

LAPORAN PRAKTIKUM INTERNET OF THINGS (IoT)

Fakultas Vokasi, Universitas Brawijaya

Praktik Pembuatan API Menggunakan Laravel 11 dan Ngrok Dan Praktik Akses API Melalui Simulasi WOKWI

Intan Tania

Fakultas Vokasi, Universitas Brawijaya

Email: Intantania2412@gmail.com

ABSTRAK

The development of Internet of Things (IoT) technology enables devices to communicate over the internet, with the Application Programming Interface (API) serving as a bridge between hardware and servers. In this practical experiment, an API was developed using Laravel 11 and tested for accessibility through WOKWI simulation to understand API development, assess communication with IoT devices, and evaluate the transmission of sensor data to a database. Ngrok was used to enable public access to the API. The experimental results indicate that the API effectively handles GET and POST requests and successfully integrates ESP32 and the DHT22 sensor in the WOKWI simulation, allowing real-time temperature and humidity data transmission and storage. This success demonstrates that a Laravel-based API can be utilized in cloud-based remote monitoring systems.

Keywords — *Internet of Things (IoT), API, Laravel 11, ESP32, Sensor DHT22*

1. Introduction

1.1 Latar Belakang

Perkembangan Internet of Things (IoT) telah membawa perubahan signifikan dalam berbagai aspek kehidupan, terutama dalam bidang industri, kesehatan, transportasi, dan rumah pintar. IoT memungkinkan perangkat untuk saling berkomunikasi melalui jaringan internet, memungkinkan pemantauan dan otomatisasi yang lebih efisien. Salah satu elemen penting dalam penerapan IoT adalah Application Programming Interface (API), yang berperan sebagai perantara antara perangkat IoT dan server untuk bertukar data secara real-time.

Dalam sistem IoT yang ideal, perangkat sensor dapat mengirimkan data secara otomatis ke server melalui API yang aman dan dapat diakses dari mana saja. Namun, pada kenyataannya, masih banyak kendala dalam pengembangan API untuk sistem IoT, terutama dalam aspek aksesibilitas, keamanan, dan integrasi dengan perangkat keras. Salah satu tantangan utama adalah bagaimana membuat API yang dapat diakses oleh perangkat IoT dari jaringan eksternal tanpa harus mengonfigurasi jaringan yang kompleks.

Seiring dengan meningkatnya kebutuhan akan sistem IoT yang efisien, diperlukan metode yang lebih praktis dan fleksibel dalam pengembangan API. Laravel 11, sebagai salah satu framework PHP yang populer, menawarkan berbagai fitur yang mendukung pengelolaan API secara lebih efisien. Selain itu, Ngrok dapat digunakan untuk membuka akses API secara publik tanpa harus mengubah konfigurasi jaringan secara manual. Dengan adanya alat ini, API yang dikembangkan dapat diuji dan diakses dari berbagai perangkat IoT, termasuk yang berada di luar jaringan lokal.

Untuk menguji integrasi antara API dan perangkat IoT, diperlukan simulasi perangkat keras agar eksperimen dapat dilakukan tanpa harus menggunakan perangkat fisik secara langsung. Dalam praktikum ini, wokwi digunakan sebagai simulator untuk ESP32 dan sensor DHT22, yang bertugas membaca data suhu dan kelembaban, lalu mengirimkan data tersebut ke server melalui API Laravel. Dengan menggunakan wokwi, pengujian dapat dilakukan secara virtual tanpa perlu membeli perangkat keras, sehingga lebih efisien dalam hal biaya dan waktu.

1.2 Tujuan Eksperimen

1. Mempelajari cara membuat API menggunakan Laravel.
2. Memahami penggunaan Ngrok untuk mengakses API.
3. Menguji akses API melalui simulasi perangkat IoT menggunakan WOKWI.
4. Mengevaluasi keberhasilan komunikasi antara API dan perangkat simulasi.

2. Methodology (Metodologi)

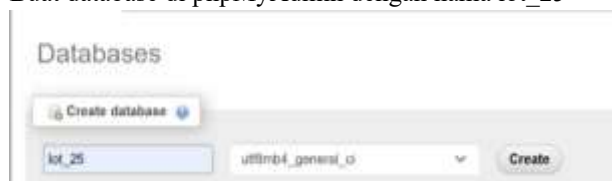
2.1 Tools & Materials (Alat dan Bahan)

- Laravel
- Ngrok
- Wokwi
- Visual Studio Code
- Postman
- PlatfromIO

2.2 Implementation Steps (Langkah Implementasi)

1. Buat program Laravel 11 dengan nama larvael_iot dengan mengetikan perintah berikut di terminal
composer create-project --prefer-dist laravel/laravel laravel_iot

2. Buat database di phpMyAdmin dengan nama `iot_25`



3. Ubah isi konfigurasi pada file .env

```
23 DB_CONNECTION=mysql
24 DB_HOST=127.0.0.1
25 DB_PORT=3306
26 DB_DATABASE=iot_25
27 DB_USERNAME=root
28 DB_PASSWORD=caberg2016
29
30 DB_CHARSET=utf8mb4
31 DB_COLLATION=utf8mb4_unicode_ci
```

4. Buat file model **TransaksiSensor.php** dengan cara menjalankan perintah berikut di terminal



5. Ubah file 2025_03_09_015135_create_transaksi_sensors_table.php yang ada di dalam folder databases/migration

```
2025_03_09_015135_create_transaksi_sensors_table.php X
laravel_20 > database > migrations > 2025_03_09_015135_create_transaksi_sensors_table.php
1  <?php
2
3  use Illuminate\Database\Migrations\Migration;
4  use Illuminate\Database\Schema\Blueprint;
5  use Illuminate\Support\Facades\Schema;
6
7  return new class extends Migration
8  {
9      public function up(): void
10     {
11         Schema::create('transaksi_sensor', function (Blueprint $table) {
12             $table->id->startingValue(1); // Menetapkan AUTO INCREMENT dimulai dari 1
13             $table->string('nama_sensor', 255); // varchar(255)
14             $table->integer('nilai1', false)->length(255); // int(255)
15             $table->integer('nilai2', false)->length(255); // int(255)
16             $table->timestamps(); // Menambahkan created_at dan updated_at
17         });
18     }
19
20     public function down(): void
21     {
22         Schema::dropIfExists('transaksi_sensors');
23     }
24 }
```

6. Ubah isi file **app/Models/TransaksiSensor.php**

```

TransaksiSensor.php X
laravel_lot > app > Models > TransaksiSensor.php
1  <?php
2
3  namespace App\Models;
4
5  use Illuminate\Database\Eloquent\Factories\HasFactory;
6  use Illuminate\Database\Eloquent\Model;
7
8  class TransaksiSensor extends Model
9  {
10     use HasFactory;
11
12     /**
13      * The table associated with the model.
14      *
15      * @var string
16      */
17     protected $table = 'transaksi_sensor';
18
19     /**
20      * The attributes that are mass assignable.
21      *
22      * @var array
23      */
24     protected $fillable = [
25         'nama_sensor',
26         'nilai1',
27         'nilai2',
28     ];
29
30     /**
31      * The attributes that should be hidden for arrays.
32      *
33      * @var array
34      */
35     protected $hidden = [];

```

7. Kemudian jalankan perintah berikut di terminal

```

PS C:\Users\tania\OneDrive\Documents\SDPE3784\Int\laravel_lot> php artisan migrate
Migrating...
Preparing database...
Creating migration table ..... 300.01ms DONE
Running migrations...
0001_01_01_000000_create_users_table ..... 90.92ms DONE
0001_01_01_000001_create_cache_table ..... 14.37ms DONE
0001_01_01_000002_create_jobs_table ..... 71.17ms DONE
2025_01_06_010133_create_transaksi_sensor_table ..... 12.14ms DONE

```

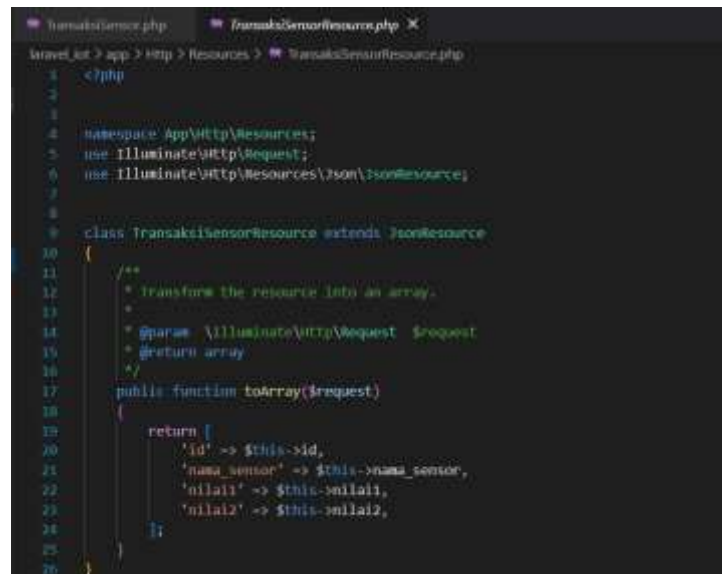
8. Buat Resource dengan menjalankan perintah di terminal
php artisan make:resource TransaksiSensorResource

```

PS C:\Users\tania\OneDrive\Documents\SDPE3784\Int\laravel_API> php artisan make:resource TransaksiSensorResource
*
Open file in editor (ctrl + click)
DONE Resource [C:\Users\tania\OneDrive\Documents\SDPE3784\Int\laravel_API\app\Http\Resources\TransaksiSensorResource.php] created successfully.

```

9. Ubah isi file **TransaksiSensorResource.php** yang ada di folder app-Http-Resources dengan isi file berikut



```

1 <?php
2
3
4 namespace App\Http\Resources;
5 use Illuminate\Http\Request;
6 use Illuminate\Http\Resources\Json\JsonResource;
7
8
9 class TransaksiSensorResource extends JsonResource
10 {
11     /**
12      * Transform the resource into an array.
13      *
14      * @param \Illuminate\Http\Request $request
15      * @return array
16      */
17     public function toArray($request)
18     {
19         return [
20             'id' => $this->id,
21             'nama_sensor' => $this->nama_sensor,
22             'nilai1' => $this->nilai1,
23             'nilai2' => $this->nilai2,
24         ];
25     }
26 }

```

10. Buat API controller dengan menjalankan perintah berikut di terminal
php artisan make:controller Api/TransaksiSensorController

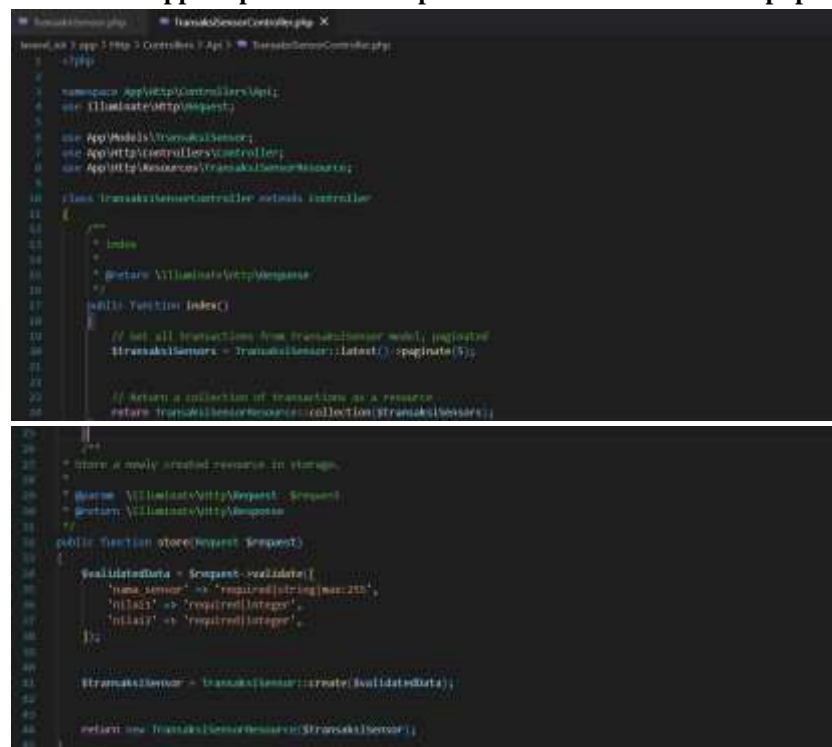


```

PS C:\Users\tania\OneDrive\Documents\SEMESTER4\Inf\laravel_API> php artisan make:controller Api/TransaksiSensorController

```

11. Ubah isi file **app/Http/Controllers/Api/TransaksiSensorController.php**



```

1 <?php
2
3 namespace App\Http\Controllers\Api;
4 use Illuminate\Http\Request;
5
6 use App\Models\TransaksiSensor;
7 use App\Http\Controllers\Controller;
8 use App\Http\Resources\TransaksiSensorResource;
9
10 class TransaksiSensorController extends Controller
11 {
12     /**
13      * index
14      *
15      * @return \Illuminate\Http\Response
16      */
17     public function index()
18     {
19         // get all transactions from transaksiSensor model; paginated
20         $transaksiSensors = TransaksiSensor::latest()->paginate(5);
21
22         // Return a collection of transactions as a resource
23         return TransaksiSensorResource::collection($transaksiSensors);
24     }
25
26     /**
27      * Store a newly created resource in storage.
28      *
29      * @param \Illuminate\Http\Request $request
30      * @return \Illuminate\Http\Response
31      */
32     public function store(Request $request)
33     {
34         $validatedData = $request->validate([
35             'nama_sensor' => 'required|string|max:255',
36             'nilai1' => 'required|integer',
37             'nilai2' => 'required|integer',
38         ]);
39
40         $transaksiSensor = TransaksiSensor::create($validatedData);
41
42         return new TransaksiSensorResource($transaksiSensor);
43     }
44 }

```

12. Buat route khusus API dengan menjalankan perintah berikut di terminal

```

88 packages you are using are looking for funding.
Use the "composer fund" command to find out more!
> @php artisan vendor:publish --tag=laravel-assets --ansi --force

[INFO] No publishable resources for tag [laravel-assets].

Found 1 security vulnerability advisory affecting 1 package.
[INFO] No publishable resources for tag [laravel-assets].

[INFO] No publishable resources for tag [laravel-assets].
[INFO] No publishable resources for tag [laravel-assets].

Found 1 security vulnerability advisory affecting 1 package.
Run "composer audit" for a full list of advisories.
[INFO] Published API routes file.

One new database migration has been published. Would you like to run all pending database migrations? (yes/no)
[yes]:
>

[INFO] Running migrations.

2025_03_07_134985_create_personal_access_tokens_table ..... 51.1ms DONE

[INFO] API scaffolding installed. Please add the [Laravel\Sanctum\HasApiTokens] trait to your User model.

```

- ```

1 1 |> php
2
3
4 use Illuminate\Auth\Middleware\Authenticate;
5 use Illuminate\Http\Request;
6 use Illuminate\Support\Facades\Route;
7
8
9 Route::get('/user', function (Request $request) {
10 return $request->user();
11 })->middleware(Authenticate::using('sanctum'));
12
13
14 //posts
15
16 Route::apiResource('posts', App\Http\Controllers\Api\TransaksiSensorController::class);

```

- [illegible]

- ```
PS C:\Users\tania\OneDrive\Documents\SEMESTER4\IoT\laravel_iot> php artisan serve
```
- INFO** Server running on [http://127.0.0.1:8000].
- Press Ctrl+C to stop the server

16. Masukkan data kedalam tabel di database. Berikut elah ada 2 baris data pada tabel transaksi sensor pada database iot 25

Showing rows 0 - 1 (2 total. Query took 0.0004 seconds.)

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

☐ Polling

☐ 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
<input type="checkbox"/> Edit <input type="button" value="Copy"/> <input type="button" value="Delete"/>	2	SensorA	190	200	2025-03-07 20:55:40	2025-03-07 20:55:40
<input type="checkbox"/> Edit <input type="button" value="Copy"/> <input type="button" value="Delete"/>	3	SensorB	87	176	2025-03-07 20:55:40	2025-03-07 20:55:40

17. Buka aplikasi postman dan jalankan untuk mulai melakukan akses api
18. Kemudian Pada bagian URL masukkan alamat server laravel <http://127.0.0.1:8000/api/posts>
Pilih method **GET** untuk mengambil data dari database, kemudian klik tombol **SEND**

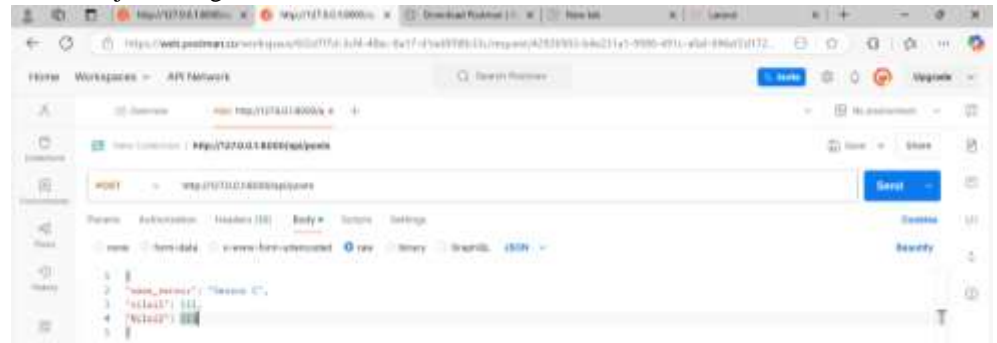
```

{
  "data": [
    {
      "id": 2,
      "nama_sensor": "SensorA",
      "nilai1": 190,
      "nilai2": 200
    },
    {
      "id": 3,
      "nama_sensor": "SensorB",
      "nilai1": 87,
      "nilai2": 176
    }
  ],
  "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": "< Previous",
        "active": false
      },
      {
        "url": "http://127.0.0.1:8000/api/posts?page=1",
        "label": "1",
        "active": true
      },
      {
        "url": null,
        "label": "Next >",
        "active": false
      }
    ],
    "path": "http://127.0.0.1:8000/api/posts",
    "per_page": 5,
    "to": 2,
    "total": 2
  }
}

```


Jika tampilan sudah seperti di atas maka API telah berfungsi untuk mengambil data dari database.

19. Berikutnya lakukan percobaan insert data ke tabel di database menggunakan API. Caranya adalah mengganti method menjadi POST kemudian pada bagian header ubah menjadi sebagai berikut:



Selanjutnya klik send dan data berhasil di-insert ke database seperti tampilan berikut



20. Check manual di phpmyadmin, pastikan data baru masuk

	id	nama_sensor	nilai1	nilai2	created_at	updated_at
<input type="checkbox"/>	2	Sensor A	100	200	2025-03-07 20:55:40	2025-03-07 20:55:40
<input type="checkbox"/>	3	Sensor B	87	176	2025-03-07 20:55:40	2025-03-07 20:55:40
<input type="checkbox"/>	4	Sensor C	111	211	2025-03-11 00:24:37	2025-03-11 00:24:37

21. Berikutnya mengonline-kan API menggunakan service ngrok sehingga API dapat diakses melalui device iot atau simulasi wokwi iot. Pastikan sudah download aplikasi ngrok kemudian lakukan registrasi. Setelah aplikasi di download lakukan ekstraksi

22. Kemudian jalankan perintah sesuai yang ada di akun ngrok seperti berikut:

```
C:\Users\tania>cd C:\Users\tania\OneDrive\Documents\ngrok
C:\Users\tania\OneDrive\Documents\ngrok>ngrok config add-authtoken 2u9GuxaRfGBY6FP1TBYvXc1zje_2maJeSVBTGRN1oPwNA1Ye
Auth token saved to configuration file: C:\Users\tania\AppData\Local\ngrok\ngrok.yml
```

23. Kemudian jalankan perintah berikut di Command Prompt untuk mengonline kan laravel melalui port 8000

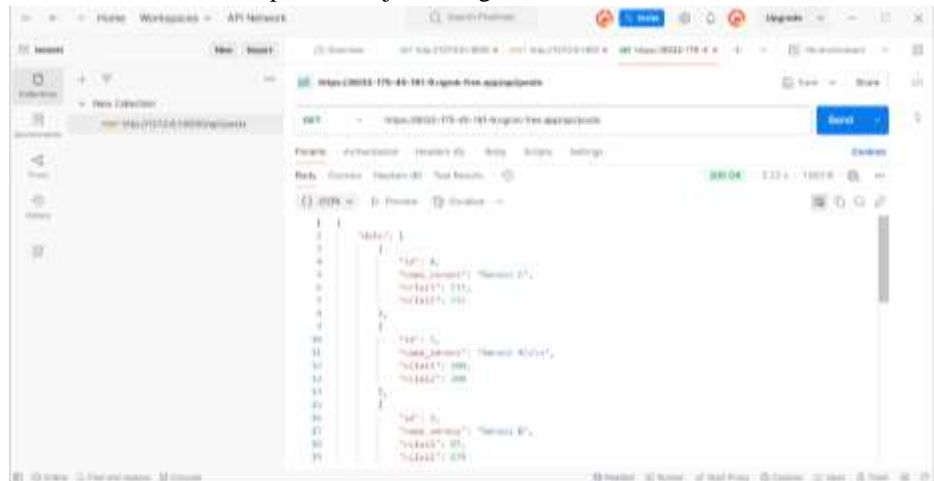
ngrok http <http://localhost:8000>


```
Command Prompt - ngrok - [X] + - v
ngrok
* Found a bug? Let us know: https://github.com/ngrok/ngrok

Session Status      online
Account             instantania2412@gmail.com (Plan: Free)
Version             3.28.0
Region              Asia Pacific (ap)
Latency              334ms
Web Interface        http://127.0.0.1:4040
Forwarding            https://87f6-175-45-191-15.ngrok-free.app -> http://localhost:8000

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

24. Kemudian lakukan percobaan menggunakan postman menggunakan URL yang diberikan oleh ngrok. Untuk melakukan percobaan GET api , maka URL harus ditambahkan alamat endpoint menjadi sebagai berikut :



25. Kemudian lakukan percobaan melakukan insert data baru melalui API dengan mengubah method menjadi **POST** dan parameter header dan body sesuai



Sampai disini API yang dibangun menggunakan laravel artinya sudah dapat berjalan dengan baik dan dapat diakses melalui URL publik.

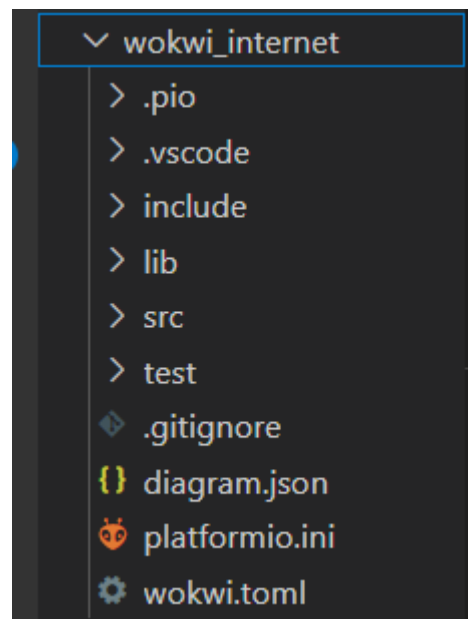
26. Selanjutnya mengakses API Melalui Simulasi WOKWI

Sebelumnya jalankan API laravel dengan perintah

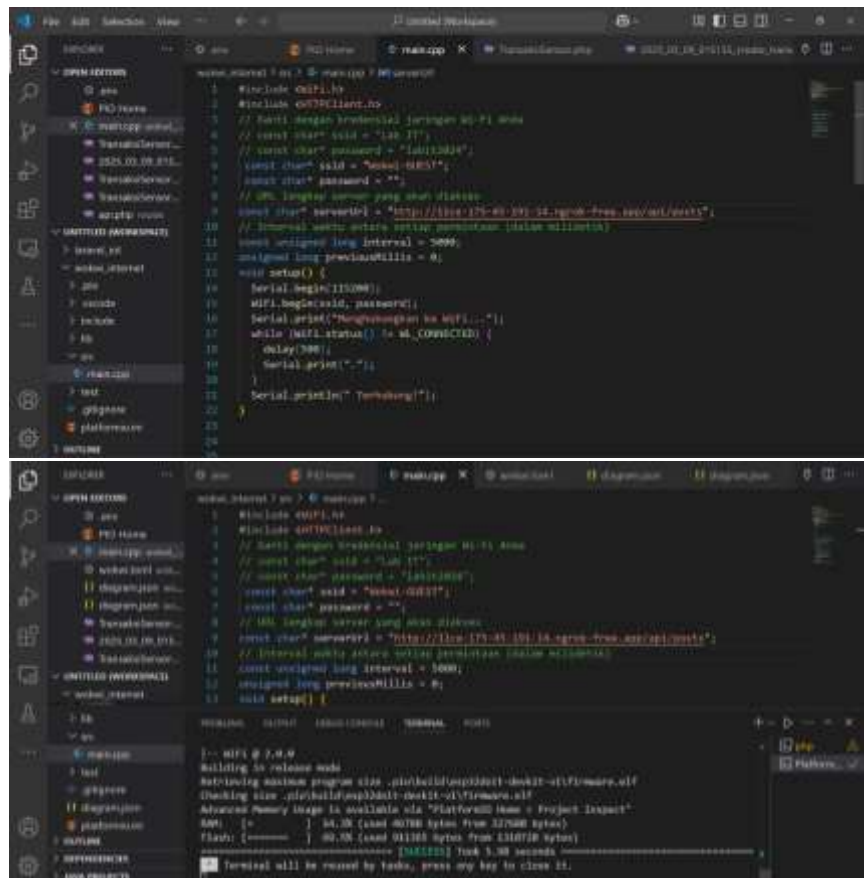
php artisan serve --host=0.0.0.0 --port=8080

Perintah diatas memastikan API laravel dapat diakses dari IP Address manapun dan memastikan bekerja pada port 8080.

27. Buat file baru wokwi simulator di platform.io dengan nama **wokwi_internet**

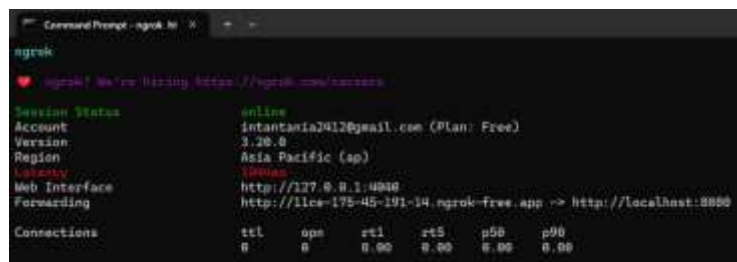


28. Ubah isi scrip **scr/main.cpp** seperti berikut



Ubah bagian serverURL sesuai dengan URL hasil dari generate perintah NGROK dengan alamat URL dalam bentuk **http** bukan **https** yaitu dengan menjalankan perintah

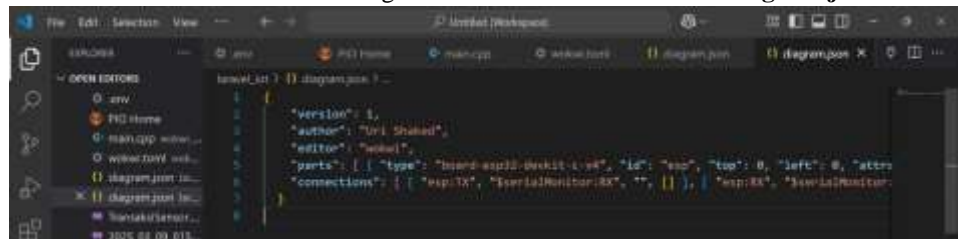
ngrok http --scheme=http 8080



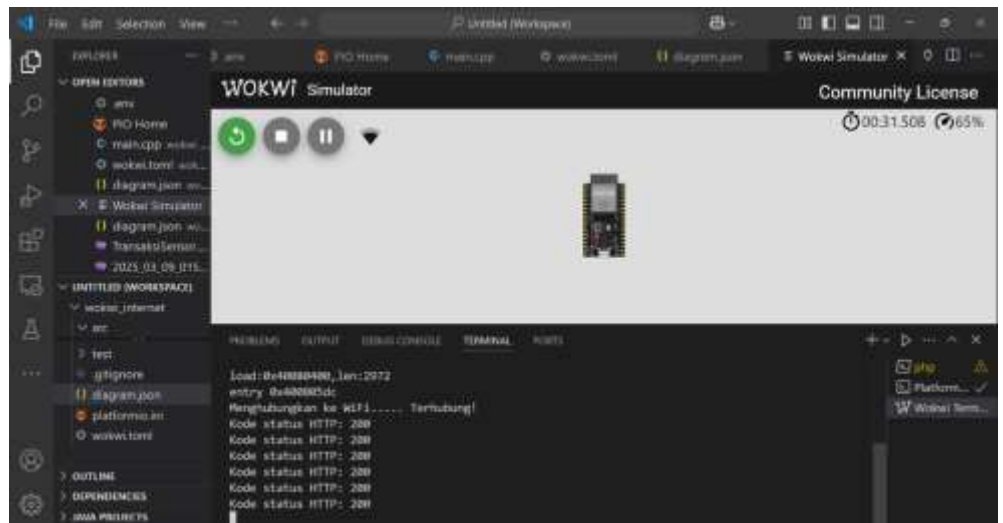
29. Buat file **wokwi.toml**



30. Buat file dengan nama **diagram.json**

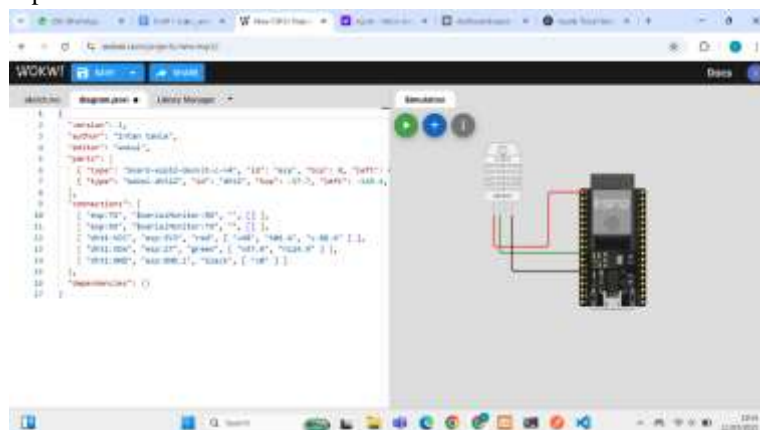


Kemudian >Wokwi Start Simulator



Pastikan *kode status Http yang di dapat adalah 200* yang menunjukkan bahwa ESP32 berhasil terhubung ke WIFI Wokwi-GUEST dan berhasil mengakses API laravel yang sudah dibuat pada bab sebelumnya.

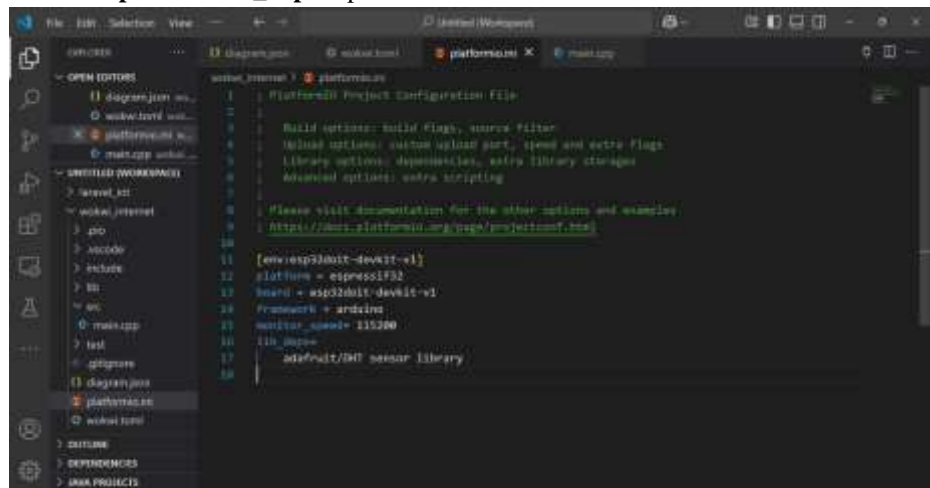
31. Selanjutnya lakukan modifikasi simulasi dengan menambahkan sensor suhu dan kelembaban di **wokwi simulator** dengan Rangkaian sensor DHT22 dan ESP32 seperti berikut



Salin kode **diagram.json** ke file diagram.json yang ada di vscode.

```
wokwi:terminal > diagram.json >
1
2 {
3   "version": 1,
4   "author": "Anonymous maker",
5   "editor": "wokwi",
6   "parts": {
7     { "type": "board-esp32-deskit-c-v1", "id": "esp", "top": 0, "left": -4.76, "attrs": {} },
8     { "type": "wokwi-dht11", "id": "dht11", "top": -114.9, "left": -120.6, "attrs": {} }
9   },
10  "connections": {
11    { "esp:TX", "serialMonitor:RX", "w", [ ] },
12    { "esp:RX", "serialMonitor:TX", "w", [ ] },
13    { "dht11:GND", "esp:GND", "black", [ "w" ] },
14    { "dht11:SDA", "esp:27", "green", [ "w" ] },
15    { "dht11:VCC", "esp:3V3", "red", [ "V5V:6", "h67.2", "V-20.8" ] }
16  },
17  "dependencies": {}
18 }
```

32. Kemudian ubah setting file **platformio.ini** dengan menambahkan 2 setting yaitu **monitor speed** dan **lib_deps** seperti berikut:



```
File Edit Selection View PlatformIO IDE
diagram.json platformio.ini main.cpp
diagram.json 1 PlatformIO Project Configuration File
2
3 Build options: build flags, source filter
4 Upload options: custom upload port, speed and extra flags
5 Library options: dependencies, extra library storage
6 Advanced options: extra scripting
7
8 Please visit documentation for the other options and examples
9 https://docs.platformio.org/page/projectconf.html
10
11 [env:esp32deskit-c-v1]
12 platform = espressif32
13 board = esp32deskit-c-v1
14 framework = arduino
15 monitor_speed = 115200
16 lib_deps =
17   Adafruit/DHT sensor library
```

33. Ubah isi file **main.cpp**

```
#include <Arduino.h>
#include <WiFi.h>
#include <HTTPClient.h>
#include <DHT.h>
#define DHTPIN 27
#define DHTTYPE DHT22
DHT dht(DHTPIN, DHTTYPE);

// Ganti dengan kredensial WiFi Anda
const char* ssid = "Mokel-SMS1";
const char* password = "";

unsigned long previousMillis = 0;
const long interval = 5000; // Interval 5 detik (5000 ms)

void setup() {
  Serial.begin(115200);
  // Hubungkan ke WiFi
  WiFi.begin(ssid, password);
  Serial.print("Menghubungkan ke WiFi");
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }

  Serial.println(" Terhubung!");
  dht.begin();
  // Tunggu sebentar agar koneksi stabil
  delay(1000);
}

void loop() {
  unsigned long currentMillis = millis();
  // Kirim POST setiap interval yang telah ditentukan
  if (currentMillis - previousMillis >= interval) {
    previousMillis = currentMillis;

    float h = round(dht.readHumidity());
    // Read temperature in Celsius (the default)
    float t = round(dht.readTemperature());

    // Check if any reads failed and exit early (to try again).
    if (isnan(h) || isnan(t)) {
      Serial.println(F("Failed to read from DHT sensor!"));
      return;
    }

    // Compute heat index in Celsius ((Fahrenheit - 32) * 0.5556 + 32)
    float hia = dht.computeHeatIndex(t, h, false);

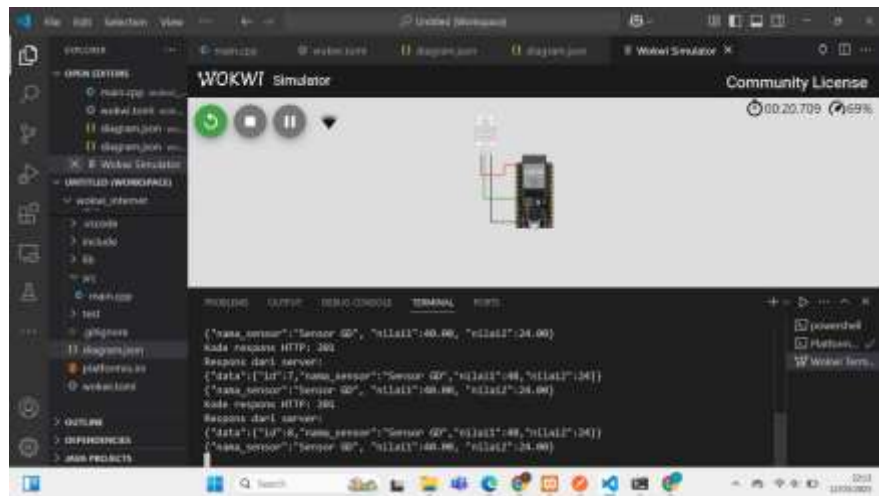
    // Inisialisasi HTTPClient
    HTTPClient http;
    String url = "http://192.168.1.34:8080/api/posts"; // Ganti dengan URL
    http.begin(url); // Pungkasikan HTTP, bukan HTTPS
    http.addHeader("Content-Type", "application/json");
    String payload = "{\"nama_sensor\":\"Sensor 00\", \"nilai\": \"\", \"nilai2\":\"\"}";

    Serial.println(payload); // Untuk melihat apakah payload sudah terbentuk dengan benar
    // Kirim POST request
    int httpResponseCode = http.POST(payload);

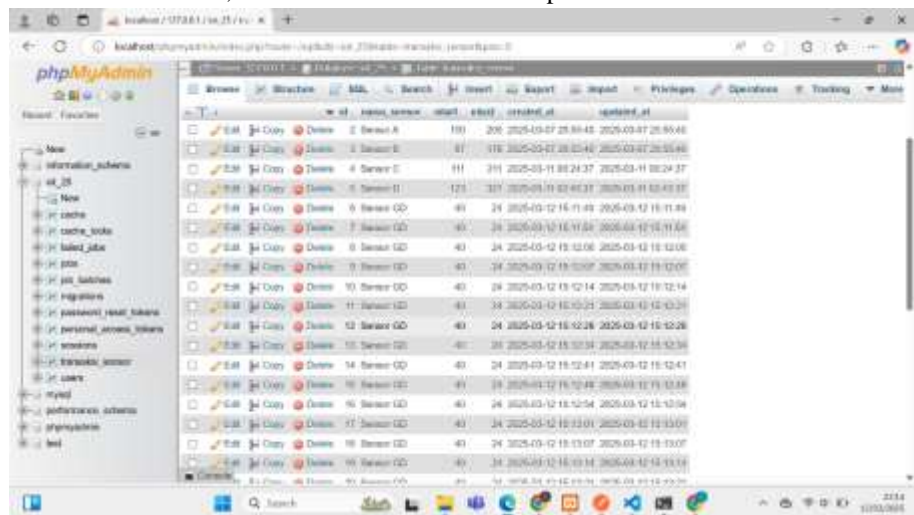
    // Tampilkan kode respons HTTP
    Serial.print("Kode respons HTTP: ");
    Serial.println(httpResponseCode);
    // Tampilkan respons dari server jika request berhasil
    if (httpResponseCode == 200 || httpResponseCode == 201) {
      String response = http.getString();
      Serial.println("Respons dari server:");
      Serial.println(response);
    } else {
      Serial.println("Gagal mengirim data");
    }

    // Tutup koneksi HTTP
    http.end();
  }
}
```

Bagian String URL uahh sesuai dengan URL nrook anda
Jalankan simulasi > **Wokwi Start Simulasi**



34. Pastikan di database, data telah muncul dan tersimpan



Setiap langkah ini dilakukan secara bertahap dan berurutan untuk memastikan sistem IoT dapat berfungsi secara optimal dalam menghubungkan perangkat sensor dengan API berbasis Laravel.

3. Results and Discussion

3.1 Eksperimental Result

Eksperimen ini dilakukan untuk menguji keberhasilan pembuatan API menggunakan Laravel 11 serta integrasinya dengan perangkat IoT melalui simulasi wokwi. Tahapan eksperimen meliputi pembuatan API, pengujian dengan Postman, penggunaan Ngrok untuk mengakses API secara publik, serta simulasi komunikasi perangkat ESP32 dengan sensor DHT22.

1. Pembuatan API menggunakan Laravel 11

Pada tahap awal, API dikembangkan menggunakan framework Laravel 11 dengan database mysql. Konfigurasi dilakukan melalui file .env, yang

menghubungkan Laravel dengan database `iot_25`. Berikut adalah langkah-langkah yang dilakukan dalam pembuatan API:

- Membuat proyek Laravel baru
- Membuat model dan migrasi database untuk tabel `transaksi_sensor`, yang akan digunakan untuk menyimpan data suhu dan kelembaban dari sensor IoT.
- Membuat resource controller untuk menangani permintaan Http GET dan POST dari perangkat IoT.
- Menjalankan Laravel server sehingga API dapat diuji.

```
PS C:\Users\tania\OneDrive\Documents\SEMESTER4\IoT\laravel_API> php artisan route:list

GET|HEAD / ..... posts.index ..... Api\TransaksiSensorController@index
GET|HEAD api/posts ..... posts.store ..... Api\TransaksiSensorController@store
POST api/posts ..... posts.show ..... Api\TransaksiSensorController@show
GET|HEAD api/posts/{post} ..... posts.update ..... Api\TransaksiSensorController@update
PUT|PATCH api/posts/{post} ..... posts.destroy ..... Api\TransaksiSensorController@destroy
DELETE api/user ..... sanctum.csrf-cookie ..... Laravel\Sanctum\CsrfCookieController@show
GET|HEAD storage/{path} ..... up .....
GET|HEAD up .....

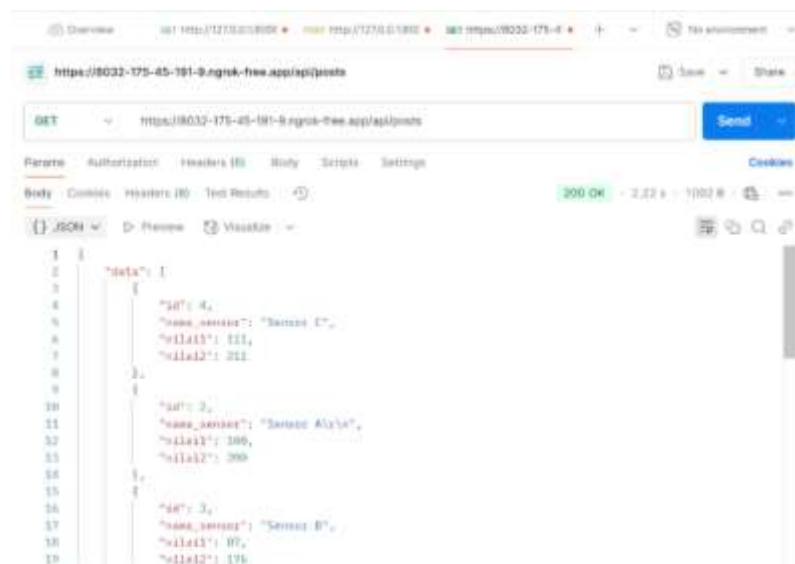
Showing [10] routes
```

2. Pengujian API menggunakan Postman

Setelah API berhasil dibuat, dilakukan pengujian menggunakan Postman untuk memastikan bahwa API dapat menerima dan mengirim data dengan benar. Pengujian dilakukan dengan skenario berikut:

- Menggunakan metode GET untuk mengambil data dari database.
- Menggunakan metode POST untuk mengirim data suhu dan kelembaban ke database.

Dari hasil pengujian, API dapat merespons dengan baik, menunjukkan bahwa struktur API dan koneksi ke database sudah benar.

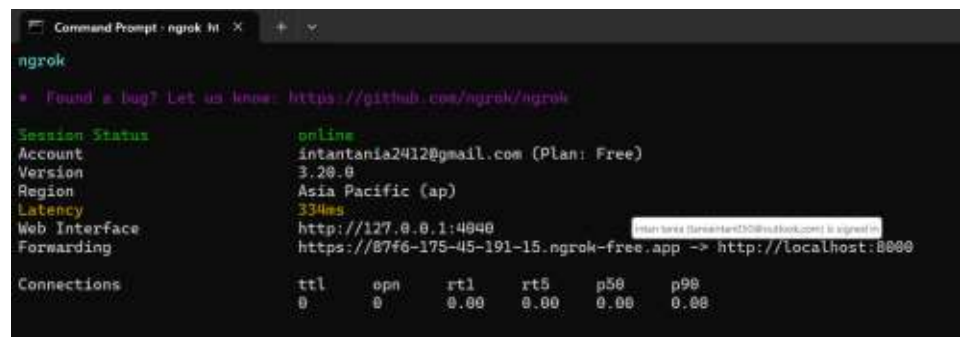


3. Menghubungkan API ke Internet menggunakan Ngrok

Agar API dapat diakses oleh perangkat IoT di luar jaringan lokal, digunakan Ngrok untuk membuat **tunnel** ke server Laravel yang berjalan di komputer lokal. Langkah-langkah yang dilakukan adalah:

- Menjalankan Laravel dengan perintah: `php artisan serve --host=0.0.0.0 --port=8000`
- Menjalankan Ngrok untuk membuka akses ke API secara publik: `ngrok http http://localhost:8000`
- Menggunakan URL yang diberikan oleh Ngrok untuk mengakses API melalui internet.

Dari hasil pengujian, API berhasil diakses melalui URL publik yang dihasilkan oleh Ngrok, sehingga perangkat IoT atau simulasi dapat mengirimkan data ke server dari luar jaringan lokal.



```
Command Prompt - ngrok ht x + v
ngrok
* Found a bug? Let us know: https://github.com/ngrok/ngrok

Session Status      online
Account             intantania2412@gmail.com (Plan: Free)
Version             3.20.0
Region              Asia Pacific (ap)
Latency              334ms
Web Interface       http://127.0.0.1:4840
Forwarding           https://87f6-175-45-191-15.ngrok-free.app -> http://localhost:8000

Connections
  ttl    opn    rt1    rt5    p50    p99
    0     0     0.00   0.00   0.00   0.00
```

4. Simulasi IoT menggunakan WOKWI dengan ESP32 dan Sensor DHT22

Setelah API berjalan dengan baik, melakukan simulasi perangkat IoT menggunakan wokwi dengan membuat rangkaian ESP32 dengan sensor DHT22. Simulasi dilakukan dengan langkah-langkah berikut:

- Menyiapkan rangkaian ESP32 dan DHT22 di wokwi.
- Menulis program Arduino yang memungkinkan ESP32 membaca data dari sensor dan mengirimkannya ke API Laravel menggunakan metode http POST.
- Menggunakan URL dari Ngrok dalam kode ESP32 agar dapat mengirim data ke API.

Dari hasil simulasi, ESP32 berhasil membaca suhu dan kelembaban dari sensor DHT22, kemudian mengirimkannya ke API Laravel dengan kode status 200 yang menunjukkan bahwa data telah diterima oleh server.



The screenshot shows the AWS IAM console with a list of 16 IAM users. The table has the following columns: Name, Access key, Status, MFA, Created at, and Last sign-in at. All users have 'Access key' status and 'MFA Off' status. The 'Created at' column shows dates from 2025-03-07 to 2025-03-14.

Name	Access key	Status	MFA	Created at	Last sign-in at
2001	Access key	Access key	MFA Off	2025-03-07 00:00:00	2025-03-07 00:00:00
2002	Access key	Access key	MFA Off	2025-03-07 00:00:00	2025-03-07 00:00:00
2003	Access key	Access key	MFA Off	2025-03-07 00:00:00	2025-03-07 00:00:00
2004	Access key	Access key	MFA Off	2025-03-07 00:00:00	2025-03-07 00:00:00
2005	Access key	Access key	MFA Off	2025-03-07 00:00:00	2025-03-07 00:00:00
2006	Access key	Access key	MFA Off	2025-03-07 00:00:00	2025-03-07 00:00:00
2007	Access key	Access key	MFA Off	2025-03-07 00:00:00	2025-03-07 00:00:00
2008	Access key	Access key	MFA Off	2025-03-07 00:00:00	2025-03-07 00:00:00
2009	Access key	Access key	MFA Off	2025-03-07 00:00:00	2025-03-07 00:00:00
2010	Access key	Access key	MFA Off	2025-03-07 00:00:00	2025-03-07 00:00:00
2011	Access key	Access key	MFA Off	2025-03-07 00:00:00	2025-03-07 00:00:00
2012	Access key	Access key	MFA Off	2025-03-07 00:00:00	2025-03-07 00:00:00
2013	Access key	Access key	MFA Off	2025-03-07 00:00:00	2025-03-07 00:00:00
2014	Access key	Access key	MFA Off	2025-03-07 00:00:00	2025-03-07 00:00:00
2015	Access key	Access key	MFA Off	2025-03-07 00:00:00	2025-03-07 00:00:00
2016	Access key	Access key	MFA Off	2025-03-07 00:00:00	2025-03-07 00:00:00

The screenshot displays the 'Users' page in the AWS IAM console. The table lists 16 IAM users, all of whom have 'Access console' enabled and 'MFA' set to 'None'. The 'Created at' column shows dates from 2020-03-07 to 2020-03-12. The 'Last sign-in' column is empty for all users.

Name	Access console	MFA	Status	Created at	Last sign-in
user-1	Enabled	None	Active	2020-03-07 20:09:40	2020-03-07 20:09:40
user-2	Enabled	None	Active	2020-03-07 20:09:40	2020-03-07 20:09:40
user-3	Enabled	None	Active	2020-03-07 20:09:40	2020-03-07 20:09:40
user-4	Enabled	None	Active	2020-03-07 20:09:40	2020-03-07 20:09:40
user-5	Enabled	None	Active	2020-03-07 20:09:40	2020-03-07 20:09:40
user-6	Enabled	None	Active	2020-03-07 20:09:40	2020-03-07 20:09:40
user-7	Enabled	None	Active	2020-03-07 20:09:40	2020-03-07 20:09:40
user-8	Enabled	None	Active	2020-03-07 20:09:40	2020-03-07 20:09:40
user-9	Enabled	None	Active	2020-03-07 20:09:40	2020-03-07 20:09:40
user-10	Enabled	None	Active	2020-03-07 20:09:40	2020-03-07 20:09:40
user-11	Enabled	None	Active	2020-03-07 20:09:40	2020-03-07 20:09:40
user-12	Enabled	None	Active	2020-03-07 20:09:40	2020-03-07 20:09:40
user-13	Enabled	None	Active	2020-03-07 20:09:40	2020-03-07 20:09:40
user-14	Enabled	None	Active	2020-03-07 20:09:40	2020-03-07 20:09:40
user-15	Enabled	None	Active	2020-03-07 20:09:40	2020-03-07 20:09:40
user-16	Enabled	None	Active	2020-03-07 20:09:40	2020-03-07 20:09:40