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

*Agatha Herma Putra*

*Fakultas Vokasi, Universitas Brawijaya*

*Email: agathaputra24@student.ub.ac.id*

**Abstract**

This report presents the implementation of API creation practice using Laravel 11 and Ngrok. The study aims to explore the development of a RESTful API with Laravel 11 and its integration with Ngrok to facilitate external access for testing and development purposes. The API was built using Laravel’s built-in features for routing, controllers, and middleware. Ngrok was used to expose the local development server to the internet, enabling real-time API testing and interaction with external services. The results indicate that Laravel 11 provides a robust framework for API development, while Ngrok simplifies external connectivity, making it an effective tool for debugging and testing API endpoints before deployment.

*Keywords : Laravel 11, API Development, Ngrok, RESTful API, Web Development, API Testing..*

**1 Introduction**

**1.1 Background**

API development plays a crucial role in modern web applications, enabling communication between different software components and services. Laravel, a popular PHP framework, provides built-in tools for creating secure and efficient APIs. However, testing APIs locally can be challenging when external access is required. This study explores how Laravel 11 can be used to create RESTful APIs and how Ngrok can be leveraged to expose the local server, facilitating easier testing and integration with external applications.

**1.2 Purpose of the experiment**

The primary purpose of this experiment is to develop a RESTful API using Laravel 11 and establish a secure tunnel using Ngrok for external access. This approach allows for efficient API testing, debugging, and interaction with third-party services. Additionally, this study aims to demonstrate the advantages of using Laravel 11’s built-in API development features and Ngrok’s tunneling capabilities.

**2.1 Methodology**

The API development was conducted using Laravel 11, a PHP framework known for its simplicity and efficiency in building web applications. The development environment was set up with Composer, and Laravel’s built-in Artisan commands were used to generate controllers, models, and routes. A RESTful API was implemented to handle HTTP requests, including GET, POST, PUT, and DELETE methods. Ngrok was configured to create a secure tunnel, allowing external access to the local Laravel server. API functionality was tested using tools like Postman and cURL to verify request handling and response accuracy. Debugging and error handling were performed using Laravel’s logging features and real-time monitoring via Ngrok’s interface. Finally, the results were analyzed to evaluate the effectiveness of this approach in API development and testing.

**2.2 Tools & Materials**

To successfully complete the API Creation Practice Using Laravel 11 and Ngrok, several tools and materials are required. These include:

1. Hardware Requirements
   * A computer or laptop with an internet connection
   * A web browser (Google Chrome, Mozilla Firefox, Microsoft Edge, or any other compatible browser)
2. Software & Online Platforms
   * Microsoft Visual Studio Code
   * Xampp
   * Postman
   * Ngrok
   * Laravel 11
   * Database phpMyAdmin
3. Additional Tools (Optional)
   * A text editor (such as Visual Studio Code or Notepad++) for working with GitHub repositories
   * A Git client (such as Git Bash or GitHub Desktop) for version control testing

These tools and materials ensure a smooth and efficient process for creating accounts and exploring the basic functionalities of both platforms.

**2.3 Implemention Steps**

Implementation for experiment making API Creation Practice Using Laravel 11 and Ngrok:

**1. Accessing the Platforms**

* Open a web browser on a computer or laptop.
* Make sure Xampp is running
* Access phpmyadmin database and create a new database with the name iot\_25



Make new Database

**2. Create Project**

* Download the packages needed to start Laravel 11 by typing the following command in the terminal/command prompt.

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

cd laravel-11

**3. Edit Project**

* Change the contents of the .env file configuration.

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

* Create a TransactionSensor.php model file by running the following command in the terminal

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

* Then change the file 2025\_02\_21\_074123\_create\_transaksi\_sensors\_table.php

Which is in the databases-migrations folder

<?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');

    }

};

* Then change the contents of the app/Models/TransaksiSensor.php file

<?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 = [];

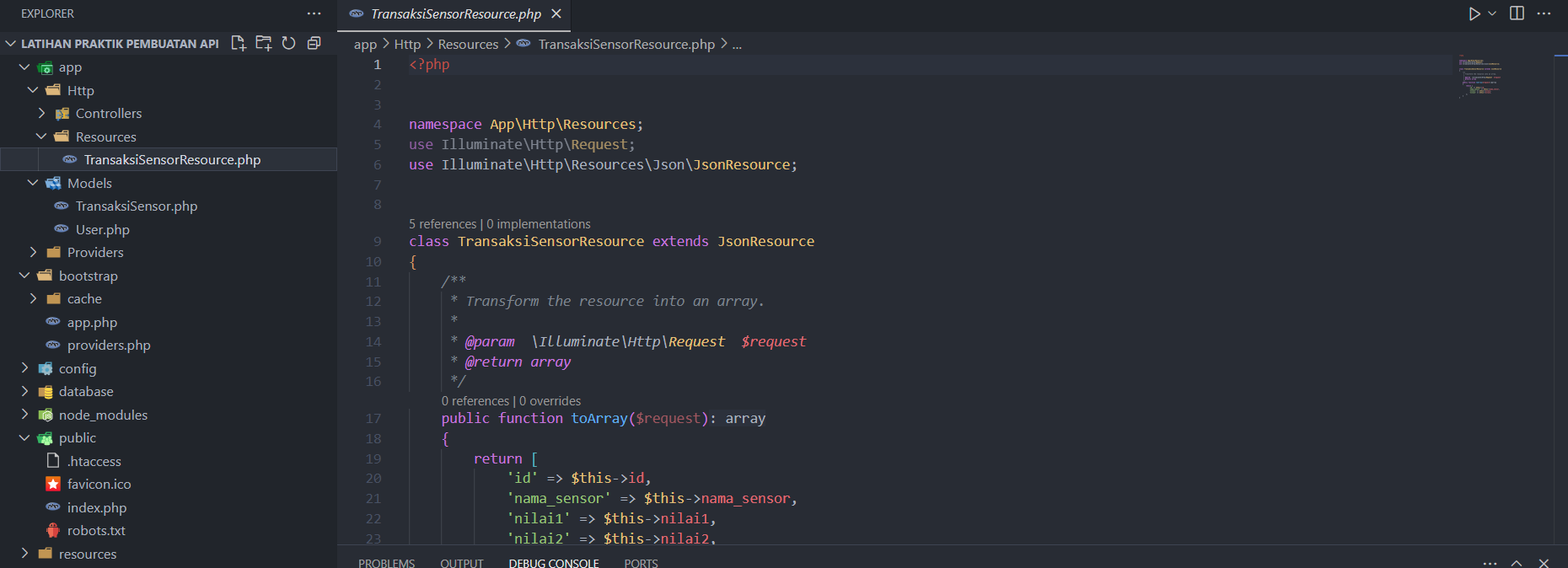
}

* Kemudian jalankan perintah berikut untuk membuat tabel :

php artisan migrate

* Create a Resource by running the command:

php artisan make:resource TransaksiSensorResource



TransaksiSensorResource

* Change the contents of the TransactionSensorResource.php file in the app-Http-Resources folder with the following file contents.

<?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,

];

}

}

* Create an API controller by running the command:

php artisan make:controller Api/TransaksiSensorController

* Change the contents of the file app/Http/Controllers/Api/TransactionSensorController.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);

}

}

* Create a special API route by running the command:

php artisan install:api

* Open the routes/api.php file and change the contents of the file to:

<?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);

* Then make sure the routes have been created by running the command:

php artisan route:list

**4. Testing with Postman and ngrok**

* Open the Postman app.
* To test access to the API, make sure the Laravel application is run with the command:

php artisan serve

* Start Ngrok and expose the local server:

ngrok http 8000

* Copy the generated Ngrok URL (e.g., https://randomid.ngrok.io).
* Open Postman and create a new request:
* Select the HTTP method (GET, POST, PUT, DELETE).
* Enter the API endpoint, replacing localhost:8000 with the Ngrok URL.
* Example: https://randomid.ngrok.io/api/users.
* If authentication is required, add headers (e.g., Authorization: Bearer <token>).
* Click **Send** and verify the response.
* This process allows the API to be tested externally without deploying it to a live server.

**3. Results and Discussion**

**3.1 Experimental Results**

* **Schema Development** The API endpoints were successfully designed and implemented in Laravel 11.
* **Functionality Testing** The API endpoints were tested using Postman and verified to handle HTTP requests correctly.
* **Code Implementation** The API system executed successfully, handling data transactions as expected.
* **Observation Results** The API functioned correctly with external access through Ngrok, allowing smooth communication with external services.
* **Evaluation** No major issues were found, but further improvements could be made to optimize API performance and security.

1. **Appendix**

**A. Links to Official Websites**

The following links were used during the experiment for account registration and platform access:

* **Wokwi**: <https://wokwi.com>
* **GitHub**: <https://github.com>
* **Platform.**io
* **Ngrok**
* **Laravel**
* **Xampp**
* **mysql**

**B. Required System Specifications**

To ensure smooth execution of the experiment, the following system requirements were met:

* **Device**: Laptop with at least 4GB RAM
* **Operating System**: Windows 10 / macOS / Linux
* **Browser**: Google Chrome (Version 100+)
* **Internet Connection**: Stable with a minimum speed of 5 Mbps