**JISKA (Jurnal Informatika Sunan Kalijaga)**
**Vol. 10, No. 3, SEPTEMBER, 2025, Pp. 341 – 350**
**ISSN: 2527 – 5836 (print) | 2528 – 0074 (online)**

# Arsitektur Microservice untuk Optimalisasi Aplikasi Eco-Maps dalam Mendukung Kampus Ramah Lingkungan

**Alam Rahmatulloh [(1)*], Rohmat Gunawan [(2)], Randi Rizal [(3)]**
Departemen Informatika, Universitas Siliwangi, Tasikmalaya, Indonesia
e-mail : {alam,rohmatgunawan,randirizal}@unsil.ac.id.
* Penulis korespondensi.
Artikel ini diajukan 30 Agustus 2024, direvisi 19 Maret 2025, diterima 22 Maret 2025, dan dipublikasikan 30 September 2025.

**Abstract**

*The implementation of environmentally friendly campus concepts has become increasingly crucial in addressing global environmental challenges. Eco-Maps is an application designed to visualize and manage sustainability efforts on campus, including energy management, waste management, and sustainable transportation initiatives. To enhance efficiency and flexibility, this study discusses the application of a microservice architecture in Eco-Maps. This architecture supports faster and more efficient development, testing, and deployment, while enabling horizontal scalability to manage high complexity and large data volumes. By separating application functions into independent services, microservices facilitate maintenance and updates while minimizing the impact of failures in individual services. This study also reviews the integration of containerization technologies, such as Docker and Kubernetes, to support microservice implementation. Through these technologies, the application can be deployed quickly and consistently across various environments, from development to production. System testing was conducted using load testing and stress testing methods, as shown in Tables 3 and 4. The results demonstrate that the average response time across ten iterations was 745.9 ms, with an average CPU usage of 44.38%. These findings confirm that processing load directly affects CPU efficiency and overall system performance.*

***Keywords: Eco-Maps, Microservice Architecture, Sustainable Campus, Docker, Kubernetes, System Performance***

**Abstrak**

Penerapan konsep kampus ramah lingkungan semakin krusial dalam menghadapi tantangan lingkungan global. Eco-Maps merupakan aplikasi yang dikembangkan untuk memvisualisasikan dan mengelola upaya keberlanjutan di kampus, seperti manajemen energi, pengelolaan limbah, dan transportasi berkelanjutan. Untuk meningkatkan efisiensi dan fleksibilitas, penelitian ini membahas penerapan arsitektur *microservice* pada Eco-Maps. Arsitektur ini mendukung pengembangan, pengujian, dan penyebaran yang lebih cepat serta efisien, sekaligus memungkinkan skalabilitas horizontal dalam mengelola kompleksitas dan volume data yang besar. Dengan memisahkan fungsi aplikasi ke dalam layanan independen, *microservice* memudahkan pemeliharaan dan pembaruan, serta meminimalkan dampak apabila terjadi gangguan pada salah satu layanan. Penelitian ini juga meninjau integrasi teknologi kontainerisasi seperti Docker dan manajemen orkestrasi Kubernetes untuk mendukung implementasi *microservice*. Melalui teknologi tersebut, aplikasi dapat dideploy secara cepat dan konsisten di berbagai lingkungan, mulai dari pengembangan hingga produksi. Pengujian sistem dilakukan menggunakan metode load test dan stress test, sebagaimana ditunjukkan pada Tabel 3 dan Tabel 4. Hasil menunjukkan bahwa rata-rata waktu respons dari sepuluh iterasi adalah 745,9 ms dengan rata-rata penggunaan CPU sebesar 44,38%. Temuan ini menegaskan bahwa beban pemrosesan memiliki pengaruh langsung terhadap efisiensi CPU dan kinerja keseluruhan sistem.

**Kata Kunci: Eco-Maps, Arsitektur Microservice, Kampus Ramah Lingkungan, Docker, Kubernetes, Kinerja Sistem**

## 1. PENDAHULUAN

Dalam era digital yang terus berkembang pesat, teknologi informasi memainkan peran yang semakin penting dalam mendukung berbagai aspek kehidupan, termasuk dalam upaya mewujudkan kampus ramah lingkungan (Abdullai et al., 2022; Abdulmouti et al., 2022; Faizah & Nugraheni, 2024; Trevisan et al., 2023). Kampus-kampus di seluruh dunia kini berupaya untuk menerapkan konsep keberlanjutan melalui berbagai inisiatif yang bertujuan untuk mengurangi dampak lingkungan, meningkatkan efisiensi energi, dan mengelola sumber daya secara lebih bijak (Khan & Ximei, 2022; Sugiarto et al., 2022). Salah satu alat yang dapat digunakan untuk mendukung inisiatif-inisiatif ini adalah aplikasi berbasis Eco-Maps, sebuah platform digital yang dirancang untuk membantu universitas dalam memantau dan mengelola infrastruktur hijau, penggunaan energi, pengelolaan limbah, serta berbagai kegiatan lain yang berkontribusi terhadap keberlanjutan kampus (Aasa et al., 2020; Amelia et al., 2023; Martins et al., 2021; Ribeiro et al., 2021).

Namun, seiring dengan meningkatnya kompleksitas data dan kebutuhan untuk memberikan layanan yang lebih cepat dan responsif, arsitektur perangkat lunak yang digunakan untuk mengembangkan aplikasi Eco-Maps sering kali menghadapi tantangan besar (Dimov et al., 2022; Palomo et al., 2018; Xiao, 2024). Arsitektur monolitik tradisional, yang menggabungkan semua fungsi aplikasi ke dalam satu kesatuan, sering kali tidak mampu menangani tuntutan ini dengan baik (Iqbal et al., 2023; Sidiq et al., 2024). Sebagai salah satu solusi, banyaknya pengembang perangkat lunak mulai beralih ke arsitektur *microservice*, sebuah pendekatan desain yang membagi aplikasi menjadi layanan-layanan kecil yang dapat dioperasikan secara independen namun tetap dapat saling berintegrasi dengan baik (Santos et al., 2024; Weerasinghe & Perera, 2024).

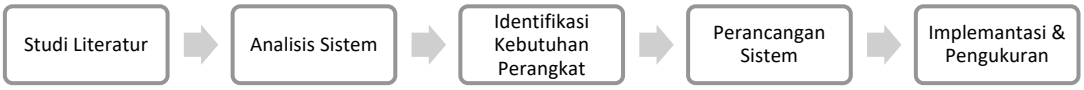
Arsitektur *microservice* menawarkan sejumlah keunggulan yang sangat relevan dengan kebutuhan pengembangan aplikasi Eco-Maps (Kautsar et al., 2023). Dengan membagi aplikasi menjadi layanan-layanan yang lebih kecil dan lebih terfokus, setiap layanan dapat dikembangkan, diuji, dan dideploy secara terpisah tanpa harus mempengaruhi layanan lain yang sudah berjalan (Telang, 2023). Hal ini tidak hanya memungkinkan pengembangan yang lebih cepat dan efisien, tetapi juga meningkatkan fleksibilitas dan skalabilitas sistem. Dalam konteks kampus ramah lingkungan, arsitektur *microservice* memungkinkan aplikasi Eco-Maps untuk terus berkembang dan beradaptasi dengan kebutuhan yang dinamis, seperti perubahan dalam manajemen energi, pengelolaan transportasi, atau integrasi dengan sistem IoT yang ada.

Penerapan arsitektur *microservice* pada aplikasi Eco-Maps juga membuka peluang untuk integrasi yang lebih baik dengan sistem lain yang mungkin sudah ada di kampus. Sebagai contoh, aplikasi ini dapat terhubung dengan sistem manajemen gedung yang cerdas, jaringan sensor IoT, serta platform analitik data yang digunakan untuk memantau dan mengoptimalkan penggunaan sumber daya di kampus. Dengan pendekatan ini, aplikasi Eco-Maps tidak hanya berfungsi sebagai alat yang berdiri sendiri, tetapi juga sebagai bagian dari ekosistem digital yang lebih besar yang mendukung upaya kampus untuk menjadi lebih ramah lingkungan. Oleh karena itu, penelitian ini bertujuan untuk mengeksplorasi penerapan arsitektur *microservice* dalam pengembangan aplikasi Eco-Maps dan bagaimana pendekatan ini dapat mengoptimalkan fungsionalitas serta kinerja aplikasi dalam mendukung visi kampus ramah lingkungan.

## 2. METODE PENELITIAN

Metode penelitian ini terdiri dari lima tahap utama, yaitu: studi literatur, analisis sistem, identifikasi kebutuhan perangkat, perancangan sistem, serta implementasi dan pengukuran, sebagaimana ditunjukkan pada Gambar 1. Tahap studi literatur dilakukan dengan menelaah berbagai jurnal dan referensi relevan untuk memperoleh dasar teori serta pemahaman mendalam terkait topik penelitian. Selanjutnya, tahap analisis sistem bertujuan untuk mengidentifikasi permasalahan yang ada dan merumuskan solusi yang sesuai. Proses kemudian dilanjutkan dengan identifikasi kebutuhan perangkat, mencakup perangkat keras maupun perangkat lunak yang diperlukan dalam penelitian. Tahap perancangan sistem meliputi penyusunan alur kerja serta struktur

sistem. Terakhir, tahap implementasi dan pengukuran dilakukan untuk menguji sistem yang telah dirancang serta mengevaluasi kinerjanya secara menyeluruh.




**Gambar 1 Metodologi Penelitian**

### 2.1 Studi Literatur

Kegiatan pada tahap ini mencakup peninjauan mendalam terhadap penelitian yang ada serta pemahaman menyeluruh mengenai seluruh informasi terkait *microservice*. Selain itu, sumber referensi yang berkaitan dengan teknologi kontainer dan arsitektur berbasis API juga ditelaah secara komprehensif. Penelaahan ini tidak hanya terbatas pada literatur umum tetapi juga mencakup jurnal-jurnal ilmiah yang relevan dan telah diterbitkan, guna memastikan pemahaman yang mendalam dan akurat terhadap konsep-konsep yang mendasari. Dengan pendekatan ini, diharapkan mampu menghasilkan landasan teoritis yang kuat untuk pengembangan lebih lanjut.

### 2.2 Analisis Sistem

Untuk mengevaluasi kinerja *microservice* dalam penelitian ini sekaligus mendukung optimalisasi aplikasi Eco-Maps dalam mewujudkan kampus ramah lingkungan, dikembangkan sebuah prototipe aplikasi yang mengintegrasikan berbagai layanan pengelolaan lingkungan kampus. Sistem informasi ini, sebagaimana digambarkan pada Gambar 2, terdiri atas empat layanan utama, yaitu Auth Service (layanan 1), Eco-Map Service (layanan 2), Waste Management Service (layanan 3), dan Energy Monitoring Service (layanan 4), serta NATS Streaming Service sebagai event-bus. Seluruh layanan terhubung dengan basis data yang digunakan, yaitu MongoDB.

Auth Service bertugas menangani seluruh aktivitas autentikasi dan manajemen pengguna. Eco-Map Service menyediakan peta interaktif kampus yang menampilkan informasi mengenai area hijau, titik daur ulang, serta lokasi-lokasi penting lainnya. Waste Management Service mengelola proses pengumpulan, pemilahan, dan pembuangan limbah secara efisien. Energy Monitoring Service memantau konsumsi energi di seluruh kampus sekaligus memberikan rekomendasi penghematan energi. Terakhir, NATS Streaming Service berfungsi sebagai penghubung utama untuk pertukaran data antar-layanan dengan mekanisme *publishing* (mengirim data) dan *listening* (menerima data).

### 2.3 Identifikasi Kebutuhan Perangkat

Proses pengukuran dalam penelitian ini memerlukan identifikasi perangkat keras dan perangkat lunak untuk memastikan sistem dapat berjalan secara optimal. Spesifikasi perangkat keras yang digunakan disajikan secara rinci pada Tabel 1, memuat informasi mengenai komponen utama, seperti prosesor, memori, dan kapasitas penyimpanan. Sedangkan spesifikasi perangkat lunak yang mendukung penelitian ditampilkan pada Tabel 2, mencakup sistem operasi, bahasa pemrograman, serta perangkat pendukung lainnya yang digunakan selama proses implementasi.

**Tabel 1 Spesifikasi Perangkat Keras**

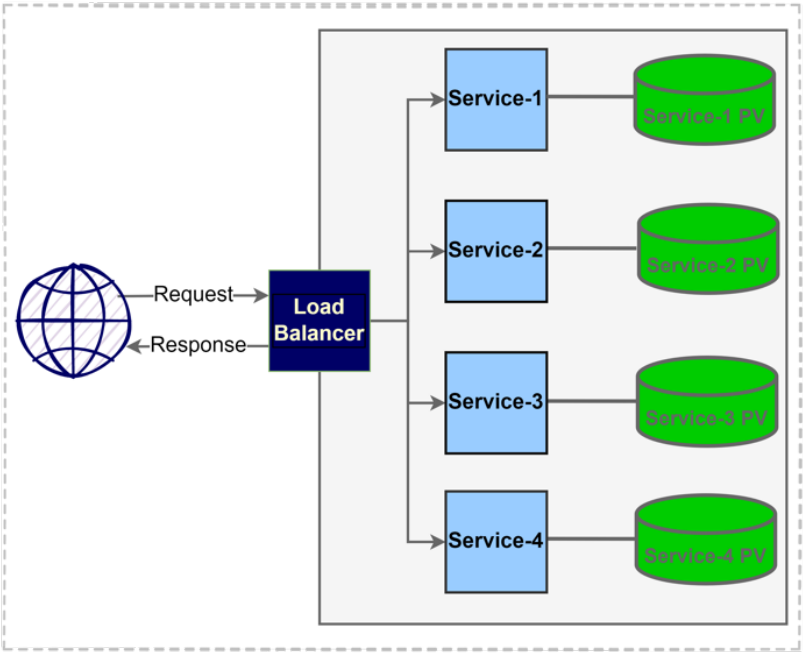
| No. | Nama | Deskripsi |
|---|---|---|
| 1 | CPU | AMD A4-9120 RADEON R3, 4 COMPUTE CORES 2C + 2G @ 2x 2.25GHz |
| 2 | GPU | AMD STONEY |
| 3 | RAM | 8GB |
| 4 | Kernel | x86_64 Linux 5.10.70-1-MANJARO |
| 5 | Disk | 110GB |
| 6 | Operating System | Manjaro 21.1.6 Pahvo (Linux) |

**Tabel 2 Spesifikasi Perangkat Lunak**

| No. | Nama | Deskripsi |
|---|---|---|
| 1 | Docker Latest Version | Application-oriented container |
| 2 | Kubernetes v1.22.2 | Container orchestrator |
| 3 | Skaffold v2 beta20 | Kubernetes support |
| 4 | NATS Streaming v0.22 | Event bus |
| 5 | Typescript v4.4.4 | Programming language |
| 6 | Node.js v17.1.0 | Runtime environment |
| 7 | Express.js v4.17.1 | App framework |
| 8 | MongoDB v5.0 | Database |
| 9 | Jest v27.1.0 | Testing framework |
| 10 | Stan.js v0.3.2 | Client communication tool |

## 2.4 Perancangan Sistem

Pada tahap ini, prototipe aplikasi berbasis layanan mikro dikembangkan dengan menerapkan prinsip komunikasi data dalam arsitektur *microservice*. Prototipe aplikasi terdiri atas empat layanan utama, yaitu Service-1, Service-2, Service-3, dan Service-4, yang masing-masing terhubung ke basis data tersendiri sebagaimana ditunjukkan pada Gambar 2. Gambar 2 memperlihatkan arsitektur *microservice* yang terdiri dari beberapa layanan (Service-1 hingga Service-4) yang dikelola oleh sebuah Load Balancer. Load Balancer berfungsi mendistribusikan permintaan (request) dari pengguna secara merata ke setiap layanan untuk mencegah terjadinya kelebihan beban pada salah satu layanan. Masing-masing layanan memiliki penyimpanan data tersendiri (Persistent Volume) yang terhubung langsung, sehingga memungkinkan penyimpanan data secara persisten. Alur permintaan dan respons ditunjukkan oleh panah: permintaan dari pengguna masuk melalui Load Balancer, kemudian diproses oleh layanan terkait, dan hasilnya dikirim kembali ke pengguna melalui Load Balancer. Dengan rancangan ini, arsitektur *microservice* diharapkan mampu memberikan efisiensi, skalabilitas, serta ketersediaan layanan yang optimal.




**Gambar 2 Communication Process in *Microservice***

### 2.5 Implementasi dan Pengukuran

Pada tahap ini, dilakukan pengujian dengan metode *load* dan *stess test*. Selain itu, waktu respons (ms), tingkat kesalahan (%) (Hong et al., 2018), dan Penggunaan CPU (ms) diukur pada arsitektur layanan mikro berbasis *microservice*. Pengukuran waktu respons dan tingkat kesalahan dilakukan dengan bantuan Apache JMeter dan API node.js untuk mengukur penggunaan CPU. Percobaan dilakukan dengan mengirimkan 100 paket data permintaan dari klien ke aplikasi pada server berbasis layanan mikro. Setiap percobaan diulang sepuluh kali dan hasil percobaan dicatat dalam tabel.

## 3. HASIL DAN PEMBAHASAN

Pada tahap ini dibahas implementasi sistem Eco-Maps beserta hasil pengukuran waktu respons dan penggunaan CPU dalam arsitektur berbasis *microservice*. Implementasi ini menunjukkan bagaimana arsitektur *microservice* mendukung integrasi serta pengelolaan data lingkungan secara efisien. Pengukuran waktu respons dilakukan untuk mengevaluasi kecepatan sistem dalam menangani permintaan API. Arsitektur *microservice* pada Eco-Maps mencakup empat layanan utama, yaitu:

1) Auth Service, dengan endpoint untuk registrasi (POST /auth/register), login (POST /auth/login), dan verifikasi token (GET /auth/verify).
2) Eco-Map Service, menyediakan API untuk memperoleh daftar lokasi ramah lingkungan (GET /eco-map/locations) dan rute menuju lokasi tertentu (GET /eco-map/route).
3) Waste Management Service, dengan endpoint untuk penjadwalan pengelolaan limbah (POST /waste-management/schedule) dan pengambilan laporan pengelolaan limbah (GET /waste-management/reports).
4) Energy Monitoring Service, menyediakan API untuk memantau konsumsi energi (GET /energy-monitoring/usage) dan melaporkan anomali energi (POST /energy-monitoring/report).

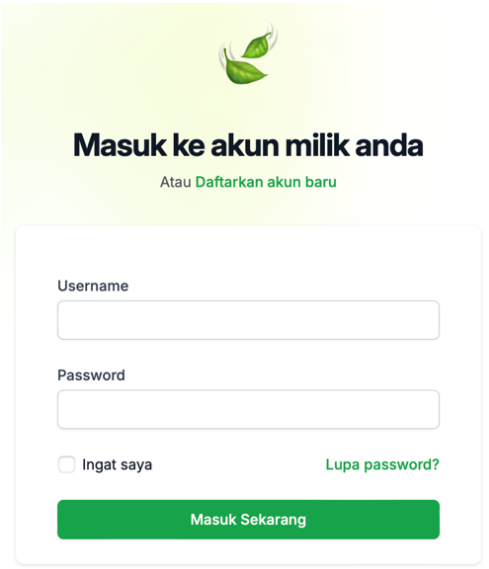
Komunikasi antar-layanan dilakukan melalui NATS Streaming menggunakan event auth.user.created, waste.management.schedule.created, dan energy.monitoring.report.submit untuk mendukung sinkronisasi data secara *event-driven*. Selain itu, pengukuran penggunaan CPU dilakukan untuk menilai efisiensi pemrosesan dalam konteks arsitektur *microservice*. Hasil pengujian menunjukkan bahwa sistem Eco-Maps berbasis *microservice* mampu memberikan kinerja optimal, baik dari segi kecepatan respons maupun efisiensi penggunaan sumber daya CPU, sehingga efektif dalam mendukung operasional sistem secara keseluruhan.

### 3.1 Implementasi Sistem Eco-Maps

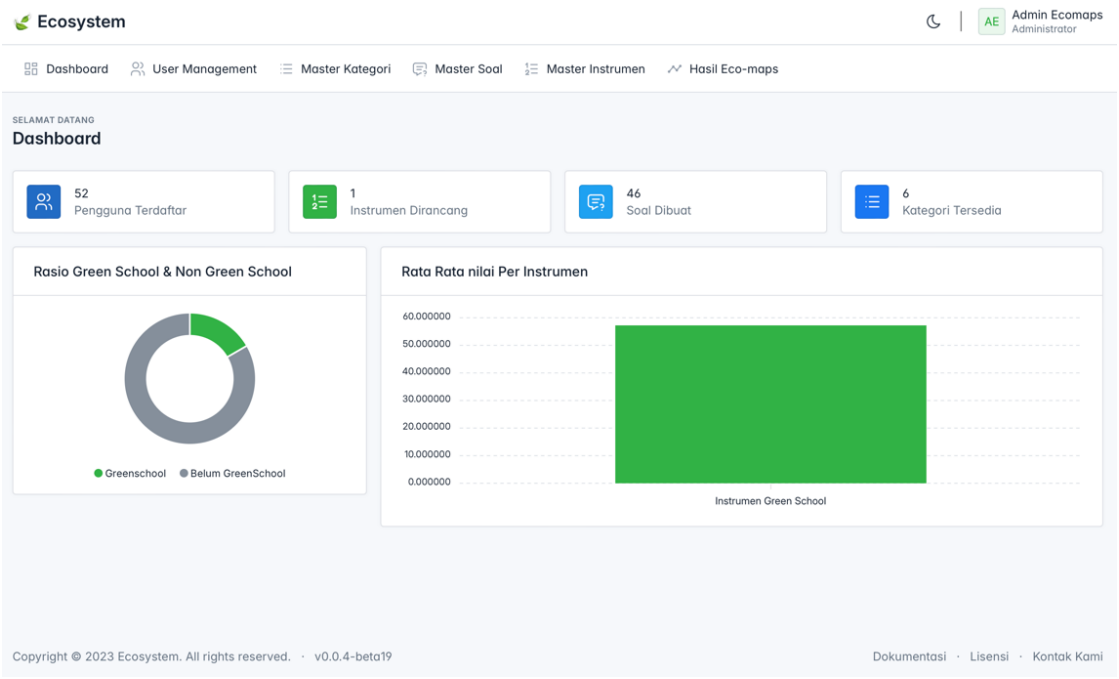
Pada tahap ini, implementasi sistem Eco-Maps dimulai dari proses login sebagaimana ditunjukkan pada Gambar 3, kemudian dilanjutkan dengan tampilan Menu Utama yang terdiri atas beberapa komponen penting. Setelah berhasil masuk, pengguna diarahkan ke menu utama yang mencakup fitur-fitur berikut (Gambar 4):

1) Dashboard, menampilkan informasi sistem secara keseluruhan.
2) User Management, untuk pengelolaan data pengguna.
3) Master Category, untuk pengaturan kategori dalam sistem.
4) Master Soal, untuk manajemen pertanyaan yang digunakan dalam proses evaluasi.
5) Master Instrument, untuk pengelolaan instrumen atau alat ukur yang digunakan.
6) Hasil Eco-Maps, untuk menampilkan hasil analisis berdasarkan data yang diproses oleh sistem.

Struktur ini dirancang dengan mempertimbangkan aspek kemudahan navigasi dan efisiensi penggunaan. Hasil pengujian menunjukkan bahwa sistem Eco-Maps dengan rancangan tersebut mampu memberikan performa yang responsif serta mendukung kebutuhan pengguna dalam manajemen data dan analisis secara efektif.




**Gambar 3 Login Eco-Maps**




**Gambar 4 Menu Utama Eco-Maps**

### 3.2 Pengujian menggunakan Stress and Load Test

Tabel 3 dan Tabel 4 menyajikan hasil pengujian Load Test dan Stress Test pada implementasi arsitektur *microservice* aplikasi Eco-Maps, yang terdiri dari empat layanan utama, yaitu Auth Service (Layanan 1), Eco-Map Service (Layanan 2), Waste Management Service (Layanan 3), dan Energy Monitoring Service (Layanan 4), serta layanan NATS Streaming sebagai event-bus. Hasil pengujian menunjukkan bahwa setiap layanan memiliki performa yang bervariasi sesuai dengan fungsinya. Auth Service dan NATS Streaming menunjukkan performa terbaik, dengan waktu respons rata-rata masing-masing 120 ms dan 110 ms, serta tingkat kesalahan minimal (*error rate* 10% dan 5%) bahkan pada pengujian beban tinggi.

Sementara itu, Eco-Map Service dan Waste Management Service mengalami *bottleneck* signifikan, ditandai dengan waktu respons tinggi (150 ms dan 180 ms) serta peningkatan *error rate* hingga 15% dan 20% saat stress test. Hal ini disebabkan oleh tingginya konsumsi memori dan latensi database yang mencapai 120 ms. Untuk penelitian selanjutnya, disarankan penerapan caching pada Eco-Map dan Waste Management, serta auto-scaling untuk menangani lonjakan beban. Selain itu, Energy Monitoring Service membutuhkan peningkatan kapasitas memori untuk menjaga performa saat beban tinggi. Secara keseluruhan, meskipun arsitektur *microservice* memberikan fleksibilitas dan skalabilitas, optimasi lebih lanjut pada database, alokasi sumber daya, dan sistem caching diperlukan untuk meningkatkan performa di seluruh layanan.

**Tabel 3 Hasil Pengujian dengan Load Test**

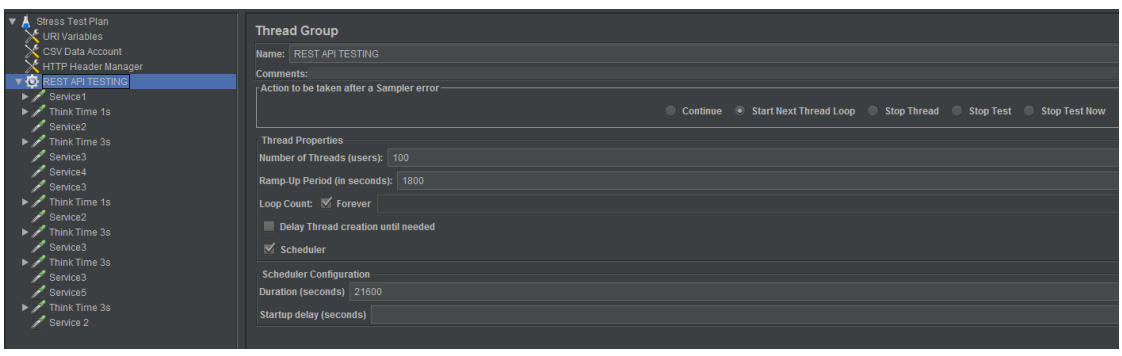
| Parameter Uji | Layanan 1 | Layanan 2 | Layanan 3 | Layanan 4 | Event Bus |
|---|---|---|---|---|---|
| Jumlah Request | 300/sec | 700/sec | 600/sec | 500/sec | 1000/sec |
| Response Time | 100 ms | 130 ms | 160 ms | 150 ms | 90 ms |
| *Error rate* | 0.2% | 0.5% | 0.8% | 0.3% | 0.1% |
| Throughput (Reg/Sec) | 300 | 690 | 580 | 490 | 1,000 |
| Memory Usage | 200 MB | 300 MB | 350 MB | 250 MB | 150 MB |
| Database Latency | 50 ms | 80 ms | 120 ms | 70 ms | - |
| Service Latency | 20 ms | 40 ms | 60 ms | 30 ms | 10 ms |
| Network Bandwidth | 100 Mbps | 150 Mbps | 120 Mbps | 110 Mbps | 200 Mbps |

**Tabel 4 Hasil pengujian dengan Stress Test**

| Parameter Uji | Layanan 1 | Layanan 2 | Layanan 3 | Layanan 4 | Event Bus |
|---|---|---|---|---|---|
| Jumlah Request | 500/sec | 1000/sec | 800/sec | 700/sec | 1200/sec |
| Response Time | 120 ms | 150 ms | 180 ms | 170 ms | 110 ms |
| *Error rate* | 10% | 15% | 20% | 12% | 5% |
| Throughput (Reg/Sec) | 490 | 970 | 760 | 680 | 1190 |
| Memory Usage | 300 MB | 450 MB | 500 MB | 400 MB | 200 MB |
| Database Latency | 50 ms | 80 ms | 120 ms | 70 ms | - |
| Service Latency | 20 ms | 40 ms | 60 ms | 30 ms | 10 ms |
| Network Bandwidth | 100 Mbps | 150 Mbps | 120 Mbps | 110 Mbps | 200 Mbps |

### 3.3 Hasil Pengukuran Waktu Respon dan Penggunaan CPU

Percobaan dilakukan dengan mengirimkan 100 paket data permintaan dari klien ke server, dengan menerapkan *microservice*. Percobaan diulang sebanyak sepuluh kali, dan dihitung nilai rata-ratanya. Waktu respons diukur menggunakan alat Apache JMeter, seperti yang ditunjukkan pada Gambar 5. Setelah sepuluh kali percobaan, data yang diperoleh dicatat, kemudian dihitung persentase perubahan data antara arsitektur komunikasi data berbasis *microservice*, seperti yang ditunjukkan pada Tabel 3.




**Gambar 5 Konfigurasi jumlah permintaan pada JMeter**

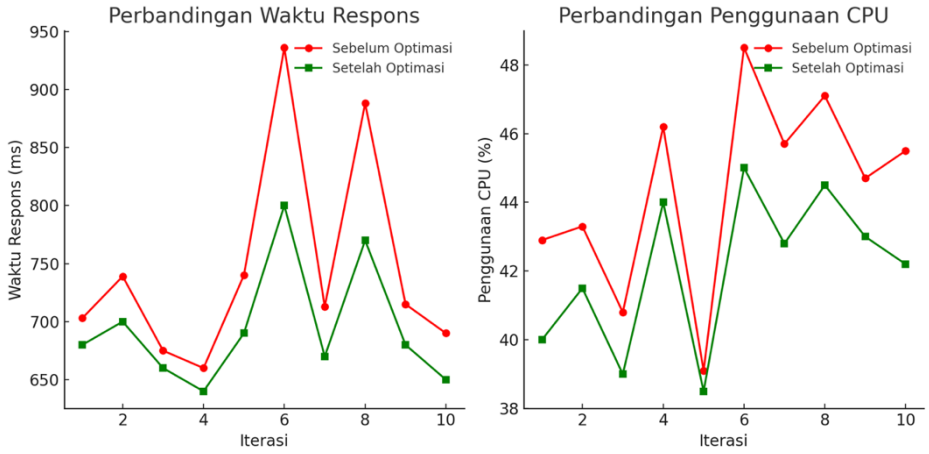
**Tabel 5 Hasil Pengukuran Waktu Respon dan Penggunaan CPU**

| Percobaan | Waktu (ms) | CPU Usage (%) |
|---|---|---|
| 1 | 703 | 42,9 |
| 2 | 739 | 43,3 |
| 3 | 675 | 40,8 |
| 4 | 660 | 46,2 |
| 5 | 740 | 39,1 |
| 6 | 936 | 48,5 |
| 7 | 713 | 45,7 |
| 8 | 888 | 47,1 |
| 9 | 715 | 44,7 |
| 10 | 690 | 45,5 |

Dari data pada Tabel 3, yang menunjukkan hasil pengukuran waktu respons dan penggunaan CPU dengan penerapan *microservice*, terlihat bahwa waktu respons bervariasi antara 660 ms hingga 936 ms, dengan iterasi keempat mencatat waktu tercepat (660 ms) dan iterasi keenam mencatat waktu terlama (936 ms). Penggunaan CPU juga menunjukkan variasi, mulai dari 39,1% pada iterasi kelima hingga 48,5% pada iterasi keenam. Secara umum, ada kecenderungan bahwa waktu respons yang lebih tinggi terkait dengan penggunaan CPU yang lebih tinggi, mengindikasikan bahwa beban pemrosesan API dapat meningkatkan kebutuhan CPU dan mempengaruhi kinerja.

Perbandingan performa dalam penelitian ini sudah mempertimbangkan kesamaan spesifikasi perangkat keras dan lunak, namun masih ada potensi ketimpangan dalam distribusi sumber daya, terutama pada layanan yang mengalami *bottleneck* seperti Eco-Map dan Waste Management. Variabilitas waktu respons dan penggunaan CPU menunjukkan adanya faktor eksternal yang dapat mempengaruhi hasil, seperti manajemen sumber daya Kubernetes dan latensi database. Untuk meningkatkan *fairness*, perlu diterapkan optimasi seperti caching, auto-scaling, dan *load balancing*. Secara keilmuan, penelitian ini berdampak bagi akademisi dalam pengembangan arsitektur *microservice*s, pengembang sistem dalam optimasi performa layanan, serta institusi pendidikan dan industri dalam implementasi smart campus.

Pada Gambar 6 menunjukkan perbandingan kinerja sistem sebelum dan setelah optimasi. Grafik kiri menampilkan waktu respons, di mana setelah optimasi terlihat penurunan waktu respons yang lebih stabil dan lebih rendah. Grafik kanan menunjukkan penggunaan CPU, yang mengalami efisiensi lebih baik setelah optimasi, mengindikasikan distribusi beban yang lebih optimal. Perbaikan ini dapat dicapai melalui penerapan caching, auto-scaling, dan *load balancing* untuk mengatasi *bottleneck* dalam layanan.




**Gambar 6 Perbandingan Waktu Respon dan Penggunaan CPU**

## 4. KESIMPULAN

Berdasarkan hasil percobaan, pengujian Load Test menunjukkan bahwa sistem memiliki performa yang stabil, dengan waktu respons rata-rata antara 100–160 ms dan *error rate* rendah (0,1%–0,8%). Pada Stress Test, layanan Auth Service dan NATS Streaming tetap menunjukkan performa optimal dengan *error rate* masing-masing 10% dan 5%, sedangkan Eco-Map Service dan Waste Management Service mengalami *bottleneck*, dengan waktu respons 150–180 ms dan *error rate* hingga 15%–20%, yang disebabkan oleh tingginya konsumsi memori dan latensi database hingga 120 ms.

Hasil pengukuran juga mencakup waktu respons dan penggunaan CPU. Waktu respons rata-rata dari sepuluh percobaan dengan penerapan API adalah 745,9 ms, dengan kisaran antara 660 ms hingga 936 ms. Iterasi keenam mencatat waktu respons tertinggi (936 ms), sedangkan iterasi keempat mencatat waktu respons terendah (660 ms). Penggunaan CPU bervariasi, dengan nilai terendah sebesar 39,1% pada iterasi kelima dan tertinggi 48,5% pada iterasi keenam. Pola ini menunjukkan bahwa waktu respons yang lebih lama cenderung terjadi saat penggunaan CPU tinggi, menandakan bahwa beban pemrosesan berdampak langsung pada efisiensi CPU dan kinerja keseluruhan sistem.

Meskipun arsitektur *microservice* memberikan fleksibilitas dan skalabilitas, optimasi lebih lanjut pada database, alokasi sumber daya, dan implementasi sistem caching diperlukan untuk meningkatkan performa seluruh layanan. Selain itu, penerapan auto-scaling juga diperlukan untuk menangani beban tinggi. Untuk penelitian selanjutnya, disarankan fokus pada optimasi arsitektur API-driven guna mengurangi fluktuasi waktu respons dan penggunaan CPU. Penelitian lanjutan dapat mencakup pengelolaan beban kerja yang lebih baik, eksplorasi teknologi optimasi seperti caching dan *load balancing*, serta perbandingan performa dengan arsitektur lain, misalnya Event-Driven Architecture (EDA). Pendekatan ini diharapkan dapat menemukan metode yang lebih efektif untuk meningkatkan efisiensi dan konsistensi kinerja sistem berbasis API-driven.

## UCAPAN TERIMA KASIH

Ucapan terima kasih kepada Lembaga Penelitian dan Pengabdian kepada Masyarakat (LPPM) Universitas Siliwangi atas dukungan dana dan berbagai bantuan yang diberikan dalam skema penelitian ini. Kami juga berterima kasih kepada seluruh tim penelitian atas kerja keras, kolaborasi, dan dedikasi tinggi dalam setiap tahap penelitian ini. Kami berharap hasilnya dapat memberikan kontribusi berarti bagi perkembangan ilmu pengetahuan dan teknologi serta bermanfaat bagi masyarakat.

## DAFTAR PUSTAKA

Aasa, O. P., Jesuleye, O. A., & Ajayi, M. O. (2020). Towards Greening Decisions on the University Campus: Initiatives, Importance and Barriers. *International Journal of Engineering and Management Research*, *10*(3), 82–88. https://doi.org/10.31033/ijemr.10.3.13

Abdullai, L., Porras, J., & Haque, S. (2022). Building a National Smart Campus to Support Sustainable Business Development: An Ecosystem Approach. *ArXiv Preprint*. http://arxiv.org/abs/2209.13613

Abdulmouti, H., Skaf, Z., & Alblooshi, S. (2022). Smart Green Campus: The Campus of Tomorrow. *2022 Advances in Science and Engineering Technology International Conferences (ASET)*, 1–8. https://doi.org/10.1109/ASET53988.2022.9735087

Amelia, A., Sundawa, B. V., Yusoff, M., Mohamad Zain, J. B., Azis, A., Suprianto, Roslina, R., & Pribadi, B. A. (2023). Performance Analysis of Environmental Monitoring System (EMS) towards POLMEDs Green Campus. *International Journal on Advanced Science, Engineering and Information Technology*, *13*(5), 1727–1732. https://doi.org/10.18517/ijaseit.13.5.19481

Dimov, A., Emanuilov, S., Bontchev, B., Dankov, Y., & Papapostolu, T. (2022). Architectural Approaches to Overcome Challenges in the Development of Data-Intensive Systems. In T.

Ahram (Ed.), *Human Factors in Software and Systems Engineering* (Vol. 61, pp. 38–43). AHFE International. https://doi.org/10.54941/ahfe1002521

Faizah, A. N., & Nugraheni, N. (2024). Pendidikan Berkelanjutan Berbasis Konservasi dan Teknologi Sebagai Aksi Nyata dalam Mewujudkan SDGs. *Jurnal Penelitian Ilmu-Ilmu Sosial*, *1*(10), 73–80. https://doi.org/10.5281/zenodo.11152410

Hong, X. J., Sik Yang, H., & Kim, Y. H. (2018). Performance Analysis of RESTful API and RabbitMQ for Microservice Web Application. *2018 International Conference on Information and Communication Technology Convergence (ICTC)*, 257–259. https://doi.org/10.1109/ICTC.2018.8539409

Iqbal, M., Syahputra, A. K., & Handoko, W. (2023). Penerapan Monolitic Arsitecture pada Aplikasi Ujian Online BERBASIS Berbasis Web. *Jurnal Pemberdayaan Sosial dan Teknologi Masyarakat*, *2*(2), 213–218. https://doi.org/10.54314/jpstm.v2i2.1107

Kautsar, I. A., Maika, M. R., Budiman, A. N., Setyawan, A. B., & Awali, J. Y. (2023). Microservice Based Architecture: The Development of Rapid Prototyping Supportive Tools for Project Based Learning. *2023 IEEE World Engineering Education Conference (EDUNINE)*, 1–6. https://doi.org/10.1109/EDUNINE57531.2023.10102884

Khan, A., & Ximei, W. (2022). Digital Economy and Environmental Sustainability: Do Information Communication and Technology (ICT) and Economic Complexity Matter? *International Journal of Environmental Research and Public Health*, *19*(19), Article ID: 12301. https://doi.org/10.3390/ijerph191912301

Martins, P., Lopes, S. I., Rosado da Cruz, A. M., & Curado, A. (2021). Towards a Smart & Sustainable Campus: An Application-Oriented Architecture to Streamline Digitization and Strengthen Sustainability in Academia. *Sustainability*, *13*(6), Article ID: 3189. https://doi.org/10.3390/su13063189

Palomo, I., Willemen, L., Drakou, E., Burkhard, B., Crossman, N., Bellamy, C., Burkhard, K., Campagne, C. S., Dangol, A., Franke, J., Kulczyk, S., Le Clec'h, S., Abdul Malak, D., Muñoz, L., Narusevicius, V., Ottoy, S., Roelens, J., Sing, L., Thomas, A., … Verweij, P. (2018). Practical Solutions for Bottlenecks in Ecosystem Services Mapping. *One Ecosystem*, *3*, Article ID: e20713. https://doi.org/10.3897/oneeco.3.e20713

Ribeiro, J. M. P., Hoeckesfeld, L., Dal Magro, C. B., Favretto, J., Barichello, R., Lenzi, F. C., Secchi, L., de Lima, C. R. M., & de Andrade Guerra, J. B. S. O. (2021). Green Campus Initiatives as Sustainable Development Dissemination at Higher Education Institutions: Students' Perceptions. *Journal of Cleaner Production*, *312*, Article ID: 127671. https://doi.org/10.1016/j.jclepro.2021.127671

Santos, L. C. dos, da Silva, M. L. P., & dos Santos Filho, S. G. (2024). Sustainability in Industry 4.0: Edge Computing Microservices as a New Approach. *Sustainability*, *16*(24), Article ID: 11052. https://doi.org/10.3390/su162411052

Sidiq, M. A. Z., Anshori, M. I., & Yaqin, R. A. (2024). Penerapan Arsitektur Monolitik pada Aplikasi Jasa Service Online Tekku Berbasis Web. *JUKI: Jurnal Komputer dan Informatika*, *6*(1), 27–36. https://doi.org/10.53842/juki.v6i1.418

Sugiarto, A., Lee, C.-W., & Huruta, A. D. (2022). A Systematic Review of the Sustainable Campus Concept. *Behavioral Sciences*, *12*(5), Article ID: 130. https://doi.org/10.3390/bs12050130

Telang, T. (2023). Microservices Architecture. In *Beginning Cloud Native Development with MicroProfile, Jakarta EE, and Kubernetes* (pp. 111–142). Apress. https://doi.org/10.1007/978-1-4842-8832-0_5

Trevisan, L. V., Eustachio, J. H. P. P., Dias, B. G., Filho, W. L., & Pedrozo, E. Á. (2023). Digital Transformation Towards Sustainability in Higher Education: State-of-the-Art and Future Research Insights. *Environment, Development and Sustainability*, *26*(2), 2789–2810. https://doi.org/10.1007/s10668-022-02874-7

Weerasinghe, L. D. S. B., & Perera, I. (2024). Reference Architecture for Microservices with an Optimized Inter-Service Communication Strategy. *2024 International Research Conference on Smart Computing and Systems Engineering (SCSE)*, 1–6. https://doi.org/10.1109/SCSE61872.2024.10550466

Xiao, X. (2024). Architectural Tactics to Improve the Environmental Sustainability of Microservices: A Rapid Review. *ArXiv Preprint*. http://arxiv.org/abs/2407.16706