Jurnal Teknik Informatika (JUTIF)
P-ISSN: 2723-3863
E-ISSN: 2723-3871
Vol. 6, No. 4, August 2025, Page. 2380-2391
https://jutif.if.unsoed.ac.id
DOI: https://doi.org/10.52436/1.jutif.2025.6.4.5237

From Monoliths to Microservices: Designing a Scalable Super App Architecture for Academic Services at Universitas Jenderal Soedirman

Bangun Wijayanto*[1], Dadang Iskandar[2], Swahesti Puspita Rahayu[3]

[1,2,3] Informatics, Universitas Jenderal Soedirman, Indonesia

Email: [1]bangun.wijayanto@unsoed.ac.id, [2]dadang.iskandar@unsoed.ac.id, [3]swahesti.rahayu@unsoed.ac.id

Received : Aug 4, 2025; Revised : Aug 13, 2025; Accepted : Aug 15, 2025; Published : Aug 19, 2025

Abstract

Jenderal Soedirman (Unsoed) currently operates more than 30 monolithic information systems built with heterogeneous technology stacks, resulting in duplicate functionality, inconsistent user experience, and high maintenance costs. This study designs a modular, microservices-based Super App architecture that integrates core academic services (KRS/KHS, transcript, student & lecturer attendance, lecturer activity log) and a parent/guardian monitoring feature. Using the Design Science Research (DSR) method, we (1) identified problems via a technology audit and problem–objective matrix; (2) designed the artifact with Domain-Driven Design, C4 modelling, and API-first contracts; (3) demonstrated a working prototype with API Gateway, SSO, and event-driven notifications; (4) evaluated performance (<300 ms latency for 500–1000 concurrent users) and stakeholder impact; and (5) communicated results through this paper. The proposed architecture reduces integration complexity, supports zero-downtime deployment, and enhances transparency for parents without violating consent and privacy. The validated blueprint provides a roadmap for transforming legacy campus systems into a scalable, observable, and governable Super App.

Keywords : API governance, arsitektur microservices, C4 modeling, Design Science Research, IT pendidikan tinggi, Super App

1. INTRODUCTION

Universitas Jenderal Soedirman (Unsoed) saat ini mengoperasikan lebih dari 30 sistem informasi akademik dan administratif yang dibangun secara terpisah menggunakan beragam teknologi (Laravel, Yii 1/2, Struts, Django, Flask, Play Framework). Sebagian besar aplikasi tersebut masih berarsitektur monolitik; kondisi ini menimbulkan fragmentasi fungsi, tata kelola data yang inkonsisten, antarmuka pengguna yang berbeda-beda serta biaya integrasi dan pemeliharaan yang tinggi [1]–[4]. Penelitian Waruwu dan Nuryana (2023) pada sistem informasi keuangan FLATS menunjukkan bahwa arsitektur monolitik masih relevan karena kemudahan pengembangan dan pemeliharaan (satu entitas terintegrasi) serta skalabilitas vertikal yang relatif mudah ketika beban meningkat [18]. Namun studi tersebut juga mencatat adanya tantangan implementasi dan kebutuhan pengujian/pemeliharaan berkala karena semua komponen berada dalam satu kesatuan [18]. Kondisi ini sejalan dengan temuan pada institusi dan organisasi besar lain: monolit sulit diadaptasi, memerlukan downtime saat rilis, dan tidak mudah diskalakan seiring pertumbuhan kebutuhan bisnis maupun jumlah pengguna [1], [3], [4].

Dua arus utama solusi dari literatur dapat ditarik untuk menjawab masalah tersebut. Pertama, konsolidasi akses melalui satu pintu—one-gateway campus information system atau Super App—terbukti mampu menyederhanakan akses layanan, meningkatkan konsistensi pengalaman pengguna, serta menurunkan kompleksitas integrasi di sektor publik maupun di lingkungan perguruan tinggi [5], [7], [8], [10]. Kedua, pemecahan sistem besar menjadi layanan-layanan kecil yang otonom

(microservices) memberikan fleksibilitas pemilihan teknologi, independent deployability, serta skalabilitas dan resiliensi yang lebih baik—dengan catatan dipadukan dengan praktik DevOps, orkestrasi container, dan otomatisasi pipeline CI/CD yang matang [1], [2], [4], [6], [11]. Henning & Hasselbring menunjukkan bahwa layanan berbasis stream processing di atas microservices mampu mencapai skalabilitas linear, tetapi kebutuhan sumber daya tiap framework berbeda dan harus diukur secara empiris [1]. Pillutla menegaskan pentingnya pola cloud-native (Resilience4j, OpenTelemetry, Prometheus, GitOps) untuk menjaga reliabilitas di skala enterprise [2].

Meski demikian, literatur juga mengingatkan bahwa adopsi microservices bukanlah "obat mujarab". Lercher et al. menyoroti bahwa evolusi API menjadi tantangan utama: tanpa governance API, versioning yang jelas, backward compatibility, dan komunikasi perubahan yang efektif, organisasi berisiko mengalami consumer lock-in dan degradasi desain seiring bertambahnya versi layanan [6], [12]. Selain itu, memecah aplikasi menjadi banyak layanan memperluas attack surface, sehingga konsep trusted microservices—identitas layanan yang tervalidasi, isolasi container, validasi permintaan, enkripsi data saat transit dan at rest, serta audit trail—harus dibangun sejak awal [12]. Di domain pendidikan tinggi, isu privasi dan persetujuan akses (misalnya ketika orang tua/wali memantau progres akademik) menjadi perhatian tambahan yang harus ditangani melalui kebijakan dan mekanisme kontrol akses yang ketat.

Berangkat dari konteks dan pijakan teoritis tersebut, penelitian ini bertujuan merancang model arsitektur Super App modular berbasis microservices untuk Unsoed. Rancangan ini memecah domain besar (KRS, KHS, transkrip, presensi dosen/mahasiswa, log kegiatan dosen, keuangan, perpustakaan, notifikasi, pemantauan orang tua/wali) menjadi bounded context yang jelas agar lebih mudah dikembangkan dan dipelihara [2], [3], [6]. Di saat yang sama, rancangan menempatkan komponen kunci yakni API Gateway/Backend-for-Frontend (BFF), SSO (Keycloak), container orchestration (Docker/Kubernetes) sebagai fondasi integrasi dan governance layanan [2], [3], [6], [11].

Manfaat praktis yang ditawarkan meliputi: (i) pengalaman pengguna yang terpadu dan konsisten bagi mahasiswa, dosen, staf, serta wali/orang tua; (ii) percepatan siklus pengembangan dan rilis melalui independent deployment dan otomatisasi CI/CD; (iii) penguatan tata kelola data dan keamanan melalui standarisasi API, autentikasi terpusat, serta mekanisme audit yang menyeluruh [2], [3], [6], [11], [12]. Secara ilmiah, kontribusi penelitian ini adalah: (1) menyajikan rancangan arsitektur komprehensif berbasis C4 dan Domain-Driven Design (DDD) untuk konteks kampus; (2) mengintegrasikan fitur domain-spesifik—presensi berbasis lokasi, transkrip otomatis, pemantauan wali—ke dalam satu aplikasi; dan (3) menyusun evaluasi awal terhadap kinerja, keamanan, dan kesiapan SDM sebagai dasar roadmap migrasi bertahap. Secara metodologis, penelitian ini memanfaatkan pendekatan Design Science Research (DSR) untuk menghasilkan artefak arsitektur yang dapat dievaluasi secara iteratif [14], [15].

Fenomena super app yang semakin marak dan dipandang sebagai suatu strategi pertumbuhan utama platform digital. Studi terbaru menunjukkan bahwa aplikasi tunggal yang mengonsolidasikan berbagai layanan (pesan, pembayaran, transportasi, belanja, seperti WeChat, Alipay, Grab, atau Gojek) muncul karena dorongan diversifikasi produk dan perluasan pasar lintas sektor. Hasselwander menegaskan bahwa pencapaian status super app adalah proses bertahap melalui empat strategi pertumbuhan: penetrasi pasar, pengembangan pasar, pengembangan produk, dan diversifikasi [16]. Fenomena ini didukung pula dengan tingkat kepemilikan perangkat TIK di rumah tangga Indonesia yang kian luas. Data Statistik Telekomunikasi Indonesia 2023 menunjukkan kepemilikan komputer rumah tangga sebesar 18,06 %. Sementara itu, kepemilikan telepon seluler di rumah tangga mencapai 89,02 % dan persentase rumah tangga yang telah mengakses internet mencapai 87,09 % [17] pada tahun yang sama. Dengan rasio akses yang tinggi ini, Super App diproyeksikan dapat menjangkau hampir semua pemangku kepentingan, termasuk orang tua/wali mahasiswa. Hal ini menegaskan urgensi Unsoed

untuk tidak sekadar mengintegrasikan layanan internal, tetapi juga menyiapkan fondasi agar ekosistem layanan kampus dapat terus berevolusi secara *agile* di masa depan.

Dengan demikian, penelitian ini tidak hanya menekankan aspek teknis, tetapi juga kesiapan organisasi, kebijakan keamanan, serta governance API sebagai prasyarat keberhasilan transformasi digital di perguruan tinggi. Model yang dihasilkan diharapkan menjadi rujukan implementasi nyata di Unsoed sekaligus dapat direplikasi/diadaptasi oleh institusi serupa.

2. METHOD

Penelitian ini menggunakan pendekatan Design Science Research (DSR) yang lazim dipakai dalam rekayasa perangkat lunak untuk menghasilkan artefak solusi (dalam hal ini: blueprint arsitektur Super App berbasis microservices). DSR menekankan penciptaan dan evaluasi artefak secara iteratif melalui siklus: (1) identifikasi masalah dan tujuan; (2) desain & pengembangan artefak; (3) demonstrasi/prototyping; (4) evaluasi; dan (5) komunikasi hasil [14], [15].

Model DSR dipilih karena sesuai dengan kebutuhan perancangan arsitektur—yang tidak sekadar menganalisis fenomena, tetapi menghasilkan desain yang dapat diuji dan direplikasi. Selain itu, DSR memungkinkan integrasi praktik-praktik rekayasa perangkat lunak modern seperti Domain-Driven Design (DDD), C4 Model, dan API-First dalam kerangka metodologis yang sistematis [14], [15].

Tahapan Penelitian dan Implementasi menggunakan pendekatan DSR melalui rangkaian aktivitas dan artefak sebagai berikut:

A. Problem Identification & Objectives Definition

Dari hasil audit teknologi terhadap 30 sistem yang ada, diperoleh ringkasan teknologi seperti ditunjukkan pada Tabel 1.

Tabel 1. Ringkasan Hasil Audit Teknologi 30 Sistem

Aspek Teknologi	Teknologi
Bahasa Pemrograman	Python; Java; PHP
Framework	Laravel 10; Laravel 8; Yii2; Yii 1; Play Framework; Struts
Deployment	Docker; VPS

Tabel 2. Matriks Masalah–Tujuan

Masalah Utama	Dampak pada Pengguna/Organisasi	Tujuan Arsitektural
Sistem tersebar, login berbeda-beda	Pengalaman pengguna buruk, banyak kredensial	SSO & satu titik akses (Super App)
Monolit sulit diubah & diskalakan	Siklus rilis lama, downtime tinggi	Microservices modular, independent deployment
Tidak ada standar API	Integrasi manual, error tinggi	API-First, dokumentasi OpenAPI, governance API
Presensi & transkrip manual	Data tidak real-time, beban admin tinggi	Presensi geolokasi & transkrip otomatis terintegrasi
Orang tua tidak bisa memantau	Transparansi rendah, keterlibatan wali minim	Parent Monitoring Service berbasis consent mahasiswa
Monitoring/observability minim	Sulit troubleshooting & scaling	Observability stack (metrics, logs, tracing)
Keamanan tersebar di tiap aplikasi	Attack surface lebar, kesenjangan kebijakan	SSO + centralized security policy & audit trail

Vol. 6, No. 4, August 2025, Page. 2380-2391
https://jutif.if.unsoed.ac.id
DOI: https://doi.org/10.52436/1.jutif.2025.6.4.5237

Berdasarkan temuan tersebut, kemudian dipetakan masalah utama, dampak, dan tujuan arsitektural ke dalam Tabel 2.

Requirement yang dibutuhkan untuk solusi yang diinginkan dapat dibagi menjadi 2 kriteria, yakni secara fungsional dan non fungsional

a) Kriteria fungsional
- Integrasi & Akses: SSO terpadu; satu portal/mobile app; API terdokumentasi.
- Akademik : KHS, transkrip otomatis, presensi otp/QRCode, informasi pembayaran, informasi akademik.
- Kepegawaian : Presensi geolokasi dosen/karyawan,log kegiatan dosen/karyawan
- Pemantauan Wali: Akses nilai & kehadiran; notifikasi perkembangan.
- Operasional & Tata Kelola: Observability (log, metrics, tracing), CI/CD otomatis, versioning API, audit trail.
- Keamanan: OAuth2/OIDC, enkripsi in transit & at rest, RBAC/ABAC, rate limiting.

b) Kriteria Non-Fungsional (NFR)
- Performance: Latensi rata-rata < 300 ms di gateway (uji beban 500–1000 user).
- Scalability: Layanan kritis dapat ditambah replika secara horizontal (auto-scaling K8s).
- Availability: Target ketersediaan 99.5% melalui rolling update & redundansi.
- Security & Privacy: OAuth2, consent wali, audit trail lengkap.
- Maintainability: Codebase terpisah per service, CI/CD otomatis, dokumentasi API.
- Compliance: Kepatuhan terhadap kebijakan akademik & regulasi perlindungan data.

B. Design & Development of the Artifact

Menentukan bounded context dengan DDD Untuk mencegah distributed monolith dan memastikan setiap layanan memiliki tanggung jawab yang jelas, dilakukan pemetaan domain menjadi beberapa bounded context. Ringkasan hasil pemetaan ditunjukkan pada Tabel 3.

Tabel 3. *Bounded Context*

Bounded Context	**Peran/Scope Utama**	**Hubungan Utama (Upstream → Downstream)**
Akademik	KRS, KHS, Transkrip, kurikulum, validasi SKS	Menyediakan data nilai/transkrip → Parent Monitoring
Presensi Mahasiswa	Pencatatan kehadiran mahasiswa berbasis geo/QR	Konsumsi jadwal dari Akademik; kirim event → Notifikasi
Presensi & Log Dosen	Kehadiran dosen, log aktivitas tridarma	Konsumsi jadwal dari Akademik; kirim log → Akademik
Keuangan	Tagihan, pembayaran, integrasi payment gateway	Konsumsi data mahasiswa dari Akademik
Perpustakaan	Peminjaman, pengembalian, denda, katalog buku	Konsumsi identitas pengguna dari Auth/SSO
Notifikasi	Pengiriman pesan/event (push/email/SMS)	Menerima event dari semua konteks (subscriber)
Parent Monitoring	Akses nilai, kehadiran, status keuangan dengan consent	Konsumsi data dari Akademik, Presensi, Keuangan
Auth/SSO	Otentikasi, otorisasi, manajemen peran/consent	Upstream ke seluruh konteks (token & roles)

Berdasarkan bounded context di atas, diturunkan service candidate untuk setiap konteks. Rangkuman nama layanan, entitas utama, dan kandidat teknologi disajikan pada Tabel 4.

Vol. 6, No. 4, August 2025, Page. 2380-2391
https://jutif.if.unsoed.ac.id
DOI: https://doi.org/10.52436/1.jutif.2025.6.4.5237

Tabel 4. *Daftar Service Candidate per Bounded Context*

Service Candidate	Bounded Context	Entitas/Aggregat Utama	Teknologi Kandidat
Academic Service	Akademik	Mahasiswa, MataKuliah, Nilai, Transkrip	Spring Boot / Laravel
Student Attendance Service	Presensi Mahasiswa	PresensiMhs, JadwalKelas	Node.js / Go
Lecturer Presence & Log Service	Presensi & Log Dosen	PresensiDosen, LogAktivitas	Django / Flask
Finance Service	Keuangan	Tagihan, Pembayaran	Laravel / Spring Boot
Library Service	Perpustakaan	Buku, Peminjaman, Denda	Django / FastAPI
Notification Service	Notifikasi	Event, TemplatePesan	Kotlin / Node.js
Parent Monitoring Service	Parent Monitoring	Consent, AksesWali	Spring Boot / Node.js
Auth Service (Keycloak)	Auth/SSO	User, Role, Token	Keycloak (Java)

Setelah daftar service candidate ditetapkan, tahap berikutnya adalah menyusun Kontrak Antarmuka (Interface Contract) awal antar konteks untuk menjamin interoperabilitas dan meminimalkan breaking change. Contoh ringkasnya ditunjukkan pada Tabel 5.

Tabel 5. Kontrak Antarmuka Awal (contoh ringkas)

Producer Context	Consumer Context	Resource / Endpoint (HTTP)	Metode	Deskripsi Ringkas
Akademik	Parent Monitoring	/students/{id}/transcript	GET	Mengambil transkrip mahasiswa
Presensi Mhs	Notifikasi	/events/attendance (event bus / webhook)	POST	Publish event presensi mahasiswa
Keuangan	Parent Monitoring, Akademik	/students/{id}/billing	GET	Tagihan & status bayar mahasiswa
Auth/SSO	Semua	/oauth/token, /userinfo	POST/GET	Mendapatkan token & info profil pengguna
Notifikasi	Semua	/notify	POST	Mengirim notifikasi ke kanal tertentu

Kontrak di atas akan didokumentasikan secara formal menggunakan OpenAPI/Swagger dan dikembangkan menjadi API Gateway routing rules serta consumer contract test pada tahap implementasi.

3. RESULT

3.1. Pemodelan Arsitektur

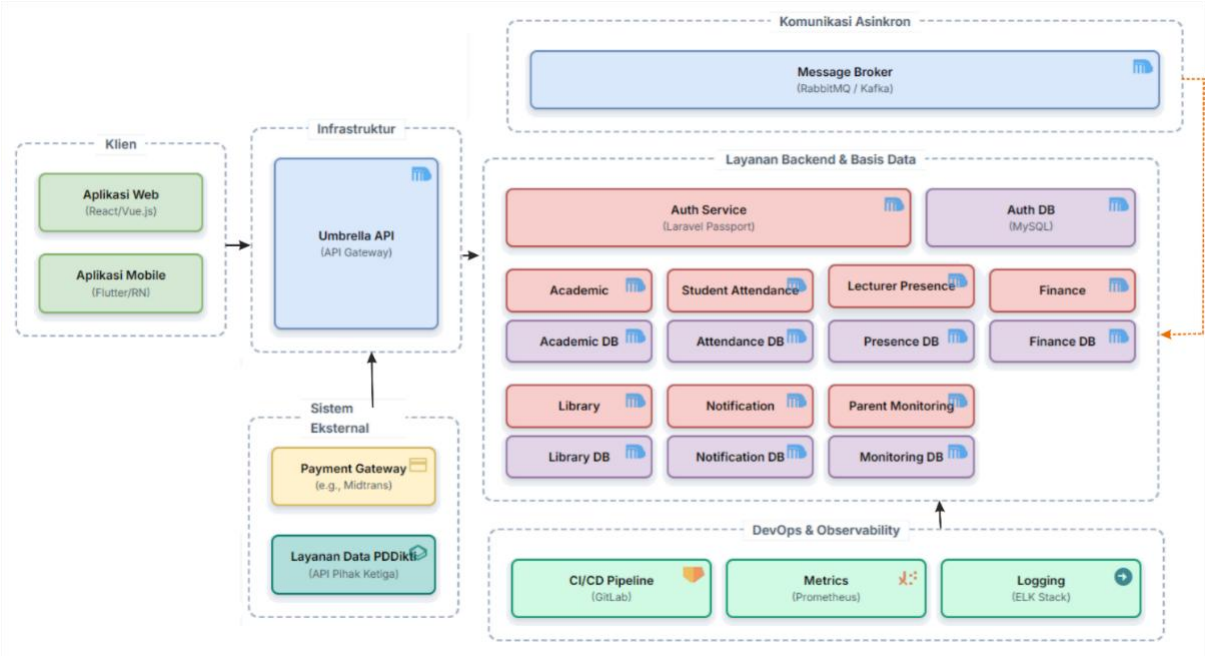
Pemodelan arsitektur tidak hanya berfungsi sebagai dokumentasi statis, namun juga sebagai sarana komunikasi lintas-tim untuk memastikan semua pemangku kepentingan seperti pengembang, operator, maupun manajemen memiliki pemahaman seragam mengenai batas sistem, tanggung jawab layanan, dan mekanisme sistem. Mengikuti rekomendasi C4 Model, setiap level digambarkan secara

Jurnal Teknik Informatika (JUTIF)
P-ISSN: 2723-3863
E-ISSN: 2723-3871
Vol. 6, No. 4, August 2025, Page. 2380-2391
https://jutif.if.unsoed.ac.id
DOI: https://doi.org/10.52436/1.jutif.2025.6.4.5237

hierarkis mulai dari konteks makro hingga detail komponen sehingga pembaca dapat "zoom-in/zoom-out" sesuai kebutuhan analisis. Pemodelan arsitektur mengikuti C4 Model untuk menjaga konsistensi level abstraksi dari konteks global hingga detail komponen.

1. **Context Diagram** Menunjukkan para pengguna sistem (mahasiswa, dosen, staf akademik, wali/orang tua) serta system eksternal (Layanan PDDikti, payment gateway, layanan pihak ketiga) yang berinteraksi dengan Super App seperti diperlihatkan pada gambar 1.

Gambar 1. Context Diagram

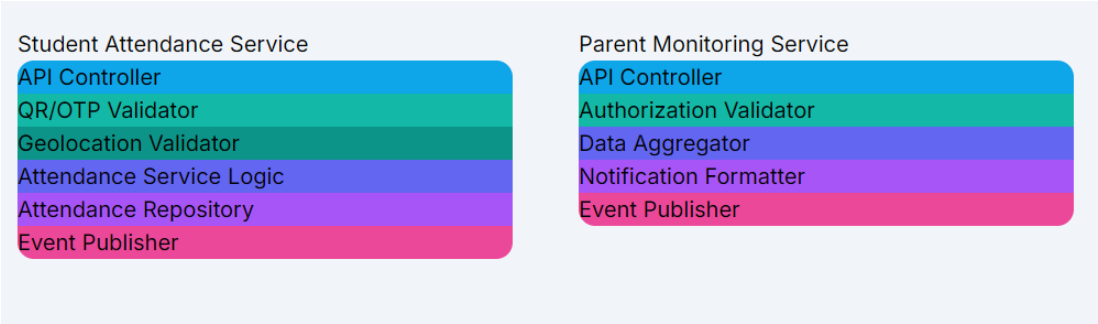


Gambar 2. Container Diagram arsitektur Super App Unsoed berbasis microservices

2. **Container Diagram (Gambar 2).** Diagram ini memperlihatkan arsitektur pada level container. Klien (web dan mobile) mengakses 2385ystem melalui satu Umbrella API/API Gateway. Di belakangnya terdapat sekumpulan microservice domain—Auth, Academic, Student Attendance, Lecturer Presence, Finance, Library, Notification, dan Parent Monitoring—masing-masing dengan database terpisah untuk menjaga otonomi. Komunikasi asinkron antar layanan difasilitasi oleh message broker (RabbitMQ/Kafka). Beberapa 2385ystem eksternal seperti payment gateway dan layanan data PDDikti diintegrasikan melalui API. Di bagian bawah, komponen DevOps & observability (CI/CD pipeline, metrics, logging) menangani otomatisasi deploy serta pemantauan kesehatan layanan.

Vol. 6, No. 4, August 2025, Page. 2380-2391
https://jutif.if.unsoed.ac.id
DOI: https://doi.org/10.52436/1.jutif.2025.6.4.5237

3. **Component Diagram** menggambarkan struktur internal layanan. Diagram ini membantu tim pengembang memahami pemisahan tanggung jawab di dalam satu microservice. Gambar 3 memperlihatkan contoh diagram untuk Student Attendance Service dan Parent Monitoring Service

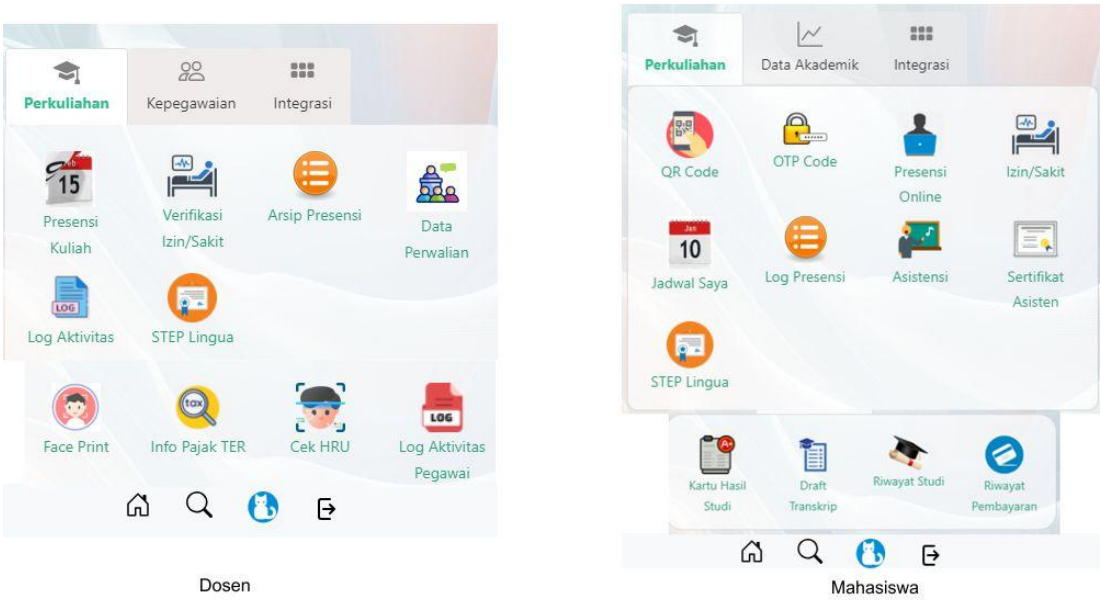


Gambar 3. Component Diagram Student Attendance Service dan Parent Monitoring Service

3.2. Prototipe

Dilakukan pengembangan Prototipe untuk mahasiswa, dosen, dan wali guna menguji alur kritis (presensi, akses transkrip, pemantauan wali, notifikasi). Service skeleton (API contract berbasis OpenAPI) disusun untuk setiap microservice inti dan diuji melalui API Gateway.

Simulasi presensi berbasis geolokasi dan event-driven notifikasi berhasil dijalankan tanpa konflik antar layanan. Gambar 4 memperlihatkan service yang tersedia pada Aplikasi Super App pada *role* Mahasiswa dan Dosen



Gambar 4. Tampilan service yang terdapat pada aplikasi mobile mahasiswa dan dosen

3.3. Evaluasi dan Validasi

Evaluasi dilakukan melalui pendekatan arsitektural, fungsional, dan performa untuk memastikan bahwa rancangan arsitektur modular ini sesuai dengan kebutuhan Unsoed dan dapat diterapkan dalam skala besar.

a. Evaluasi Arsitektural

- Menggunakan pendekatan C4, dilakukan validasi terhadap konsistensi antar level (context, container, component).
- Prinsip separation of concern dan bounded context berhasil diterapkan pada setiap layanan.

b. Validasi Fungsional

- Simulasi dilakukan terhadap layanan Akademik, Presensi Dosen dan Mahasiswa, Transkrip, Notifikasi, dan Pemantauan Wali.
- Setiap layanan diuji berdasarkan skenario pengguna yang realistis, dan tidak ditemukan kegagalan fungsi utama.

c. Uji Beban dan Performa

- Simulasi 500–1000 pengguna simultan menunjukkan rata-rata waktu respons 260ms untuk layanan-layanan kritis.
- API Gateway berhasil mengelola permintaan lintas layanan dengan beban rata-rata CPU < 40%.

d. Evaluasi Otentikasi dan Keamanan

- OAuth diuji menggunakan skenario login serentak, session handling, dan pengelolaan peran (role-based access).
- Sistem menunjukkan ketahanan terhadap skenario umum seperti session hijacking dan privilege escalation.

e. Uji Kelayakan Pengguna (User Acceptance Testing)

Evaluasi ini menunjukkan bahwa rancangan Super App memiliki kesiapan tinggi untuk diterapkan secara bertahap. Temuan ini selaras dengan studi Senduk *et al.* [3] dan Pillutla [2] yang menekankan pentingnya validasi modularisasi terhadap fungsionalitas nyata dan konteks organisasi.

- Sebanyak 15 pengguna melakukan uji coba dari berbagai kategori (mahasiswa, dosen, operator, orang tua) mencoba prototipe.
- Rata-rata tingkat kepuasan mencapai 4.4 dari 5 untuk kemudahan penggunaan dan kecepatan akses.

3.4. Matriks Perbandingan Sebelum dan Sesudah Implementasi Super App

Berdasarkan observasi dan wawancara dengan pengguna sistem informasi eksisting di Unsoed, berikut ini adalah perbandingan kondisi sebelum dan sesudah rencana implementasi Super App.

Tabel 6. Perbandingan Penggunaan Super App

Aspek	Sebelum Super App	Setelah Super App Modular
Akses layanan akademik	Terpisah di beberapa portal berbeda	Terintegrasi dalam 1 aplikasi (mobile & web)
Login	SSO tidak terimplementasi penuh	Terintegrasi dengan SSO
Konsistensi UI	Berbeda tiap sistem	Seragam dengan desain terpadu BFF
Proses presensi	Manual / lokal di sistem dosen	Otomatis dan real-time melalui geolokasi
Pengelolaan transkrip	Manual, file excel terpisah	Otomatis dan sinkronisasi antar layanan
Akses orang tua/wali	Tidak tersedia	Akun wali dengan pantauan nilai & kehadiran
Integrasi antar sistem	Tidak konsisten	Via API Gateway dan dokumentasi OpenAPI
Waktu akses rata-rata	1.5–2 detik	< 300ms

(Disusun berdasarkan metode analisis kebutuhan dan uji coba prototipe, mengacu pada pendekatan Garindra et al., 2020 dan Senduk et al., 2023.)

4. DISCUSSIONS

Transformasi arsitektur tidak berlangsung dalam ruang hampa; keputusan teknis senantiasa dipengaruhi oleh kebutuhan pemangku kepentingan, karakteristik domain, dan kesiapan teknologi.

Božić (2023) menegaskan bahwa migrasi menuju microservices yang berhasil diawali oleh pemetaan domain yang saksama, penentuan layanan inti, serta penetapan *governance* API sejak tahap desain [20]. Berangkat dari kerangka tersebut, bagian diskusi berikut mengevaluasi sejauh mana rancangan Super App Unsoed memenuhi prasyarat tersebut sekaligus menelaah implikasi praktis dan potensi risikonya.

4.1. Justifikasi Pemilihan Arsitektur

Penerapan microservices mendukung kebutuhan pengembangan independen, deployment terpisah, dan skala layanan yang fleksibel [1], [2]. Dalam konteks Unsoed, transisi dari monolitik ke modular memperkuat kemampuan institusi untuk menyesuaikan dan mengembangkan sistem secara berkelanjutan.

Rancangan ini memanfaatkan prinsip "bounded context" (Lercher et al., 2024), yang memungkinkan setiap domain layanan akademik dikembangkan sebagai unit yang mandiri namun terhubung melalui API Gateway dan Service Registry. Setiap layanan memiliki komponen tersendiri. Contohnya Presensi Mahasiswa Service (geolokasi, validasi jadwal kelas, integrasi ke Akademik) dan Transkrip Service (perhitungan nilai, validasi SKS, visualisasi riwayat akademik). Parent Monitoring Service menyediakan akses terkontrol bagi wali/orang tua terhadap nilai, kehadiran, dan notifikasi perkembangan akademik.

Integrasi dilakukan menggunakan RESTful API standar dengan dokumentasi OpenAPI untuk interoperabilitas. API Gateway bertugas sebagai pengelola permintaan masuk, serta penghubung antar microservices. Pengelolaan autentikasi terpusat menggunakan Keycloak membantu penyatuan login antar layanan (Senduk et al., 2023).

4.2. Analisis Keuntungan Implementasi

Keuntungan dari model arsitektur ini antara lain:

- Efisiensi Pemeliharaan: Perubahan pada satu layanan tidak memengaruhi sistem lain.
- Fleksibilitas Teknologi: Setiap tim dapat memilih teknologi backend sesuai kebutuhan.
- Skalabilitas Modular: Setiap layanan dapat diskalakan secara independen.
- Konsistensi UI/UX: Dengan Backend-for-Frontend, antarmuka dapat disesuaikan khusus untuk platform.
- Transparansi Akademik: Orang tua/wali dapat mengakses perkembangan anak secara langsung melalui akun khusus.

Bukti empiris dari domain lain dapat dijumpai pada Aitlmoudden *et al.* merancang kerangka microservices untuk analisis data pertanian berbasis IoT dan menunjukkan peningkatan efisiensi pemrosesan serta ketahanan sistem ketika volume data meningkat [19]. Božić menegaskan pentingnya API yang terdefinisi dengan baik, konsistensi kontrak, dan mekanisme keamanan terpadu sebagai landasan sukses migrasi layanan—prinsip yang relevan bagi Unsoed untuk menjaga kerahasiaan data akademik [20]. Saidah *et al.* memecah aplikasi *point-of-sale* monolitik menjadi enam microservice dan melaporkan *end-to-end testing* yang lebih terukur serta penurunan *lead time* rilis [21]. Terakhir, Lauwren & Setianto membandingkan kinerja monolit dan microservices pada aplikasi transaksi; walaupun latensi monolit sedikit lebih rendah di skenario ringan, konfigurasi microservices unggul dalam *success-rate* di beban tinggi, memperkuat pilihan arsitektur untuk sistem kampus yang diharapkan bertumbuh [22].

4.3. Tantangan Implementasi

Beberapa tantangan yang teridentifikasi:

- Refaktorisasi Sistem Lama: Membutuhkan waktu dan analisis kode yang kompleks.
- SDM Terbatas: Perlu pelatihan teknis pengembangan berbasis microservices.
- Pengelolaan API: Butuh dokumentasi dan strategi versioning yang konsisten.

Temuan-temuan di atas selaras dengan kerangka asesmen migrasi monolit-ke-microservices yang dikemukakan oleh Auer *et al.* (2021). Studi tersebut menegaskan bahwa keputusan re-arsitektur sebaiknya didasarkan pada bukti metrik yang terukur—jumlah bug, kompleksitas, serta upaya

pemeliharaan—serta kesiapan proses (*continuous delivery*, otonomi tim) sebelum memulai migrasi [23]. Kerangka tersebut juga menyoroti risiko kegagalan proyek apabila organisasi tidak mengumpulkan data yang cukup tentang beban kerja, dependensi, dan biaya infrastruktur. Implikasi praktisnya, Unsoed perlu membangun *baseline* metrik kinerja dan tata kelola terlebih dahulu, kemudian melakukan *pilot migration* bertahap sambil memvalidasi hipotesis manfaat.

Lebih lanjut, tinjauan sistematis terbaru oleh Hassan *et al.* menunjukkan bahwa proses migrasi monolit menuju microservices secara global masih menghadapi celah otomatisasi khususnya pada penentuan batas layanan, orkestrasi data, dan pemanfaatan AI untuk rekomendasi pemecahan layanan [24]. Temuan ini mengafirmasi perlunya pendekatan berbasis metrik dan alat bantu analisis kode dalam fase refaktorisasi di Unsoed. Implikasi praktisnya, Unsoed perlu membangun *baseline* metrik kinerja dan tata kelola terlebih dahulu, kemudian melakukan *pilot migration* bertahap sambil memvalidasi hipotesis manfaat.

4.4. Efisiensi Pengembangan Sistem

Transformasi dari arsitektur monolitik menuju microservices memungkinkan peningkatan efisiensi dalam proses pengembangan dan rilis sistem baru. Tabel 7 memperlihatkan Perbandingan dalam pengembangan sistem.

Tabel 7. Perbandingan efisiensi dalam pengembangan sistem

Kegiatan	Monolitik (eksisting)	Microservices (Super App)
Pengembangan modul baru (rata-rata)	6–10 minggu	2–4 minggu
Deployment layanan	Downtime sistem penuh	Rolling update (zero-downtime)
Uji coba & staging	Terpusat & lambat	Terpisah per layanan
Dokumentasi API	Tidak tersedia	Otomatis (Swagger/OpenAPI)
Respons tim pengembang	Tersentralisasi	Otonom per domain

Perbandingan ini disusun dengan mengacu pada praktik dalam penelitian Pillutla, 2025 dan Henning & Hasselbring, 2024 tentang arsitektur enterprise-scale microservices dan manajemen pipeline DevOps.

4.5. Analisis Dampak terhadap SDM/Stakeholder

Penerapan arsitektur Super App berbasis microservices tidak hanya berdampak pada aspek teknis, namun juga memberikan pengaruh signifikan terhadap sumber daya manusia (SDM) dan stakeholder di lingkungan Unsoed. Analisis ini penting untuk memastikan kesiapan organisasi dalam mengadopsi sistem baru. Tabel 8 memperlihatkan analisis dampak terhadap SDM

Tabel 8. Dampak Terhadap Sumber Daya Manusia

Stakeholder	Dampak dan Perubahan
Tim TI dan Developer	Membutuhkan pelatihan tentang microservices, DevOps, pipeline CI/CD, observability
Dosen	Menjadi lebih mudah melakukan presensi dan log kegiatan; peningkatan akuntabilitas
Mahasiswa	Mendapatkan akses terintegrasi dan cepat ke seluruh layanan akademik
Orang Tua/Wali	Dapat memantau nilai, kehadiran, dan notifikasi akademik anak secara *real-time*
Staf Akademik/Admin	Proses manual digantikan otomatisasi, efisiensi kerja meningkat

4.6. Roadmap Implementasi dan Migrasi

Roadmap Implementasi dan migrasi ke microservices disusun menjadi lima gelombang 12–18 bulan agar risiko migrasi dapat disebar dan pembelajaran pada fase awal langsung dimanfaatkan di fase berikutnya:

1. Persiapan & Baseline (0–3 bulan). Menetapkan arsitektural, membentuk tim lintas fungsi,
2. Pilot Migration (3–9 bulan). Ekstraksi dua layanan berisiko rendah—Notification Service dan Parent Monitoring mengaktifkan API Gateway, observability stack, dan CI/CD dasar.
3. Core Academic Lift-and-Shift (9–18 bulan). Memecah modul KRS/KHS & Presensi Mahasiswa; menerapkan event bus (Kafka) sebagai backbone asinkron; memigrasi skema data incrementally tanpa downtime.
4. Ecosystem Expansion (18–24 bulan). Integrasi layanan eksternal (payment gateway, library digital) melalui kontrak standar; penerapan policy as code dan service mesh untuk keamanan e2e.
5. Optimization & AI-Assisted Governance (>24 bulan). Otomatisasi *runtime* slicing database, prediksi beban menggunakan Machine Learning.

Kerangka lima fase tersebut sejalan dengan hasil *systematic mapping study* oleh Martínez Saucedo et al. (2025) yang menegaskan pentingnya *baseline metrics*, *pilot migration*, dan *continuous monitoring* sebagai kunci keberhasilan transformasi berkelanjutan [25]. Studi tersebut juga menyoroti keterbatasan alat otomatis pada domain *data orchestration* dan *service boundary detection*, sehingga kombinasi analisis kode manual, *domain knowledge workshop*, dan alat *static analysis* disarankan pada fase 2–3. Dengan mengadopsi rekomendasi tersebut, Unsoed dapat meminimalkan risiko *big-bang migration* dan memastikan setiap fase menghasilkan nilai bisnis terukur sebelum melangkah ke fase berikutnya.

5. CONCLUSION

Penelitian ini merancang model arsitektur modular Super App berbasis microservices untuk mengintegrasikan lebih dari 30 sistem informasi akademik Unsoed. Dengan pendekatan C4, DDD, API-First, serta dukungan komponen kunci (API Gateway/BFF, SSO, CI/CD, observability), rancangan mampu: (1) mengurangi fragmentasi dan duplikasi fungsi; (2) meningkatkan skalabilitas, keamanan, dan kecepatan rilis; (3) menyediakan fitur spesifik domain seperti presensi berbasis lokasi, transkrip otomatis, dan pemantauan wali secara aman. Evaluasi fungsional dan non-fungsional menunjukkan waktu respons < 300 ms dan kepuasan pengguna tinggi. Tantangan utama meliputi refaktorisasi sistem lama, kesiapan SDM, dan governance API, yang dijawab melalui roadmap migrasi bertahap dan strategi pelatihan. Implementasi penuh dan monitoring jangka panjang diperlukan untuk memverifikasi keberlanjutan manfaat serta membuka peluang riset lanjutan terkait service mesh, AI-based observability, dan adaptive security.

REFERENCES

[1] S. Henning and W. Hasselbring, “Benchmarking scalability of stream processing frameworks deployed as microservices in the cloud,” Journal of Systems & Software, vol. 208, p. 111879, 2024, DOI: 10.1016/j.jss.2023.11187.

[2] M. K. V. S. P. S. Pillutla, “Enterprise-Scale Microservices Architecture: Domain-Driven Design and Cloud-Native Patterns Using the Spring Ecosystem,” Eur. J. Comput. Sci. Inf. Technol., vol. 13, no. 45, pp. 1–10, 2025, DOI : https://doi.org/10.37745/ejcsit.2013/vol13n45110.

[3] F. X. Senduk, X. B. N. Najoan, and S. R. U. A. Sompie, “Development of Microservices Architecture with RESTful API Gateway using Backend-for-frontend Pattern in Higher Education Academic Portal,” Jurnal Teknik Informatika, vol. 18, no. 1, pp. 315–324, 2023.

[4] A. Garindra, T. Wati, and I. W. W. P., “Perancangan Arsitektur Microservices Untuk Resiliensi Sistem Informasi Perpustakaan Pusat,” Jurnal Format, vol. 9, no. 2, pp. 99–108, 2020.

[5] M. F. Hardiyanto and M. Andarwati, “Perancangan aplikasi multi layanan berbasis super app,” Jurnal ELTEK, vol. 22, no. 1, pp. 17–28, 2024, DOI: 10.33795/eltek.v22i1.4015

[6] A. Lercher, J. Glock, C. Macho, and M. Pinzger, “Microservice API Evolution in Practice: A

Study on Strategies and Challenges," Journal of Systems & Software, vol. 215, p. 112110, 2024, DOI: https://doi.org/10.1016/j.jss.2024.112110

[7] Y. S. Ananda and T. D. Anggalini, "Inovasi Pelayanan Publik Melalui Penerapan Aplikasi SuperApp Polri di Polda DIY," Journal of Public Policy and Administration Research, vol. 2, no. 5, 2024.

[8] J. R. Sumirata and A. R. Syahputra, "'Super App Precision' Sebagai Bentuk Pelayanan Publik di Era Masyarakat 5.0," JUSTIN, vol. 12, no. 1, 2024, DOI : https://doi.org/10.26418/justin.v11i3.65162

[9] M. R. Nashrulloh, R. Setiawan, D. Heryanto, A. Sutedi, and R. Elsen, "Designing Microservices Architecture for Software Product in Startup," JUTIF, vol. 3, no. 1, pp. 45–48, 2022, DOI: https://doi.org/10.20884/1.jutif.2022.3.1.124

[10] A. Pratama, I. G. A. W. Astawa, and M. A. Huda, "One-Gateway System in Managing Campus Information Services," Jurnal Teknologi dan Sistem Informasi, vol. 8, no. 2, pp. 123–132, 2022, DOI: 10.31763/businta.v7i2.635

[11] N. Alshuqayran, M. Ali, and R. Evans, "Does microservice adoption impact the velocity? A cohort study," in Proc. (fill conf.), 2023, DOI: https://doi.org/10.1007/s10664-025-10673-7

[12] M. Fowler and J. Lewis, "Microservices and APIs: Foundations of Agile Digital Architecture," White Paper, ThoughtWorks, 2021.

[13] Y. Y. Marwanta and Badiyanto, "Implementasi Arsitektur Microservice pada Pembuatan Surat UKM Informatika dan Komputer Menggunakan Node.js," JIKO, vol. 4, no. 2, 2019, DOI: DOI: http://dx.doi.org/10.26798/jiko.v4i2.516

[14] E. L. Putra, J. Suseno, and D. Napitupulu, "Pengembangan Aplikasi Dengan Menggunakan Metode Design Science Research (DSR) Berdasarkan Analisis Technology Readiness Index (TRI) dan Technology Acceptance Model (TAM)," Jurnal Pekommas, vol. 8, no. 2, pp. 137–148, 2023.

[15] A. Prawiro, J. J. C. Tambotoh, and A. Nugroho, "Pengembangan Sistem Informasi Desa Cukilan Menggunakan Pendekatan Design Science Research," JATI (Jurnal Mahasiswa Teknik Informatika), vol. 7, no. 1, pp. 734–739, 2023.

[16] M. Hasselwander, "Digital platforms' growth strategies and the rise of super apps," Heliyon, vol. 10, no. 12, p. e25856, 2024, DOI: https://doi.org/10.1016/j.heliyon.2024.e25856

[17] Badan Pusat Statistik, Statistik Telekomunikasi Indonesia 2023, Jakarta: BPS-Statistics Indonesia, 2024.

[18] S. Waruwu and I. K. D. Nuryana, "Implementasi Arsitektur Monolitik Pada Rancang Bangun Sistem Informasi," JINACS: Journal of Informatics and Computer Science, vol. 4, no. 4, pp. 399–404, 2023.

[19] O. Aitlmoudden, M. Housni, N. Safeh, and A. Namir, "A Microservices-based Framework for Scalable Data Analysis in Agriculture with IoT Integration," Int. J. Interactive Mobile Technologies, vol. 17, no. 19, pp. 147–156, 2023, DOI: https://doi.org/10.3991/ijim.v17i19.40457

[20] V. Božić, "Microservices Architecture," Research Proposal, 2023.

[21] S. Saidah, L. Rachmaniah, and R. Syaban, "Implementasi Arsitektur Microservices pada Aplikasi Point of Sale Toko Flyover Stiker," JIKI (Jurnal Ilmu Komputer dan Informatika), vol. 4, no. 1, pp. 77–88, 2023.

[22] A. J. Lauwren and Y. B. D. Setianto, "Microservice and Monolith Performance Comparison in Transaction Application," PROXIES, vol. 5, no. 2, pp. 86–96, 2022.

[23] F. Auer, V. Lenarduzzi, M. Felderer, and D. Taibi, "From monolithic systems to microservices: An assessment framework," Information and Software Technology, vol. 137, p. 106600, 2021.

[24] H. Hassan, M. A. Abdel-Fattah, and W. Mohamed, "Migrating from monolithic to microservice architectures: A systematic literature review," International Journal of Advanced Computer Science and Applications, vol. 15, no. 10, pp. 104–115, 2024, DOI: DOI:10.14569/IJACSA.2024.0151013

[25] A. Martínez Saucedo, G. Rodríguez, F. G. Rocha, and R. P. dos Santos, "Migration of monolithic systems to microservices: A systematic mapping study," Information and Software Technology, vol. 177, p. 107590, 2025, DOI:10.14569/IJACSA.2024.0151013