Implementasi Arsitektur *Microservices* dan OAuth 2.0 untuk Optimalisasi Performa API SIAKAD di STIMATA

Rendi Saputra[#*1], Linda Suvi Rahmawati[#2], Indah Dwi Mumpuni[#3]
[#]Program Studi D-III Sistem Informasi, STMIK PPKIA Pradnya Paramita, Malang, Indonesia
*rendi_22310011@stimata.ac.id

Info Artikel

Diajukan: *-*
Diterima: *-*
Diterbitkan: *-*

Keywords:
microservices; OAuth 2.0; Academic Information System; REST API; performance testing; load testing

Kata Kunci:
microservices; OAuth 2.0; Sistem Informasi Akademik; REST API; pengujian performa; load testing

Abstract

This research addresses performance degradation and complex authentication management in the existing Academic Information System (SIAKAD) at STMIK PPKIA Pradnya Paramita (STIMATA), especially when the number of users increases and services are accessed through multiple platforms. The proposed solution is to redesign the system using a microservices architecture and implement OAuth 2.0 as a centralized authentication mechanism, with REST API as the communication interface between services. The experiment compares two implementations, namely a monolithic architecture and a microservices-based architecture, which are tested under equivalent environments using K6 and Artillery for load and stress testing and Prometheus–Grafana for performance monitoring. The main performance indicators observed are latency, throughput, error rate, and utilization of CPU and memory resources. The results show that the microservices-based system achieves millisecond-level response times, higher throughput, and more efficient resource usage compared to the monolithic system, particularly under high workloads where the monolithic implementation experiences significant degradation. In addition, the adoption of OAuth 2.0 successfully simplifies cross-platform login processes by enabling centralized access management for web and mobile applications. These findings confirm that the combination of microservices architecture and OAuth 2.0 provides an effective solution to improve performance, security, and user experience of SIAKAD at STIMATA.

Abstrak

Penelitian ini membahas permasalahan penurunan performa dan kompleksitas manajemen autentikasi pada Sistem Informasi Akademik (SIAKAD) STMIK PPKIA Pradnya Paramita (STIMATA), terutama ketika jumlah pengguna meningkat dan layanan diakses melalui berbagai platform. Solusi yang diusulkan adalah merancang ulang sistem menggunakan arsitektur microservices serta menerapkan OAuth 2.0 sebagai mekanisme autentikasi terpusat, dengan REST API sebagai antarmuka komunikasi antar layanan. Eksperimen dilakukan dengan membandingkan dua implementasi, yaitu arsitektur monolith dan arsitektur berbasis microservices, yang diuji dalam lingkungan setara dengan bantuan K6 dan Artillery untuk pengujian beban dan stress, serta Prometheus dan Grafana untuk pemantauan performa. Indikator utama yang diamati meliputi latensi, throughput, tingkat kegagalan, serta pemanfaatan sumber daya CPU dan memori. Hasil pengujian menunjukkan bahwa sistem berbasis microservices mampu mencapai waktu respon pada orde milidetik, throughput yang lebih tinggi, dan penggunaan sumber daya yang lebih efisien dibandingkan sistem monolith, terutama pada skenario beban tinggi ketika implementasi monolith mengalami degradasi kinerja yang signifikan. Selain itu, penerapan OAuth 2.0 berhasil menyederhanakan proses login lintas platform dengan menyediakan pengelolaan akses terpusat untuk aplikasi web dan mobile. Temuan ini mengonfirmasi bahwa kombinasi arsitektur microservices dan OAuth 2.0 merupakan solusi yang efektif untuk meningkatkan performa, keamanan, dan pengalaman pengguna SIAKAD di STIMATA.

Cara mensitasi artikel:
Anwar, K., Arifin, S., & Satria, P. (2020). Judul Artikel. *Jurnal Teknologi Informasi (JTI), x*(x), x–xx. https://doi.org/10.33474/jti.v5i2.xxxxx

PENDAHULUAN

Sistem Informasi Akademik (SIAKAD) merupakan komponen inti dalam pengelolaan data akademik perguruan tinggi, seperti pendaftaran mata kuliah, pengelolaan nilai, dan penyimpanan data mahasiswa. [1]. Di STMIK PPKIA Pradnya Paramita (STIMATA), SIAKAD yang berjalan menghadapi berbagai kendala, antara lain performa yang menurun saat beban pengguna meningkat serta keterbatasan dalam memenuhi kebutuhan fungsional mahasiswa, dosen, dan staf administrasi. Kondisi tersebut mengakibatkan proses administrasi menjadi kurang efisien dan menurunkan kualitas layanan akademik.

Selain masalah performa, STIMATA juga menghadapi tantangan dalam manajemen autentikasi pengguna di berbagai platform, karena belum tersedianya portal terintegrasi yang memadai. Mahasiswa dan pengguna lain harus mengingat banyak kombinasi nama pengguna dan kata sandi untuk mengakses layanan yang berbeda, sehingga meningkatkan risiko kesalahan dan praktik keamanan yang lemah.[2]. Ketiadaan mekanisme autentikasi terpusat juga menyulitkan pengelolaan hak akses dan pengendalian keamanan data secara menyeluruh.

Perkembangan teknologi arsitektur perangkat lunak menawarkan pendekatan *microservices* yang memecah aplikasi menjadi layanan-layanan kecil yang mandiri dan saling berkomunikasi melalui API. Pendekatan ini dinilai lebih fleksibel, mudah diskalakan, dan mendukung

pengembangan bertahap dibandingkan arsitektur *monolith* yang menyatukan seluruh fungsi dalam satu kesatuan aplikasi. [3] Di sisi lain, protokol OAuth 2.0 menyediakan mekanisme autentikasi dan otorisasi terpusat yang cocok diterapkan pada ekosistem aplikasi web dan mobile modern.

Berdasarkan latar belakang tersebut, penelitian ini membangun ulang API SIAKAD dengan pendekatan arsitektur *microservices* dan autentikasi OAuth 2.0 untuk meningkatkan performa dan fleksibilitas sistem di STIMATA [4]. Rumusan masalah yang dikaji meliputi: bagaimana merancang sistem dengan performa tinggi, bagaimana menyederhanakan akses pengguna di berbagai platform kampus, dan bagaimana mengukur keberhasilan implementasi dari sisi performa, keamanan, dan pengalaman pengguna. Tujuan penelitian adalah mengembangkan SIAKAD berbasis *microservices* untuk mengoptimalkan performa, merancang layanan autentikasi terpusat menggunakan OAuth 2.0, serta menganalisis peningkatan performa dan kualitas layanan melalui pengujian terukur terhadap implementasi baru dibandingkan sistem *monolith*.

METODE

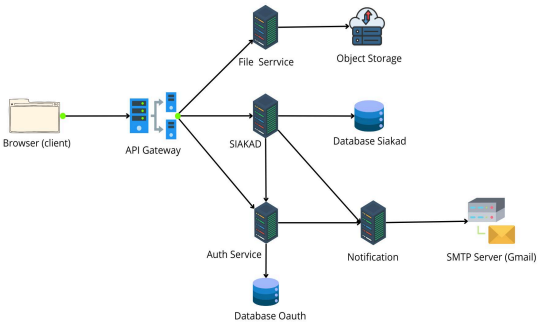
Penelitian ini menerapkan kerangka kerja berbasis arsitektur *microservices* dalam pengembangan Sistem Informasi Akademik (SIAKAD) dengan tujuan meningkatkan performa, skalabilitas, serta efisiensi pengelolaan autentikasi pengguna di lingkungan STMIK PPKIA Pradnya Paramita [4]. Pendekatan ini dilakukan dengan memecah sistem monolitik menjadi layanan-layanan mandiri yang saling terintegrasi melalui REST API dan dikelola menggunakan autentikasi terpusat berbasis OAuth 2.0 [2], [6]. Kerangka kerja tersebut digunakan sebagai dasar dalam perancangan sistem dan sebagai acuan evaluasi performa melalui perbandingan antara arsitektur *monolith* dan *microservices*.

Kerangka kerja yang digunakan dalam penelitian ini diwujudkan dalam bentuk arsitektur sistem berbasis *microservices* yang dirancang untuk memisahkan fungsi-fungsi utama SIAKAD ke dalam layanan-layanan mandiri. Pendekatan ini bertujuan untuk meningkatkan modularitas sistem, mempermudah pengelolaan autentikasi terpusat, serta memungkinkan pengujian performa dilakukan secara lebih terisolasi pada setiap komponen layanan.

A. Arsitektur Sistem

Arsitektur sistem pada penelitian ini mengadopsi pendekatan *microservices*, di mana setiap layanan utama SIAKAD dikembangkan dan dijalankan secara independen. Sistem melayani berbagai jenis klien, seperti aplikasi web dan aplikasi mobile, yang berkomunikasi dengan backend melalui sebuah API Gateway sebagai titik masuk utama.

Arsitektur sistem yang digunakan dalam penelitian ini ditunjukkan pada Gambar 1, yang menggambarkan interaksi antara klien, API Gateway, layanan *microservices*, serta mekanisme autentikasi terpusat berbasis OAuth 2.0.



Gambar. 1 Arsitektur *microservices* SIAKAD

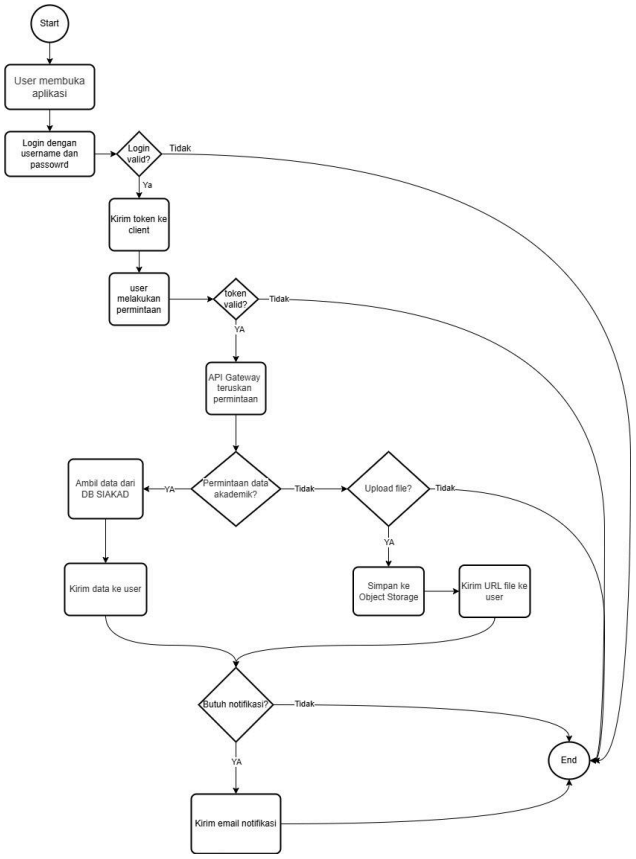
Secara konseptual, kerangka kerja solusi terdiri dari beberapa komponen utama, yaitu klien (aplikasi web atau mobile), API Gateway berbasis NGINX, layanan autentikasi OAuth 2.0, layanan inti SIAKAD, layanan manajemen berkas, layanan notifikasi, serta lapisan penyimpanan data yang memuat basis data SIAKAD, basis data OAuth, dan *object storage*. Klien mengirimkan permintaan ke API Gateway, yang kemudian meneruskan permintaan tersebut ke layanan yang sesuai untuk diproses, sebagaimana ditunjukkan pada Gambar 1.

Pada tahap awal, setiap permintaan pengguna yang masuk ke API Gateway diverifikasi oleh layanan autentikasi untuk memastikan bahwa hanya pengguna dengan kredensial sah yang dapat mengakses layanan internal. Setelah proses autentikasi berhasil, permintaan diteruskan ke layanan terkait, misalnya layanan inti SIAKAD untuk operasi terhadap data akademik atau layanan manajemen berkas untuk pengunggahan dan pengunduhan dokumen.

B. Alur kerja arsitektur

Alur proses sistem pada penelitian ini digambarkan menggunakan flowchart untuk memperjelas tahapan operasional sistem berdasarkan arsitektur *microservices* yang telah dijelaskan sebelumnya. Flowchart ini menunjukkan urutan aktivitas yang terjadi ketika pengguna berinteraksi dengan SIAKAD, mulai dari proses autentikasi hingga pemrosesan permintaan layanan dan pengiriman respon ke pengguna.

Alur kerja operasional sistem SIAKAD berbasis *microservices* ditunjukkan pada Gambar 2.



Gambar. 2 Flowchart SIAKAD berbasis *microservices*

Berdasarkan Gambar 2, proses sistem dimulai ketika pengguna mengakses aplikasi dan melakukan proses login dengan memasukkan kredensial. Permintaan login dikirimkan ke Auth Service untuk dilakukan proses autentikasi. Jika autentikasi berhasil, sistem mengembalikan token akses ke klien sebagai identitas pengguna.

Token akses tersebut kemudian disertakan dalam setiap permintaan layanan yang dikirimkan oleh pengguna melalui API Gateway. API Gateway memverifikasi validitas token sebelum meneruskan permintaan ke layanan yang sesuai. Apabila permintaan berkaitan dengan data akademik, maka permintaan diproses oleh Core Service yang berinteraksi dengan basis data SIAKAD.

Untuk permintaan yang melibatkan pengelolaan berkas, sistem meneruskan permintaan ke File Management Service yang berinteraksi dengan *Object Storage*. Selain itu, pada kondisi tertentu sistem juga memicu Notification Service untuk mengirimkan notifikasi kepada pengguna melalui email. Setelah seluruh proses selesai, hasil pemrosesan dikembalikan ke klien sebagai respon akhir.

Alur kerja operasional ini menjadi dasar dalam penyusunan skenario pengujian performa, di mana setiap tahapan proses diuji untuk mengevaluasi respons sistem pada berbagai kondisi beban.

C. Rancangan Pengujian

Rancangan pengujian pada penelitian ini bertujuan untuk mengevaluasi kinerja sistem SIAKAD berbasis arsitektur *microservices* dan membandingkannya dengan sistem berbasis arsitektur *monolith*. Evaluasi difokuskan pada aspek performa dan efisiensi penggunaan sumber daya melalui skenario pengujian beban yang terkontrol dan dapat direplikasi.

Objek pengujian terdiri dari dua sistem SIAKAD yang memiliki fungsi identik, yaitu sistem berbasis monolith dan sistem berbasis *microservices*. Kedua sistem menggunakan dataset dummy akademik yang sama untuk memastikan bahwa perbedaan hasil pengujian hanya dipengaruhi oleh perbedaan arsitektur sistem.

Parameter pengujian meliputi jumlah virtual user, kecepatan respon, durasi pengujian, serta jenis operasi API yang dijalankan. Skenario pengujian mencakup beban normal, peningkatan beban secara bertahap, serta pengujian beban tinggi.

Indikator performa yang diukur meliputi *throughput*, waktu respon, dan *error rate* [7]. Selain itu, penggunaan sumber daya seperti CPU dan memori juga dipantau untuk menilai efisiensi masing-masing arsitektur dalam menangani beban kerja [7].

Seluruh pengujian dijalankan dalam lingkungan terkontrol menggunakan Docker dan Kubernetes untuk memastikan kesetaraan konfigurasi [8]. Pengujian beban dilakukan menggunakan K6 dan Artillery [7], sedangkan pemantauan sumber daya dilakukan menggunakan Prometheus dan Grafana [9], [10]. Rancangan pengujian ini digunakan untuk memperoleh hasil perbandingan performa yang objektif antara arsitektur *monolith* dan *microservices* pada sistem SIAKAD.

HASIL DAN PEMBAHASAN

Pengujian performa dilakukan untuk mengevaluasi perbedaan karakteristik kinerja antara sistem SIAKAD berbasis arsitektur *monolith* dan sistem SIAKAD berbasis arsitektur *microservices*. Pengujian mencakup berbagai skenario beban, mulai dari beban normal hingga beban tinggi, dengan fokus pada pengukuran waktu respon, *throughput*, tingkat kegagalan permintaan, serta penggunaan sumber daya sistem. Hasil pengujian menunjukkan adanya perbedaan yang signifikan antara kedua arsitektur, khususnya ketika sistem berada pada kondisi beban tinggi. Arsitektur *microservices* mampu mempertahankan performa yang lebih stabil dan responsif dibandingkan arsitektur *monolith*, yang cenderung mengalami penurunan kinerja seiring meningkatnya jumlah permintaan. Temuan ini menjadi dasar dalam pembahasan lebih lanjut mengenai efektivitas penerapan *microservices* dalam meningkatkan performa dan skalabilitas sistem SIAKAD.

Berdasarkan pengujian yang telah dilakukan, skenario pengujian performa yang digunakan dalam penelitian ini disajikan pada Tabel 1.

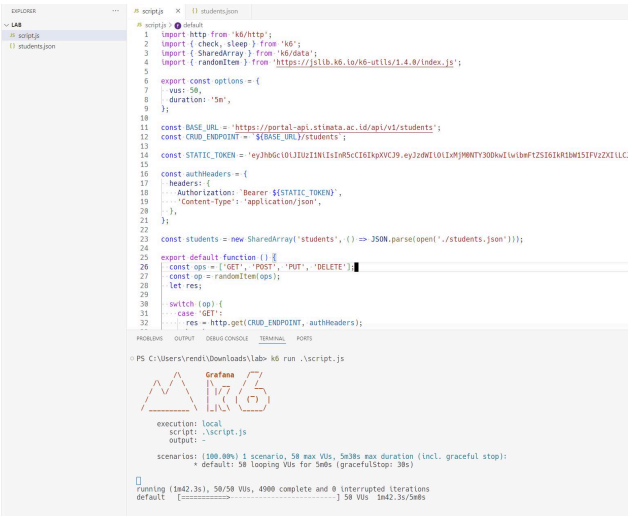
Tabel 1 Skenario Pengujian Performa

No	Skenario	Tujuan	Parameter	Metrik
1	Beban normal	Evaluasi performa dasar	50 VU, 5 menit	Latensi, throughput
2	Beban bertahap	Menilai titik jenuh	+10 VU / 60 detik	Error rate
3	Beban tinggi	Uji stabilitas ekstrem	1000 req/detik	Success rate
4	Login serentak	Evaluasi OAuth	200 user	Latensi login
5	Monitoring resource	Efisiensi sistem	CPU, Memori	Penggunaan CPU, dan Memori

A. Proses Pengujian

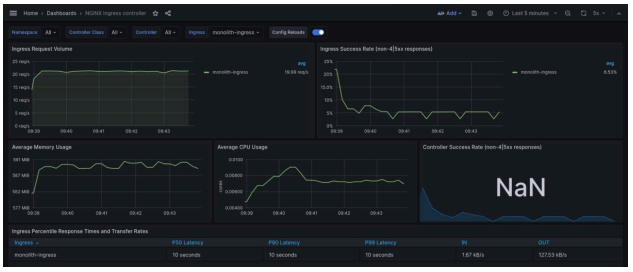
Proses pengujian pada penelitian ini dilakukan untuk memastikan bahwa sistem SIAKAD berbasis arsitektur monolith dan *microservices* diuji pada kondisi yang setara. Pengujian dilakukan dengan mengirimkan permintaan ke sistem melalui *endpoint* API yang telah ditentukan, mencakup proses autentikasi dan akses layanan akademik.

Load testing dilakukan menggunakan **K6** untuk mensimulasikan sejumlah pengguna dan permintaan secara bersamaan. Setiap permintaan yang dikirimkan oleh K6 diproses oleh sistem sesuai dengan arsitektur yang digunakan dan dicatat untuk mengukur waktu respon, *throughput*, serta tingkat kegagalan permintaan. Selain itu, penggunaan sumber daya sistem seperti CPU dan memori juga dipantau selama pengujian berlangsung. Alur proses pengujian sistem menggunakan K6 ditunjukkan pada Gambar 3.



Gambar 3 Proses Pengujian Sistem Menggunakan K6

B. Hasil Pengujian Dengan Beban Normal

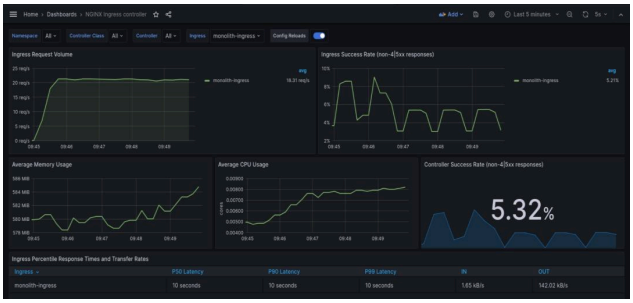


Gambar 4 Hasil Pengujian Beban Normal Arsitektur *Monolith*

Hasil pengujian beban normal pada sistem SIAKAD ditunjukkan pada Gambar 4 dan Gambar 5. Pada pengujian ini, sistem mampu memproses permintaan dengan tingkat keberhasilan yang baik, dengan waktu respon yang berada pada kondisi normal selama pengujian berlangsung. Selama proses pengujian, permintaan dapat ditangani secara konsisten tanpa terjadinya kegagalan yang signifikan.

Gambar 5 Hasil Pengujian Beban Normal Arsitektur *Microservices*

C. Hasil Pengujian Dengan Beban Bertahap



Gambar 6 Hasil Pengujian Beban Bertahap Arsitektur *Monolith*

Hasil pengujian beban bertahap pada sistem SIAKAD ditunjukkan pada Gambar 6 dan Gambar 7. Pada pengujian ini, sistem diuji dengan peningkatan jumlah permintaan secara bertahap selama periode pengujian. Hasil yang diperoleh menunjukkan bahwa sistem tetap dapat memproses permintaan yang masuk seiring dengan bertambahnya beban, dengan perubahan waktu respon yang terlihat mengikuti peningkatan jumlah permintaan.

Gambar 7 Hasil Pengujian Beban Bertahap Arsitektur *Microservices*

D. Hasil Pengujian Dengan Beban Tinggi

Gambar 8 Hasil Pengujian Beban Tinggi Arsitektur *Monolith*

Pada skenario pengujian beban tinggi, hasil pengujian menunjukkan perbedaan tingkat keberhasilan pemrosesan permintaan antara kedua arsitektur. Seperti ditunjukkan pada Gambar 8 dan Gambar 9, aplikasi SIAKAD berbasis arsitektur monolith mengalami penurunan tingkat keberhasilan permintaan hingga sebesar 4,82%.

Sebaliknya, pada sistem SIAKAD berbasis arsitektur microservices, seluruh permintaan pada skenario pengujian ini dapat diproses dengan tingkat keberhasilan 100%, dengan nilai latensi yang tetap berada pada kisaran 4 millisecond selama pengujian berlangsung.

Gambar 9 Hasil Pengujian Beban Tinggi Arsitektur Microservices

E. Hasil Pengujian Dengan Beban Tinggi

Gambar 10 Hasil Pengujian Login Serentak

Pada skenario pengujian keempat, seperti gambar 4.7 200 virtual users melakukan login secara bersamaan dengan maksimal waktu request 1 menit aplikasi menunjukan latensi yang relatif sama namun dengan tingkat keberhasilan yang berbeda dimana tingkat keberhasilan aplikasi microservices 100% dan aplikasi monolith < 50%.

F. Hasil Pengujian Resource Usage

Pada skenario pengujian kelima yang mengacu pada hasil pengujian skenario satu sampai empat, penggunaan memory dan cpu relatif sama penggunaan memory pada arsitektur microservices maksimal di 578MB dan monolith 587MB.

Tabel 2 Hasil pengujian

No	Skenario Pengujian	Monolith	Microservices
1	Beban normal	Semua request diproses	Rata-rata latensi maksimal 5

		dalam 10 detik/*request*	milliseconds; performa lebih responsif
2	Beban bertahap	Terjadi penurunan response time dan tingkat keberhasilan hanya 5%	Tetap stabil dengan tingkat keberhasilan 100% dan latensi 4 *milliseconds*
3	Beban tinggi	Tingkat keberhasilan turun hingga 4.82%	Semua request berhasil (100%) dengan latensi tetap 4 *milliseconds*
4	Login serentak	Latensi mirip dengan *microservices*, tetapi tingkat keberhasilan < 50%	Latensi serupa tetapi keberhasilan login 100%
5	Monitoring resource	penggunaan memory maksimal 578MB	penggunaan memory maksimal 587MB

KESIMPULAN DAN SARAN

Berdasarkan hasil penelitian dan pengujian yang telah dilakukan, dapat disimpulkan bahwa penerapan arsitektur microservices pada sistem SIAKAD mampu memberikan peningkatan performa dibandingkan dengan arsitektur *monolith*. Hasil pengujian menunjukkan bahwa sistem berbasis *microservices* memiliki kemampuan yang lebih baik dalam menangani peningkatan beban permintaan, khususnya pada skenario pengujian beban tinggi, di mana seluruh permintaan dapat diproses dengan tingkat keberhasilan 100% dan latensi yang tetap rendah. Sebaliknya, sistem berbasis arsitektur *monolith* mengalami penurunan tingkat keberhasilan pemrosesan permintaan secara signifikan pada kondisi beban tinggi, yang menunjukkan keterbatasan dalam hal skalabilitas dan stabilitas performa.

Berdasarkan temuan tersebut, penerapan arsitektur microservices dinilai lebih sesuai untuk digunakan pada sistem SIAKAD yang melayani banyak pengguna secara simultan. Untuk pengembangan selanjutnya, disarankan agar sistem SIAKAD berbasis *microservices* dikembangkan dengan pemisahan layanan yang lebih spesifik. Layanan penilaian dan layanan kurikulum dapat dipisahkan menjadi service mandiri untuk meningkatkan modularitas dan fokus pengelolaan. Pemisahan ini diharapkan mampu meningkatkan fleksibilitas sistem serta membuka peluang pengembangan fitur lanjutan berbasis data, seperti dukungan rekomendasi akademik, guna mendukung proses pengelolaan dan pengambilan keputusan akademik.

UCAPAN TERIMA KASIH

Penulis mengucapkan terima kasih kepada seluruh pihak yang telah memberikan dukungan dan kontribusi selama proses pelaksanaan penelitian dan penyusunan artikel ini. Ucapan terima kasih secara khusus disampaikan kepada Ibu Linda Suvi Rahmawati, S.Kom., MMSI selaku Dosen Pembimbing I dan Ibu Dr. Indah Dwi Mumpuni, S.Kom., MM selaku Dosen Pembimbing II atas bimbingan, arahan, serta masukan yang sangat berharga sejak tahap perencanaan hingga penyelesaian penelitian.

Penulis juga mengucapkan terima kasih kepada teman-teman yang telah memberikan dukungan, diskusi, dan bantuan selama proses penelitian. Selain itu, apresiasi yang sebesar-besarnya disampaikan kepada orang tua dan keluarga atas doa, dukungan moral, serta motivasi yang senantiasa diberikan.

Ucapan terima kasih turut disampaikan kepada seluruh staf dan sivitas akademika STMIK PPKIA Pradnya Paramita (STIMATA) atas dukungan fasilitas, data, dan lingkungan akademik yang mendukung sehingga penelitian ini dapat diselesaikan dengan baik.

REFERENSI

[1] S. A. Makian and A. M. S. Kiat, “Inovasi Pemanfaatan Sistem Informasi Akademik (SIAKAD): Studi Literatur,” *J. Ilmu Sos. Dan Ilmu Polit.*, vol. 01, no. 01, 2025.

[2] A. Raysa and I. Muslem, “Implementasi Single Sign On (SSO) Menggunakan Protokol OAuth Pada Sistem Informasi Kampus,” vol. 5, no. 2, 2024.

[3] M. S. Alikhan and I. M. Suartana, “Desain dan Pengembangan Backend Aplikasi Bantucari Menggunakan Microservices,” *J. Inform. Comput. Sci. JINACS*, pp. 314–321, Jan. 2023, doi: 10.26740/jinacs.v4n03.p314-321.

[4] A. M. Husein, A. J. Simanjuntak, C. J. Sinaga, M. M. Tampubolon, and P. N. C. Situmorang, “SIAKAD Mobile With API Service To Improve Academic Services,” *Sinkron*, vol. 8, no. 2, pp. 641–650, Mar. 2024, doi: 10.33395/sinkron.v8i2.13519.

[5] Ugan Suganda, “Analisis Perancangan Arsitektur Microservices pada Sistem Informasi Akademik Stie Dharma Negara dengan Pendekatan Framework Togaf-ADM,” *J. Ekon. Kop. Kewirausahaan*, vol. 15, Agustus 2024.

[6] R. Kurniawan, “Perancangan dan Implementasi Sistem Otentikasi OAuth 2.0 dan PKCE Berbasis Extreme Programming (XP),” *J. Pendidik. Dan Teknol. Indones.*, vol. 2, no. 2, Art. no. 2, Feb. 2022, doi: 10.52436/1.jpti.141.

[7] Agung Suprapto and Dimas Sasongko, “EVALUASI PERFORMA WEBSITEBERDASARKAN PENGUJIAN BEBAN DAN STRESS MENGGUNAKAN LOADIMPACT (STUDI KASUS WEBSITEIAIN SALATIGA),” *J. Ilm. NERO*, vol. 6, Apr. 2021.

[8] A. Poniszewska-Marańda and E. Czechowska, “Kubernetes Cluster for Automating Software Production Environment,” *Sensors*, vol. 21, no. 5, Art. no. 5, Jan. 2021, doi: 10.3390/s21051910.

[9] Professor, Department of Communication Systems Engineering, University of Science and Technology, Khartoum, Sudan and M. D. Elradi, “Prometheus & Grafana: A Metrics-focused Monitoring Stack,” *J. Comput. Allied Intell.*, vol. 3, no. 3, pp. 28–39, June 2025, doi: 10.69996/jcai.2025015.

[10] Department of ISE and K. Sai, “Enhanced Visibility for Real-time Monitoring and Alerting in Kubernetes by Integrating Prometheus,Prometheus, Grafana, Loki, and Alerta,” *INTERANTIONAL J. Sci. Res. Eng. Manag.*, vol. 08, no. 06, pp. 1–5, June 2024, doi: 10.55041/IJSREM35639.