Modernisasi Sistem Informasi Akademik: Transisi Arsitektur Monolitik ke Microservices dengan Integrasi Vue.js, Golang, dan Keycloak

Bahteramon Bintang Sanjaya Manurung[1)], I Komang Setia Buana[2)], Freddy Matius Herdian Hutasoit*[3)]

1) Rekayasa Kriptografi, Politeknik Siber dan Sandi Negara, bahteramon.bintang@student.poltekssn.ac.id
2) Rekayasa Kriptografi, Politeknik Siber dan Sandi Negara, komang.setia@poltekssn.ac.id
3) Rekayasa Keamanan Siber, Politeknik Siber dan Sandi Negara, freddy.matius@student.poltekssn.ac.id

Riwayat Artikel

Dikirim 27 Sep 2024
Diterima 9 Des 2025
Diterbitkan 19 Des 2025

Kata kunci:

Sistem Informasi
Microservices
Keamanan
Vue.js
Golang
Keycloak

Keywors:

Information System
Microservices
Security
Vue.js
Golang
Keycloak

Abstrak

Penelitian ini membahas pengembangan sistem informasi akademik modern berbasis *microservices* sebagai solusi atas keterbatasan sistem monolitik di Politeknik Siber dan Sandi Negara yang terkendala fleksibilitas, integrasi data, dan keamanan. Tujuan utamanya adalah merancang sebuah sistem yang modular dan aman dengan mengimplementasikan arsitektur *microservices* yang mengintegrasikan RESTful API, Vue.js sebagai *framework frontend*, Golang untuk backend, serta Keycloak sebagai sistem autentikasi *Single Sign-On* (SSO). Metode pengembangan yang digunakan adalah *prototype*, yang mencakup perancangan, implementasi bertahap, hingga evaluasi. Hasil penelitian menunjukkan bahwa arsitektur yang diusulkan berhasil mengatasi keterbatasan sistem lama. Sistem baru ini berhasil melewati pengujian keamanan berdasarkan standar OWASP Top 10: 2021 dan mendapatkan persepsi pengguna yang sangat positif melalui *User Acceptance Testing* (UAT), di mana sistem dinilai lebih responsif dan memberikan pengalaman pengguna yang lebih baik. Penelitian ini berkontribusi pada upaya transformasi digital di lingkungan pendidikan tinggi dengan menyajikan solusi pengelolaan data akademik yang terintegrasi, aman, dan efisien.

Abstract

This research discusses the development of a modern microservices-based academic information system as a solution to the limitations of the monolithic system at the State Cyber and Cryptography Polytechnic, which faced challenges in flexibility, data integration, and security. The main objective is to design a modular and secure system by implementing a microservices architecture that integrates a RESTful API, Vue.js as the frontend framework, Golang for the backend, and Keycloak as a Single Sign-On (SSO) authentication system. The prototype method was used for development, covering design, phased implementation, and evaluation. The results show that the proposed architecture successfully overcomes the limitations of the legacy system. The new system passed security testing based on the OWASP Top 10:2021 standard and received highly positive user perception through User Acceptance Testing (UAT), where it was rated as more responsive and providing a better user experience. This study contributes to digital transformation efforts in higher education by presenting an integrated, secure, and efficient academic data management solution.

1. PENDAHULUAN

Di tengah era transformasi digital, kebutuhan akan sistem informasi akademik yang fleksibel, skalabel, dan mudah diintegrasikan menjadi sangat penting bagi institusi pendidikan tinggi. Politeknik Siber dan Sandi Negara, sebagai institusi di bidang teknologi informasi, memerlukan sistem yang optimal untuk mengelola data akademik dan kemahasiswaan. Namun, sistem yang saat ini digunakan, yaitu Aplikasi Manajemen Data Akademik dan Kemahasiswaan Terintegrasi (MASTER), masih berbasis arsitektur monolitik, yang menghadirkan keterbatasan dalam hal pemeliharaan, penyesuaian, dan integrasi data [1]. Arsitektur ini menyebabkan perubahan pada satu bagian kode berpotensi berdampak luas, menghambat integrasi dengan aplikasi lain, dan membuat antarmuka pengguna kurang responsif [2].

Penelitian terdahulu telah memberikan landasan yang kuat untuk modernisasi sistem ini. Implementasi modul-modul inti pada sistem MASTER dengan pendekatan REST API terbukti dapat meningkatkan interoperabilitas dan fleksibilitas dalam pengelolaan data [3, 4]. Selain itu, dari aspek keamanan, sistem yang ada saat ini masih rentan karena hanya menggunakan *username* dan *password* sederhana [5]. Untuk mengatasi hal ini, penelitian merekomendasikan penggunaan autentikasi berbasis *Single Sign-On* (SSO) sebagai langkah mitigasi untuk memperkuat keamanan sistem informasi [6]. Di sisi lain, pengalaman pengguna yang kurang optimal akibat antarmuka yang sudah usang juga menjadi perhatian utama, di mana peningkatan pada aspek ini dapat menjadi pendorong keberhasilan penerapan sistem baru [7].

Oleh karena itu, penelitian ini dilakukan untuk mengatasi keterbatasan tersebut dengan mengusulkan pengembangan sistem informasi akademik modern berbasis *microservices*. Alasan utama pemilihan arsitektur ini adalah untuk meningkatkan fleksibilitas, skalabilitas, dan keamanan. Tujuan dari penelitian ini adalah merancang dan membangun sebuah sistem yang mengintegrasikan teknologi terkini seperti Vue.js untuk antarmuka yang responsif, Golang untuk *backend* berkinerja tinggi [8], RESTful API untuk komunikasi antar layanan [4], serta Keycloak sebagai sistem otentikasi dan otorisasi terpusat untuk menjamin keamanan [9]. Pengujian keamanan sistem akan dilakukan dengan merujuk pada standar OWASP untuk mengidentifikasi dan mengatasi kerentanan [10]. Target terukur mencakup implementasi kontrol akses yang ketat (A01: Broken Access Control), proteksi terhadap injeksi (A03: Injection), validasi konfigurasi keamanan (A05: Security Misconfiguration), dan penerapan mekanisme autentikasi modern yang terpusat (A07: Identification and Authentication Failures) untuk menggantikan sistem lama. Pengembangan sistem ini diharapkan dapat menjadi model transformasi digital yang efektif di lingkungan pendidikan tinggi. Adapun tujuan dari penelitian ini adalah:

a) Mengembangkan sistem informasi akademik berbasis web yang modern dan aman untuk meningkatkan fleksibilitas pengelolaan data, pengalaman pengguna, serta keamanan data akademik.
b) Menerapkan arsitektur *microservices* untuk meningkatkan fleksibilitas, keamanan, dan kemudahan integrasi data akademik.
c) Membangun antarmuka pengguna yang responsif dan intuitif menggunakan *framework* Vue.js, serta menerapkan autentikasi dan otorisasi berbasis Keycloak dengan fitur *Single Sign-On* (SSO) untuk memastikan keamanan data.

2. LANDASAN TEORI

2.1. Arsitektur *Microservices*

Arsitektur *microservices* adalah sebuah pendekatan untuk pengembangan aplikasi sebagai kumpulan layanan-layanan kecil yang independen dan terdistribusi. Setiap layanan dirancang untuk menjalankan satu fungsi bisnis spesifik, memiliki basis datanya sendiri, dan berkomunikasi dengan layanan lain melalui antarmuka yang terdefinisi dengan baik, umumnya menggunakan *Application Programming Interface* (API) [11]. Keunggulan utama dari arsitektur ini adalah

peningkatan modularitas. Meskipun praktik ideal (seperti dalam NIST SP 800-204) merekomendasikan basis data terpisah per layanan (*database-per-service*), penelitian ini mengadopsi pendekatan basis data tunggal yang ter-segmentasi secara logis untuk menyederhanakan implementasi. Dalam pendekatan ini, setiap layanan tetap memiliki "kepemilikan" data yang jelas atas skema atau tabel yang relevan. Hal ini memberikan fleksibilitas dan skalabilitas yang jauh lebih tinggi dibandingkan dengan arsitektur monolitik, di mana seluruh komponen aplikasi terikat erat dalam satu unit tunggal yang kaku [1, 2]. Pendekatan ini sangat sesuai untuk sistem yang kompleks karena memecah masalah besar menjadi bagian-bagian yang lebih mudah dikelola.

2.2. Teknologi Pendukung

Sistem yang dikembangkan dalam penelitian ini ditopang oleh beberapa teknologi kunci yang terintegrasi dalam arsitektur microservices:

a. *Representational State Transfer* (REST): REST adalah gaya arsitektur yang digunakan untuk merancang API yang fleksibel dan efisien untuk komunikasi antar aplikasi perangkat lunak melalui internet [12, 13, 14]. Dalam sistem ini, RESTful API berfungsi sebagai protokol komunikasi standar antar *microservices*, serta antara *backend* dan *frontend*. Penggunaan metode HTTP standar seperti GET, POST, PUT, dan DELETE memungkinkan interaksi tanpa status (*stateless*), yang mendorong keterpisahan (*loose coupling*) antara klien dan server, sehingga meningkatkan fleksibilitas dan reusabilitas komponen [15, 16, 17].

b. Vue.js: Dipilih sebagai *framework frontend*, Vue.js adalah *framework* JavaScript progresif yang memungkinkan pembuatan antarmuka pengguna yang interaktif dan responsif. Arsitekturnya yang berbasis komponen sangat mendukung pengembangan aplikasi yang modular dan mudah dipelihara [18]. Salah satu fitur unggulannya adalah sistem pengikatan data reaktif dua arah, yang menyederhanakan proses pengembangan dengan menyinkronkan perubahan antara antarmuka pengguna dan model data secara otomatis. Hal ini sangat bermanfaat untuk aplikasi yang memerlukan pembaruan konten dinamis tanpa memuat ulang seluruh halaman [19].

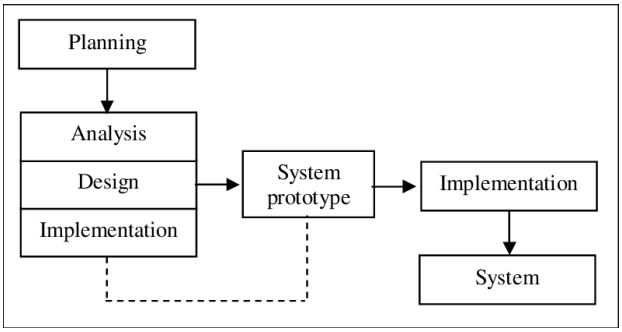
c. Golang: Bahasa pemrograman ini digunakan untuk membangun layanan *backend*. Golang dipilih karena performanya yang tinggi, efisiensi, dan dukungan bawaan untuk konkurensi, yang memungkinkannya menangani banyak permintaan secara simultan tanpa kompleksitas yang sering ditemukan pada bahasa lain [1, 8]. Pustaka standarnya yang kuat menyediakan dukungan luas untuk membangun server web dan menangani permintaan HTTP, menjadikannya pilihan ideal untuk mengembangkan API RESTful yang skalabel dan mudah dipelihara dalam lingkungan *microservices*.

d. Keycloak: Sebagai solusi manajemen identitas dan akses *open-source*, Keycloak menyediakan kerangka kerja yang kuat untuk autentikasi dan otorisasi terpusat. Implementasi Keycloak dengan fitur *Single Sign-On* (SSO) dan dukungan terhadap protokol standar seperti OAuth 2.0 dan OpenID Connect secara signifikan meningkatkan keamanan sistem [9, 20]. Dengan memusatkan manajemen pengguna, Keycloak menyederhanakan proses login dan memastikan bahwa kebijakan kontrol akses diterapkan secara konsisten di seluruh layanan, yang merupakan aspek krusial dalam arsitektur terdistribusi.

3. METODE PENELITIAN

Penelitian ini menggunakan pendekatan kualitatif dan kuantitatif deskriptif. Pendekatan kualitatif digunakan untuk menganalisis kelemahan sistem MASTER yang ada melalui studi literatur, observasi, dan wawancara, yang hasilnya menjadi dasar perumusan kebutuhan sistem baru. Pendekatan kuantitatif diterapkan untuk mengevaluasi sistem yang dikembangkan melalui UAT.

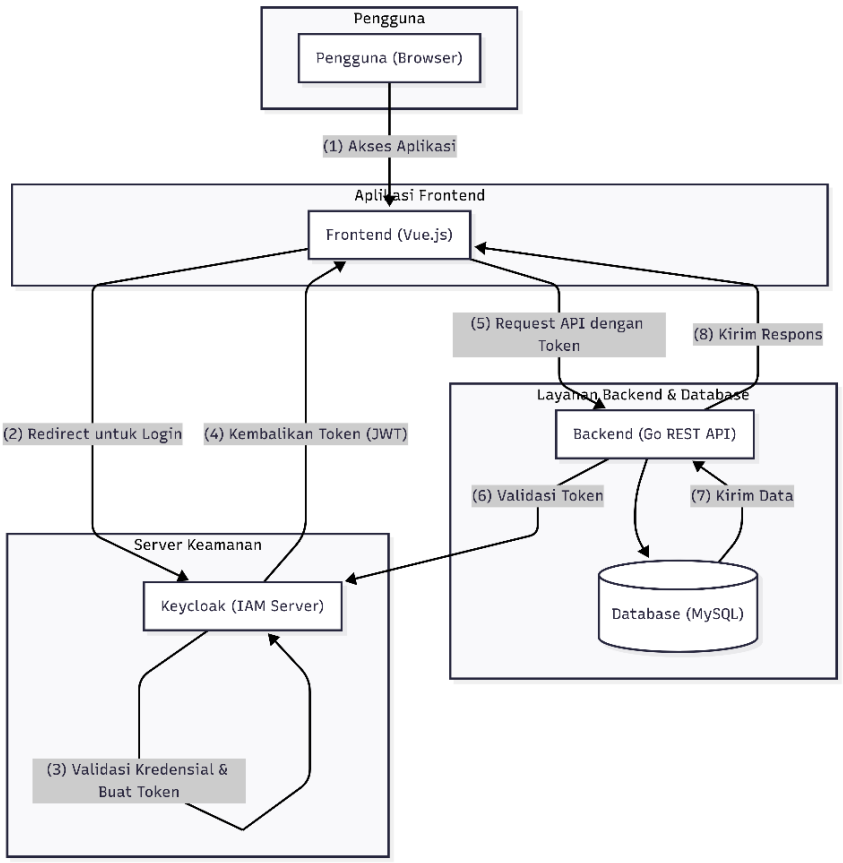
Instrumen UAT menggunakan kuesioner dengan skala Likert 5 poin (1 = Sangat Tidak Setuju hingga 5 = Sangat Setuju). Butir-butir pertanyaan (item) dirancang spesifik untuk setiap peran guna mengukur konstruk utama, meliputi: (a) kemudahan dan keamanan proses login (SSO), (b) persepsi terhadap antarmuka (UI) dan kemudahan navigasi, (c) fungsionalitas fitur-fitur utama (seperti melihat jadwal dan mengelola data), serta (d) kepuasan pengguna secara keseluruhan. Pengambilan data dilakukan menggunakan non-probability sampling dengan komposisi responden terdiri dari 21 pengguna yang mewakili empat peran utama: 12 Mahasiswa, 2 Dosen, 3 Staf AAK, dan 4 Staf Unit TI.



Gambar 1. Metode Penelitian *Prototype*

Metode pengembangan sistem yang digunakan adalah Metode Prototype. Metode ini dipilih karena memungkinkan pengembangan sistem secara iteratif dengan melibatkan pengguna secara aktif dalam setiap tahap, sehingga dapat memastikan sistem yang dibangun sesuai dengan kebutuhan sebelum implementasi penuh. Tahapan penelitian yang ada pada Gambar. 1 dilakukan mengikuti alur metode prototype dengan penjelasan sebagai berikut:

1. *Planning* (Perencanaan): Mengumpulkan data awal dari evaluasi sistem yang ada, hasil wawancara, dan studi literatur untuk mengidentifikasi permasalahan utama dan merumuskan kebutuhan sistem baru.
2. *Analysis* (Analisis): Menganalisis kebutuhan fungsional dan non-fungsional sistem secara mendalam, termasuk memetakan fitur yang diperlukan oleh setiap peran pengguna dan mengidentifikasi aspek keamanan serta integrasi.
3. *Design* (Perancangan): Membuat *blueprint* sistem yang mencakup desain arsitektur *microservices*, perancangan basis data, dan desain antarmuka pengguna (UI).
4. *Implementation* (Implementasi Awal & Umpan Balik): Mengembangkan *prototype* fungsional pertama yang kemudian diuji oleh pengguna untuk mendapatkan umpan balik awal. Berdasarkan masukan tersebut, dilakukan penyempurnaan secara iteratif.
5. *System* (Implementasi Final & Evaluasi): Mengembangkan sistem secara menyeluruh berdasarkan *prototype* yang telah disempurnakan. Tahap akhir melibatkan pengujian keamanan, pengukuran performa, dan UAT final untuk memvalidasi sistem secara keseluruhan.



Gambar 2. Gambaran Umum Arsitektur Sistem

Arsitektur sistem dirancang dengan pendekatan microservices yang memisahkan komponen-komponen utama untuk meningkatkan fleksibilitas dan skalabilitas seperti yang ada pada Gambar 2. Interaksi antara komponen-komponen utama, yaitu Pengguna Frontend (Vue.js), Backend (Go), Database (MySQL), dan server autentikasi terpusat (Keycloak), diatur melalui alur komunikasi yang jelas untuk memastikan keamanan dan efisiensi.

4. HASIL DAN PEMBAHASAN

4.1. Implementasi Arsitektur *Microservices*

Sistem berhasil diimplementasikan sesuai dengan rancangan arsitektur *microservices* yang memisahkan antara *frontend*, *backend*, dan layanan autentikasi. *Frontend* dibangun menggunakan Vue.js yang menghasilkan antarmuka pengguna yang modern dan responsif. Layanan *backend*, yang dikembangkan dengan Golang, menyediakan *endpoint* API untuk setiap modul fungsional (data induk, perkuliahan, dan penilaian) dan berinteraksi dengan basis data MySQL melalui GORM. Seluruh proses autentikasi dan otorisasi dikelola secara terpusat oleh Keycloak, yang memastikan bahwa setiap permintaan ke *backend* divalidasi melalui token JWT. Pemisahan ini membuktikan bahwa arsitektur *microservices* dapat diimplementasikan secara efektif untuk menciptakan sistem yang modular dan terstruktur.

4.2. Hasil Pengujian Keamanan

Pengujian keamanan dilakukan untuk mengevaluasi tingkat ketahanan aplikasi terhadap berbagai ancaman umum dengan mengacu pada standar OWASP Top 10:2021. Proses pengujian difokuskan pada identifikasi kerentanan kritis yang berpotensi memengaruhi kerahasiaan, integritas, dan ketersediaan sistem. Pengujian pada aplikasi MASTER lama dilakukan sebagai

baseline untuk memperoleh gambaran awal kondisi keamanan sistem sebelum dilakukan modernisasi arsitektur dan mekanisme pengamanan.

Tabel 1 Hasil Tes Keamanan Aplikasi MASTER Lama

Test ID	Kategori OWASP	Skenario	Hasil yang Diharapkan	Hasil Aktual	Kesimpulan
OLD-001	A01: *Broken Access Control*	Menguji adanya token anti-CSRF pada formulir untuk mencegah aksi ilegal.	Formulir dilindungi oleh token anti-CSRF.	ZAP tidak menemukan token anti-CSRF pada formulir yang ada.	Teridentifikasi - Rentan
OLD-002	A02: *Cryptographic Failures*	Memeriksa apakah server memaksa penggunaan koneksi aman (HTTPS).	Server menerapkan *header Strict-Transport-Security* (HSTS).	ZAP menemukan bahwa *header* HSTS tidak ada.	Teridentifikasi - Rentan
OLD-003	A03: *Injection*	Mengirim payload SQL Injection ke input yang terekspos.	Sistem menolak *payload* dan tidak mengeksekusi *query* berbahaya.	Pemindaian ZAP tidak menemukan kerentanan injeksi level dasar.	Tidak Teridentifikasi - Aman (Dasar)
OLD-004	A04: *Insecure Design*	Review arsitektur aplikasi untuk kelemahan desain.	Arsitektur yang digunakan modern dan aman.	Aplikasi menggunakan arsitektur monolitik yang kurang fleksibel dan aman.	Teridentifikasi - Rentan
OLD-005	A05: *Security Misconfiguration*	Memindai server untuk header keamanan yang hilang.	Server mengirimkan semua *header* keamanan yang direkomendasikan.	ZAP menemukan banyak *header* keamanan yang hilang, termasuk CSP dan X-Frame-Options.	Teridentifikasi - Rentan
OLD-006	A06: *Vulnerable Components*	Memeriksa versi komponen/library yang digunakan.	Tidak ada komponen usang atau rentan yang digunakan.	Tidak dapat diuji oleh ZAP karena memerlukan akses ke *source code*.	Tidak Teruji
OLD-007	A07: *Authentication Failures*	Memeriksa keamanan atribut *cookie* sesi.	*Cookie* menggunakan atribut SameSite untuk melindungi dari CSRF.	ZAP menemukan bahwa *cookie* sesi tidak memiliki atribut SameSite.	Teridentifikasi - Rentan
OLD-008	A08: *Data Integrity Failures*	Memastikan integritas dependensi saat instalasi/pembaruan.	Proyek menggunakan mekanisme *file lock* untuk verifikasi.	Tidak dapat diuji oleh ZAP tanpa akses ke *source code*.	Tidak Teruji
OLD-009	A09: *Security Logging Failures*	Memverifikasi apakah aktivitas penting dicatat oleh sistem.	Sistem memiliki *logging* dan *monitoring* yang memadai.	Tidak dapat diuji oleh ZAP karena memerlukan akses sisi server.	Tidak Teruji
OLD-010	A10: SSRF	Menguji apakah server bisa dipaksa membuat permintaan HTTP.	Server menolak permintaan yang tidak sah.	Pemindaian ZAP tidak menemukan kerentanan SSRF level dasar.	Tidak Teridentifikasi - Aman (Dasar)

Hasil pengujian keamanan pada aplikasi MASTER lama yang disajikan pada Tabel 1 menunjukkan adanya beberapa kerentanan dengan tingkat risiko menengah hingga tinggi. Kerentanan tersebut terutama berkaitan dengan kelemahan pada kontrol akses, validasi input, serta konfigurasi keamanan aplikasi. Temuan ini mengindikasikan bahwa pendekatan arsitektur yang digunakan pada sistem lama belum sepenuhnya memenuhi prinsip keamanan aplikasi web modern,

sehingga diperlukan perbaikan menyeluruh melalui penerapan arsitektur dan mekanisme keamanan yang lebih terstruktur.

Tabel 2 Hasil Tes Keamanan Aplikasi MASTER Baru

Test ID	Kategori OWASP	Skenario	Hasil yang Diharapkan	Hasil Aktual	Kesimpulan
NEW-001	A01: *Broken Access Control*	Pengguna dengan peran mahasiswa mencoba mengakses endpoint API khusus admin (misalnya /dosen).	ZAP Active Scan tidak menemukan celah akses. *Backend* menolak akses dengan status 403 Forbidden.	Hasil Active Scan ZAP tidak menemukan celah, membuktikan *middleware* otorisasi Keycloak efektif.	Tidak Teridentifikasi - Aman
NEW -002	A02: *Cryptographic Failures*	Lalu lintas jaringan dicegat saat proses otentikasi.	Proses otentikasi ke Keycloak terbukti menggunakan enkripsi HTTPS.	Proses otentikasi ke Keycloak terbukti menggunakan enkripsi HTTPS.	Tidak Teridentifikasi - Aman
NEW -003	A03: *Injection*	*Payload SQL Injection* dikirim melalui kolom input.	ZAP Active Scan tidak menemukan adanya kerentanan injeksi.	Hasil Active Scan ZAP tidak menemukan alert *SQL Injection*, membuktikan penggunaan GORM efektif.	Tidak Teridentifikasi - Aman
NEW -004	A04: *Insecure Design*	Review arsitektur aplikasi untuk kelemahan desain fundamental.	Pilihan arsitektur terbukti memitigasi banyak risiko dibanding desain monolitik.	Pilihan arsitektur microservices dengan otentikasi terpusat (Keycloak) merupakan desain yang aman secara inheren.	Tidak Teridentifikasi - Aman
NEW -005	A05: *Security Misconfiguration*	Server dipindai untuk *header* keamanan yang hilang dan konfigurasi CORS.	ZAP menemukan miskonfigurasi, namun berhasil diperbaiki dan diverifikasi.	Hasil pemindaian ZAP membuktikan masalah CORS & *Missing Security Headers* telah berhasil diperbaiki.	Tidak Teridentifikasi - Aman
NEW -006	A06: *Vulnerable Components*	Dependensi proyek diperiksa terhadap daftar kerentanan yang diketahui.	Verifikasi manual melalui *file lock* menunjukkan penggunaan versi terbaru yang aman.	Semua dependensi (*library*) yang digunakan adalah versi stabil terbaru, dibuktikan dengan file lock dan hasil audit dependensi.	Tidak Teridentifikasi - Aman
NEW -007	A07: *Authentication Failures*	Token JWT yang tidak valid atau kedaluwarsa digunakan untuk mengakses API.	ZAP tidak menemukan celah validasi token. Backend menolak token tidak valid.	ZAP gagal menembus mekanisme sesi dan validasi token JWT yang diatur oleh Keycloak.	Tidak Teridentifikasi - Aman
NEW -008	A08: *Data Integrity Failures*	Proses instalasi dependensi diverifikasi keasliannya.	Proyek menggunakan go.sum & pnpm-lock.yaml yang memverifikasi *hash* dependensi.	Integritas dependensi dijamin oleh file ock (go.sum dan pnpm-lock.yaml) yang memverifikasi hash setiap library.	Tidak Teridentifikasi - Aman
NEW -009	A09: *Security Logging Failures*	Verifikasi pencatatan aktivitas API dan otentikasi.	Verifikasi manual menunjukkan backend Gin mencatat akses API & Keycloak mencatat otentikasi.	Logging permintaan HTTP dicatat oleh framework Gin, dan *logging* otentikasi dimonitor oleh Keycloak.	Tidak Teridentifikasi - Aman

NEW -010	A10: SSRF	Server diuji apakah bisa dipaksa membuat permintaan ke domain tak terduga.	ZAP *Active Scan* tidak menemukan adanya kerentanan SSRF dasar.	*Active Scan* ZAP mencakup pengujian SSRF dasar dan tidak menemukan adanya kerentanan.	Tidak Teridentifikasi - Aman

Sebagai pembanding, hasil pengujian keamanan pada aplikasi MASTER baru yang ditunjukkan pada Tabel 2 memperlihatkan peningkatan signifikan dibandingkan sistem sebelumnya. Seluruh skenario pengujian yang dilakukan menggunakan OWASP *Zed Attack Proxy* (ZAP) tidak menemukan kerentanan kritis maupun berisiko tinggi. Hal ini menunjukkan bahwa implementasi arsitektur *microservices* serta penerapan mekanisme autentikasi dan otorisasi terpusat berhasil mengatasi kelemahan yang teridentifikasi pada sistem lama. Dengan demikian, sistem MASTER baru dinilai memiliki tingkat kesiapan keamanan yang lebih baik dalam menghadapi ancaman aplikasi web sesuai standar OWASP Top 10:2021.

4.3. Hasil Penerimaan Pengguna (*User Acceptance Testing* - UAT)

UAT dilakukan untuk mengukur fungsionalitas, kemudahan penggunaan, dan kepuasan pengguna terhadap sistem baru. Pengujian ini melibatkan responden dari berbagai peran, yaitu Mahasiswa, Dosen, Staf AAK, dan Unit TI. Responden diminta untuk menjalankan serangkaian skenario tugas dan kemudian mengisi kuesioner dengan skala Likert (1-5) dengan hasil seperti ditampilkan pada Tabel 3.

Tabel 3. Hasil Responden Pertanyaan

Unit Kerja	Jumlah Responden	Pertanyaan 1	Pertanyaan 2	Pertanyaan 3	Pertanyaan 4	Pertanyaan 5	Pertanyaan 6	Pertanyaan 7
Dosen	2	2.00	5.00	5.00	4.50	5.00	4.50	5.00
Mahasiswa	12	5.00	4.75	4.83	4.83	4.75	4.92	4.83
Unit TI	4	4.00	4.75	5.00	5.00	4.75	4.00	4.25
AAK	3	5.00	4.33	4.67	4.33	4.67	4.67	4.67

Hasil UAT menunjukkan tingkat penerimaan yang sangat tinggi. Secara keseluruhan, responden dari semua kelompok memberikan skor rata-rata yang sangat positif (mendekati 5.0) untuk semua aspek yang dinilai. Sistem baru dinilai lebih responsif, intuitif, dan memberikan pengalaman pengguna yang jauh lebih baik dibandingkan sistem sebelumnya. Proses *login* melalui SSO juga dinilai lebih praktis dan aman.

4.4. Pembahasan

Hasil implementasi dan pengujian secara kolektif menunjukkan bahwa transisi dari arsitektur monolitik ke *microservices* berhasil menjawab permasalahan yang diidentifikasi pada sistem lama. Arsitektur *microservices* tidak hanya memberikan fondasi teknis yang lebih fleksibel dan skalabel, tetapi juga secara langsung berkontribusi pada peningkatan keamanan dan kepuasan pengguna. Hasil pengujian keamanan OWASP yang solid membuktikan bahwa pemilihan teknologi modern seperti Keycloak dan praktik pengembangan yang aman mampu memitigasi risiko secara efektif. Di sisi lain, skor UAT yang tinggi mengonfirmasi bahwa perbaikan teknis ini berhasil diterjemahkan menjadi pengalaman pengguna yang lebih baik, yang merupakan salah satu tujuan utama dari penelitian ini.

5. KESIMPULAN

Setelah melalui seluruh tahapan penelitian yang mencakup perancangan arsitektur, implementasi teknis, serta serangkaian pengujian keamanan dan fungsionalitas terhadap sistem informasi akademik modern berbasis *microservices*, maka dapat ditarik beberapa kesimpulan utama yang menjawab tujuan penelitian sebagai berikut:

a. Transisi dari arsitektur monolitik ke *microservices* berhasil diimplementasikan, menghasilkan sebuah sistem yang secara fundamental lebih fleksibel, modular, dan mendukung integrasi data yang lebih baik. Keberhasilan pengembangan modul-modul yang terpisah dan berkomunikasi melalui RESTful API membuktikan kelayakan arsitektur ini untuk modernisasi sistem akademik.
b. Penerapan Keycloak dengan fitur *Single Sign-On* (SSO) dan pengujian keamanan berbasis standar OWASP Top 10:2021 terbukti mampu memperkuat aspek keamanan sistem secara signifikan. Hasil pengujian menunjukkan bahwa sistem yang dikembangkan tidak memiliki kerentanan kritis, serta menyediakan mekanisme autentikasi terpusat yang aman dan andal.
c. Persepsi pengguna terhadap sistem baru sangat positif. Berdasarkan hasil *User Acceptance Testing* (UAT), sistem yang dikembangkan dinilai lebih fungsional, mudah digunakan, dan responsif. Hal ini menunjukkan bahwa perbaikan pada arsitektur dan teknologi berhasil memberikan pengalaman pengguna yang lebih baik dibandingkan dengan sistem sebelumnya.

Penelitian ini memiliki beberapa batasan yang penting untuk dicatat. Pertama, evaluasi UAT menggunakan metode *non-probability sampling* dengan ukuran sampel yang kecil dan tidak seimbang (2 Dosen, 12 Mahasiswa, 3 AAK, 4 TI). Selain itu, validitas instrumen kuesioner tidak diuji secara statistik menggunakan uji reliabilitas (seperti *Cronbach's Alpha*), sehingga hasil kepuasan pengguna bersifat perseptif dan tidak dapat digeneralisasi.

Kedua, dari sisi keamanan, pengujian (DAST) difokuskan pada *endpoint* yang terekspos ke pengguna. Penelitian ini belum mencakup pengujian keamanan komunikasi internal *service-to-service* (antar-layanan), seperti implementasi *mutual TLS* (mTLS) atau *token exchange*, dan belum melakukan pengujian DAST terotentikasi penuh untuk setiap peran pengguna. Batasan-batasan ini dapat menjadi acuan untuk penelitian selanjutnya.

REFERENSI

[1] Y. Hong and D. Kim, "AN INNOVATIVE METHODOLOGY FOR TRANSITIONING FROM MONOLITH TO MICROSERVICES," *ICIC Express Letters*, vol. 17, no. 4, pp. 463-470, Apr. 2023, doi: 10.24507/icicel.17.04.463.

[2] G. Blinowski, A. Ojdowska, and A. Przybylek, "Monolithic vs. Microservice Architecture: A Performance and Scalability Evaluation," *IEEE Access*, vol. 10, pp. 20357-20374, 2022, doi: 10.1109/ACCESS.2022.3152803.

[3] B. N. Silva, M. Khan, and K. Han, "Integration of Big Data analytics embedded smart city architecture with RESTful web of things for efficient service provision and energy management," *Future Generation Computer Systems*, vol. 107, pp. 975-987, Jun. 2020, doi: 10.1016/J.FUTURE.2017.06.024.

[4] R. Sinaga, "Pengembangan Model Penilaian Kepatuhan Salah Satu Perguruan Tinggi Terhadap Standar ISO 27001:2022," *Jurnal Teknik Informatika dan Sistem Informasi*, vol. 9, no. 3, 2024, doi: 10.28932/jutisi.v9i3.6850.

[5] I. P. A. E. D. Udayana, "Integrasi Sistem Single Sign On Pada Sistem Informasi Akademik, Web Information System Dan Learning Management System Berbasis Central Authentication Service," *Jurnal RESISTOR (Rekayasa Sistem Komputer)*, vol. 1, no. 1, 2018, doi: 10.31598/jurnalresistor.vlil.265.

[6] Gemawaty, C. A., & Yuliani, Y. (2024). MANAJEMEN IDENTITAS DAN AKSES DALAM KEAMANAN SISTEM INFORMASI (PENDEKATAN LITERATURE REVIEW). Jurnal Manajamen Informatika Jayakarta, 4(4), 396-403.

[7] Z. Zhang, M. Chabbi, A. Welc, and T. Sherwood, "Optimistic concurrency control for real-world go programs," in *2021 USENIX Annual Technical Conference*, 2021.

[8] A. Chatterjee and A. Prinz, "Applying Spring Security Framework with Keycloak-Based OAuth2 to Protect Microservice Architecture APIs: A Case Study," *Sensors*, vol. 22, no. 5, Mar. 2022, doi: 10.3390/s22051703.

[9] O'rinboev, A. (2023). Optimizing Performance in a Dental Queue Web App. *Development of pedagogical technologies in modern sciences*, *2*(9), 5-9.

[10] J. Bogner, S. Kotstein, and T. Pfaff, "Do RESTful API design rules have an impact on the understandability of Web APIs?," *Empir Softw Eng*, vol. 28, no. 6, Nov. 2023, doi: 10.1007/s10664-023-10367-y.

[11] N. Robles, N. Potes, K. Garcés, J. L. C. Izquierdo, and J. Cabot, "Exploratory Analysis of the Structural Evolution of public REST APIs," in ***Congresso Ibero-Americano em Engenharia de Software (CIbSE)***, SBC, Apr. 2023, pp. 92–106.

[12] B. O. Emeka, S. Hidaka, and S. Liu, "A Practical Model Driven Approach for Designing Security Aware RESTfulWeb APIs Using SOFL," *IEICE Trans Inf Syst*, vol. E106.D, no. 5, pp. 986–1000, 2023, doi: 10.1587/transinf.2022EDP7194.

[13] P. Sinha* and K. A. Kumar, "REST APIs for Emerging Social Media Platforms," *International Journal of Innovative Technology and Exploring Engineering*, vol. 9, no. 5, pp. 652–659, Mar. 2020, doi: 10.35940/ijitee.E2608.039520.

[14] A. F. Rochim, A. Rafi, A. Fauzi, and K. T. Martono, "As-RaD System as a Design Model of the Network Automation Configuration System Based on the REST-API and Django Framework," *Kinetik: Game Technology, Information System, Computer Network, Computing, Electronics, and Control*, pp. 291–298, Nov. 2020, doi: 10.22219/kinetik.v5i4.1093.

[15] P. DYMORA, M. MAZUREK, and M. NYCZ, "Comparison of Angular, React, and Vue Technologies in the Process of Creating Web Applications on the User Interface Side," *Journal of Education, Technology and Computer Science*, vol. 4, no. 34, pp. 210–222, Dec. 2023, doi: 10.15584/jetacomps.2023.4.21.

[16] L. Gan, Y. Huang, and Y. Cheng, "Research on intelligent learning platform system based on Spring Boot," in *Proceedings of the 2022 International Conference on Computer Science, Information Engineering and Digital Economy (CSIEDE 2022)*, Atlantis Press, Jan. 2023. doi: 10.2991/978-94-6463-108-1_19.

[17] Y. Nakamura, T. Yamauchi, and T. Norimatsu, "Policy-Based Method for Applying OAuth 2.0-Based Security Profiles," *IEICE Trans Inf Syst*, vol. E106.D, no. 9, pp. 1364–1379, Sep. 2023, doi: 10.1587/transinf.2022ICP0004.

[18] A. Ahmad et al., "The Second-Factor Authentication System at CERN," ***EPJ Web of Conferences***, vol. 295, p. 04025, 2024.

[19] I. S. Matiushin, V. V Korkhov, I. Matiushin, and V. Korkhov, "Distributed Computing and Grid Technologies in Science and Education."

[20] E. Wündisch et al., "Development of a Trusted Third Party at a Large University Hospital: Design and Implementation Study," *JMIR Med Inform*, vol. 12, Jan. 2024, doi: 10.2196/53075.