TUGAS SIB ARKATAMA MULTI SOLUSINDO
IOT PRAKTIK

ID Kegiatan : 7582873
Nama : Evy Nur Imamah
Kelas : IoT 1

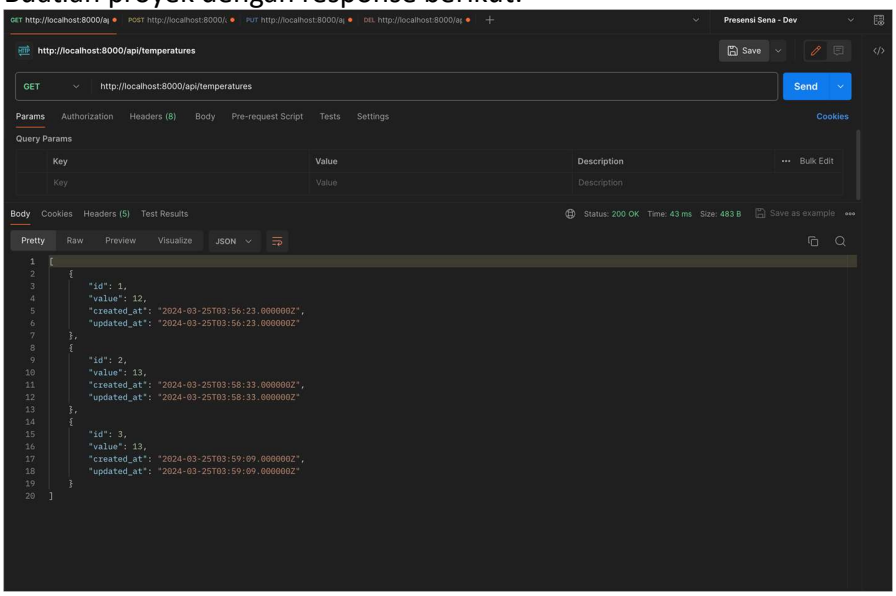
Kumpulkan dalam Format PDF!

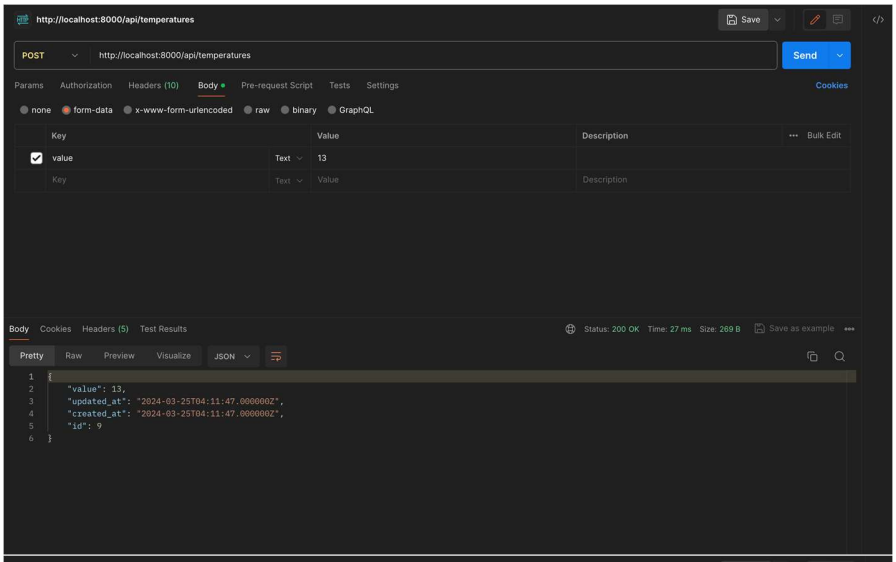
TOOLS!

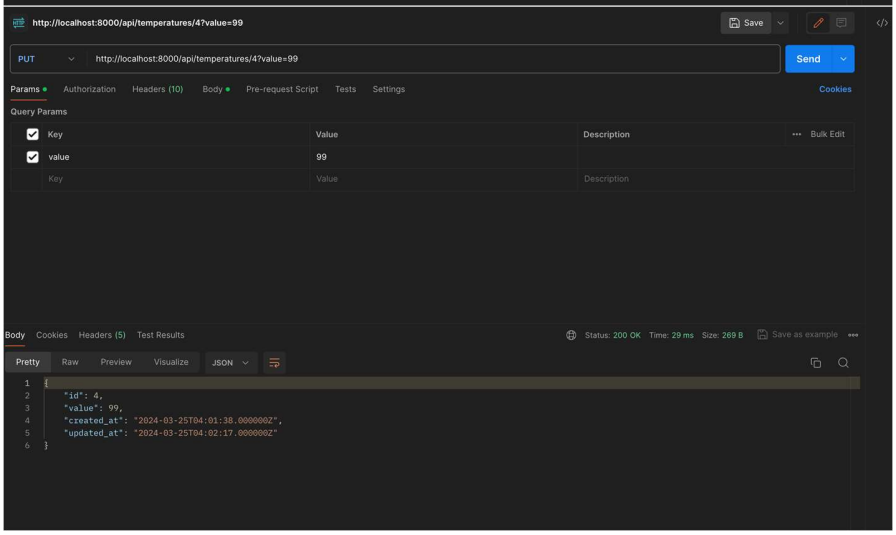
1. https://www.postman.com/
2. https://laragon.org/
3. https://getcomposer.org/
4. https://nodejs.org/en

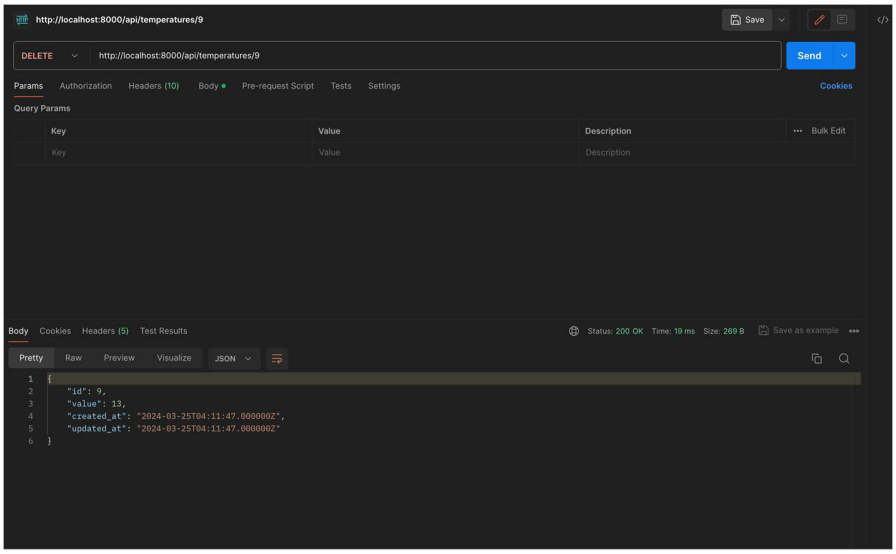
Soal!

1. Buatlah proyek dengan response berikut:




http://localhost:8000/api/temperatures
Save
POST
http://localhost:8000/api/temperatures
Send
Params Authorization Headers (10) Body Pre-request Script Tests Settings
Cookies
none form-data x-www-form-urlencoded raw binary GraphQL
Key Value Description Bulk Edit
value Text 13
Key Text Value Description
Body Cookies Headers (5) Test Results
Status: 200 OK Time: 27 ms Size: 269 B
Save as example
Pretty Raw Preview Visualize JSON
{
"value": 13,
"updated_at": "2024-03-25T04:11:47.000000Z",
"created_at": "2024-03-25T04:11:47.000000Z",
"id": 9
}


http://localhost:8000/api/temperatures/4?value=99
Save
PUT
http://localhost:8000/api/temperatures/4?value=99
Send
Params Authorization Headers (10) Body Pre-request Script Tests Settings
Cookies
Query Params
Key Value Description Bulk Edit
value 99
Key Value Description
Body Cookies Headers (5) Test Results
Status: 200 OK Time: 29 ms Size: 269 B
Save as example
Pretty Raw Preview Visualize JSON
{
"id": 4,
"value": 99,
"created_at": "2024-03-25T04:01:38.000000Z",
"updated_at": "2024-03-25T04:02:17.000000Z"
}


http://localhost:8000/api/temperatures/9
Save
DELETE
http://localhost:8000/api/temperatures/9
Send
Params
Authorization
Headers (10)
Body
Pre-request Script
Tests
Settings
Cookies
Query Params
Key
Value
Description
Bulk Edit
Key
Value
Description
Body
Cookies
Headers (5)
Test Results
Status: 200 OK
Time: 19 ms
Size: 269 B
Save as example
Pretty
Raw
Preview
Visualize
JSON
{
"id": 9,
"value": 13,
"created_at": "2024-03-25T04:11:47.000000Z",
"updated_at": "2024-03-25T04:11:47.000000Z"
}

Deskripsi Proyek:

Proyek ini bertujuan untuk membuat sebuah endpoint API menggunakan framework Laravel yang akan digunakan dalam konteks aplikasi Internet of Things (IoT). Endpoint ini akan memungkinkan pengguna untuk melakukan operasi CRUD (Create, Read, Update, Delete) terhadap data temperatur yang dikumpulkan dari berbagai sensor. Data temperatur akan disimpan dalam sebuah tabel database yang disebut **`temperatures`**.

Spesifikasi Proyek:

a. Endpoint API /api/temperatures:
    - Menyediakan operasi CRUD untuk data temperatur.
    - Menggunakan resources Laravel untuk membuat endpoint.
b. Operasi CRUD:
    - Create: Membuat data temperatur baru dengan mengirimkan nilai temperatur melalui permintaan POST.
    - Read: Menampilkan daftar semua data temperatur atau detail data temperatur berdasarkan ID melalui permintaan GET.
    - Update: Memperbarui data temperatur berdasarkan ID dengan mengirimkan nilai temperatur yang baru melalui permintaan PUT atau PATCH.
    - Delete: Menghapus data temperatur berdasarkan ID melalui permintaan DELETE.
c. Spesifikasi Teknis:
    - Nama tabel database: temperatures.
    - Kolom pada tabel:
        - id (Primary Key, Auto Increment)
        - value (Nilai temperatur)
        - created_at (Tanggal dan waktu pembuatan data)
        - updated_at (Tanggal dan waktu terakhir pembaharuan data)

Langkah-langkah Proyek:

1. Persiapan Lingkungan Pengembangan:
    - Jalan Laragon.
    - Persiapkan proyek IoT yang sudah diberikan pada Mentoring Sebelumnya dan menjalankan proyek tersebut dengan **`php artisan serve`**.
2. Desain Database:
    - Buatlah migrasi database untuk tabel temperatures dengan perintah **`php artisan make:migration create_temperatures_table`**.
    - Tentukan struktur tabel dan kolom yang sesuai dengan spesifikasi proyek.
    - Jalankan migrasi untuk membuat tabel di database dengan perintah **`php artisan migrate`**.
3. Buat Endpoint API:
    - Buatlah model **`Temperature`** menggunakan artisan command **`` `php artisan make:model Temperature` ``**.
    - Definisikan struktur model **`Temperature`** dengan menambahkan atribut-atribut yang sesuai dengan kolom pada tabel **`temperatures`**.
    - Buatlah resource controller untuk mengelola data temperatur dengan perintah **`` `php artisan make:controller -r api/TemperatureController` ``**.
    - Implementasikan logika CRUD dalam controller yang telah dibuat.

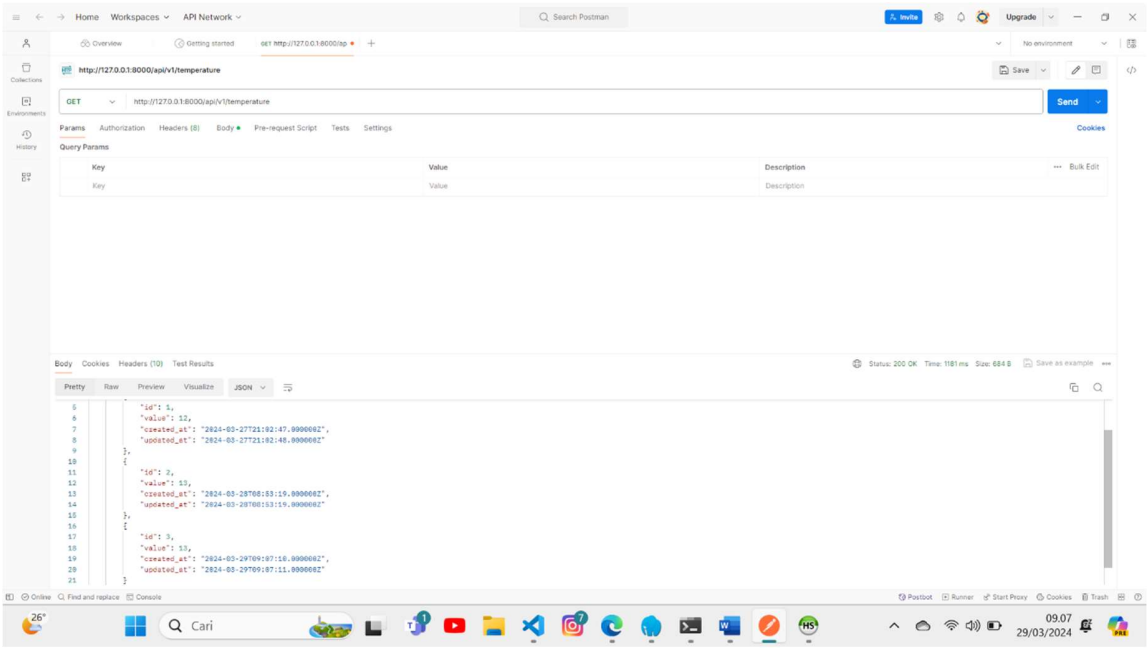
- Atur rute untuk endpoint API dalam file routes/api.php.

4. Validasi Input:
   - Pastikan untuk melakukan validasi input untuk setiap operasi CRUD menggunakan fitur validasi Laravel.
   - Gunakan middleware untuk memastikan data yang masuk ke endpoint API telah divalidasi dengan benar.
5. Testing Endpoint API:
   - Uji setiap operasi CRUD pada endpoint API menggunakan perangkat lunak pengujian Postman.
   - Pastikan endpoint berfungsi dengan baik dan sesuai dengan spesifikasi proyek.

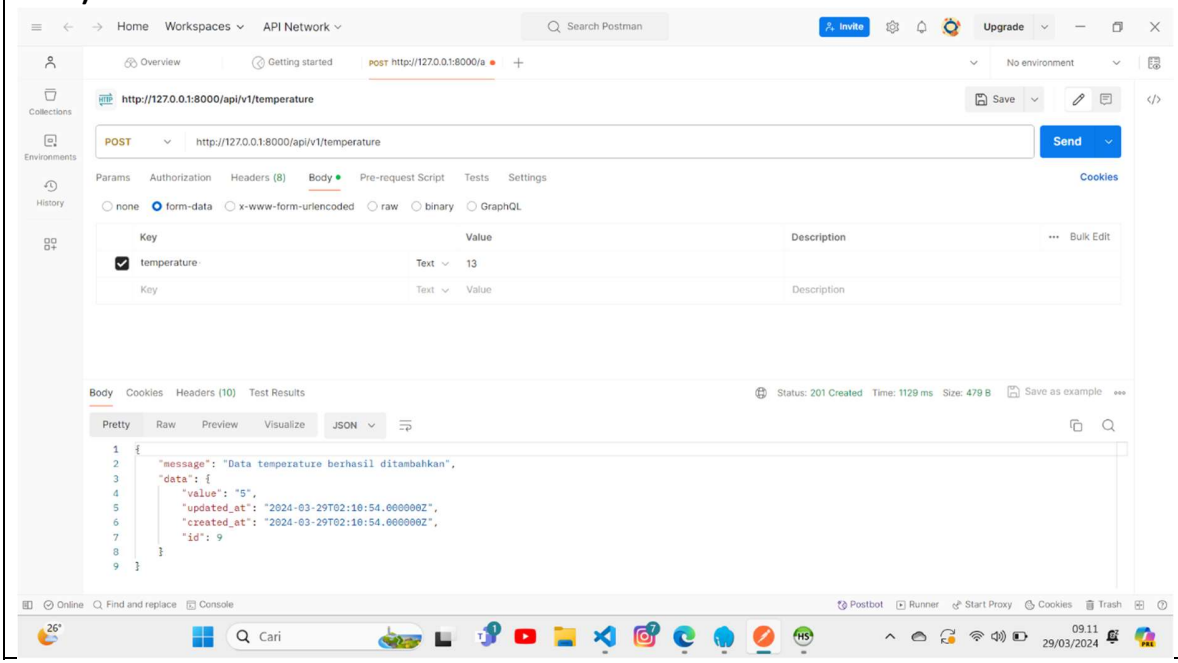
Jawab!

1. Screenshoot Postman (*gambar/screenshoot)

a) Get



b) Post



2. Kode Program Migrasi (.php)

```php
<?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('temperature_controllers', function (Blueprint
$table) {
            $table->id();
            $table->timestamps();
        });
    }

    /**
     * Reverse the migrations.
     */
    public function down(): void
    {
        Schema::dropIfExists('temperature_controllers');
    }
};
```

3. Kode Program Model (.php)

```php
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Model;
use App\Http\Controllers\PostController;

class Temperature extends Model
{
    use HasFactory;
    // protected $guarded = [];
    protected $fillable=['value'];
}
```

4. Kode Program Controller (.php)

```php
<?php
```

```php
namespace App\Http\Controllers\Api;

use App\Http\Controllers\Controller;
use App\Models\Temperature;
// use Dotenv\Validator;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Validator;

class TemperatureController extends Controller
{
    function getTemperature()
    {
        // variabel di php tidak perlu tipe data,
        // nama variable diawali tanda $
        // mengambil semua data temperature
        $temperature = Temperature::all();
        // mengambalikan response dalam bentuk JSON
        // dan status code 200
        return response()->json([
            "message"   => "Data temeperature berhasil diambil",
            "data"      => $temperature
        ], 200);
    }

    function insertTemperature(Request $request)
    {
        //1. Mengambil data request
        $value = $request->temperature;

        //2.Menyimpan data requeat ke database
        $temperature = Temperature::create([
            'value' => $value
        ]);

        //3. Mengembalikan response json
        //dengan status code 200/201
        return response()->json([
            "message" => "Data temperature berhasil ditambahkan",
            "data"    => $temperature
        ], 201);
    }

    public function deleteTemperature(Request $request)
    {
```

```
        Temperature::findorfail($request->id)->delete();

        return response()->json([
            'success' => true,
            "message" => "Data berhasil dihapus"
        ], 201);
    }

    public function putTemperature(Request $request)
    {
        $temperature = Temperature::findOrFail($request->id);

        $temperature->value = $request->value;

        $temperature->update();

        return response()->json([
            "message" => "Data berhasil diubah",
            "data"    => $temperature
        ], 201);
    }
}
```

5. Kode Program Route (.php)

```
<?php

use App\Http\Controllers\Controller;
use App\Http\Controllers\PostController;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Route;
use App\Http\Controllers\Api\AuthController;
use App\Http\Controllers\Api\SensorController;
use App\Http\Controllers\Api\LampController;
use App\Http\Controllers\Api\SensorLogController;
use App\Http\Controllers\Api\UserController;
use App\Http\Controllers\Api\DeviceController;
use App\Http\Controllers\Api\TemperatureController;
/*
|--------------------------------------------------------------------------
------
| API Routes
|--------------------------------------------------------------------------
------
|
```

```
| Here is where you can register API routes for your application.
These
| routes are loaded by the RouteServiceProvider and all of them will
| be assigned to the "api" middleware group. Make something great!
|
*/

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

Route::get('v1/temperature', [TemperatureController::class,
'getTemperature']);
Route::post('v1/temperature', [TemperatureController::class,
'insertTemperature']);
Route::put('v1/temperature', [TemperatureController::class,
'putTemperature']);
Route::delete('v1/temperature', [TemperatureController::class,
'deleteTemperature']);
```