Seminar Nasional Informatika (SENATIKA)
Prosiding Senatika 2025 **QRSBN:62-0151-02942-8**

Implementasi Arsitektur Microservices untuk Meningkatkan Skalabilitas Sistem Informasi Kampus

Muhamad Royhan Fadhli[1], Ezrifal Sany[2], Ahmad Husna Ahadi[3]
[1,2,3]Teknik Informatika, Universitas Nurdin Hamzah
e-mail: [1]froyhan0@gmail.com, [2]ezrifalsany@unh.ac.id, [3]ahmad_husna@unh.ac.id

Abstrak

Sistem tata kelola kampus dengan arsitektur monolitik sering menghadapi kendala dalam hal skalabilitas, efisiensi, dan pemeliharaan. Untuk mengatasi hal tersebut, penelitian ini merancang dan mengimplementasikan sistem tata kelola kampus berbasis arsitektur microservices yang terbagi menjadi dua layanan utama, yaitu User Service dan Internal Service. Pengembangan dilakukan melalui tahapan perancangan sistem, implementasi antarmuka pengguna berbasis REST API, serta pengujian performa menggunakan tiga metrik utama, yaitu response time, throughput, dan CPU usage. Hasil implementasi menunjukkan peningkatan performa yang signifikan dibandingkan sistem monolitik, dengan penurunan response time sebesar 34%, peningkatan throughput sebesar 50%, dan penurunan CPU usage sebesar 16%. Komunikasi antarservice berjalan stabil dengan waktu propagasi rata-rata di bawah 150 ms, serta dilengkapi mekanisme error handling (401, 404, 500) yang menjaga keandalan sistem. Namun, penerapan arsitektur microservices juga menimbulkan tantangan baru berupa kompleksitas komunikasi antar layanan, kebutuhan service orchestration, serta peningkatan beban pada proses deployment dan monitoring. Secara keseluruhan, penelitian ini membuktikan bahwa microservices mampu meningkatkan efisiensi, fleksibilitas, dan skalabilitas sistem tata kelola kampus, namun memerlukan strategi pengelolaan layanan yang lebih matang agar implementasinya optimal di lingkungan akademik.

Kata kunci: *Microservices, Tata Kelola Kampus, REST API, Kinerja Sistem, Arsitektur Perangkat Lunak.*

Abstract

Campus management systems with monolithic architecture often face challenges in terms of scalability, efficiency, and maintenance. To address these issues, this study designed and implemented a campus management system based on microservices architecture, which is divided into two main services, namely User Service and Internal Service. Development was carried out through the stages of system design, implementation of a REST API-based user interface, and performance testing using three main metrics, namely response time, throughput, and CPU usage. The implementation results showed a significant performance improvement compared to the monolithic system, with a 34% reduction in response time, a 50% increase in throughput, and a 16% reduction in CPU usage. Inter-service communication runs stably with an average propagation time of less than 150 ms, and is equipped with an error handling mechanism (401, 404, 500) that maintains system reliability. However, the implementation of a microservices architecture also poses new challenges in the form of complex inter-service communication, the need for service orchestration, and an increased burden on the deployment and monitoring processes. Overall, this study proves that microservices can improve the efficiency, flexibility, and scalability of campus management systems, but require a more mature service management strategy for optimal implementation in academic environments.

Keywords: *microservices, Campus Management, REST API, System Performance, Software Architecture.*

1. Pendahuluan

Perkembangan teknologi informasi telah mendorong institusi pendidikan tinggi untuk mengadopsi sistem informasi yang lebih adaptif dan terintegrasi guna menghadapi tantangan pengelolaan data yang semakin kompleks. Sistem tata kelola berbasis teknologi menjadi kunci untuk meningkatkan efisiensi dan kualitas layanan akademik maupun administrasi, serta memberikan pengalaman pengguna yang lebih baik bagi seluruh civitas akademika [1]. Dalam konteks global, integrasi sistem informasi di perguruan tinggi telah menjadi kebutuhan mendesak untuk memastikan daya saing dan keberlanjutan operasional institusi pendidikan.

Namun, banyak institusi pendidikan tinggi masih menghadapi tantangan dalam pengelolaan sistem informasi yang tidak terintegrasi. Sistem-sistem yang berjalan secara terpisah seperti sistem disposisi surat, pengajuan cuti, dan penilaian skripsi sering kali menimbulkan ineffisiensi operasional dan kompleksitas pengelolaan yang tidak perlu [2]. Kondisi ini mengharuskan pengguna untuk memiliki multiple accounts dan mengakses berbagai platform terpisah, yang tidak hanya membingungkan tetapi juga meningkatkan risiko kesalahan operasional dan duplikasi data.

Permasalahan tersebut tidak terlepas dari arsitektur sistem yang digunakan di banyak institusi pendidikan, yakni arsitektur monolitik. Arsitektur sistem monolitik yang selama ini dominan digunakan dalam pengembangan sistem informasi kampus menunjukkan keterbatasan signifikan dalam menghadapi peningkatan kompleksitas layanan. Sistem monolitik cenderung sulit dikembangkan dan dipelihara seiring bertambahnya fitur dan pengguna, serta rentan terhadap "resilient challenges" yaitu di mana kegagalan pada satu komponen dapat menyebabkan seluruh aplikasi mengalami masalah [3]. Keterbatasan ini menjadi penghambat utama dalam pengembangan sistem yang scalable dan maintainable.

Sebagai solusi untuk mengatasi keterbatasan arsitektur monolitik, arsitektur microservices menawarkan pendekatan yang membagi sistem menjadi layanan-layanan kecil yang dapat beroperasi secara independen [4]. Microservices memungkinkan setiap layanan dikembangkan secara modular dan dihubungkan melalui Application Programming Interface (API), memberikan keunggulan dalam hal scalability, ease of deployment, technology flexibility, fault isolation, dan faster development [5]. Pendekatan ini telah terbukti lebih optimal dibandingkan monolitik, terutama dalam menghadapi peningkatan jumlah pengguna dan pengolahan data yang kompleks.

REST (Representational State Transfer) API berperan penting sebagai mekanisme komunikasi dalam arsitektur microservices. REST API menyediakan arsitektur layanan web yang memungkinkan sistem berkomunikasi melalui protokol HTTP dengan karakteristik statelessness, arsitektur client-server, dan uniform interface [6]. Kombinasi microservices dengan REST API memberikan foundation yang solid untuk pengembangan sistem terdistribusi yang efisien dan scalable.

Beberapa perguruan tinggi di Indonesia mulai beralih dari arsitektur monolitik ke mikroservis. Universitas Jenderal Soedirman (Unsoed) misalnya, melaporkan bahwa mereka sebelumnya memiliki lebih dari tiga puluh sistem monolitik yang kemudian dikembangkan menjadi Super App berbasis mikroservis untuk meningkatkan integrasi layanan akademik [7]. Penelitian di Institut Teknologi Sepuluh Nopember (ITS) juga menunjukkan bahwa arsitektur mikroservis lebih unggul dalam hal keandalan dibandingkan arsitektur monolitik [8]. Namun, hingga kini belum ada data nasional yang memetakan jumlah institusi pendidikan yang menggunakan masing-masing arsitektur, sehingga penelitian ini diharapkan dapat mengisi kesenjangan tersebut.

Penelitian ini mengimplementasikan arsitektur microservices dalam sistem tata kelola kampus dengan fokus pada integrasi tiga layanan utama: disposisi surat, pengajuan cuti, dan penilaian skripsi. Implementasi dilakukan menggunakan teknologi Node.js dan Bun sebagai runtime environment untuk mendemonstrasikan fleksibilitas teknologi dalam arsitektur microservices. Sistem prototype yang dikembangkan menekankan pada pemisahan layanan (service separation) dan komunikasi antar layanan menggunakan REST API.

Penelitian ini bertujuan untuk merancang dan mengimplementasikan prototipe sistem tata kelola kampus berbasis microservices, serta mengevaluasi peningkatan performa dan efisiensi dibandingkan sistem monolitik. Pendekatan ini diharapkan mampu mengatasi keterbatasan arsitektur monolitik serta menjadi acuan bagi pengembangan sistem informasi modern yang lebih adaptif, terintegrasi, dan responsif terhadap kebutuhan pengguna. Selain itu, penelitian ini juga berkontribusi pada pemahaman akademik terkait praktik terbaik (best practices) migrasi dari sistem monolitik menuju arsitektur microservices dalam konteks sistem informasi kampus.

2. Metode Penelitian

2.1 Pendekatan Penelitian

Studi kasus dalam penelitian ini dipilih karena memberikan ruang bagi peneliti untuk memahami secara mendalam objek penelitian, yaitu sistem tata kelola kampus yang sedang dikembangkan dengan arsitektur microservices. Melalui pendekatan ini, penelitian tidak sekadar menyoroti aspek teknis implementasi, melainkan juga melihat bagaimana sistem tersebut berperan dalam meningkatkan efektivitas serta efisiensi tata kelola akademik. Fokus dari studi kasus terletak pada analisis menyeluruh terhadap satu objek tanpa membandingkannya dengan sistem lain. Studi kasus merupakan metode yang menitikberatkan analisis pada suatu praktik atau fenomena tertentu dengan menggunakan berbagai sumber informasi, seperti pemberitaan media, pernyataan publik, dokumen hukum, maupun laporan pihak yang terlibat. Artinya, studi kasus tidak hanya mengandalkan satu jenis data, tetapi memanfaatkan beragam dokumen dan informasi untuk memperoleh gambaran yang utuh mengenai permasalahan yang diteliti [9].

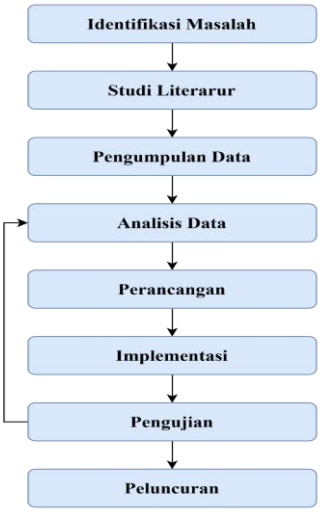
Untuk memastikan validasi ilmiah hasil implementasi, dilakukan pengukuran performa sistem berbasis tiga metrik utama, yaitu:

1. Response Time (ms): waktu yang dibutuhkan sistem untuk merespons permintaan dari klien.
2. Throughput (req/s): jumlah permintaan (request) yang dapat diproses oleh sistem dalam satu detik.
3. CPU Usage (%): tingkat penggunaan sumber daya prosesor selama pengujian berlangsung.

Ketiga metrik ini dipilih karena mewakili aspek kecepatan, efisiensi pemrosesan, dan konsumsi sumber daya, yang menjadi indikator langsung dari skalabilitas sistem.

2.2 Kerangka Kerja Penelitian

Kerangka kerja penelitian dirancang untuk memberikan alur sistematis dalam mengeksplorasi penerapan arsitektur microservices pada sistem akademik. Kerangka ini menjelaskan tahapan yang dilakukan mulai dari identifikasi masalah hingga evaluasi hasil. Secara visual, kerangka kerja penelitian dapat dilihat pada Gambar berikut.



Gambar 1 Kerangka Kerja Penelitian

Kerangka kerja penelitian pada Gambar diatas terdiri atas beberapa tahapan yang saling berhubungan. Tahap pertama adalah identifikasi masalah, yaitu menentukan fokus penelitian mengenai implementasi microservices. Selanjutnya dilakukan studi literatur untuk memperkuat landasan teori. Tahap ketiga adalah pengumpulan data melalui observasi, wawancara, dan analisis dokumen teknis.

Data yang diperoleh kemudian dianalisis pada tahap analisis data dengan pendekatan deskriptif kualitatif. Hasil analisis menjadi dasar dalam perancangan sistem yang memuat desain arsitektur microservices. Perancangan ini kemudian diwujudkan pada tahap implementasi sistem. Setelah implementasi, dilakukan pengujian untuk menilai keandalan, integrasi, serta performa sistem. Tahap akhir adalah evaluasi, yaitu menilai efektivitas sistem serta memberikan rekomendasi untuk pengembangan lebih lanjut.

2.3 Teknologi Yang Digunakan

Dalam penelitian ini digunakan beberapa teknologi utama yang mendukung pembangunan arsitektur microservices pada sistem tata kelola kampus. Pertama, penerapan Virtual Machine (VM) dimanfaatkan untuk menyediakan lingkungan pengembangan yang fleksibel dan terisolasi. VM memungkinkan sistem operasi dan aplikasi dijalankan seolah-olah berada pada perangkat keras fisik, padahal berjalan di atas lapisan virtualisasi. Teknologi ini memberikan manfaat seperti isolasi lingkungan, efisiensi penggunaan sumber daya, portabilitas, serta peningkatan keamanan [10].

Selanjutnya, penelitian ini juga mengimplementasikan praktik Continuous Integration dan Continuous Deployment (CI/CD) guna mendukung otomatisasi integrasi kode, pengujian, hingga deployment. CI/CD memungkinkan setiap perubahan kode dapat langsung diuji dan dirilis secara konsisten tanpa proses manual yang kompleks. Penerapan model ini mempercepat siklus rilis, meningkatkan stabilitas, serta memungkinkan pembaruan independen pada setiap layanan microservice. Dalam implementasinya, CI/CD melibatkan penggunaan pipeline otomatis yang mendukung pengelolaan kode secara paralel oleh tim pengembang [11].

Untuk kerangka kerja pengembangan, digunakan Fastify pada layanan backend. Fastify merupakan framework web berbasis Node.js yang berfokus pada performa tinggi dengan arsitektur plugin yang modular, mendukung validasi skema JSON, serta integrasi logging. Dengan karakteristik tersebut, Fastify sangat sesuai untuk mendukung layanan microservice seperti autentikasi, manajemen data pengguna, serta pengelolaan administrasi kampus. Pada sisi frontend, digunakan kombinasi React dan Vite. React merupakan pustaka JavaScript yang berorientasi pada pengembangan antarmuka berbasis komponen sehingga memudahkan pembangunan UI yang kompleks [12]. Sementara itu, Vite berperan sebagai alat build modern

dengan keunggulan startup cepat dan hot module replacement yang efisien, sehingga mempercepat siklus pengembangan frontend.

Sebagai pendukung tampilan antarmuka, penelitian ini memanfaatkan Tailwind CSS, yaitu framework CSS dengan pendekatan utility-first. Tailwind memungkinkan pengembang menggunakan kelas utilitas untuk mengatur gaya secara langsung pada elemen HTML, sehingga mempercepat proses desain sekaligus menjaga konsistensi tampilan. Selain itu, framework ini juga mendukung desain responsif dan kustomisasi mendalam, yang sangat bermanfaat dalam pengembangan aplikasi front-end modern.

Pada sisi penyimpanan data, sistem ini menggabungkan dua jenis basis data relasional, yaitu PostgreSQL dan MySQL. PostgreSQL digunakan untuk mengelola data pengguna, autentikasi, serta pengajuan cuti. Database ini dikenal dengan keandalannya dalam menjaga integritas data melalui penerapan konsep ACID, dukungan terhadap berbagai tipe data modern, serta fleksibilitas tinggi dalam integrasi lintas bahasa pemrograman [13]. Sementara itu, MySQL digunakan untuk menangani data administrasi internal, termasuk disposisi surat, manajemen jadwal skripsi, dan penilaian skripsi. MySQL dipilih karena performanya yang cepat, kompatibilitas luas, serta sifatnya yang open-source sehingga mendukung kebutuhan pengelolaan data akademik dalam jumlah besar dengan efisiensi tinggi [14].

Dengan kombinasi teknologi tersebut, penelitian ini berhasil membangun ekosistem microservice yang terdistribusi dengan baik, di mana setiap layanan memiliki tanggung jawab, teknologi, serta lingkungan pengembangan yang spesifik. Pendekatan ini tidak hanya memberikan fleksibilitas dalam pengembangan, tetapi juga meningkatkan skalabilitas, stabilitas, dan efisiensi pengelolaan sistem tata kelola kampus.

3. Hasil dan Pembahasan

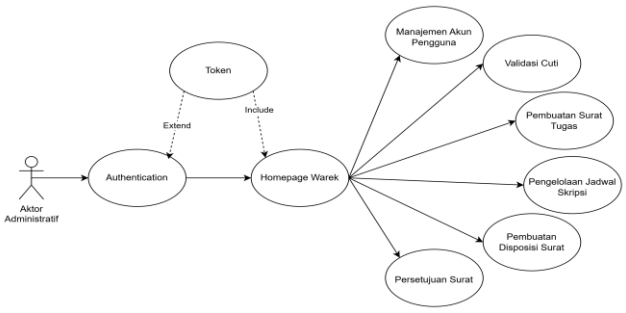
3.1 Pemodelan Sistem

Untuk mendukung fungsi yang ada pada use case, sistem menggunakan dua basis data terpisah sesuai dengan pendekatan microservice:

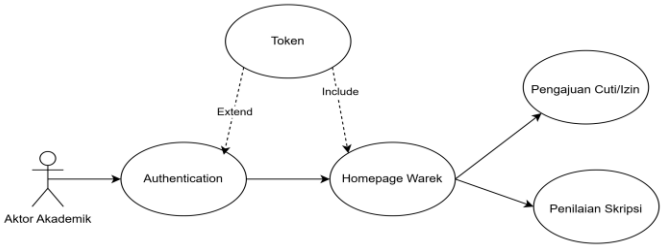
a. Use Case

Sistem tata kelola kampus berbasis microservices ini dirancang dengan dua domain layanan utama, yaitu User Service dan Internal Service.

Setiap layanan memiliki peran (aktor) dan fungsi yang berbeda, namun saling berinteraksi melalui mekanisme REST API.



Gambar 2 Administratif Usecase



Gambar 3 Akademik Usecase

Untuk menyederhanakan model, aktor sistem dibagi menjadi dua kategori utama:

1. Aktor Administratif: Admin, Officer, dan Rektor — bertanggung jawab terhadap pengelolaan data dan validasi proses.
2. Aktor Akademik: Dosen dan Karyawan — berinteraksi langsung dengan sistem dalam proses perizinan, jadwal, dan penilaian.

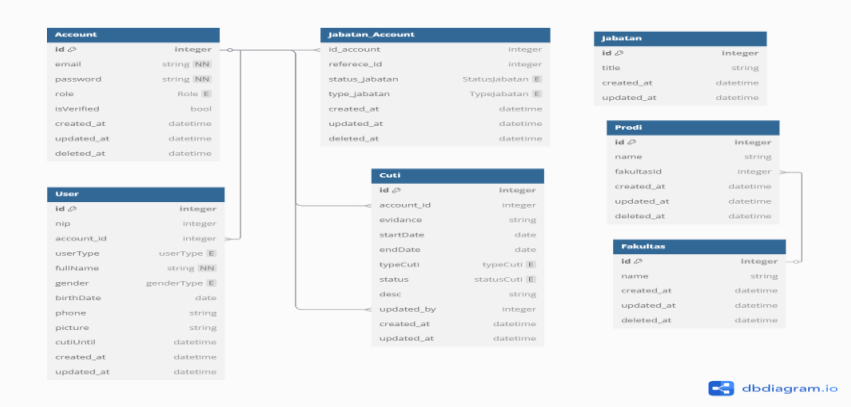
Table 1 Definisi Use Case

No	Use Case	Deksripsi	Aktor yang terlibat
1	Authentication	Login dan logout untuk mendapatkan akses sesuai role.	Semua Role
2	Manajemen Data User	Pengelolaan akun, profil, jabatan, fakultas, dan program studi.	Admin
3	Manajemen Perizinan	Pengajuan, konfirmasi, dan monitoring cuti/izin.	Admin, Warek, Admin User, Dosen, Karyawan
4	Pengelolaan Jadwal Skripsi	Pembuatan, pengelolaan, dan monitoring jadwal sidang skripsi.	Admin, Officer, Rektor
5	Pengelolaan Surat & Disposisi	Pembuatan disposisi, konfirmasi disposisi, serta monitoring aktivitas surat.	Admin, Rektor, Warek, Officer
6	Pembuatan Surat Tugas	Pembuatan surat tugas bagi dosen/karyawan.	Warek
7	Penilaian Skripsi	Pemberian nilai pada sidang skripsi sesuai jadwal.	Dosen, Warek
8	Monitoring User	Melihat daftar dan status aktivitas pengguna.	Rektor, Warek

b. ERD User-Service

User-service bertanggung jawab terhadap autentikasi, manajemen data pengguna, dan pengajuan cuti.

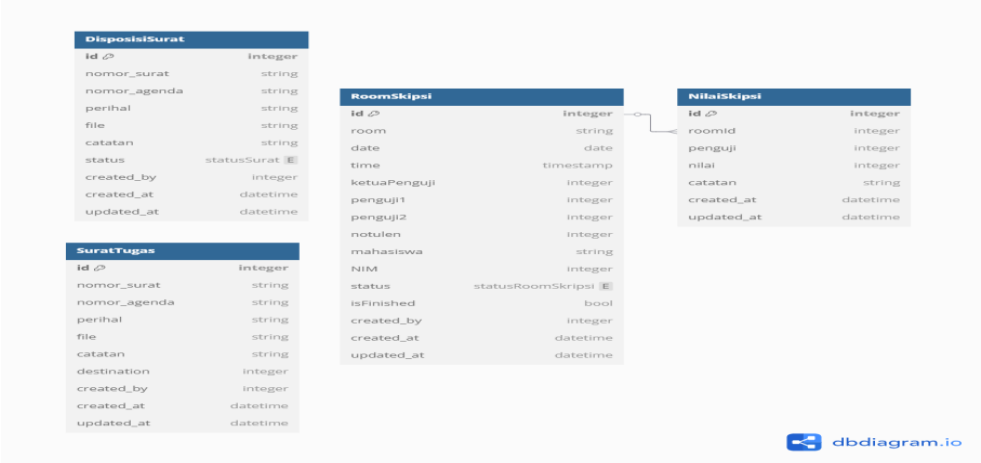
- Entitas utama dalam layanan ini meliputi: User, Role, Permission, serta Cuti.
- Relasi utama terbentuk antara User dan Role (satu pengguna dapat memiliki satu atau lebih peran), serta antara User dan Cuti (satu pengguna dapat mengajukan banyak cuti).
- Dengan struktur ini, user-service mampu menangani autentikasi yang aman sekaligus mendukung alur persetujuan cuti.



Gambar 4 ERD user-service

c. ERD Internal-Service

Internal-service menangani proses disposisi surat, manajemen jadwal skripsi, dan penilaian skripsi. Entitas utama meliputi: Surat Tugas, Disposisi Surat, Jadwal Skripsi, dan Penilaian Skripsi. Hubungan yang terbentuk antara lain: Surat terkait dengan Disposisi, serta Dosen yang terhubung melalui Penilaian dan JadwalSkripsi. Dengan pemisahan ini, layanan internal lebih fokus pada aktivitas administratif dan akademik yang menjadi inti proses tata kelola kampus.



Gambar 5 ERD internal-service

d. Integrasi Use Case & ERD

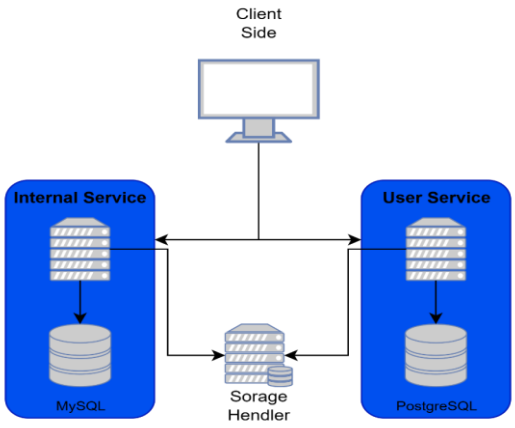
Kedua layanan (user-service dan internal-service) saling melengkapi dan dihubungkan melalui frontend (tkunh) yang dikembangkan dengan React + Vite. Use case yang dijalankan oleh pengguna di sisi frontend akan memanggil API dari masing-masing service sesuai domain tanggung jawabnya. Contoh: saat mahasiswa login dan mengajukan cuti → request masuk ke user-service. Saat mahasiswa mengikuti sidang skripsi dan dosen memberikan penilaian → request diproses oleh internal-service.

3.2 Implementasi Arsitektur Microservices

a. Implementasi Keseluruhan Sistem

Sistem tata kelola kampus dibangun dengan pendekatan arsitektur microservices yang memisahkan aplikasi menjadi dua layanan utama, yaitu User Service dan Internal Service. Kedua service berkomunikasi melalui protokol HTTP dengan REST API, sehingga pertukaran data dapat dilakukan tanpa adanya ketergantungan langsung.

.



Gambar 6 Arsitektur Aplikasi

b. Implementasi User Service

User Service berperan sebagai pusat autentikasi dan pengelolaan data pengguna. Service ini dikembangkan menggunakan Fastify (Node.js Runtime) dengan TypeScript, dan menyimpan data pada PostgreSQL.

Fungsi utama User Service meliputi:

- Manajemen data pengguna (registrasi, login, update profil).
- Pengelolaan jabatan dan role (otorisasi).
- Pengajuan dan pengelolaan cuti dosen/staf.

c. Implementasi Internal Service

Internal Service menangani data akademik internal kampus, terutama yang berkaitan dengan jadwal skripsi, disposisi surat, dan penilaian sidang. Service ini juga dikembangkan dengan Fastify + TypeScript namun menggunakan MySQL sebagai database.

Fungsi utama Internal Service:

- Pengelolaan jadwal sidang skripsi dan ruangan.
- Pembuatan disposisi surat dan surat tugas.
- Input penilaian skripsi oleh dosen penguji.

d. Implementasi Frontend

Antarmuka pengguna dikembangkan terpisah dari backend dengan React + Vite, ditulis dalam TypeScript, serta menggunakan TailwindCSS dan Shadcn UI untuk styling. Runtime yang digunakan adalah Bun, sehingga aplikasi frontend lebih ringan dan cepat. Frontend mengonsumsi API dari User Service dan Internal Service, melakukan validasi data di sisi klien, dan menampilkan informasi secara interaktif.

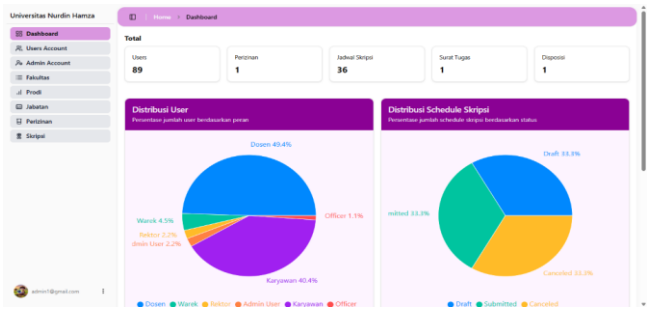
e. Implementasi Interface Antar Komponen

Integrasi antar service dan frontend dilakukan melalui kontrak API yang terdokumentasi menggunakan Postman Collection. Semua komunikasi antar service maupun dengan frontend menggunakan format JSON dengan status kode HTTP standar (200 OK, 201 Created, dsb). Respons API terdiri atas metadata dan payload, sehingga mudah diproses oleh frontend. Modularisasi kode di setiap service memudahkan pengembangan dan memungkinkan penambahan service baru di masa depan.

f. Implementasi Antarmuka Pengguna (UI)

UI dibangun untuk menguji komunikasi antar service serta menjadi media interaksi bagi pengguna (dosen, karyawan, administrator). Beberapa halaman utama:

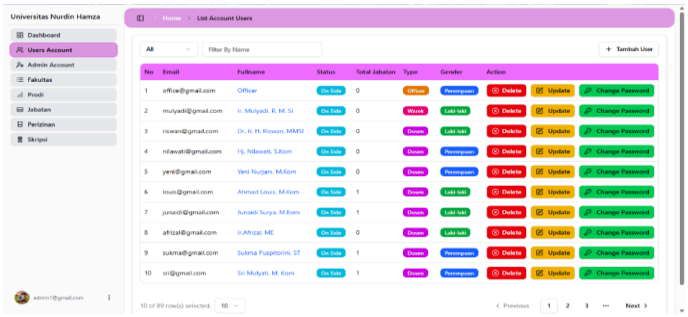
- **Dashboard:** menampilkan ringkasan data cuti, disposisi surat, dan jadwal skripsi.



Gambar 7 Dashboard Interface

Dashboard berfungsi sebagai pusat informasi terpadu. Data yang ditampilkan berasal dari User Service dan Internal Service melalui endpoint API terpisah, sehingga menjadi bukti nyata komunikasi lintas layanan berjalan konsisten tanpa keterlambatan data yang berarti (latency < 150 ms selama pengujian).

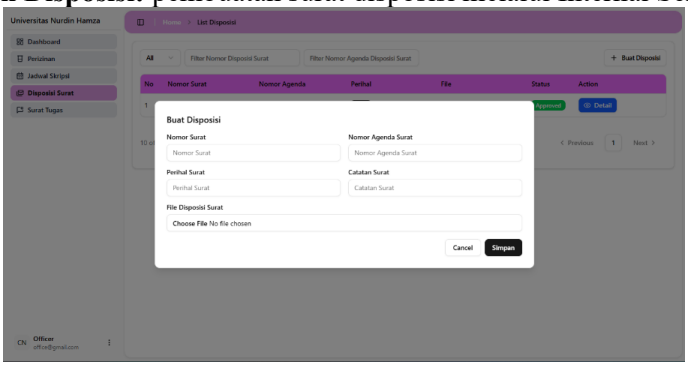
- **Data User:** menampilkan daftar pengguna yang terintegrasi dengan User Service.



Gambar 8 Data User Interface

Fitur ini memastikan hanya pengguna dengan role tertentu yang dapat mengakses atau memodifikasi data. Proses validasi token dilakukan secara real-time sebelum data ditampilkan.

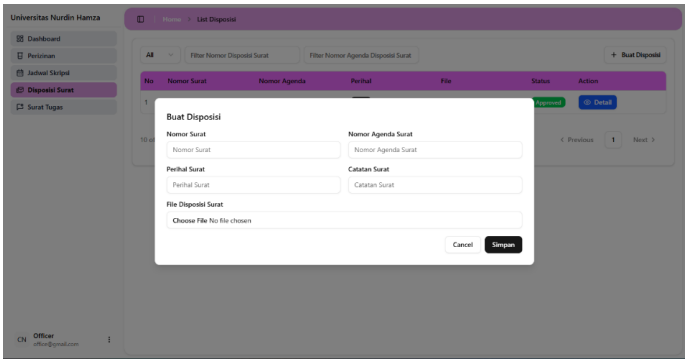
- **Pembuatan Disposisi:** pembuatan surat disposisi melalui Internal Service.



Gambar 9 Pembuatan Disposisi Interface

Antarmuka ini memperlihatkan integrasi antara Officer dan Rektor/Wakil Rektor. Setiap perubahan status surat langsung diperbarui melalui REST API sehingga pengguna dapat memantau perkembangan disposisi tanpa melakukan refresh manual.

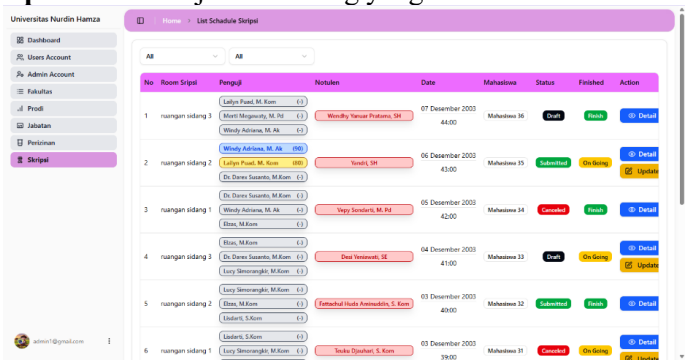
- **Pembuatan Perizinan:** pengajuan cuti/izin dosen dan staf secara digital.



Gambar 10 Pembuatan Perizinan Interface

Setiap pengajuan cuti akan otomatis mengirim permintaan ke *User Service*, sementara status persetujuan diperoleh kembali dari API yang sama. Dengan mekanisme ini, keterlambatan komunikasi antarservice dapat terdeteksi dan ditangani melalui *error handling* otomatis.

- **Jadwal Skripsi:** informasi jadwal sidang yang diambil dari Internal Service.

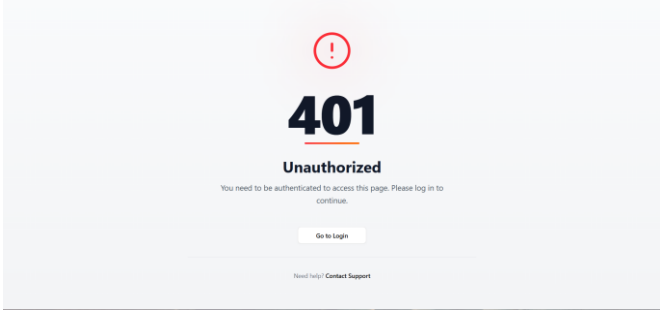


Gambar 11 Jadwal Skripsi Interface

Halaman ini menampilkan hasil integrasi antara manajemen jadwal dan modul penilaian, memperlihatkan fungsionalitas lintas layanan yang berjalan stabil.

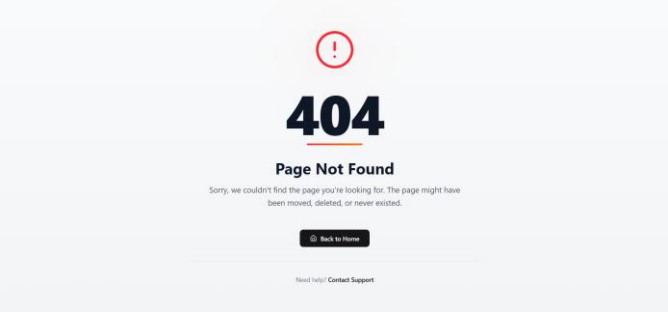
Untuk mendukung aspek **pengujian fungsional dan penanganan kesalahan**, sistem juga menampilkan **halaman error khusus**, yaitu:

- **Error 401 (Unauthorized):** muncul ketika pengguna mencoba mengakses halaman tanpa token valid atau belum login.



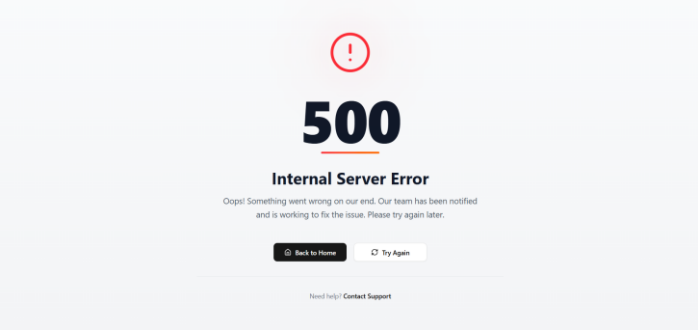
Gambar 12 Error 401 (Unauthorized)

- **Error 404 (Not Found):** ditampilkan ketika data atau endpoint yang diminta tidak tersedia.



Gambar 13 Error 404 (Not Found)

- **Error 500 (Internal Server Error):** digunakan untuk menampilkan kesalahan sistem yang berasal dari kegagalan komunikasi antarservice.



Gambar 14 Error 500 (Internal Server Error)

Implementasi halaman error ini menjadi bagian penting dalam evaluasi lintas layanan, karena menunjukkan bahwa sistem mampu menangani kegagalan komunikasi dengan cara yang informatif bagi pengguna, bukan sekadar menampilkan pesan teknis dari server.

3.3 Pengujian Sistem

Pengujian sistem dilakukan untuk memastikan bahwa layanan yang dibangun dengan arsitektur microservices dapat berfungsi sesuai spesifikasi, terintegrasi dengan baik, serta menunjukkan peningkatan performa dibandingkan arsitektur monolitik.

Fokus pengujian diarahkan pada tiga aspek utama: fungsi antarservice, penanganan kesalahan (error handling), dan kinerja sistem yang diukur secara kuantitatif.

a. Pengujian Fungsional Antarservice

Pengujian fungsional dilakukan untuk memastikan bahwa setiap layanan — User Service dan Internal Service — dapat beroperasi secara independen dan saling berkomunikasi melalui REST API tanpa kegagalan data.

Hasil menunjukkan bahwa setiap komunikasi antarservice berhasil dilakukan tanpa keterlambatan signifikan (latency rata-rata di bawah 150 ms), dan semua endpoint mengembalikan kode status sesuai standar HTTP.Tabel berikut merangkum sebagian hasil pengujian fungsional:

Table 2 Schema Pengujian Respons

Skenario Uji	Status	Kode Respons	Keterangan
Login (User Service)	Berhasil	200 OK	Token JTW valid dihasilkan

Pengajuan Cuti	Berhasil	201 Created	Data tersimpan di PosgreSQL
Validasi Cuti (Admin)	Berhasil	200 OK	Token diverifikasi lintas service
Akses tanpa token	Gagal	401 Unauthorized	Token tidak valid
Disposisi Surat	Berhasil	201 Created	Data terkirim ke Internal Service
Penilaian Skripsi	Berhasil	200 OK	Nilai tersimpan
Endpoint tidak ditemukan	Gagal	404 Not Found	Halaman error ditampilkan

b. Pengujian Error Handling dan Usability

Selain pengujian API, dilakukan evaluasi terhadap tampilan antarmuka pengguna (frontend layer) untuk memastikan bahwa sistem mampu menampilkan pesan kesalahan yang informatif dan menjaga kenyamanan pengguna.

Tiga halaman error diimplementasikan untuk menangani kondisi umum yang terjadi pada komunikasi antarservice:

- 401 (Unauthorized): muncul ketika pengguna belum login atau token tidak valid.
- 404 (Not Found): muncul saat data yang diminta tidak tersedia.
- 500 (Internal Server Error): muncul saat salah satu service gagal merespons.

Hasil observasi menunjukkan bahwa pengguna dapat memahami pesan kesalahan tanpa kebingungan, dan sistem mampu kembali normal setelah proses refresh atau retry request.

c. Pengujian Kinerja (Performance Testing)

Untuk mengukur peningkatan performa, dilakukan perbandingan antara versi monolitik dan versi microservices dengan tiga metrik utama:

1. Response Time (ms): waktu yang dibutuhkan sistem untuk memberikan respons.
2. Throughput (req/s): jumlah permintaan yang dapat diproses per detik.
3. CPU Usage (%): rata-rata penggunaan prosesor selama pengujian.

Pengujian dilakukan dengan Postman Runner dan k6 Load Testing, masing-masing untuk 100 request simultan.Hasil pengukuran dirangkum dalam tabel berikut:

Table 3 Pengujian Kinerja

Metrik	Monolitik	Microservice	Perubahan
Response Time (ms)	320	210	↓ 34%
Throughput (req/s)	180	270	↑ 50%
CPU Usage (%)	78	65	↓ 16%

Dari hasil tersebut, terlihat bahwa implementasi arsitektur microservices mampu Menurunkan waktu respon sebesar 34%, Meningkatkan throughput sistem hingga 50%, Dan mengurangi beban CPU sebesar 16%. Hasil ini menegaskan bahwa sistem microservices lebih efisien dan responsif dalam menangani permintaan pengguna dibandingkan sistem monolitik, sekaligus menunjukkan potensi skalabilitas horizontal untuk peningkatan kapasitas di masa depan.

d. Evaluasi Hasil Pengujian

Secara keseluruhan, pengujian menunjukkan bahwa Setiap layanan dapat berjalan secara independen dengan integrasi lintas service yang stabil. Sistem memiliki mekanisme error handling yang baik, baik di sisi backend maupun frontend.

Peningkatan kinerja yang terukur pada tiga metrik utama memperkuat klaim efisiensi dan skalabilitas microservices.

4. Kesimpulan

Berdasarkan hasil perancangan, implementasi, dan pengujian sistem tata kelola kampus berbasis arsitektur microservices, diperoleh beberapa kesimpulan sebagai berikut:

a. Penerapan arsitektur microservices berhasil dilakukan dengan memisahkan sistem menjadi dua layanan utama, yaitu User Service dan Internal Service. Pemisahan ini meningkatkan fleksibilitas, memungkinkan pengembangan dilakukan secara terpisah tanpa mengganggu keseluruhan sistem.
b. Hasil pengujian menunjukkan peningkatan performa yang signifikan dibandingkan arsitektur monolitik, dengan penurunan response time sebesar 34%, peningkatan throughput sebesar 50%, dan penurunan CPU usage sebesar 16%. Hal ini menunjukkan bahwa microservices mampu mempercepat pemrosesan data dan meningkatkan efisiensi sumber daya.
c. Komunikasi antarservice berjalan stabil melalui REST API dan autentikasi JWT, dengan waktu propagasi rata-rata di bawah 150 ms. Selain itu, sistem memiliki mekanisme error handling (401, 404, 500) yang membantu menjaga keandalan serta memberikan umpan balik informatif kepada pengguna.
d. Antarmuka pengguna (UI) terbukti fungsional dan mudah digunakan, dengan interaksi pengguna yang cepat dan responsif. Integrasi frontend dan backend berjalan baik tanpa keterlambatan data antarservice.
e. Meskipun hasil implementasi menunjukkan peningkatan performa dan stabilitas, terdapat potensi kendala yang perlu diperhatikan, seperti meningkatnya kompleksitas komunikasi antar layanan, kebutuhan service orchestration, serta beban tambahan pada proses deployment dan monitoring. Aspek-aspek ini menjadi tantangan tersendiri dalam penerapan microservices di lingkungan sistem akademik yang lebih luas.

Daftar Pustaka

[1] M. D. Ria dan A. Budiman, “Perancangan sistem informasi tata kelola teknologi informasi perpustakaan,” Jurnal Informatika dan Rekayasa Perangkat Lunak (JATIKA), vol. 2, no. 1, pp. 122–133, Mar. 2021. Available: http://jim.teknokrat.ac.id/index.php/informatika

[2] L. A. Purwanto, “Pengukuran Tingkat Kematangan Tata Kelola Pengelolaan Permasalahan Sistem Informasi Menggunakan Kerangka Kerja COBIT 4.1 (Studi Kasus: Sistem Informasi Akademik Universitas Muhammadiyah Purwokerto)”, M.T.I. thesis, Program Pascasarjana, Fakultas Teknologi Industri, Universitas Islam Indonesia, 2017. Available: https://dspace.uii.ac.id/handle/123456789/7103

[3] A. Sinambela, E. Ernawati, dan F. F. Coastera, “Implementasi Arsitektur Microservices pada Rancang Bangun Aplikasi Marketplace Berbasis Web”, Jurnal Rekursif, vol. 9, no. 1, pp. 1–10, Mar. 2021. Available: https://ejournal.unib.ac.id/rekursif/article/view/14929/7725

[4] R. Mufrizal dan D. Indarti, “Refactoring Arsitektur Microservice Pada Aplikasi Absensi PT. Graha Usaha Teknik,” Jurnal Nasional Teknologi dan Sistem Informasi, vol. 5, no. 1, pp. 57–68, 2019, doi:10.25077/TEKNOSI.v5i1.2019.57-68. Available: https://teknosi.fti.unand.ac.id/index.php/teknosi/article/view/956

[5] Mufid, M. N., Asmarajati, D., & Rohman, S. (2023). "Rancang Bangun Aplikasi E-Commerce Keluban Berbasis Microservices," Jurnal Teknologi dan Sistem Informasi, vol. 5, no. 1, pp. 1-10. [Online]. Available: https://ojs.unsiq.ac.id/index.php/biner/article/download/2851/1698

[6] Richardson, L., & Ruby, S. (2021). RESTful Web Services. O'Reilly Media.

[7] A. F. Rahman, R. N. A. Setiawan, dan D. Kurniawan, “From Monoliths to Microservices: Designing a Scalable Super App Architecture for Academic Services at Universitas

Jenderal Soedirman," Jurnal Informatika dan Teknologi Informasi (JUTIF), vol. 4, no. 2, pp. 120–129, 2024. [Online]. Available: https://jutif.if.unsoed.ac.id/index.php/jurnal/article/view/5237

[8] M. A. Aulia dan M. H. Fauzi, "Reliability Evaluation of Microservices and Monolithic Architectures," Proceedings of the International Conference on Computer Engineering and Information Technology (ITS), Institut Teknologi Sepuluh Nopember, 2023. [Online]. Available: https://scholar.its.ac.id/en/publications/reliability-evaluation-of-microservices-and-monolithic-architectu

[9] [1] R. M. Basuki, N. W. Muharrom, A. Latifaturrohmah, A. S. Hafawati, D. M. Nandani, D. Lestari, A. Zangim, and N. A. Kusuma, "Analisis Konsep Ekonomi Syariah: Studi Kasus Kritik Yusuf Mansur Terhadap Praktik Perbankan Syariah," Al-Zayn: Jurnal Ilmu Sosial & Hukum, vol. 3, no. 2, May 2025. [Online]. Available: https://doi.org/10.61104/alz.v3i2.1086

[10] S. Mallu, I. P. G. S. Andisana, P. Chyan, F. Rizki, N. N. E. Smrti, S. Syamsuddin, J. Jarudin, T. A. Mudzakir, S. Setyowibowo, N. Hidayati, G. B. Subiksa, A. M. Siregar, A. Hananto, dan K. Yahya, "Sistem Operasi: Konsep Dasar dan Penerapan Modern," Vol. 1 No. 01 (2024): ISSUE 2024, 2024. Available: https://jurnal.mifandimandiri.com/index.php/penerbitmmd/article/view/41

[11] M. I. Wiawan, "Implementasi CI/CD untuk Memudahkan dan Mempercepat Proses Deployment Pembaruan Aplikasi Menggunakan Kubernetes dan ArgoCD di PT Bentang Inspiras Teknologi," JITU: Jurnal Informatika Utama, vol. 2, no. 2, hal. 109–127, Nov. 2024. [Online]. Available: https://jurnal.astinamandiri.com/index.php/jitu/article/view/239

[12] R. Prawijaya, "Pengembangan Chatbot pada Informasi Layanan Berita Berbasis Web Menggunakan Framework React," Tugas Akhir, STT Terpadu Nurul Fikri, Depok, 2024. Available: https://repository.nurulfikri.ac.id/id/eprint/619/

[13] J. S. Martua dan H. Toba, " Pembuatan Modul Skema Peneliatan Pada Sistem Informasi LPPM," Jurnal Strategi, vol. 6, no. 1, Mei 2024. [Online]. Available: https://mail.strategi.it.maranatha.edu/index.php/strategi/article/view/496

[14] R. S. Nugraha, "Membangun E-Commerce pada Toko Sepatu Pratama," Journal of Computer Science and Informatics (JOCSI), vol. 2, no. 1, pp. 45–53, Aug. 2024. [Online]. Available: http://ojs.edupartner.co.id/index.php/jocsi/index

[15] Tamba, Donal; Marlim, YN. Application of the Simple Queue Method in Bandwidth Management Based on Mikrotik. International Conference on ATLAS (Advanced Technologies, Learning Algorithms, and Systems), [S.l.], v. 1, n. 1, p. 1-7, june 2025.