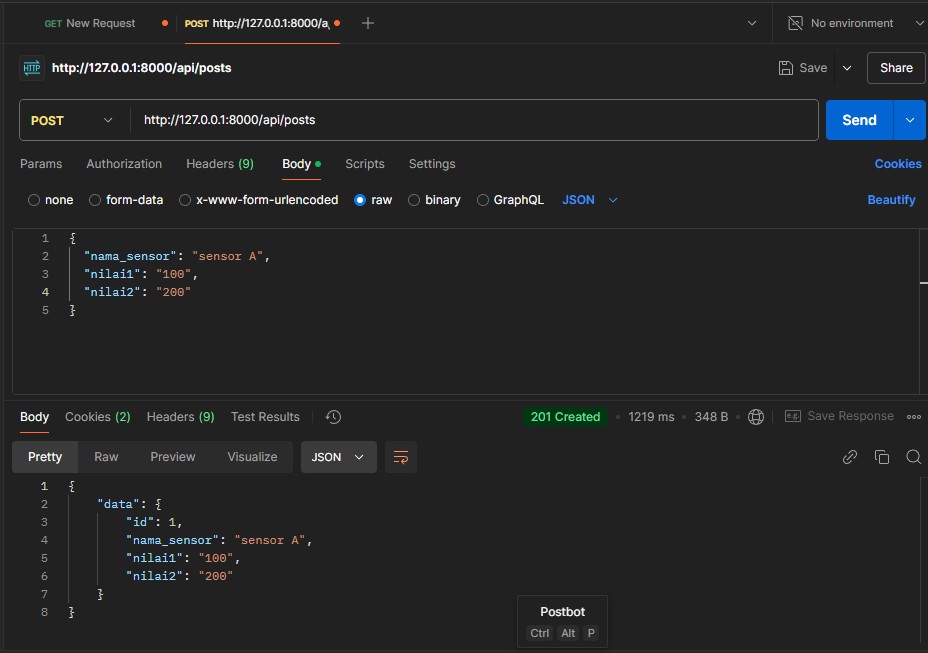
1. Kemudian pastikan routes telah terbentuk dengan menjalankan scema **php artisan route:list**



1. Download POSTMAN dan lakukan scema **php artisan serve** pada terminal



1. Kemudian lakukan pengambilan data melalui postman dengan melakukan scema berikut;

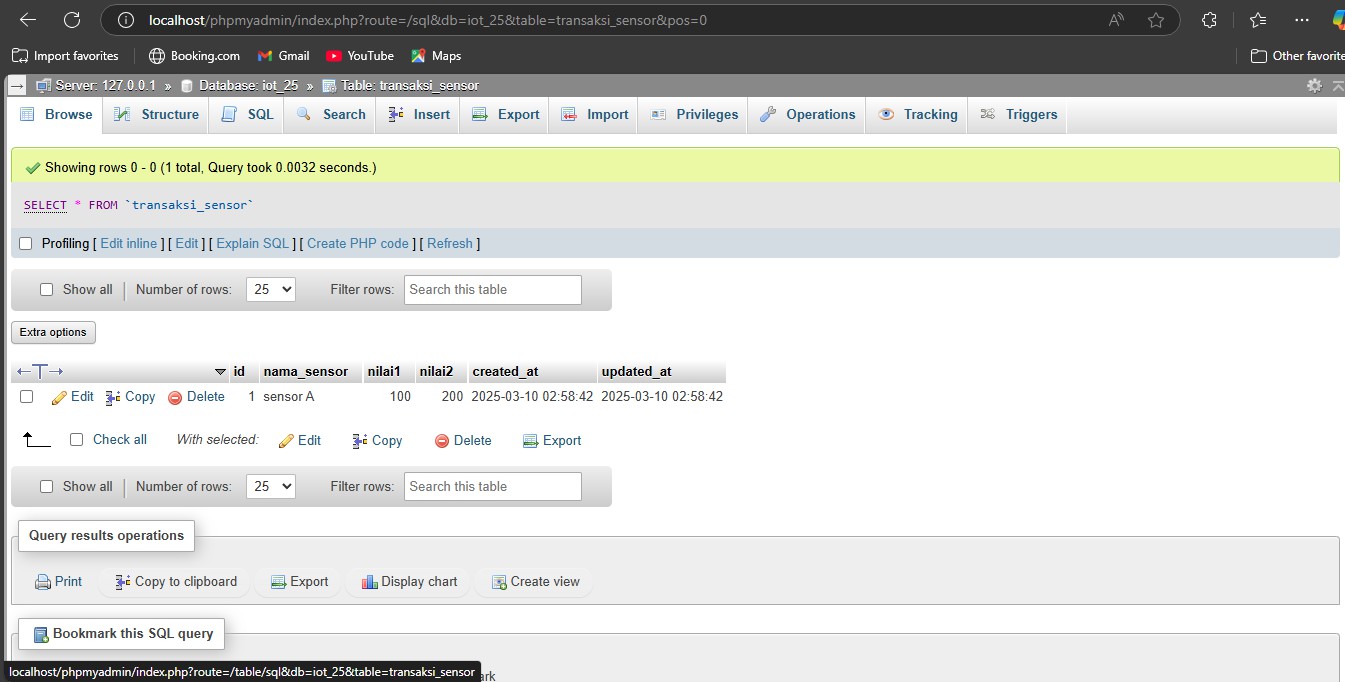


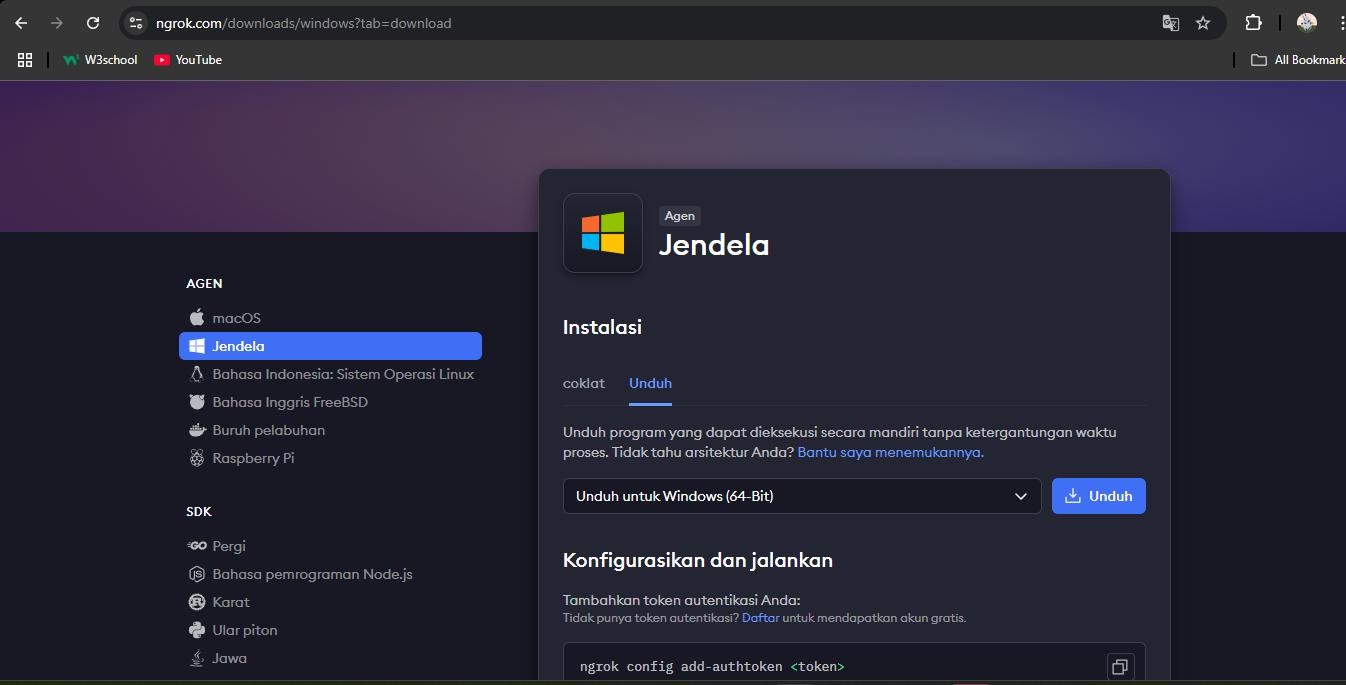
{

"nama\_sensor": "sensor A", "nilai1": "100",

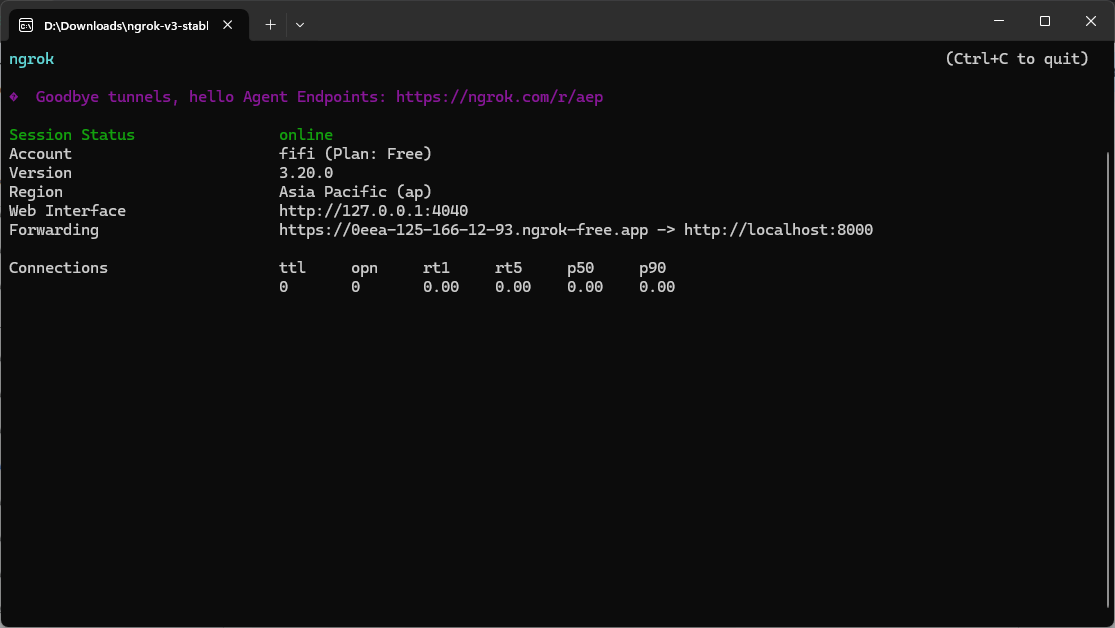
"nilai2": "200"

}

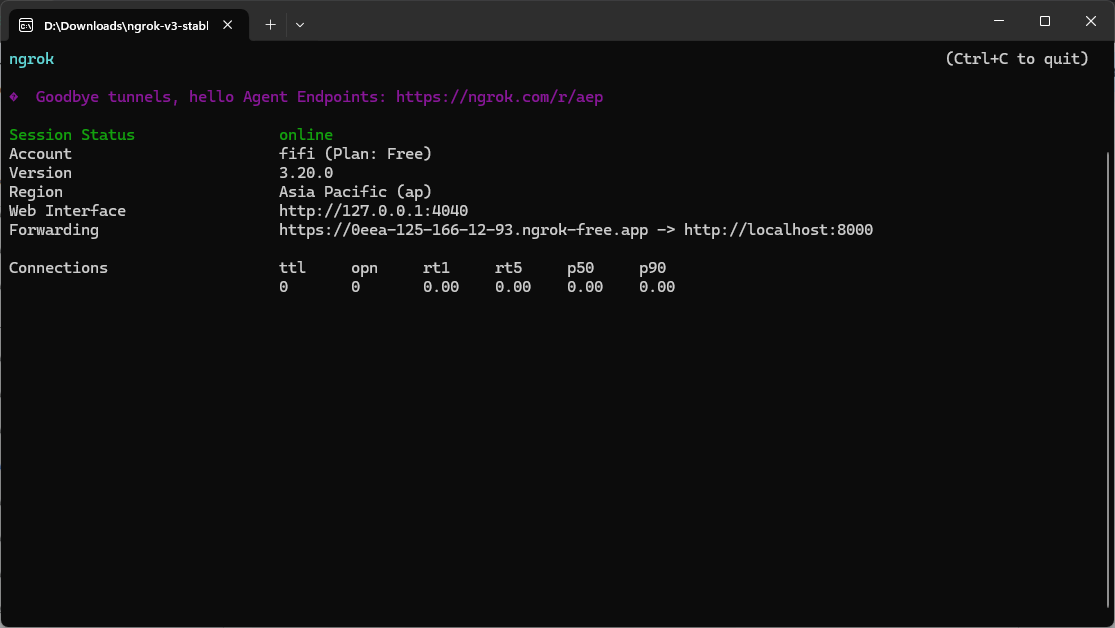
1. Setelah itu lakukan pengecekan database pada localhost
2. Langkah berikutnya adalah mengonline-kan API menggunakan service ngrok sehingga API dapat diakses melalui device iot atau simulasi wokwi iot
   1. Download aplikasi ngrok

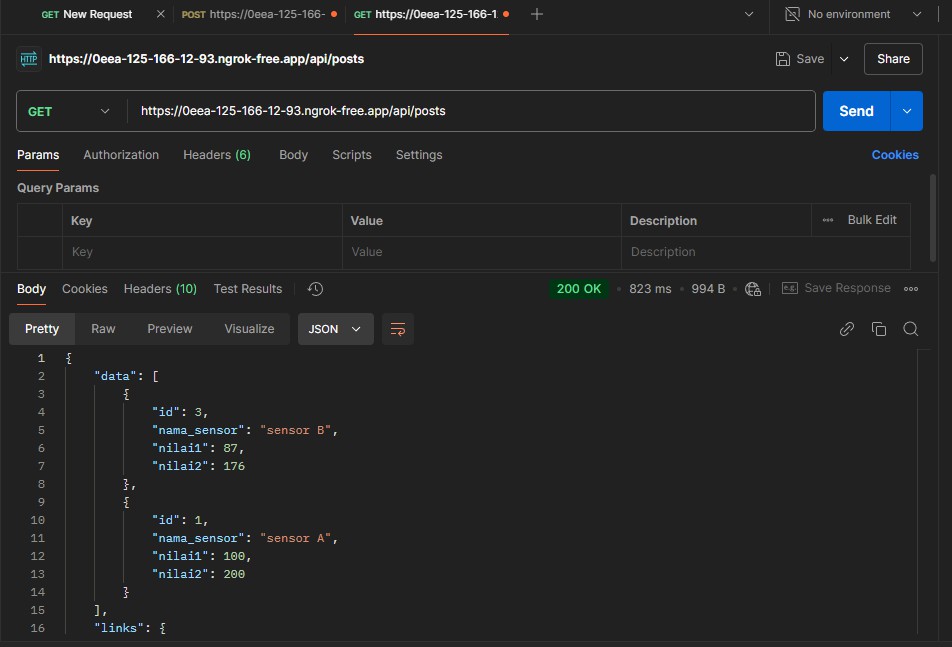


* 1. Kemudian lakukan konfigurasi pada file ngrok yang sudah di ekstrak
  2. Kemudian jalankan perintah berikut untuk mengonline kan laravel melalui port 8000 dengan scema berikut:

**ngrok http** [**http://localhost:8000**](http://localhost:8080/)

* 1. Dengan link yang diberikan oleh ngrok kita dapat melakukan percobaan munggunakan postman denan menanbahkan **/api/posts**



****