*TRANSMISI: JURNAL ILMIAH TEKNIK ELEKTRO, 27, (1), JANUARI 2025*
*p-ISSN 1411-0814 e-ISSN 2407-6422*

# PENGEMBANGAN ARSITEKTUR REST API UNTUK INTEGRASI DATA REAL-TIME PADA WEBSITE PEMANTAUAN KUALITAS UDARA LAHAN PERTANIAN

Satrio Bisma Bramantyo[1], Ika Novita Dewi[1*)], Ivan Muhammad Reza[2], Filmada Ocky Saputra[2] dan Zainal Arifin Hasibuan[2]

[1]Program Studi Sistem Informasi, Universitas Dian Nuswantoro, Semarang, Indonesia
[2]Center of Excellence in Science and Technology, Universitas Dian Nuswantoro, Semarang, Indonesia

[*)] *E-mail: ikadewi@dsn.dinus.ac.id*

## Abstrak

Transformasi teknologi dalam sektor pertanian semakin berkembang dengan adopsi perangkat Internet of Things (IoT) yang mampu menghasilkan data *real-time* terkait kualitas udara, seperti konsentrasi CO2, NO2, dan CH4. Meskipun data yang dihasilkan sangat berharga, namun tantangan besar masih dihadapi dalam hal penyimpanan dan manajemen data yang terfragmentasi. Penelitian ini bertujuan untuk mengembangkan arsitektur REST API yang mampu mengintegrasikan perangkat IoT, basis data, dan antarmuka pengguna dalam bentuk website pemantauan kualitas udara lahan pertanian. REST API digunakan untuk memvalidasi, memproses, dan memformat data yang kemudian dikirimkan ke pusat database melalui protokol HTTP standar (GET, POST). Selain itu, protokol WebSocket diterapkan untuk memastikan komunikasi dua arah yang memungkinkan transmisi data secara real-time antara perangkat IoT dan antarmuka pengguna. Hasil pengujian menunjukkan bahwa arsitektur ini mampu memberikan informasi yang akurat dan cepat kepada petani, mendukung pengambilan keputusan yang lebih baik dalam pengelolaan lahan, serta berkontribusi pada pengurangan emisi gas rumah kaca.

*Kata kunci: IoT, REST API, WebSocket, kualitas udara, lahan pertanian, pemantauan real-time*

## *Abstract*

*The technological transformation in the agricultural sector is advancing with the adoption of Internet of Things (IoT) devices capable of generating real-time data related to air quality, such as CO2, NO2, and CH4 concentrations. Despite the value of the data produced, significant challenges remain in terms of fragmented data storage and management. This research aims to develop a REST API architecture capable of integrating IoT devices, databases, and a user interface in the form of a website for monitoring agricultural land air quality. The REST API is used to validate, process, and format data, which is then sent to a central database using standard HTTP protocols (GET, POST). Additionally, WebSocket protocols are applied to ensure bidirectional communication, enabling real-time data transmission between IoT devices and the user interface. Testing results show that this architecture is capable of providing accurate and timely information to farmers, supporting better decision-making in land management, and contributing to the reduction of greenhouse gas emissions.*

*Keywords: IoT, REST API, WebSocket, air quality, agricultural land, real-time monitoring*

## 1. Pendahuluan

Sektor pertanian mengalami transformasi signifikan yang didorong oleh munculnya teknologi modern, salah satunya adalah Internet of Things (IoT). IoT memiliki peranan penting dalam proses revolusi teknologi pertanian dengan menyediakan kemampuan pengumpulan dan pemantauan data secara real-time seperti data kualitas udara, kelembapan [1], suhu, dan kesehatan tanaman. Markets and Markets melaporkan bahwa pangsa pasar IoT di sektor pertanian diperkirakan akan tumbuh dari USD 11.4 miliar pada tahun 2021 menjadi USD 18.1 miliar pada tahun 2026, dengan tingkat pertumbuhan tahunan sebesar 9.8% [2]. Salah satu adopsi teknologi pertanian dengan penggunaan IoT adalah pemantauan kesehatan tanaman dan prediksi hasil dengan tujuan untuk meningkatkan produktivitas pertanian [3]. Manfaat lain penggunaan teknologi sensor dalam IoT di bidang pertanian dapat mengurangi penggunaan air irigasi hingga 30% dengan tetap menjaga kualitas dan kuantitas hasil pertanian [4]. Di sisi lain, emisi gas rumah kaca seperti CO2, NO2, dan CH4 yang dihasilkan dari praktik pertanian seperti pembakaran lahan dan penggunaan pupuk urea dapat berkontribusi pada pemanasan global dan perubahan iklim [5]. Gas-gas ini

mempercepat efek rumah kaca, yang mengakibatkan kenaikan suhu bumi, perubahan pola cuaca, serta degradasi lahan pertanian. Oleh karena itu, pemantauan emisi ini penting untuk mengurangi dampak negatif terhadap lingkungan dan mendukung pertanian yang lebih berkelanjutan.

Salah satu bentuk pemantauan emisi untuk mengukur kualitas udara di lahan pertanian yang mudah diakses oleh petani adalah website. Salah satu website yang tengah dikembangkan dalam penelitian kami adalah E-Asia. E-Asia merupakan *dashboard* pemantauan emisi gas yang dapat diakses oleh petani melalui web. Melalui website ini, petani dapat memperoleh informasi penting terkait pantauan kualitas udara, seperti tingkat konsentrasi CO2, NO2, dan CH4. Selain itu, mereka juga dapat melihat **status lahan dan jenis tanaman yang sedang dikelola**. Data yang ditampilkan di website E-Asia berasal dari perangkat IoT, seperti sensor, yang mengirimkan informasi secara *real-time* ke basis data untuk keperluan penyimpanan dan pengelolaan data [6]. Dengan antarmuka pengguna yang ramah, website ini dapat membantu petani mengambil keputusan yang lebih tepat untuk menjaga produktivitas sekaligus mengurangi dampak lingkungan.

Meskipun perangkat IoT menghasilkan data yang berlimpah dan bernilai, tantangan besar tetap ada pada sifat penyimpanan dan manajemen data yang terfragmentasi. Saat ini, integrasi antara perangkat IoT, basis data, dan antarmuka pengguna melalui website E-Asia masih belum dimanfaatkan secara optimal. Salah satu penyebab utama adalah terbatasnya platform terpadu yang mampu mengintegrasikan data dari perangkat IoT pertanian, basis data, serta antarmuka pengguna, yang memperlambat adopsi teknologi ini secara efektif. Penelitian ini mengusulkan integrasi antara perangkat IoT, basis data, dan antarmuka pengguna untuk pengelolaan kualitas udara di lahan pertanian melalui pengembangan *Application Programming Interface* (API). API merupakan *framework* yang menghubungkan perangkat IoT ke pusat basis data sehingga memungkinkan tampilan visualisasi data yang intuitif melalui antarmuka pengguna website yang ramah pengguna [7]. API juga memastikan proses transmisi dan penyimpanan data yang aman melalui enkripsi dan protokol komunikasi [8]. API ini juga mendukung skalabilitas, memungkinkan sistem untuk menangani jumlah data yang semakin meningkat seiring dengan penambahan perangkat IoT dan lebih banyak pengguna yang mengakses sistem [8].

Pengembangan API telah dilakukan dalam beberapa bidang, seperti arsip data [9][10][11], data kesehatan [12], dan sistem rekomendasi [13]. Salah satu arsitektur yang dapat digunakan dalam API adalah *Representational State Transfer* API (RestAPI). Rest API melakukan validasi, proses dan memformat penyimpanan data untuk disimpan ke pusat basis data [14]. Perangkat IoT yang dipasang di lahan pertanian menghasilkan data secara *real-time*, yang kemudian dikirimkan ke basis data melalui RestAPI menggunakan protokol standar HTTP seperti GET dan POST. Selain itu, protokol komunikasi dua arah seperti WebSocket juga diterapkan untuk memastikan data dapat dikirim dan diterima secara *real-time*, sehingga memungkinkan petani mendapatkan informasi yang akurat dan cepat tentang kualitas udara di lahan pertanian mereka. Dengan mengintegrasikan RestAPI dan WebSocket, penelitian ini bertujuan untuk mengembangkan arsitektur yang andal dan efisien dalam mendukung pengelolaan kualitas udara melalui sebuah website yang mudah diakses. Penelitian ini akan memberikan kontribusi signifikan terhadap adopsi teknologi IoT di sektor pertanian dengan fokus pada pemantauan kualitas udara secara *real-time*.

## 2. Metode

Penelitian ini menerapkan beberapa tahapan dalam mengintegrasikan sistem monitoring kualitas udara di lahan pertanian dengan RestAPI seperti yang terlihat dalam Gambar 1. Dalam Gambar 1, tahapan penelitian yang dilakukan meliputi identifikasi masalah, pengembangan API, hasil dan pengujian API dan pengujian sistem.


**Gambar 1. Alur Penelitian**

Langkah awal dalam penelitian adalah melakukan identifikasi masalah. Identifikasi masalah dilakukan dengan melakukan eksplorasi terhadap perangkat IoT dan website E-Asia. Masalah yang teridentifkasi adalah bagaimana melakukan integrasi data agar data sensor dari perangkat IoT dapat secara *real-time* ditampilkan pada website E-Asia.

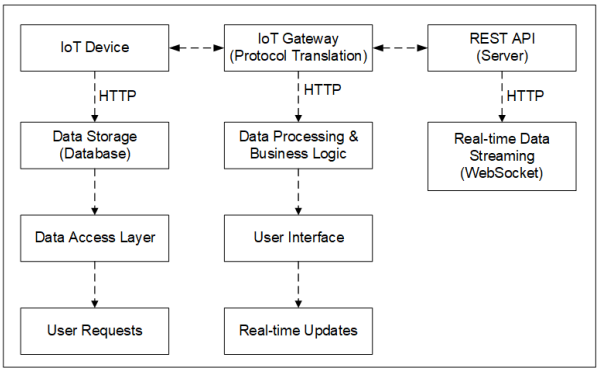

**Gambar 2. Rancangan integrasi data real-time**

Pengembangan API dengan RestAPI dilakukan dengan menentukan kebutuhan data, membuat desain Endpoint,

melakukan set up environment, implementasi Endpoint, integrasi database, dan pengujian. Alur pengembangan dapat dilihat dalam Gambar 2. Gambar 2 menjelaskan alur dalam mengembangkan integrasi data dengan RestAPI, sehingga pengguna dapat melakukan akses data secara *real-time*.

### 2.1. Integrasi Data

Alur data dalam proses integrasi data untuk pemantauan data secara real-time dapat dilihat dalam Gambar 3.




**Gambar 3. Alur integrasi data**

Dalam proses alur data, perangkat IoT pertama-tama mengirimkan data, yang berupa data *longitude*, *latitude*, CO2, NO2, dan CH4, ke IoT Gateway. IoT Gateway kemudian meneruskan informasi tersebut ke RestAPI Server. Setelah menerima data, RestAPI Server memproses dan memperbarui *data storage* (basis data) sesuai kebutuhan menggunakan metode POST. Data yang diterima oleh basis data ini disimpan dalam tabel bernama '*device*'. Layer ini berperan penting dalam memproses dan memvalidasi data yang diterima melalui Rest API. Data yang masuk akan diperiksa untuk memastikan kelengkapannya, menghilangkan data yang tidak sesuai atau rusak, serta memastikan formatnya benar sebelum disimpan ke dalam basis data. Selanjutnya, *business logic* diterapkan sebagai aturan dan logika spesifik dari aplikasi dijalankan. Misalnya, akan muncul peringatan jika tingkat CO2 melebihi ambang batas tertentu.

Setalah data tersimpan ke dalam basis data, halaman antar muka pengguna (*user interface*) dalam website akan melakukan permintaan data ke endpoint dari basis data menggunakan metode GET. Untuk memastikan pengguna mendapatkan informasi terkini, RestAPI Server kemudian melakukan streaming pembaruan data *real-time* ke *user interface* menggunakan Socket.io, Socket.io dipilih karena memiliki fitur tambahan seperti *fallback* dibandingkan dengan Websocket bawaan atau *server-sent events (SSE)*. Pada tahap akhir, halaman website menampilkan pembaruan secara *real-time* dan memproses permintaan pengguna, memungkinkan interaksi dengan sistem dan memberikan pengalaman pengguna yang responsif. Dengan menggunakan Socket.io, website hanya perlu melakukan *request* sekali dan data baru akan dikirimkan secara otomatis setiap kali ada pembaruan data. Hal ini memungkinkan pemantauan data secara instan dan efektif tanpa perlu melakukan permintaan berulang kali.

### 2.2. Identifikasi kebutuhan

Identifikasi kebutuhan dilakukan dengan menentukan kebutuhan fungsional dan kebutuhan non-fungsional. Kebutuhan fungsional dilakukan dengan menentukan fitur utama API, jenis data yang akan diakses atau dimodifikasi, serta desain endpoint yang mencakup URL, metode HTTP, dan parameter yang diperlukan. Kebutuhan non-fungsional ditentukan dengan mengidentifikasi keamanan, kinerja, dan skalabilitas, termasuk kebutuhan untuk otentikasi, enkripsi data, dan kemampuan API untuk menangani pertumbuhan pengguna.

### 2.3. Desain endpoint

Endpoint merupakan perancangan titik akses yang akan disediakan oleh API untuk berinteraksi dengan berbagai sumber daya. Pada tahap ini, perlu ditentukan URL yang akan digunakan untuk setiap resource, misalnya '/users' untuk data pengguna, serta metode HTTP yang akan digunakan (GET, POST). Sedangkan format data yang akan dikembalikan oleh API dalam bentuk JSON.

### 2.4. Arsitektur Backend

Penelitian ini menggunakan lingkungan pengembangan dengan bahasa pemrograman JavaScript dan framework Express.js.

Arsitektur *backend* dikembangkan dengan Express.js dengan bahasa pemrograman JavaScript. Tahap awal yang dilakukan adalah inisialisasi *library* yang diperlukan seperti Express, CORS, body-Parser, HTTP, bcrypt untuk hashing password, dan Socket.IO. Kode program untuk inisialisasi *library* dapat dilihat dalam Gambar 4.

```
const express = require('express');
const cors = require('cors');
const bodyParser = require('body-parser');
const http = require('http');
const bcrypt = require('bcryptjs');
const { Server } = require('socket.io');
const db = require('./connect');
const app = express();
const PORT = 3001;
```

**Gambar 4. Inisialisasi *library***

Proses *request* data dilakukan dengan memformat data respon dengan *url-uncoded* dan JSON, serta *cors* menggunakan *bodyParser* untuk mengizinkan akses lintas domain. Kode program terdapat dalam Gambar 5.

```
app.use(bodyParser.urlencoded({ extended: true }));
app.use(bodyParser.json());
app.use(cors());
```

**Gambar 5. Kode program proses permintaan data**

Socket.io mengelola dan menyimpan *socket* dari *client* yang terhubung. Ketika *client* sudah terhubung akan muncul pesan '*client connected*' pada *console.log* dan ketika terputus muncul pasan '*client disconnect*'. Kode program dapat dilihat dalam Gambar 6.

```
const sockets = {};
io.on('connection', function (socket) {
  console.log('Client connected');
  sockets[socket.id] = socket;
  socket.on('message', function (message) {
    console.log('received: %s', message);
  });
  socket.on('disconnect', function () {
    console.log('Client disconnected');
    delete sockets[socket.id];
  });
```

**Gambar 6. Kode program Socket.IO connection handling**

Server digunakan dengan menentukan port 3001 sesuai dengan setup yang telah dibuat. Kode program untuk menjalankan server terdapat dalam Gambar 7.

```
server.listen(PORT, () => {
  console.log(`Server is running on port ${PORT}`);
});
```

**Gambar 7. Kode program server running**

Konfigurasi untuk koneksi file connect yang berisi struktur koneksi basis data terdiri dari *IP database*, *user*, *password*, dan nama *database*. Jika berhasil terkoneksi, maka akan muncul pesan '*connected*' pada *console.log*, dan jika error maka akan menampilkan error pada console.log. Kode program konfigurasi koneksi basis data terdapat dalam Gambar 8.

```
db.connect(err => {
  if (err) {
    console.log(err.message);
  } else {
    console.log('connected');
  }
});
```

**Gambar 8. Kode program koneksi basis data**

Fungsi response http dilakukan menggunakan format JSON. Response ini menerima tiga parameter yaitu, *statuscode*, *data*, dan *object response*. Kode program fungsi response terdapat dalam Gambar 9.

```
function response(statusCode, data, res) {
  res.status(statusCode).json({ data: data });
}
```

**Gambar 9. Kode program fungsi response**

Endpoint dengan metode GET diakses dengan perintah *query* '*select * from device*' untuk mengambil semua data dari tabel device. Hasil *query* kemudian ditangani dalam *callback*: jika terjadi kesalahan, pesan *error* dicetak ke konsol dan respons dengan status 500 serta pesan *error* dikirim ke *client*. Jika hasil *query* sesuai dengan format yang diharapkan, fungsi *response* dipanggil untuk mengirimkan data dengan status 200. Kode program endpoint GET dapat dilihat dalam Gambar 10.

```
app.get('/device/data', (req, res) => {
  const sql = `SELECT * FROM device`;
  db.query(sql, (error, result) => {
    if (error) {
      console.error(error);
      return res.status(500).json({ success: false,
message: error.message });
    }
    if (Array.isArray(result.rows)) {
      response(200, result.rows, res);
    } else {
      console.error('Data is not in expected
format');
      return res.status(500).json({ success: false,
message: 'Data is not in expected format' });
    }
  });
});
```

**Gambar 10. Kode program endpoint GET**

```
app.post('/device/input', (req, res) => {
  const { IdDevice, long, lat, ch4, co2, no2 } =
req.body;
  const sql = `INSERT INTO device (IdDevice, long,
lat, ch4, co2, no2) VALUES ($1, $2, $3, $4, $5,
$6)`;
  const values = [IdDevice, long, lat, ch4, co2,
no2];
  db.query(sql, values, (error, result) => {
    if (error) {
      console.error(error);
      return res.status(500).json({ success: false,
message: error.message });
    }
    for (const socketId in sockets) {
      if (sockets.hasOwnProperty(socketId)) {
        sockets[socketId].emit('update', { IdDevice,
long, lat, ch4, co2, no2, timestamp: new Date() });
      }
    }
```

**Gambar 11. Kode program endpoint POST**

Endpoint dengan metode POST digunakan untuk menerima data device dari permintaan *client.* Data yang dikirim dalam *request body* (IdDevice, longitude, latitude,

ch4, co2, no2) diambil dan dimasukkan ke dalam tabel device melalui perintah *query insert*. Jika terjadi kesalahan selama *query*, pesan *error* dicetak ke konsol dan respons dengan status 500 serta pesan *error* dikirim ke *client*. Jika *query* berhasil, data yang baru ditambahkan dikirim ke semua *client* yang terhubung melalui *socket* dengan mengirimkan *event update*. Akhirnya, respons dengan status 200 dan pesan sukses dikirim ke *client*. Kode program penambahan data dapat dilihat dalam Gambar 11.

### 2.5. Integrasi basis data

Integrasi basis data dalam pengembangan API memungkinkan penyimpanan dan pengambilan data secara efisien. Dalam proses ini, API dihubungkan dengan basis data untuk memastikan data yang dikirim atau diterima dapat disimpan dan dikelola dengan baik. Implementasi query menggunakan operasi CRUD (*Create*, *Read*, *Update*, *Delete*) dilakukan untuk mengelola data dengan cara yang sistematis. Sistem basis data yang digunakan adalah PostgreSQL.

Sistem pengelolaan basis data dibuat dengan menggunakan PostgreSQL terdiri dari dua tabel, yaitu tabel *users* dan tabel *device*. Tabel users terdiri dari empat atribut, yaitu *userID*, *username*, *email* dan *password*, serta *IdDevice* sebagai *secondary key* untuk terhubung ke tabel *device*. Sedangkan tabel *device* terdiri dari sembilan atribut, yaitu *IdDevice*, *DeviceName*, *Type*, *longitude*, *latitude*, *timestamp*, CH4, CO2, dan NO2.

### 2.6. Pengujian

Pengujian dilakukan dengan menguji API secara keseluruhan untuk memastikan bahwa semua *endpoint* berfungsi bersama dengan baik. API juga diuji dari perspektif pengguna akhir untuk memastikan bahwa seluruh alur kerja aplikasi berfungsi dengan baik

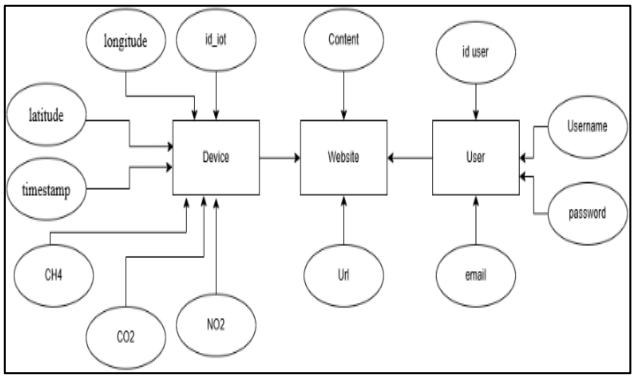
## 3. Hasil dan Pembahasan

### 3.1. Perancangan Arsitektur Sistem

Dalam mengembangkan sistem perlu dilakukan perancangan arsitektur sistem. Perancangan arsitektur sistem atau Enterprise Architecture Planning(EAP) adalah metode yang digunakan untuk merencanakan dan mengkoordinasikan arsitektur informasi dalam suatu organisasi. EAP bertujuan mengintegrasikan strategi bisnis dengan teknologi informasi untuk mendukung proses bisnis secara efektif[15]. Dengan begitu sistem yang akan dibangun dapat berjalan sesuai dengan rencana.

### 3.2. Perancangan Database

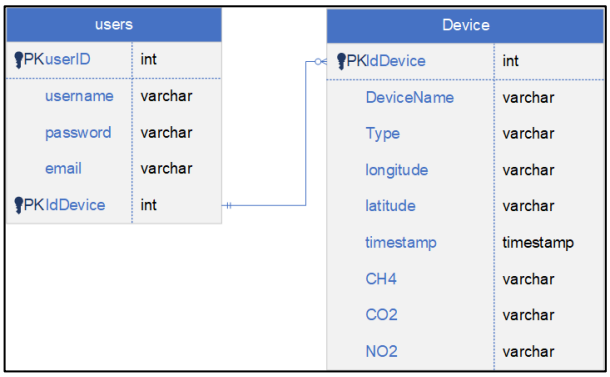
Untuk mengetahui struktur data dan hubungan antar entitas dalam sistem, dibentuklah *Entity Relation Diagram (ERD)* sebagai media untuk membantu dalam merancang sistem informasi. Gambar 12 menunjukkan ERD yang dirancang untuk sistem pemantauan kualitas udara di lahan pertanian.




**Gambar 12. ERD Sistem Integrasi Database**

### 3.3. Implementasi Database

Setelah merancang hubungan antar entitas sistem, kemudian membuat skema relasi berdasarkan ERD yang telah dibuat. Pada tahap ini setiap atribut ditetapkan type data berdasarkan kebutuhan input. Skema ini menunjukkan bahwa IdDevice menjadi *foreign key* yang digunakan untuk membangun hubungan antar tabel. Ini menunjukkan bahwa hanya pengguna yang terdaftar yang dapat mengakses data dari perangkat IoT. Untuk skema relasi tabel users dan device dalam sistem pemantauan kualitas udara dapat dilihat pada Gambar 13.




**Gambar 13. Skema Relasi Tabel Users dan Device**

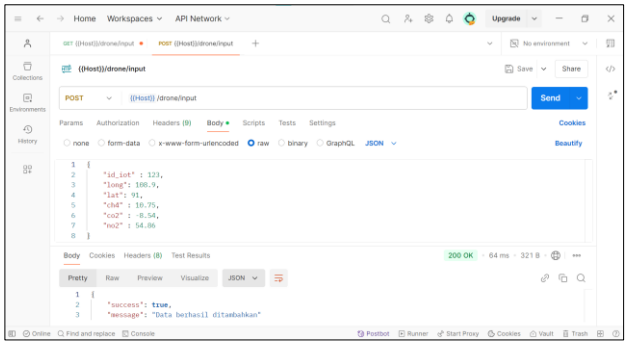
### 3.4. Implementasi Endpoint

#### 3.4.1. Mengirim data dari perangkat menuju database

Endpoint ini digunakan untuk mengirimkan data yang dikumpulkan dari perangkat Internet of Things (IoT) ke dalam basis data.

Gambar 14 menunjukan pengujian API dengan endpoint /drone/input yang berfungsi untuk mengirim data dari perangkat IoT menuju database dilakukan dengan method POST. Ketika perangkat IoT dapat menerima data dari sensor, perangkat IoT melakukan permintaan POST ke

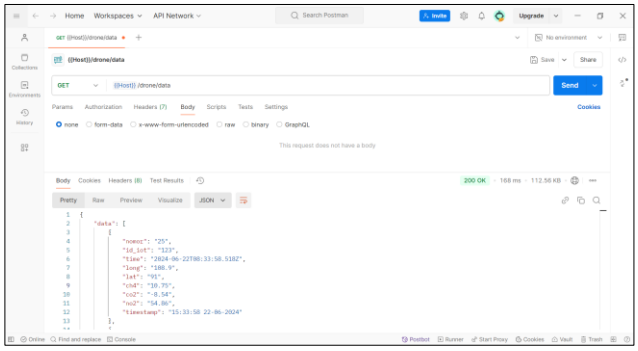
server kemudian server menerima permintaan dan memproses penyimpanan dalam database. Jika pengiriman data berhasil maka akan muncul pesan “data berhasil ditambahkan”.



**Gambar 14. POST data dari IoT ke Database**

### 3.4.2. Mengambil data dari database menuju dashboard

Endpoint ini digunakan untuk mengambil data dari basis data dan menampilkannya di dashboard website, yang dapat digunakan untuk analisis atau pemantauan.

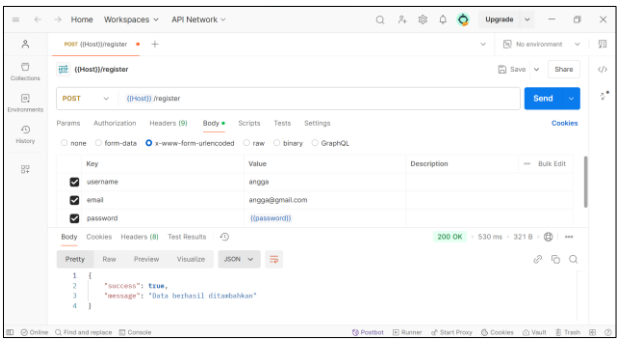


**Gambar 15. GET data dari Database ke Website**

Gambar 15 menunjukan pengujian API dengan endpoint /drone/data untuk melakukan pengambilan data dari database, hal ini merupakan proses menampilkan data pada dashboard website E-Asia. Ketika website melakukan permintaan GET, database akan mengirim data menggunakan format JSON.

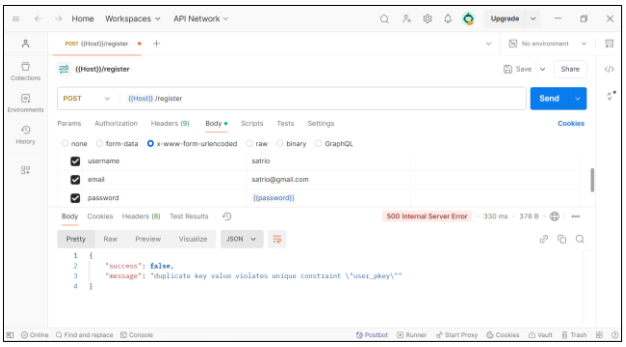
### 3.4.3. Mengirim data ketika register menuju database

Endpoint ini memungkinkan pengguna untuk mengirimkan data mereka dari website ke dalam basis data, seperti saat mereka melakukan pendaftaran akun melalui website. Disediakan kolom untuk username, email, dan password yang dapat diisi sesuai data pengguna.



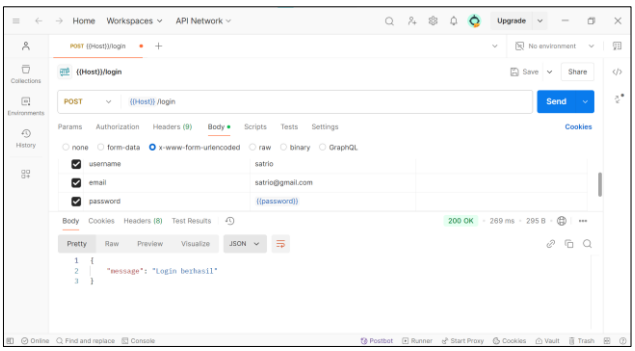
**Gambar 16. POST Data User Ketika Register**

Pada endpoint register memiliki 2 kondisi, jika user memasukan data register dan data tersebut belum ada pada database, maka akan muncul pesan “*Data berhasil ditambahkan*” seperti gambar 16, namun jika user memasukan data dan data tersebut memiliki kesamaan dengan data di dalam database maka akan muncul pesan “*duplicate key value violates unique constraint \"user_pkey\"*” seperti gambar 17.



**Gambar 17. POST Data User Ketika Data Sudah Ada**

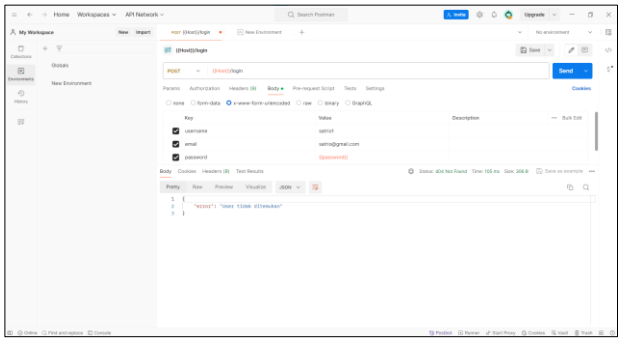
### 3.4.4. Mengirim data untuk verifikasi login



**Gambar 18. POST data untuk verifikasi login**

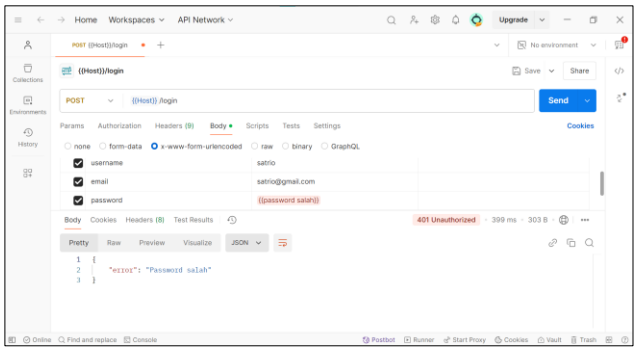
Endpoint ini digunakan untuk memvalidasi data pengguna saat mereka mencoba untuk login, dengan mengirimkan usename dan password pengguna ke server untuk diperiksa.

Gambar 18 menunjukan ketika pengujian endpoint /login yang dilakukan pengguna jika ingin masuk ke dalam website E-Asia. Ketika pengguna memasukan username dan password dengan benar, maka pengguna dapat masuk ke dalam website dan console akan mengirimkan pesan "login berhasil".



**Gambar 19. POST data dari IoT ke Database**

Namun jika pengguna salah memasukan username atau username tidak berada di dalam database, maka pengguna tidak dapat melakukan login dan console akan mengirimkan pesan "user tidak ditemukan" seperti pada gambar 19 dan pengguna tidak dapat masuk ke dalam website.




**Gambar 20. POST data dari IoT ke Database**

Jika pengguna salah dalam mengisi password maka console akan mengirimkan pesan "Password salah" seperti gambar 20 dan pengguna tidak dapat masuk ke dalam website.

### 3.5. Pengujian

Pengujian yang dilakukan menggunakan table hasil pengujian endpoint yang diambil berdasarkan hasil dari tahapan implementasi. Pengujian ini dilakukan untuk memastikan apakah semua endpoint yang telah dirancang hasilnya telah sesuai atau belum. Hasil pengujian dalam Tabel 1 menunjukkan bahwa semua skenario berhasil dilaksankan dengan baik dan sesuai, sehingga penelitian ini telah berhasil dalam integrasi sistem pemantauan kualitas udara lahan pertanian.

**Table 1. Hasil pengujian Implementasi endpoint**

| Endpoint | Skenario | Implementasi | Hasil |
|---|---|---|---|
| POST data dari Login | Data terkirim ke basis data | Menggunakan POST pada bagian login untuk dapat masuk ke halaman website | Sesuai |
| GET data dari basis data | Data dapat diambil dari basis data | Data dapat diambil dari basis data dan tampil *secara real-time* di website | Sesuai |
| POST data dari perangkat IoT | Data terkirim ke basis data | POST data dari IoT dapat tersimpan ke table device di basis data | Sesuai |
| POST data dari register | Data terkirim ke basis data | POST data dari register dapat tersimpan ke table users di basis data | Sesuai |

## 4. Kesimpulan

Penelitian ini berhasil mengembangkan arsitektur REST API yang mengintegrasikan perangkat Internet of Things (IoT), basis data, dan antarmuka pengguna dalam website pemantauan kualitas udara. Sistem ini memvalidasi, memproses, dan memformat data dari perangkat IoT, mengirimkan ke basis data dan menampilkan data dalam antarmuka berupa website melalui protokol HTTP seperti GET dan POST. Inovasi yang diterapkan adalah penerapan protokol WebSocket, yang memungkinkan komunikasi dua arah dan transmisi data secara real-time, dibandingkan dengan menggunakan protokol HTTP standar yang harus *me-refresh* website setiap akan memuat data baru. Sehingga petani dapat memantau kualitas udara, termasuk gas berbahaya seperti CO2, NO2, dan CH4 secara langsung. Hasil pengujian menunjukkan bahwa semua endpoint berfungsi sesuai dengan rencana, menandakan sistem ini dapat diandalkan dan efektif dalam memberikan informasi yang cepat dan akurat. Dengan demikian, sistem ini mendukung pengambilan keputusan yang lebih baik dalam pengelolaan lahan dan berkontribusi pada pengurangan emisi gas rumah kaca di sektor pertanian.

## Referensi

[1]. A. Sofwan, Y. Wafdulloh, M. R. Akbar, and B. Setiyono, "Sistem Pengaturan dan Pemantauan Suhu dan Kelembapan pada Ruang Budidaya Jamur Tiram Berbasis IoT (Internet of Things)," *Transmisi*, vol. 22, no. 1, pp. 1–5, 2020, doi: 10.14710/transmisi.22.1.1-5.

[2]. Markets and Markets, "Agricultural IoT Market by Hardware, Application (Precision Farming, Precision Forestry, Precision Livestock, Precision Aquaculture, Smart Greenhouse), Farm Sizr, Production Stage and Geography," 2021.

[3]. A. Jabbari, A. Humayed, F. A. Reegu, M. Uddin, Y. Gulzar, and M. Majid, "Smart Farming Revolution: Farmer's Perception and Adoption of Smart IoT Technologies for Crop Health Monitoring and Yield Prediction in Jizan, Saudi Arabia," *Sustain.*, vol. 15, no. 19, pp. 1–19, 2023, doi: 10.3390/su151914541.

[4]. L. García, L. Parra, J. M. Jimenez, J. Lloret, and P. Lorenz, "IoT-based smart irrigation systems: An overview on the recent trends on sensors and iot systems for irrigation in precision agriculture," *Sensors (Switzerland)*, vol. 20, no. 4, 2020, doi: 10.3390/s20041042.

[5]. A. H. Utomo, E. Andrianto, and ..., "Alat Ukur Tingkat Pencemaran Udara (Karbon Monoksida (Co) dan Nitrogen Dioksida (No2)) pada Sektor Pertanian Berbasis Iot," *J. Kecerdasan Buatan …*, vol. 2, no. 1, pp. 54–69, 2023.

[6]. B. L. Hartawati, "Implementasi IoT Data Storage Dengan Menggunakan Sistem Basis Data Terdistribusi Berbasis MySQL Cluster," vol. 5, no. 7, pp. 2986–2993, 2021, [Online]. Available: http://j-ptiik.ub.ac.id

[7]. E. Erwin *et al.*, *Pengantar & Penerapan Internet of Things : Konsep dasar & Penerapan IoT di berbagai Sektor*. PT. Sonpedia Publishing Indonesia, 2023.

[8]. A. Sopian, S. Mulyati, M. Syafrullah, and T. Fatimah, "Implementasi Restful Api Pada Aplikasi Monitoring Perangkat Jaringan Komputer Di Universitas Budi Luhur," in *2nd Seminar Nasional Mahasiswa Fakultas Teknologi Informasi (SENAFTI)*, 2023, pp. 508–517.

[9]. I. Sontana, A. Rahmatulloh, and A. N. Rachman, "Application Programming Interface Google Picker Sebagai Penyimpanan Data Sistem Informasi Arsip Berbasis Cloud," *J. Nas. Teknol. dan Sist. Inf.*, vol. 5, no. 1, pp. 25–32, 2019, doi: 10.25077/teknosi.v5i1.2019.25-32.

[10]. L. P. Leo, A. Ambarwari, and S. D. Putra, "Rancang Bangun Web Service Api Dan Dokumentasi Rest Api Web Portal Unit Kegiatan Mahasiswa Di Politeknik Negeri Lampung," *ROUTERS J. Sist. dan Teknol. Inf.*, vol. 1, no. 1, pp. 9–18, 2022, doi: 10.25181/rt.v1i1.2700.

[11]. W. Galindra Wardhana, I. Arwani, and B. Rahayudi, "Implementasi Teknologi Restful Web Service Dalam Pengembangan Sistem Informasi Perekaman Prestasi Mahasiswa Berbasis Website (Studi Kasus: Fakultas Teknologi Pertanian Universitas Brawijaya)," *J. Pengemb. Teknol. Inf. dan Ilmu Komput.*, vol. 4, no. 2, pp. 680–689, 2020.

[12]. B. Baharuddin, H. Wakkang, and B. Irianto, "Implementasi Web Service Dengan Metode Rest Api Untuk Integrasi Data Covid 19 Di Sulawesi Selatan," *J. Sintaks Log.*, vol. 2, no. 1, pp. 236–241, 2022, doi: 10.31850/jsilog.v2i1.1035.

[13]. A. M. N. Riady, P. Paniran, and I. M. B. Suksmadana, "Perancangan Backend Api Berbasis Rest-Api pada Aplikasi Rekomendasi Resep Makanan," *Mars J. Tek. Mesin, Ind. Elektro Dan Ilmu Komput.*, vol. 2, no. 3, 2024.

[14]. B. Badieah, A. Mujib, M. Y. Madrah, A. Riansyah, and N. M. Syaifuddin, "Implementation of RESTful Web Service on Indonesian's Integrated Breastfeeding Donor Information System," *Sistemasi*, vol. 11, no. 2, p. 455, 2022, doi: 10.32520/stmsi.v11i2.1797.

[15]. M. Zulfani and A. W. R. Emanuel, "Perancangan Model Enterprise Architecture Planning pada PT. Bestari Kalimantan," *J. JTIK (Jurnal Teknol. Inf. dan Komunikasi)*, vol. 8, no. 2, pp. 341–350, 2024, doi: 10.35870/jtik.v8i2.1795.