Implementasi Mikroservis Untuk Monitoring Kualitas Air Kolam Budidaya Ikan Berbasis IoT

1st Nathanael Dwi Cahyo
School of Electrical Engineering
Telkom University
Bandung, Indonesia
nathanaeldwi@student.telkomuniversity.ac.id

2nd Muhammad Rifqi Alhisyam
School of Electrical Engineering
Telkom University
Bandung, Indonesia
alhisyam@student.telkomuniversity.ac.id

3rd Muhammad Hasbi Nurhadi
School of Electrical Engineering
Telkom University
Bandung, Indonesia
hasbinurhadi@student.telkomuniversity.ac.id

4th Luthfia Caesa Roselina
School of Electrical Engineering
Telkom University
Bandung, Indonesia
luthfiaacr@student.telkomuniversity.ac.id

5th Sofia Naning Hertiana
School of Electrical Engineering
Telkom University
Bandung, Indonesia
sofiananing@telkomuniversity.ac.id

6th Danu Dwi Sanjoyo
School of Electrical Engineering
Telkom University
Bandung, Indonesia
danudwj@telkomuniversity.ac.id

***Abstrak* — Implementasi sistem monitoring kualitas air berbasis IoT menjadi penting untuk mendukung ekosistem dan produktivitas ikan. Metode manual sering tidak efisien, sementara sistem IoT yang ada kurang fleksibel dan skalabel. Penelitian ini mengusulkan arsitektur mikroservis berbasis IoT yang mengintegrasikan sensor pH, suhu, dan kekeruhan dengan *Cloud Firestore* untuk penyimpanan data. Mikroservis membagi tugas ke dalam beberapa container, memungkinkan pengolahan data yang fleksibel dan terdistribusi. Hasil pengujian menunjukkan akurasi tinggi (pH 92,47%, suhu 98,27%, kekeruhan 97,46%), penggunaan data rendah (5,8 MB/hari), dan latensi rata-rata 499 ms. Sistem ini efektif dan berpotensi diterapkan dalam budidaya ikan.**

***Kata kunci*— Budidaya ikan, *Internet of Things*, Kualitas air, Mikroservis, Sistem monitoring**

I. PENDAHULUAN

A. Latar Belakang

Seiring dengan perkembangan teknologi informasi, kebutuhan akan pengelolaan jaringan yang efisien dan fleksibel semakin meningkat. Di tengah tantangan tersebut, arsitektur jaringan monolitik telah muncul sebagai solusi yang sangat potensial. Arsitektur monolitik memungkinkan penempatan semua fungsi dan fitur secara terpusat dalam satu aplikasi tunggal yang besar. Penggunaan arsitektur monolitik umumnya digunakan sebagai otak dari seluruh jaringan. Namun, pendekatan ini memiliki beberapa kelemahan yang signifikan. Salah satunya adalah ketergantungan pada satu titik kegagalan tunggal yang dapat menyebabkan kerentanan kinerja yang buruk dalam skala yang besar. Selain itu, perubahan dalam konfigurasi atau kebijakan jaringan sering kali memerlukan pemeliharaan atau peningkatan pada keseluruhan sistem, yang dapat menjadi proses yang rumit dan berisiko [1].

Untuk mengatasi kelemahan tersebut, pendekatan baru telah muncul dengan menggunakan model arsitektur mikroservis. Dengan model ini, fungsi layanan yang sebelumnya terkonsentrasi dalam satu kontrol layanan yang besar, dipisahkan menjadi sejumlah layanan kecil yang independen [2]. Setiap mikroservis bertanggung jawab atas tugas-tugas spesifik dalam pengelolaan jaringan, seperti manajemen layanan atau aliran data. Penggunaan arsitektur mikroservis membawa beberapa manfaat yang signifikan. Pertama, memecah fungsi kontrol menjadi layanan kecil yang dapat mengurangi dampak dari kegagalan tunggal, meningkatkan keandalan dan ketahanan jaringan secara keseluruhan. Kedua, memungkinkan perubahan dan peningkatan pada satu bagian dari sistem tanpa mempengaruhi keseluruhan jaringan, mempercepat waktu penerapan dan mengurangi resiko kesalahan. Selain itu, dengan skala yang lebih kecil dan independen, mikroservis memfasilitasi penyesuaian dan skalabilitas yang lebih baik dalam lingkungan jaringan yang berubah dengan cepat [3].

Dengan demikian, transisi dari arsitektur monolitik ke model arsitektur mikroservis menawarkan solusi yang lebih efisien, handal, dan fleksibel untuk pengelolaan jaringan modern. Dengan mengadopsi arsitektur mikroservis penelitian ini melakukan uji coba dengan penerapan arsitektur mikroservis dalam jaringan *internet of things* monitoring kolam budidaya ikan, sehingga jaringan dapat lebih responsif terhadap perubahan lingkungan kolam ikan, lebih mudah diatur, dan lebih andal dalam menghadapi tantangan yang semakin kompleks dalam dunia jaringan yang terus berkembang. Penelitian ini bertujuan untuk mengembangkan sistem menggunakan arsitektur mikroservis. Penelitian ini akan dilakukan di PT Helmi Farm Mandiri, sebuah kawasan pengembangan budidaya ikan di Universitas Telkom Bandung, Jawa Barat. Langkah ini tidak hanya memperbaiki kinerja jaringan, tetapi juga meningkatkan skalabilitas dan efisiensi operasional secara keseluruhan, menghadirkan infrastruktur jaringan yang siap menghadapi tantangan masa depan [4].

B. Analisis Masalah

Dari hasil latar belakang dapat disimpulkan bahwa ada beberapa analisa masalah

1. Ketergantungan pada Arsitektur Monolitik. Arsitektur monolitik memiliki satu titik kegagalan yang dapat menyebabkan kinerja jaringan menjadi tidak stabil, terutama dalam skala besar.
2. Kesulitan dalam Pemeliharaan dan Skalabilitas. Perubahan atau peningkatan pada sistem monolitik memerlukan pemeliharaan seluruh sistem, yang rumit dan berisiko serta menghambat fleksibilitas dalam pengelolaan jaringan.
3. Kebutuhan akan Jaringan yang Lebih Fleksibel dan Responsif**.** Dalam konteks monitoring kolam budidaya ikan, jaringan harus dapat beradaptasi dengan perubahan lingkungan dengan cepat, yang sulit dicapai menggunakan pendekatan monolitik.

C. Tujuan

Berdasarkan latar belakang dan analisis masalah yang ada, ada beberapa tujuan dibuatnya penelitian terkait diantaranya, pengembangan layanan IoT dengan arsitektur mikroservis mengatasi kelemahan arsitektur monolitik dengan memisahkan fungsi kontrol menjadi layanan independen. Pendekatan ini meningkatkan keandalan, fleksibilitas, dan skalabilitas jaringan serta memungkinkan perubahan tanpa mengganggu sistem secara keseluruhan. Dengan mikroservis, aplikasi IoT budidaya ikan menjadi lebih efisien, handal, dan siap menghadapi tantangan jaringan di masa depan.

II. KAJIAN TEORI

Implementasi mikroservis dalam layanan budidaya ikan berbasis IoT harus mempertimbangkan berbagai aspek penting, termasuk izin dan regulasi, standar teknis, serta interoperabilitas. Regulasi seperti Peraturan Menteri Kelautan dan Perikanan Nomor 26 Tahun 2021 mewajibkan pemantauan kualitas air secara berkala, sementara standar teknis mikroservis harus memastikan modularitas, skalabilitas, dan isolasi layanan untuk meningkatkan keandalan sistem. Selain itu, aspek interoperabilitas perlu diperhatikan agar layanan dapat berintegrasi dengan perangkat IoT dan sistem lain sesuai dengan regulasi yang berlaku.

Monitoring dan manajemen layanan mikroservis juga menjadi faktor krusial dalam memastikan kinerja sistem tetap optimal. Sistem ini harus dilengkapi dengan pemantauan *real time* serta audit berkala untuk memastikan kepatuhan terhadap regulasi dan standar teknis. Selain itu, konservasi lingkungan perlu diperhatikan dengan memastikan teknologi IoT yang diterapkan mendukung pengelolaan sumber daya air yang berkelanjutan, sejalan dengan prinsip-prinsip ekologi dalam peraturan perikanan.

Aspek hak dan kewajiban pengguna juga perlu diperjelas melalui edukasi yang memadai agar pemanfaatan layanan IoT dapat dilakukan dengan benar dan sesuai aturan. Dengan mempertimbangkan semua aspek ini secara menyeluruh, implementasi mikroservis dalam budidaya ikan berbasis IoT diharapkan tidak hanya memenuhi standar keamanan dan regulasi, tetapi juga mendorong inovasi serta keberlanjutan industri perikanan di Indonesia.

A. Batasan dan Spesifikasi

1. Sensor mikrokontroler dapat mengumpulkan data, perangkat mikrokontroler harus mampu mengumpulkan data mengenai kondisi air di kolam ikan melalui sensor pH, sensor suhu, dan sensor kekeruhan air. Selain itu, mikrokontroler dapat memproses data dengan kecepatan lebih dari 160 MHz, dan akurasi lebih dari 90% memastikan analisis yang cepat dan efisien.
2. Perangkat mikrokontroler dapat terhubung dengan internet, perangkat mikrokontroler harus terhubung dengan koneksi internet, dengan kecepatan pengiriman data lebih dari 150 Mbps.
3. Perangkat mikrokontroler terintegrasi dengan *database*, perangkat mikrokontroler harus terintegrasi dengan *database* agar dapat mengirim data yang telah didapatkan ke *database* kurang dari 500 milidetik [5].
4. *Website* dapat menampilkan data secara *real time*, Dalam pembudidayaan ikan mengetahui kualitas air secara *real time* sangatlah penting, oleh karena itu sensor harus dapat mengirim data ke *database* secara *real time* dan *website* dapat menampilkan data yang didapatkan oleh sensor mikrokontroler secara *real time* kepada pengguna dengan latensi kurang dari 1 detik [6].
5. *Website* membagi layanan dan *database* secara terpisah, penggunaan arsitektur mikroservis atau bisa disebut membagi layanan secara terpisah, dapat sangat berguna dalam pengembangan budidaya ikan berbasis IoT, karena jika salah satu layanan dari sistem mengalami masalah, masalah tersebut tidak akan mempengaruhi komponen-komponen lainnya.
6. Mikroservis, adalah pendekatan arsitektur perangkat lunak yang membagi aplikasi menjadi layanan kecil, independen, dan modular. Setiap layanan memiliki tanggung jawabnya sendiri dan berkomunikasi dengan layanan lain melalui API (*Application Programming Interface*). Pendekatan ini menawarkan beberapa keuntungan untuk sistem pemantauan kualitas air *real time*, yaitu dari segi skalabilitas, ketahanan dan kemudahan pengembangan dan pemeliharaan. Dengan waktu autoregenerasi kurang dari 13 detik.

B. Metode Uji Pengukuran Spesifikasi

TABEL 1
(Spesifikasi dan Verifikasi)

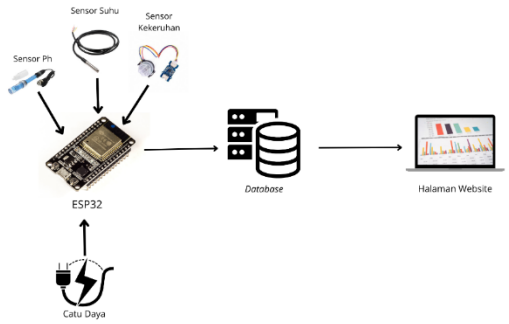
Spesifikasi	Mekanisme Pengukuran	Prosedur Pengukuran
Mikrokontroler mengumpulkan data sensor pH, suhu, dan kekeruhan (>160 MHz, akurasi >90%)	Uji kecepatan dan akurasi data	Hubungkan sensor ke mikrokontroler, amati kecepatan pengambilan dan pemrosesan data
Mikrokontroler harus terkoneksi internet (>150 Mbps)	Cek koneksi melalui serial monitor di Arduino IDE	Instal Arduino IDE, hubungkan mikrokontroler, atur *baud rate*, amati status koneksi

Mikrokontroler mengirim data ke *database* dalam <500 ms	Uji pengiriman data dengan HTTP GET	Program mikrokontroler kirim permintaan GET, amati respons *database*
Sensor mengirim data *real time* dengan latensi <1 detik	Uji kecepatan pengiriman data ke *database* dan tampilan di *website*	Hubungkan sensor, amati latensi dan pemrosesan data
Menggunakan mikroservis untuk ketahanan sistem	Monitor lalu lintas layanan dan *database*	Gunakan Docker, cek status layanan dan alokasi sumber daya
Mikroservis autoregenerasi dalam <13 detik	Evaluasi kinerja dan skalabilitas sistem	Pantau latensi antar-mikroservis, sinkronisasi data, dan ketersediaan layanan

III. METODE

A. *Desain Sistem*

Desain sistem yang akan dirancang terdiri dari tiga subsistem utama yaitu subsistem *hardware* sebagai masukan, lalu subsistem pengiriman data ke *database*, yang terakhir subsistem aplikasi atau *web application* sebagai keluaran. Berikut merupakan hubungan antar subsistem yang dirancang menjadi satu sistem.



GAMBAR 1
(Desain Sistem secara Umum)

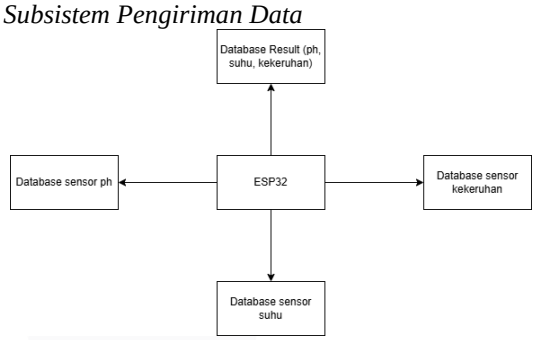
Pada Gambar 1, sistem pemantauan kualitas air kolam ikan berbasis IoT terdiri dari tiga subsistem utama: hardware, pengiriman data, dan aplikasi. Subsistem hardware menggunakan sensor pH, suhu, dan kekeruhan untuk mengumpulkan data kualitas air, yang kemudian dikirimkan secara *real time* melalui subsistem pengiriman data menggunakan ESP32 ke *database* terintegrasi. Selanjutnya, subsistem aplikasi mengolah dan menampilkan data melalui dashboard web untuk memudahkan pengguna dalam analisis dan pengambilan keputusan. Metode penelitian mencakup perancangan, implementasi, dan pengujian setiap subsistem guna memastikan akurasi sensor, keandalan transmisi data, serta efektivitas visualisasi informasi. Integrasi ketiga subsistem ini menciptakan sistem pemantauan yang efisien dan mudah diakses guna mendukung keberlanjutan budidaya ikan.

i. Subsistem Hardware

GAMBAR 2
(Subsistem *Hardware*)

Berdasarkan Gambar 2, subsistem hardware berperan dalam pengumpulan data awal dari sensor pH, suhu, dan kekeruhan. Data yang dihasilkan berupa nilai mentah ADC (Analog-to-Digital Conversion) kemudian dikonversi oleh ESP32 menjadi satuan yang dapat digunakan, seperti suhu (°C), kekeruhan (NTU), dan pH. Proses ini memastikan data yang akurat dan terkalibrasi untuk tahap pengolahan berikutnya, mendukung pengambilan keputusan yang tepat.

ii. Subsistem Pengiriman Data



GAMBAR 3
(Subsistem Pengiriman Data)

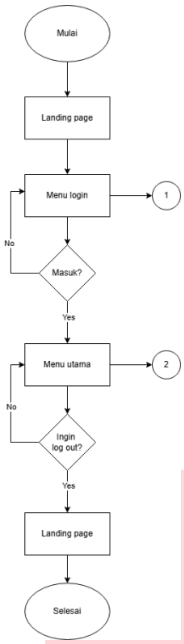
Berdasarkan Gambar 3 yang menunjukkan bahwa data yang telah diolah oleh ESP32 akan dikirimkan ke masing-masing *database* sensor, yaitu *database* sensor pH, *database* sensor suhu, *database* sensor kekeruhan, dan *database* result untuk menyimpan 3 nilai sensor dalam satu waktu, maka subsistem ini mencerminkan pendekatan arsitektur berbasis mikroservis yang terdistribusi.

iii. Subsistem Aplikasi

Pada subsistem aplikasi, akan dibagi menjadi dua bagian utama, yaitu frontend dan backend, yang masing-masing memiliki peran dan fungsi spesifik untuk mendukung operasional aplikasi secara keseluruhan.

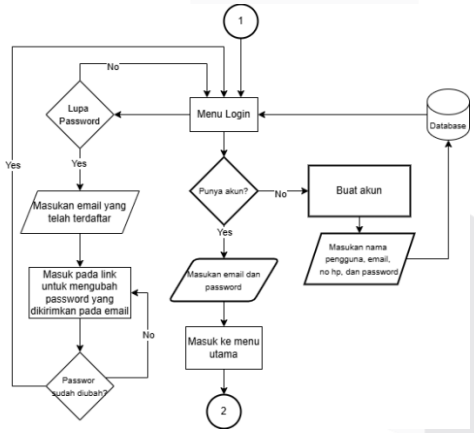
1. Frontend

Bagian ini berfungsi sebagai antarmuka pengguna (UI/UX) yang memungkinkan interaksi langsung dengan sistem. Frontend dirancang agar intuitif dan responsif, menampilkan data *real time*, grafik visualisasi kualitas air (pH, suhu, kekeruhan), serta notifikasi status. Selanjutnya, diagram alir akan menjelaskan alur interaksi pengguna dengan aplikasi, merinci langkah-langkah utama dalam operasional sistem.



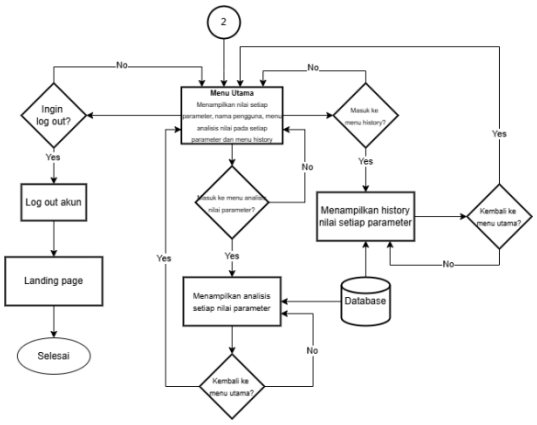
GAMBAR 4
(Alur Penggunaan Aplikasi)

Pada diagram alur di Gambar 4, proses dimulai dari *landing page*, di mana pengguna dapat memilih untuk log in atau mendaftar jika belum memiliki akun. Setelah masuk, pengguna diarahkan ke menu utama yang menyediakan berbagai fitur aplikasi. Jika memilih log out, sistem akan mengembalikan pengguna ke *landing page* untuk keluar dengan aman. Diagram ini menggambarkan jalur navigasi yang jelas, memastikan pengalaman pengguna yang lancar dan intuitif.



GAMBAR 5
(Alur Menu Login)

Diagram alur pada Gambar 5 menunjukkan langkah-langkah setelah pengguna membuka halaman log in. Pengguna dapat memilih untuk memulihkan kata sandi, mendaftar akun baru, atau langsung masuk dengan kredensial yang benar. Setelah berhasil masuk, pengguna diarahkan ke menu utama untuk mengakses fitur aplikasi. Proses ini dirancang agar fleksibel, memastikan akses yang mudah dan aman sesuai kondisi akun masing-masing.

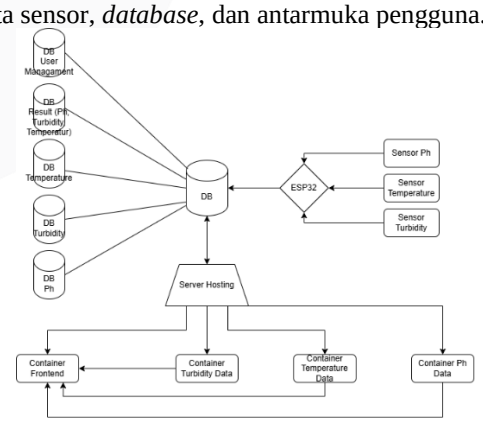


GAMBAR 6
(Alur Menu Utama)

Diagram alur pada Gambar 6 menunjukkan menu utama aplikasi yang menampilkan informasi *real time* tentang pH, suhu, dan kekeruhan air kolam. Pengguna dapat memantau kualitas air dan mengakses fitur history untuk melihat data sensor dalam bentuk tabel serta analisis yang menyajikan tren perubahan dalam grafik. Jika pengguna memilih log out, mereka akan diarahkan kembali ke *landing page*. Dengan tata letak yang terorganisir dan antarmuka intuitif, aplikasi ini mempermudah pemantauan dan pengelolaan kualitas air kolam.

2. Backend

Bagian backend bertanggung jawab mengelola lalu lintas data antara mikroservis sensor, *database*, dan API yang terhubung ke frontend. Backend memastikan data dari ESP32 diproses, disimpan, serta disediakan melalui API yang aman dan efisien. Selain itu, backend menangani validasi data, autentikasi pengguna, konfigurasi sistem, dan komunikasi antar-mikroservis. Dalam Docker, setiap mikroservis dikemas dalam container untuk memastikan layanan berjalan terpisah tetapi tetap terintegrasi. Gambar 3.7 menggambarkan interaksi antar-mikroservis melalui API dalam mengelola data sensor, *database*, dan antarmuka pengguna.

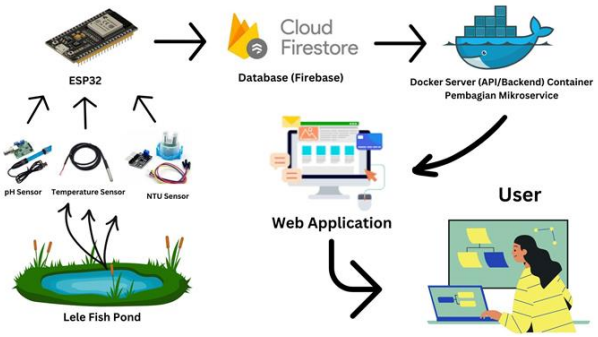


GAMBAR 7
(Alur Data dan Pembagian Mikroservis)

Diagram alur pada Gambar 7 menunjukkan alur data dan pembagian mikroservis dalam sistem. Data dari ESP32 dikirim ke masing-masing *database* sensor (pH, suhu, kekeruhan, dan hasil

keseluruhan). Setelah disimpan, server hosting mendistribusikan layanan melalui mikroservis terisolasi, yaitu mikroservis frontend, turbidity, temperature, dan pH. Masing-masing mikroservis mengelola dan mengolah data spesifik (kekeruhan, suhu, dan pH) dan berinteraksi dengan pengguna melalui frontend. Sistem ini modular, memungkinkan pengembangan dan pengujian independen dari setiap mikroservis, yang meningkatkan keandalan dan kinerja aplikasi secara keseluruhan.

B. Implementasi



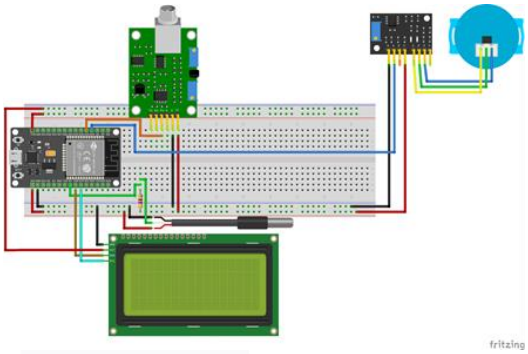
GAMBAR 8
(Desain Sistem Keseluruhan)

Pada Gambar 8 di atas, implementasi sistem ini memanfaatkan perangkat keras dan perangkat lunak yang diintegrasikan secara menyeluruh guna memberikan pemantauan *real time* terhadap kondisi air kolam. Komponen utama yang digunakan mencakup perangkat keras seperti mikrokontroler dan sensor, serta perangkat lunak yang diorganisasikan melalui arsitektur mikroservis dengan kontainerisasi Docker. Desain alur pada gambar tersebut menunjukkan keseluruhan proses implementasi mikroservis untuk *monitoring* kualitas air kolam budidaya ikan berbasis IoT.

Perangkat monitoring diletakkan, seperti yang ditunjukkan pada Gambar 9, di lokasi yang strategis di dekat kolam budidaya ikan. Pastikan perangkat terhubung ke sumber listrik yang stabil untuk menjaga kelancaran operasional. Selain itu, pastikan perangkat telah tersambung ke jaringan Wi-Fi yang telah dikonfigurasi sebelumnya dalam program ESP32. Koneksi Wi-Fi yang stabil sangat penting untuk memastikan data dari sensor dapat dikirimkan secara *real time* ke server dan ditampilkan pada aplikasi. Sebelum pemasangan, lakukan verifikasi ulang terhadap konfigurasi jaringan, seperti SSID dan password Wi-Fi, agar perangkat dapat terhubung secara otomatis tanpa gangguan.

GAMBAR 9
(Desain dan Implementasi Alat)

i. Perangkat Keras (Hardware)



GAMBAR 10
(Sirkuit Diagram)

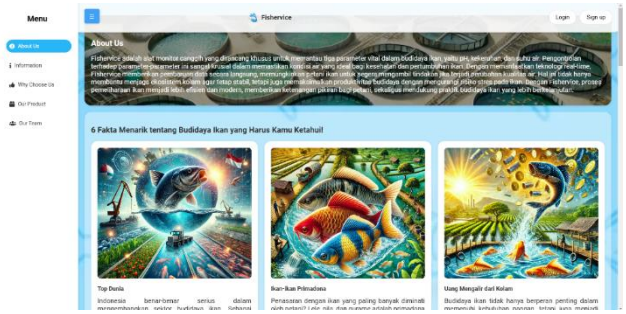
Pada Gambar 10 di atas, adalah sirkuit diagram dari projek ini, sirkuit dirancang untuk membaca parameter kualitas air seperti suhu, pH, dan tingkat kekeruhan (NTU) menggunakan sensor yang terhubung ke mikrokontroler ESP32. Data yang diperoleh kemudian ditampilkan pada layar LCD dan dikirimkan secara berkala ke Firestore melalui koneksi Wi-Fi. ESP32 diatur untuk berkomunikasi dengan berbagai sensor dan komponen melalui pin digital maupun analog. Beberapa pin penting yang digunakan adalah pin GPIO5 untuk koneksi data sensor suhu DS18B20 melalui protokol OneWire, pin GPIO33 untuk membaca tegangan dari sensor pH melalui input analog, pin GPIO34 untuk membaca tegangan dari sensor kekeruhan melalui input analog, pin GPIO18 (SDA) dan GPIO19 (SCL) untuk komunikasi I2C dengan LCD I2C.

GAMBAR *11*
(Implementasi Sirkuit Diagram)

Pada Gambar 11 diatas, merupakan implementasi dari sirkuit diagram yang telah dirancang seperti pada Gambar 10. Sensor suhu DS18B20 menggunakan protokol OneWire untuk komunikasi data. Data dari sensor diteruskan melalui pin digital GPIO5 pada ESP32, yang terhubung ke jalur data sensor suhu. Sensor pH menghasilkan tegangan analog yang sesuai dengan tingkat keasaman air. Tegangan ini dibaca melalui pin analog GPIO33 pada ESP32. Nilai ADC kemudian diolah menggunakan persamaan kalibrasi untuk menghasilkan nilai pH yang sebenarnya. Sensor kekeruhan bekerja dengan prinsip fotodioda untuk mendeteksi jumlah cahaya yang tersebar akibat partikel di dalam air. Nilai tegangan dari sensor ini dibaca melalui pin analog GPIO34 dan kemudian dikonversi menjadi nilai NTU menggunakan fungsi pemetaan dalam kode. LCD I2C menggunakan protokol komunikasi I2C untuk menampilkan data. LCD ini dihubungkan ke pin SDA (GPIO18) dan SCL (GPIO19) pada ESP32. Dengan konfigurasi alamat I2C 0x27, ESP32 dapat mengirimkan perintah dan data ke LCD untuk menampilkan informasi yang diperlukan.

ii. Aplikasi (Software)

Pada Gambar 7 Arsitektur sistem berbasis mikroservis ini dirancang untuk pemantauan kualitas air kolam ikan menggunakan sensor pH, suhu, dan kekeruhan. Data dari sensor diproses oleh ESP32 dan disimpan dalam empat *database* terpisah, termasuk satu untuk pemantauan *real time*. Sistem backend berbasis mikroservis terdiri dari container-container yang menangani frontend dan pemrosesan data tiap sensor secara independen. Keunggulan arsitektur ini terletak pada skalabilitas, fleksibilitas, dan keandalannya, memungkinkan pengembangan, perbaikan, serta penskalaan tanpa mengganggu keseluruhan sistem.



GAMBAR *12*
(Tampilan *Landing page*)

GAMBAR 13
(Tampilan *Landing page* 2)

Pada Gambar 12 dan 13 adalah *website* fisherviсe yang dirancang sebagai implementasi sistem pemantauan kualitas air kolam ikan berbasis mikroservis. *Website* ini memiliki beberapa halaman utama yang menyediakan informasi tambahan. Pada halaman Monitoring, data *real time* dari sensor suhu, pH, dan kekeruhan air ditampilkan dan diperbarui secara otomatis melalui Firebase. Dengan demikian, pengguna dapat langsung melihat kondisi terkini kolam ikan. Selain itu, fitur Analyst menyediakan indikator visual yang membantu dalam memahami tingkat optimal masing-masing parameter kualitas air.

Halaman Home berfungsi sebagai *landing page* yang memperkenalkan Fisherviсe serta menjelaskan pentingnya pemantauan kualitas air dalam budidaya ikan. Selain itu, halaman ini juga menyajikan berbagai artikel informatif, seperti fakta menarik dan tips dalam menjaga kualitas air kolam. Dengan arsitektur berbasis mikroservis dan integrasi Firebase, Fisherviсe memastikan sistem berjalan secara efisien, memungkinkan pemantauan kondisi air yang akurat dan mendukung pengelolaan kolam ikan secara lebih efektif.

IV. HASIL DAN PEMBAHASAN

A. Pengujian Sensor

TABEL 2
(Akurasi dan Error Sensor)

Sensor	Nilai	
	Rata-rata Error	Rata-rata Akurasi
Ph	7,53%	92,47%
Suhu	1,72%	98,28%
Kekeruhan	2,54%	97,46%

Tabel 2 menunjukkan hasil pengujian tiga jenis sensor yang digunakan dalam sistem pemantauan kualitas air berbasis IoT. Setiap sensor diuji dengan 240 data pengukuran, lalu dibandingkan dengan alat referensi pada setiap sensor. Hasil perbandingan ini dihitung berdasarkan rata-rata error dan rata-rata akurasi dari masing-masing sensor.

- Sensor Suhu (DS18B20) diuji dengan membandingkan hasilnya dengan Digital Thermometer. Hasil pengujian menunjukkan rata-rata error 1,72%, dengan akurasi mencapai 98,28%.
- Sensor pH (PH-4502C) diuji terhadap pH Air Analyzer. Sensor ini memiliki rata-rata error 7,53%, dengan akurasi sebesar 92,47%.
- Sensor Kekeruhan (Turbidity SEN-0175) dibandingkan dengan Turbidity Air Analyzer, menghasilkan rata-rata error 2,54%, dengan akurasi 97,46%.

B. Pengujian Penggunaan Kuota

TABEL *3*
(Penggunaan Kuota Sensor)

Rata-rata	
60,19 KB/15 Menit	239,91 KB/Jam

Tabel 3 merupakan hasil dari 240 data, rata-rata penggunaan kuota tercatat 60,19 KB/15 menit atau 239,91 KB/jam. Variasi pemakaian disebabkan oleh perbedaan jumlah parameter yang diukur, fluktuasi data (suhu, pH, kekeruhan), serta kestabilan koneksi internet. Sinyal lemah dapat meningkatkan retransmisi, tetapi fluktuasi ini masih wajar dan tidak mengganggu sistem. Dengan estimasi 175 MB/bulan untuk operasional penuh, sistem ini hemat dan ideal untuk budidaya ikan. Kuota yang ringan memungkinkan penggunaan paket data terjangkau serta integrasi lebih banyak perangkat tanpa meningkatkan biaya internet secara signifikan.

C. Pengujian Latensi

TABEL *4*
(Latensi Sensor)

Parameter	Rata-rata (ms)	Latensi Maksimum (s)	Latensi Minimum (ms)
Kekeruhan	502	1,096	102
Suhu	489	1,229	111
pH	507	1,370	101
Gabungan	499	1,370	101

Pada tabel 4 diatas, dilakukan selama 5 hari dengan 180 sample data. Hasilnya menunjukkan bahwa rata-rata latensi untuk pengukuran kekeruhan adalah 502 milidetik, dengan latensi maksimum 1,096 detik dan minimum 102 milidetik. Untuk suhu, rata-rata latensi tercatat 489 milidetik, dengan nilai tertinggi 1,229 detik dan terendah 111 milidetik. Sementara itu, pengukuran pH memiliki rata-rata latensi 507 milidetik, dengan latensi maksimum 1,370 detik dan minimum 101 milidetik. Secara keseluruhan, rata-rata latensi gabungan adalah 499 milidetik, dengan latensi maksimum 1,370 detik dan minimum 101 milidetik. Hasil ini menunjukkan bahwa sensor memiliki performa yang baik, dengan latensi di bawah 1 detik yang masih sesuai untuk aplikasi monitoring kualitas air secara *real time*. Selain itu, kestabilan latensi terlihat dari rentang nilai yang konsisten tanpa anomali ekstrem, memastikan data dapat diproses tanpa jeda yang mengganggu.

D. Pengujian Penggunaan Memori dan CPU ESP32

Pada pengujian ini, akan dilakukan pengukuran terhadap penggunaan memori dan beban kerja CPU ESP32 selama proses pengambilan, pengolahan, dan pengiriman data dari sensor pH, kekeruhan, dan suhu hingga data tampil di LCD dan dikirmkan ke *database*. Pengujian penggunaan memori bertujuan untuk mengetahui efisiensi alokasi memori agar sistem berjalan optimal tanpa mengalami kehabisan sumber daya. Sementara itu, pengujian penggunaan CPU dilakukan untuk memantau beban kerja prosesor ESP32, guna memastikan kinerja tetap stabil dan tidak mengalami *overload* selama operasional.

TABEL 5
(Penggunaan Memori dan CPU ESP32)

Parameter	Rata-rata
Penggunaan Memori Internal (Byte)	299176
Penggunaan Memori Eksternal (Byte)	2691030
Penggunaan CPU (%)	49,3

Berdasarkan hasil pengujian pada Tabel 5 diatas dengan 90 sampel data, rata-rata penggunaan memori internal pada ESP32 mencapai 299 KB, yang digunakan untuk eksekusi program, penyimpanan variabel, dan data sementara. Memori eksternal rata-rata tercatat sebesar 2,69 MB, berfungsi untuk menyimpan data lebih besar seperti file dalam SPIFFS atau LittleFS. Sementara itu, rata-rata penggunaan CPU sebesar 49,3% menunjukkan pemanfaatan prosesor dalam menangani komunikasi, pengolahan data sensor, dan eksekusi algoritma. Hasil ini menunjukkan bahwa penggunaan memori dan CPU pada ESP32 cukup optimal untuk sistem monitoring *real time* multi-*database*.

E. Pengujian Layanan Mikroservis

Pengujian layanan mikroservis dilakukan untuk mengevaluasi performa dan stabilitas dari setiap container. Pengujian dilakukan sebanyak 30 kali yang menghasilkan data berupa tabel berikut.

TABEL 6
(Uji Layanan Mikroservis)

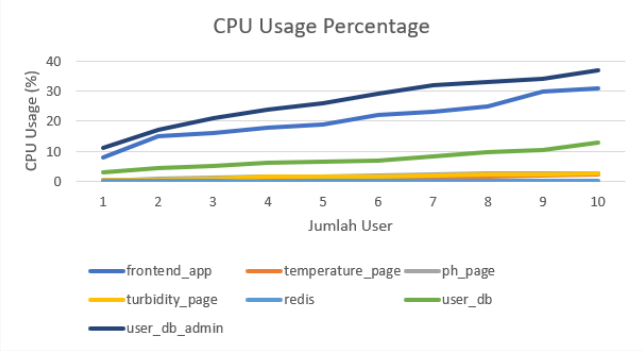
Nama *Container*	**Waktu *Startup* (detik)**	**Waktu *Shutdown* (detik)**	**Status *Container* Berjalan (Ya/Tidak)**	**Waktu *autoregenerasi* (detik)**
frontend_app	3.4	2.2	Ya	4.8
user_db	1.9	2.1	Ya	3.2
user_db_admin	1.8	2.4	Ya	3.6
ph_page	3.4	2.6	Ya	4.6
turbidity_page	3.2	2.3	Ya	4.5
temperature_page	2.9	2.2	Ya	4.5
redis	1.6	0.5	Ya	2.0

Berdasarkan Tabel 6, merupakan rata-rata hasil pengujian menunjukkan bahwa semua *container* beroperasi

dengan stabil. `Redis` memiliki waktu startup tercepat (1.6 detik) dan shutdown tercepat (0.5 detik), sementara *container* lainnya berkisar antara 1.9 hingga 3.4 detik untuk *startup* dan 2.1 hingga 2.6 detik untuk *shutdown.* Waktu *autoregenerasi* tertinggi tercatat pada `frontend_app` (4.8 detik), sedangkan `redis` paling cepat pulih (2.0 detik). Semua *container* tetap berjalan selama pengujian.

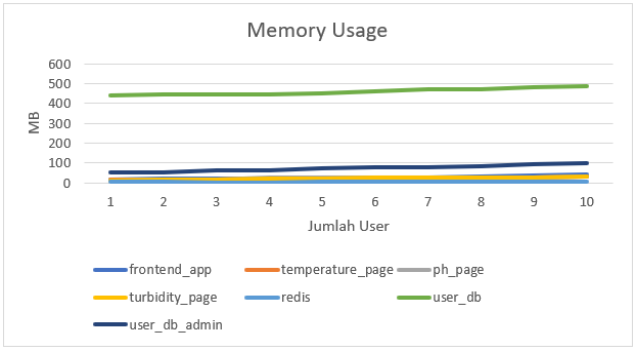
F. Pengujian Penggunaan Sumber Daya Mikroservis

Pengujian penggunaan sumber daya dilakukan dengan menambahkan 1 hingga 10 pengguna secara bertahap, dengan mencatat performa setiap *container* secara terpisah. Selain itu, kami melakukan pengujian terhadap masing-masing *container* 5 kali untuk 1 *user*, dan berkelanjutan hingga 10 *user*, hingga total pengujian ada 350 data dengan hasil rata-rata pengujian seperti berikut.



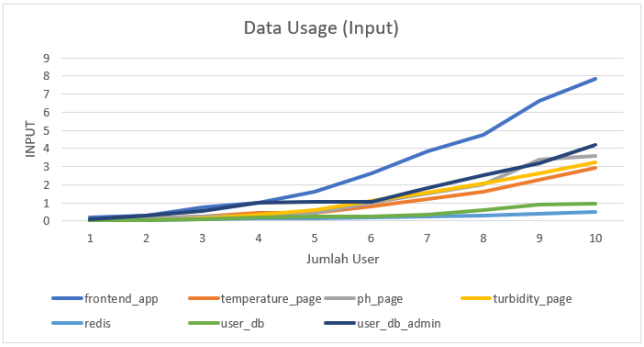
GAMBAR 14
(*CPU Usage Percentage*)

Berdasarkan Gambar 14, *user_db_admin* memiliki penggunaan CPU tertinggi, mencapai 37% pada 10 pengguna, diikuti oleh *frontend_app* yang meningkat hingga 31%. *user_db* menunjukkan kenaikan stabil dari 3% ke 13%, sementara *temperature_page*, *ph_page*, *turbidity_page*, dan *redis* tetap rendah dan konstan di kisaran 0-2%.



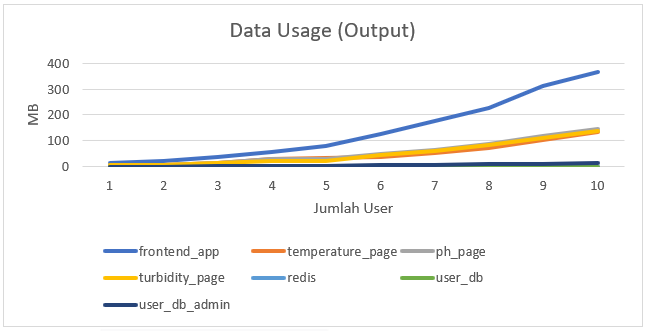
GAMBAR 15
(*Memory Usage*)

Berdasarkan Gambar 15, Penggunaan memori meningkat seiring bertambahnya pengguna. *frontend_app* naik dari 16.21 MB ke 40.32 MB, sementara *temperature_page*, *ph_page*, dan *turbidity_page* juga mengalami kenaikan signifikan. *Redis* tetap stabil dengan fluktuasi kecil, sedangkan *user_db* dan *user_db_admin* menunjukkan peningkatan terbesar, masing-masing mencapai 484.94 MB dan 98.89 MB.



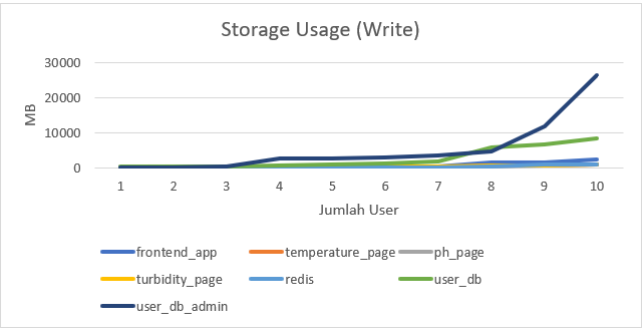
GAMBAR *16*
(*Data Usage*(*Input*))

Berdasarkan Gambar 16, *data usage input* pada berbagai *container* meningkat seiring bertambahnya pengguna. *frontend_app* mengalami lonjakan signifikan dari 0.17 MB ke 7.84 MB, sementara *turbidity_page* dan *ph_page* juga meningkat tajam. *Redis* dan *user_db* menunjukkan kenaikan yang lebih moderat.



GAMBAR 17
(*Data Usage* (*Output*))

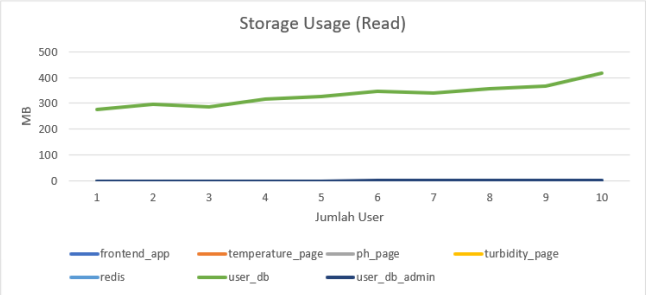
Berdasarkan Gambar 17, pengujian menunjukkan peningkatan data usage output seiring bertambahnya pengguna. *frontend_app* mengalami lonjakan dari 14 MB ke 367 MB pada 10 pengguna. *temperature_page*, *ph_page*, dan *turbidity_page* naik stabil hingga sekitar 133–147 MB, sementara *redis*, *user_db*, *dan user_db_admin* mencatat kenaikan kecil dengan output maksimal 0.3–15.3 MB.



GAMBAR 18
(*Storage Usage* (*Write*))

Berdasarkan Gambar 18, pengujian *storage usage (write)* menunjukkan bahwa *user_db_admin* memiliki penggunaan tertinggi, mencapai 26 GB pada 10 pengguna. *frontend_app* juga mengalami lonjakan signifikan dari 32.03 MB ke 2350 MB. *Temperature_page*, *ph_page*, dan *turbidity_page* mengalami peningkatan bertahap, dengan *turbidity_page* meningkat lebih cepat. *Redis* tetap stabil

dengan penggunaan rendah, sementara *user_db* meningkat signifikan hingga 8.4 GB. Secara keseluruhan, *user_db_admin* mencatat lonjakan terbesar dalam penggunaan *storage*.



GAMBAR 19
(*Storage Usage* (*Read*))

Berdasarkan Gambar 19 menunjukkan bahwa *user_db* memiliki *storage usage (read)* tertinggi, mencapai 417.2 MB pada 10 pengguna, dengan peningkatan konsisten dari 277.2 MB. *user_db_admin* dan *redis* mengalami kenaikan kecil, sementara *frontend_app*, *temperature_page*, *ph_page*, dan *turbidity_page* tidak memiliki aktivitas *storage usage (read)*, menandakan bahwa mereka tidak menulis data ke disk selama pengukuran.

G. Pembahasan

Monitoring kualitas air merupakan salah satu elemen penting dalam mendukung keberhasilan budidaya ikan lele. Standar kualitas air yang ideal untuk ikan lele mencakup tingkat kekeruhan antara 0 hingga 50 NTU, pH optimal dalam rentang 6,5 hingga 8, dan suhu air yang paling sesuai berada pada kisaran 25°C hingga 30°C[7]. Berdasarkan hasil pengujian yang penulis peroleh nilai kekeruhan yang didapat antara 19 hingga 32 NTU, pH 5,08 hingga 9,76 dan suhu 24°C hingga 27°C. Kualitas air menunjukkan kekeruhan dan suhu sudah sesuai standar, namun pH air berada di luar rentang ideal sehingga diperlukan penyesuaian pH, jika pH air rendah ditambahkan batu kapur pada kolam, jika pH tinggi ditambahkan larutan lemon pada kolam dan dilakukan pengelolaan, dengan cara jika suhu air kolam tinggi perlu ditambahkan air dengan suhu normal agar lingkungan tetap optimal untuk pertumbuhan ikan. Sedangkan jika nilai NTU tidak ideal, perlu dilakukan penggantian air. Dengan menjaga parameter-parameter ini tetap stabil, kesehatan ikan dapat terjamin, dan tingkat kematian ikan dapat diminimalkan. Untuk mencapai tujuan ini, solusi yang diusulkan oleh penulis adalah merancang sistem berbasis Internet of Things (IoT) dengan arsitektur mikroservis yang memungkinkan pemantauan kualitas air secara *real time*. Sistem ini memungkinkan pengelola kolam untuk memantau data kualitas air kapan saja melalui internet[8].

Berdasarkan hasil pengujian, sistem monitoring IoT berbasis mikroservis menunjukkan performa yang cukup baik dalam hal modularitas, integrasi, dan skalabilitas [9]. Waktu startup yang rata-rata hanya 2,6 detik dan shutdown 2 detik menunjukkan bahwa sistem mampu berjalan dengan cepat, sedangkan waktu autoregenerasi yang mencapai 3,9 detik sedikit lebih lambat, karena adanya proses sinkronisasi ulang sebelum layanan dapat kembali berjalan normal [10, 11]. Dari segi penggunaan sumber daya, container user_db_admin mencatat penggunaan CPU tertinggi hingga 37% dan memori sebesar 98 MB saat digunakan oleh 10 pengguna, sementara frontend_app menggunakan CPU hingga 31% dengan konsumsi memori sekitar 40 MB. Penggunaan CPU yang relatif tinggi pada kedua container ini dapat disebabkan oleh tingginya interaksi pengguna, terutama dalam mengakses dan mengelola data [12]. Di sisi lain, container seperti temperature_page, ph_page, turbidity_page, dan redis menunjukkan efisiensi yang baik dengan penggunaan CPU stabil di sekitar 2% dan memori antara 12 MB hingga 30 MB [12].

Namun, tantangan utama terdapat pada container user_db, yang mencatat penggunaan memori tertinggi mencapai 484,9 MB dengan aktivitas storage read sebesar 417,2 MB pada 10 pengguna. Hal ini mengindikasikan adanya akses data yang sangat intensif, karena query *database* yang belum sepenuhnya dioptimalkan melalui teknik seperti indexing atau caching [13]. Selain itu, container user_db_admin menunjukkan aktivitas storage write yang sangat tinggi, mencapai 26 GB, yang menandakan adanya proses penulisan data dalam jumlah besar, baik dalam bentuk logging maupun transaksi *database* yang berat [14]. Beban penyimpanan yang besar ini dapat berdampak pada performa sistem secara keseluruhan, terutama jika jumlah pengguna bertambah. Oleh karena itu, beberapa optimasi yang dapat dilakukan meliputi penerapan caching untuk mengurangi beban query, penggunaan load balancer untuk mendistribusikan permintaan pengguna, serta optimalisasi mekanisme logging agar tidak membebani penyimpanan [15]. Dengan peningkatan ini, sistem dapat lebih efisien dalam menangani beban kerja yang lebih besar dan memastikan skalabilitas yang lebih baik.

Selain konsumsi CPU dan memori, data usage juga mengalami peningkatan yang signifikan seiring bertambahnya jumlah pengguna. Frontend_app mencatat penggunaan data output tertinggi, yaitu 367 MB pada 10 pengguna, serta data input sebesar 7,846 MB, yang menunjukkan tingginya volume pertukaran data antara frontend dan backend. Container lainnya, seperti temperature_page, ph_page, dan turbidity_page, juga mengalami lonjakan data input hingga 3,598 MB pada 10 pengguna, dengan masing-masing mencatat data output sebesar 133,6 MB, 147 MB, dan 139,4 MB. Distribusi sumber daya yang stabil ini mengindikasikan bahwa arsitektur mikroservis mampu mendukung fleksibilitas dalam pembaruan dan pengembangan sistem lebih lanjut, seperti integrasi sensor baru atau fitur analisis data yang lebih kompleks. Namun, frontend_app masih memerlukan optimalisasi agar lebih efisien dalam menangani beban data yang besar, misalnya dengan menerapkan teknik kompresi data, optimasi caching, atau mekanisme pengelolaan request yang lebih baik. Dengan perbaikan ini, sistem dapat lebih responsif dan efisien dalam menghadapi pertumbuhan jumlah pengguna dan volume data yang semakin meningkat.

Hasil pengujian menunjukkan bahwa sistem monitoring berbasis IoT ini mampu mengumpulkan dan mengirim data dari sensor pH, suhu, dan kekeruhan sebanyak 144 kali per hari atau setiap 10 menit sekali. Data yang dikirim berhasil ditampilkan secara *real time* di dashboard dengan tingkat sinkronisasi mencapai 100%, menandakan bahwa sistem memiliki keandalan tinggi dalam mengelola data. Pengujian pada mikrokontroler ESP32 menunjukkan efisiensi dalam konsumsi internet, dengan rata-rata penggunaan data sebesar

239,91 KB per jam atau sekitar 5,75 MB per hari, yang tergolong hemat untuk sistem berbasis IoT. Selain itu, latensi rata-rata sebesar 499 milidetik menunjukkan bahwa sistem cukup responsif dalam menangani komunikasi antara sensor dan server [16]. Dari segi akurasi, sensor pH mencatat tingkat akurasi sebesar 92,47%, sensor suhu 98,27%, dan sensor kekeruhan 97,46%. Meskipun terdapat sedikit fluktuasi akibat aktivitas ikan dan perubahan cuaca, tingkat akurasi ini tetap berada dalam batas yang sangat baik untuk sistem monitoring lingkungan perairan [17-20].

Selain performa komunikasi dan akurasi sensor, pengujian juga mencatat penggunaan memori internal pada ESP32 mencapai 299 KB, yang digunakan untuk proses eksekusi program, penyimpanan variabel, dan data sementara. Sementara itu, rata-rata penggunaan memori eksternal sebesar 2,69 MB menunjukkan pemanfaatan ruang penyimpanan dalam SPIFFS atau LittleFS untuk menyimpan data lebih besar serta program yang disimpan dalam flash memory sebelum dieksekusi. Penggunaan CPU rata-rata sebesar 49,3% menunjukkan efisiensi prosesor dalam menangani berbagai tugas, termasuk komunikasi, pemrosesan data sensor, dan eksekusi algoritma [21]. Dengan performa yang masih dalam batas ideal ini, ESP32 masih memiliki kapasitas yang cukup untuk pengembangan lebih lanjut, seperti penambahan fitur atau integrasi sensor tambahan [21]. Hal ini menunjukkan bahwa sistem berbasis ESP32 dapat dikembangkan lebih lanjut tanpa mengalami penurunan performa yang signifikan, menjadikannya solusi yang fleksibel dan skalabel untuk aplikasi monitoring IoT dalam berbagai kondisi lingkungan.

V. KESIMPULAN

Implementasi arsitektur mikroservis berbasis IoT meningkatkan efisiensi dalam sistem monitoring kualitas air kolam ikan. Dengan pembagian tugas dalam container mikroservis yang independen, sistem dapat memproses data secara fleksibel tanpa saling memengaruhi, bahkan saat terjadi kegagalan pada salah satu layanan. Pendekatan ini lebih unggul dibandingkan arsitektur monolitik untuk monitoring berbasis IoT. Sistem monitoring yang dikembangkan menunjukkan akurasi tinggi dengan rata-rata 92,47% untuk pH, 98,27% untuk suhu, dan 97,46% untuk kekeruhan. Dengan penggunaan data rendah 5,8 MB/hari dan latensi rata-rata 499 milidetik, sistem memungkinkan pemantauan kondisi air secara *real time*, memberikan waktu yang cukup untuk tindakan preventif. ESP32 digunakan dengan konsumsi 299 KB memori internal, 2,69 MB memori eksternal, dan CPU 49,3%, tetap ideal untuk pengembangan fitur tambahan tanpa penurunan performa.

Kombinasi ESP32 dan Docker terbukti optimal, dengan ESP32 menawarkan performa tinggi dan efisiensi daya, serta Docker memungkinkan pengelolaan layanan mikroservis yang fleksibel. Arsitektur ini mendukung skalabilitas, memungkinkan peningkatan jumlah perangkat atau sensor tanpa mengorbankan stabilitas sistem. Dashboard monitoring menyajikan data *real time* dalam format visual yang mudah dipahami, seperti grafik dan tabel, mendukung analisis cepat dan respons tepat terhadap perubahan kualitas air. Mayoritas pengguna memberikan respon positif (94,8%), menilai *website* monitoring sebagai mudah digunakan, informatif, dan intuitif.

REFERENSI

[1] Ponce Mella, F., Márquez, G., & Astudillo, H. 2019, September 10. *Migrating from monolithic architecture to microservices: A Rapid Review*. *International Conference of the Chilean Computer Science Society*, SCCC 2019.

[2] Nugroho, A. C. (2023). Tinjauan Naratif Tentang Optimasi Perangkat Lunak Sistem dengan Arsitektur Mikroservis. Jurnal Ilmu Data, 3(1).

[3] Arzo, Sisay Tadesse. "*Journal of Network and Systems Management*." *MSN: A Playground Framework for Design and Evaluation of MicroServices-Based sdN Controller*, 2021, p. 11.

[4] Gos, K., & Zabierowski, W. 2020. *The Comparison of Microservice and Monolithic Architecture* (hlm. 153). 2020 IEEE XVIth International Conference on the Perspective Technologies and Methods in MEMS Design (MEMSTECH).

[5] Redha, F. S. (2023). Perbandingan Performa *Web Services* Yang Dibangun Menggunakan Arsitektur *Monolithic* dan *Microservices* pada Sistem *Point of Sales*. JATISI (Jurnal Teknik Informatika dan Sistem Informasi), 10(1), 406-420.

[6] Siagian, N., Tamba, T. E., Situmorang, H. H. O., & Samosir, H. (2021). Aplikasi Apotek Berbasis *Web* Menggunakan Arsitektur *Microservices* (Studi Kasus Apotek Glen, Kab. Toba). *Journal of Applied Technology and Informatics Indonesia*, 1(2), 22-28.

[7] M. T. Atilla, N. Azhari, E. Sulistyo, dan Irwan, "Sistem Kontrol dan Monitoring Kualitas Air pada Budidaya Ikan Lele dengan Media Kolam Berbasis IoT," Prosiding Seminar Nasional Inovasi Teknologi Terapan (SNITT), 2022, pp. 1–6.

[8] M. Sium, J. Morshed, M. Hasan, dan T. Rahman, "*Remote Sensing Kit for Contamination Event Detection in Water," 2019 IEEE Region 10 Humanitarian Technology Conference (R10-HTC),* Depok, Indonesia, November 2019, pp. 175-180.

[9] Tapia Leon, F., Mora, M., Fuertes, W., Aules, H., Flores, E., & Toulkeridis, T. 2020. *From Monolithic Systems to Microservices: A Comparative Study of Performance. Applied Sciences*, 10, 5797.

[10]Ponce Mella, F., Márquez, G., & Astudillo, H, *"Migrating from monolithic architecture to microservices: A Rapid Review", International Conference of the Chilean Computer Science Society, SCCC,* 2019.Syafiqoh, U., Sunardi, S., & Yudhana, A. (2018). Pengembangan *Wireless* Sensor Network Berbasis *Internet of Things* untuk Sistem Pemantauan Kualitas Air dan Tanah Pertanian. Teknik Elektro, Universitas Ahmad Dahlan, Yogyakarta.

[11] Pankowski, A., & Powroźnik, P. (2023). *Comparison of application container orchestration platforms. Journal of Computer Sciences Institute*, 29, 383-390.

[12] Putri, A. R., Munadi, R., & Negara, R. M. (2020). *Performance analysis of multi services on container Docker, LXC, and LXD. Bulletin of Electrical Engineering and Informatics*, 9(5), 2008-2011

[13] Potdar, A. M., Narayan, D. G., Kengond, S., & Mulla, M. M. (2020). *Performance evaluation of docker container and virtual machine. Procedia Computer Science*, 171, 1419-1428.

[14] Hermadi, I., & Nurhadryani, Y. (2023). Analisis Uji Performa Aplikasi Dari Hasil Implementasi *Refactoring*

Arsitektur Monolitik Ke Mikroservis dengan *Decomposition* dan *Strangler Pattern*. Jurnal Sistem Cerdas, 6(3), 189-203.
[15] Dwiyatno, S., Rachmat, E., Sari, A. P., & Gustiawan, O. (2020). Implementasi virtualisasi server berbasis *docker container*. PROSISKO: Jurnal Pengembangan Riset Dan Observasi Sistem Komputer, 7(2), 165-175.
[16] Smith, J. (2020). *Real-Time Monitoring Systems: Performance Metrics and Benchmarks. Springer Publishing.*
[17] M. T. Atilla, N. Azhari, E. Sulistyo, dan Irwan, "Sistem Kontrol dan Monitoring Kualitas Air pada Budidaya Ikan Lele dengan Media Kolam Berbasis IoT," Prosiding Seminar Nasional Inovasi Teknologi Terapan (SNITT), 2022, pp. 1–6.
[18] Manurung, C. T. H., Arifin, J., Syifa, F. T., & Rochmanto, R. A. (2022). Pemanfaatan ESP32 sebagai Sistem Pemantauan Kualitas Air Keran Siap Minum secara *Real-Time* Menggunakan Aplikasi. *Utilization of ESP32 as A Real-Time Ready-to-Drink Tap Water Quality Monitoring System Using an Application*. Prodi S1 Teknik Telekomunikasi, Prodi Teknik Elektro, Institut Teknologi Telkom Purwokerto.
[19] Syafiqoh, U., Sunardi, S., & Yudhana, A. (2018). Pengembangan *Wireless* Sensor Network Berbasis *Internet of Things* untuk Sistem Pemantauan Kualitas Air dan Tanah Pertanian. Teknik Elektro, Universitas Ahmad Dahlan, Yogyakarta.
[20] Nur'aeni Lateko, Salmawaty Tansa, Raghel Yunginger, dan Iskandar Z. Nasibu, "Monitoring Kualitas Air Sungai (Kekeruhan, Suhu, TDS, pH) Menggunakan Mikrokontroler Atmega 328," Jambura *Journal of Electrical and Electronics Engineering* (JJEEE), vol. 6, no. 1, Januari 2024, pp. 1–7.
[21] Sabbatini, M. (2024). *Hardening IoT Devices: An Analysis of the ESP32 Microcontroller.*