Perancangan dan Evaluasi Arsitektur *Microservices* Menggunakan *Microservices Migration Pattern* dan *Microservices Scorecard*

Kurnia Ramadhan Putra[1*], Asep Rizal Nurjaman[2], Aldira Fitri Haniifah[3]
[1,2,3] Program Studi Sistem Informasi, Fakultas Teknologi Industri, Institut Teknologi Nasional Bandung
[1]kurniaramadhan@itenas.ac.id, [2]aseprizal@itenas.ac.id, [3]aldirafithanifah@gmail.com
*Penulis Korespondensi
(Kurnia ramadhan putra, Institut Teknologi Nasional Bandung, [1]kurniaramadhan@itenas.ac.id)

ABSTRAK

Perkembangan teknologi informasi mendorong pergeseran dari arsitektur monolitik ke *microservices* yang lebih modular dan fleksibel. Sistem Informasi Akademik (SIKAD) di Institut Teknologi Nasional (Itenas) Bandung saat ini masih menggunakan arsitektur monolitik, sehingga menghadapi kendala skalabilitas, pemeliharaan, dan ketergantungan antar modul. Penelitian ini bertujuan merancang serta mengevaluasi arsitektur *microservices* untuk SIKAD dengan menerapkan *Microservices Migration Pattern* dan menilai kualitas rancangan menggunakan *Microservices Scorecard*. Metode penelitian meliputi studi literatur, wawancara, dokumentasi, dan observasi terhadap sistem yang ada saat ini. Hasil evaluasi menunjukkan rancangan arsitektur *microservices* mencapai skor kesiapan 81,7% yang menunjukkan kategori *high readiness*, sehingga layak untuk diimplementasikan selanjutnya. Temuan ini memberikan kontribusi berupa pendekatan terukur dalam migrasi ke *microservices* di lingkungan perguruan tinggi, yang berimplikasi pada peningkatan skalabilitas, fleksibilitas, dan kemudahan pemeliharaan sistem.

Kata kunci: Arsitektur *Microservices*, Migrasi Monolitik, *Microservices Scorecard*, Sistem Informasi Akademik

ABSTRACT

The advancement of information technology has driven a shift from monolithic architectures to more modular and flexible microservices. The Academic Information System (SIKAD) at the Institut Teknologi Nasional (Itenas) Bandung currently employs a monolithic architecture, which poses challenges in scalability, maintenance, and inter-module dependencies. This study aims to design and evaluate a microservices architecture for SIKAD by applying Microservices Migration Pattern and assessing the design quality using the Microservices Scorecard. The research methods include literature review, interviews, documentation, and observation of the current system. Evaluation results indicate that the proposed microservices architecture achieves a readiness score of 81.7%, categorizing it as high readiness and suitable for subsequent implementation. These findings contribute a measurable approach to microservices migration in higher education environments, with implications for improved scalability, flexibility, and system maintainability.

Keywords: *Microservices Architecture, Monolithic Migration, Microservices Scorecard, Academic Information System*

1. PENDAHULUAN

Perkembangan teknologi informasi yang pesat telah mendorong perubahan paradigma dalam pengembangan perangkat lunak, dari pendekatan monolitik menuju arsitektur yang lebih modular dan fleksibel. Arsitektur monolitik yang menggabungkan seluruh logika aplikasi dalam satu unit besar, kini dianggap kurang adaptif dan efisien, terutama dalam menghadapi kebutuhan skalabilitas, kecepatan pengembangan, serta pemeliharaan yang semakin kompleks [1]. Oleh karena itu, arsitektur berbasis layanan seperti *Service-Oriented Architecture* (SOA) dan lebih lanjut *microservices architecture* (MSA) semakin diminati karena kemampuannya memecah sistem menjadi layanan kecil yang mandiri dan saling berkomunikasi melalui protokol ringan, sehingga memudahkan pengembangan paralel serta integrasi lintas teknologi [2], [3].

Dalam konteks pendidikan tinggi, Sistem Informasi Akademik (SIKAD) berperan vital dalam pengelolaan data dan proses akademik, baik dosen maupun mahasiswa. Di Institut Teknologi Nasional (Itenas) Bandung, SIKAD saat ini masih menerapkan arsitektur monolitik, di mana seluruh fungsionalitas diintegrasikan dalam satu aplikasi tunggal. Hal ini menimbulkan berbagai kendala, seperti *downtime* sistem ketika ada perubahan pada satu modul, keterlambatan pengembangan fitur baru, serta risiko kesalahan manual akibat proses yang kurang otomatis dan minim dokumentasi. Kondisi ini menjadi penghambat efisiensi pengelolaan akademik dan berpotensi mengurangi kepuasan pengguna sistem.

Beberapa penelitian terdahulu telah membahas transisi dari arsitektur monolitik ke *microservices* mengemukakan *Microservices Migration Pattern* sebagai panduan praktis yang menggabungkan konsep teoretis dan temuan empiris dari industri [4]. Kemudian pendekatan konseptual dalam buku *Building Microservices* menunjukkan peningkatan resiliensi sistem setelah migrasi ke MSA [5], [6], serta keunggulan MSA dalam hal responsivitas aplikasi dibandingkan arsitektur monolitik [7]. Namun, sebagian besar penelitian ini fokus pada proses migrasi dan implementasi, tanpa evaluasi komprehensif terhadap kualitas rancangan arsitektur *microservice* itu sendiri.

Di Indonesia sendiri, penelitian terkait penerapan arsitektur microservices juga telah dilakukan pada beberapa domain seperti sistem akademik, *e-commerce*, dan layanan pemerintahan [8]–[10]. Namun, pendekatan yang digunakan umumnya masih terbatas pada pemodelan layanan dan pengujian performa dasar tanpa panduan migrasi yang terstandardisasi. Evaluasi rancangan arsitektur pada umumnya hanya mencakup aspek teknis seperti waktu respons, tampilan, atau hasil integrasi API, serta dilakukan pada prototipe aplikasi jumlah layanan yang ditentukan sendiri yaitu 4 sampai 5 layanan tanpa melihat kondisi nyata yang sebenarnya. Selain itu, sebagian besar studi tersebut tidak menyertakan evaluasi formal berbasis kerangka penilaian arsitektur dan tidak melibatkan praktisi sebagai evaluator. Keterbatasan kuantitatif ini menunjukkan bahwa kualitas arsitektur microservices yang dihasilkan belum terukur secara menyeluruh.

Penelitian yang dilakukan oleh Lecrivain, dkk (2025) menggunakan *Microservices Scorecard* sebagai metode evaluasi menyeluruh terhadap rancangan MSA, termasuk aspek teknis dan organisasi [11]. Namun, penerapannya pada konteks sistem informasi akademik di Indonesia masih sangat terbatas. Scorecard ini menilai berbagai dimensi seperti skalabilitas, independensi layanan, operability, maintainability, coupling, serta kesiapan organisasi, yang belum digunakan dalam penelitian lokal sebelumnya. Dengan demikian, terdapat gap penting berupa kekurangan evaluasi arsitektur yang komprehensif dan terukur, padahal kualitas rancangan sangat menentukan keberhasilan migrasi jangka panjang.

Berdasarkan data laporan *Microservices Adoption* pada tahun 2020, sebesar 77% perusahaan telah mengadopsi arsitektur *microservices* [12], yang mengindikasikan bahwa hal tersebut menjadi tren baik untuk perusahan maupun perguruan tingi yang banyak memberikan pelayanan melalui sistem, sehingga harus kualitas sistem informasinya. Pada era digital saat ini, kebutuhan untuk memiliki sistem akademik yang handal, fleksibel, dan dapat dikembangkan dengan cepat menjadi sangat penting agar dapat mendukung kegiatan belajar mengajar dan administrasi akademik secara efektif.

Urgensi migrasi SIKAD Itenas ke arsitektur MSA juga didorong oleh tuntutan transformasi digital di lingkungan kampus, di mana pengembangan fitur baru harus responsif terhadap kebutuhan pengguna, *downtime* harus diminimalisir, serta pemeliharaan sistem dapat dilakukan secara modular. Dengan pendekatan MSA, diharapkan institusi dapat mempercepat inovasi teknologi tanpa mengorbankan kestabilan layanan yang krusial bagi operasional akademik.

Oleh karena itu, penelitian ini bertujuan untuk merancang arsitektur *microservices* pada Sistem Informasi Akademik Itenas (SIKAD) dengan menggunakan *Microservices Migration Pattern* sebagai panduan migrasi dari arsitektur monolitik. Selanjutnya, rancangan yang dihasilkan akan dievaluasi secara komprehensif menggunakan *Microservices Scorecard*, dengan validasi dari praktisi berpengalaman, untuk memastikan kualitas, kesiapan, dan kematangan sistem sebelum implementasi penuh dilakukan.

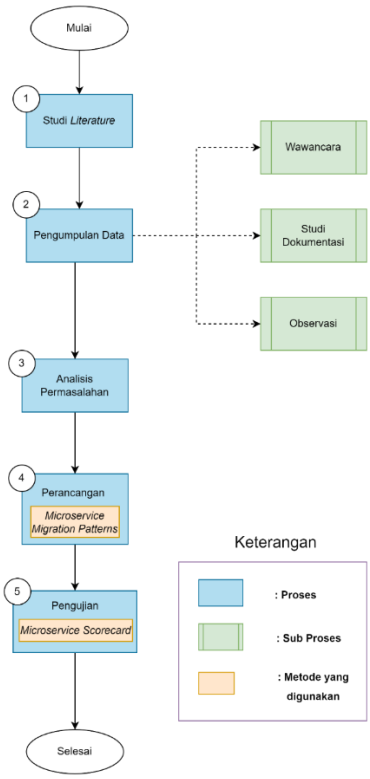
.

2. METODOLOGI PENELITIAN

Metodologi penelitian ini dirancang secara sistematis untuk memastikan proses perancangan arsitektur microservices berjalan terarah dan terukur. Tahapan metodologi terdiri dari 5 tahapan utama yang dimulai dari studi literatur, pengumpulan data, analisis permasalahan, perancangan solusi berbasis *Microservice Migration Pattern*, hingga tahap pengujian menggunakan *Microservice Scorecard*. Alur dari metodologi penelitian tersebut dapat dilihat pada Gambar 1.

Tahap studi literatur meliputi kajian teori terkait arsitektur microservices, pola migrasi, dan metode evaluasi sistem, untuk mengidentifikasi konsep fundamental dan instrumen evaluasi yang relevan. Tahap pengumpulan data dilakukan melalui wawancara dengan pengelola SIKAD, studi dokumentasi sistem, dan observasi proses bisnis, guna memahami kebutuhan pengguna, kendala arsitektur monolitik, serta konteks organisasi.

Data yang terkumpul dianalisis untuk menemukan akar masalah dan keterbatasan sistem lama. Hasil analisis menjadi dasar perancangan arsitektur microservices, dengan Microservice Migration Pattern sebagai panduan pemecahan fungsionalitas, identifikasi bounded context, dan penyusunan layanan modular. Pemilihan pattern disesuaikan dengan karakteristik proses bisnis, kompleksitas layanan, dan kebutuhan integrasi sistem internal maupun eksternal..



Gambar 1. Alur Metodologi Penelitian

Tahap berikutnya, rancangan arsitektur dievaluasi menggunakan Microservice Scorecard, yang mencakup tujuh kriteria utama yaitu Granularity and Reuse, Independence, Service Architecture, Alignment, Ownership, Automation, dan Governance, dengan total sekitar 34 sub-kriteria untuk menilai aspek teknis maupun non-teknis. Penilaian dilakukan melalui wawancara terstruktur, dokumentasi teknis, dan expert judgment, menggunakan skala 4 tingkat: Not Assessed, Not Ready (0), Partially Ready (1), dan Fully Ready (2). Skor tiap sub-kriteria diolah menjadi skor total yang mencerminkan kesiapan, kematangan, dan kelayakan rancangan microservices.

Untuk memastikan kualitas evaluasi, dua pakar arsitektur microservices dan SOA dilibatkan untuk validasi, dengan membandingkan hasil antar evaluator dan meninjau konsistensi antara temuan analisis dan skor Scorecard. Triangulasi data dari wawancara, dokumentasi, dan expert judgment memperkuat keandalan hasil evaluasi.

Penelitian memiliki keterbatasan, antara lain jumlah evaluator yang terbatas sehingga potensi subjektivitas tetap ada, fokus evaluasi pada rancangan bukan implementasi, dan hasil yang sangat bergantung pada konteks organisasi. Keterbatasan ini menjadi dasar bagi penelitian lanjutan, seperti evaluasi berbasis implementasi nyata, pengujian performa, atau penambahan jumlah evaluator.

2.1 Pendekatan Penelitian

Penelitian ini menggunakan studi kasus dan desain evaluatif untuk merancang dan mengevaluasi arsitektur microservices pada Sistem Informasi Akademik (SIKAD) Itenas. Studi kasus memungkinkan analisis mendalam terhadap karakteristik, tantangan, dan kebutuhan sistem, sedangkan desain evaluatif memastikan rancangan arsitektur dapat diuji sebelum implementasi, menggunakan Microservices Scorecard untuk menilai tingkat kesiapan desain [13].

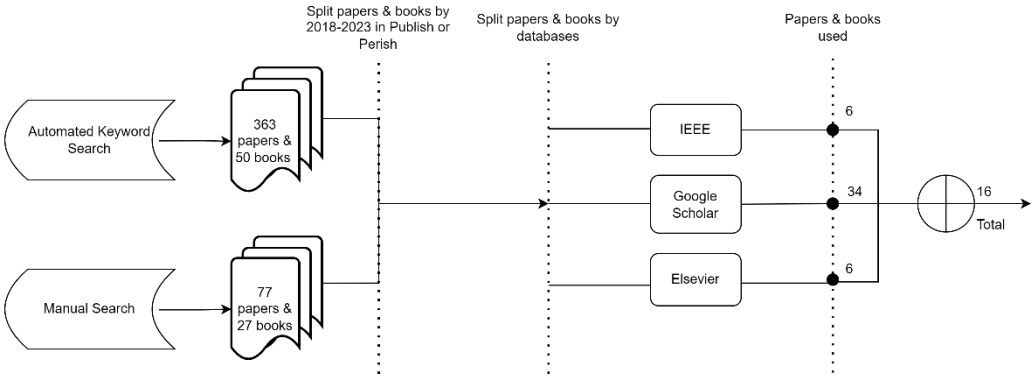
Transformasi dari arsitektur monolitik ke microservices mengacu pada Microservices Migration Pattern, yang memandu migrasi secara bertahap mulai dari identifikasi bounded context hingga desain komunikasi antar layanan. Kombinasi kedua pendekatan ini memungkinkan penelitian menghasilkan desain arsitektur yang sesuai prinsip microservices sekaligus memberikan bukti kuantitatif dan kualitatif mengenai kesiapan rancangan sebagai acuan implementasi di masa depan.

2.2 Tahapan Penelitian

Tahapan penelitian ini disusun untuk memastikan proses perancangan dan evaluasi arsitektur *microservices* pada SIKAD Itenas berjalan secara sistematis. Tahapan tersebut dapat dilihat pada Gambar 2, yang menggambarkan alur kegiatan mulai dari pengumpulan landasan teori hingga evaluasi akhir rancangan.

2.2.1 Studi Literatur

Kegiatan studi literatur dilakukan untuk memperoleh landasan teori dan praktik terkait *microservices*, arsitektur monolitik, *migration pattern*, dan metode evaluasi *microservices* yang mana penelurusanya dapat dilihat pada Gambar 2.



Gambar 2. Penelusuran Studi Literatur

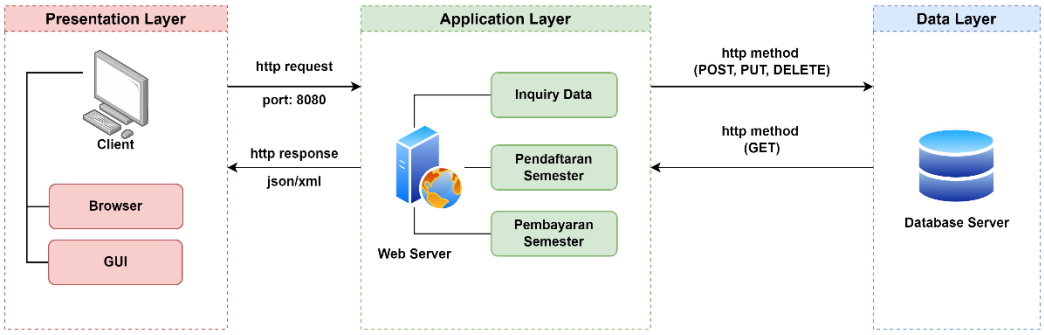
Proses penelurusan literatur dilakukan melalui dua pendekatan, yaitu *automated keyword search* menggunakan tool Harzing's Publish or Perish dan *manual search* menggunakan Google. Pencarian otomatis menggunakan kata kunci yang relevan menghasilkan 363 artikel dan 50 buku, sedangkan pencarian manual menghasilkan 77 artikel dan 27 buku. Seluruh publikasi yang ditemukan kemudian disaring berdasarkan kriteria inklusi dan eksklusi, serta dilakukan penghapusan duplikasi antar sumber. Basis data yang digunakan meliputi IEEE, Google Scholar, dan Elsevier. Setelah proses seleksi akhir, diperoleh sejumlah publikasi yang layak digunakan sebagai referensi utama penelitian.

2.2.2 Pengumpulan Data

Pengumpulan data dilakukan untuk memperoleh gambaran menyeluruh mengenai kondisi SIKAD Itenas, kebutuhan pengguna, dan tantangan teknis. Data diperoleh dari wawancara terstruktur dengan mahasiswa, staf administrasi, dan tim pengembang untuk mengidentifikasi permasalahan pengguna, kebutuhan fitur, dan ekspektasi sistem; studi dokumentasi sistem, diagram alur proses bisnis, spesifikasi teknis, dan dokumentasi pemeliharaan untuk memetakan arsitektur saat ini, aliran data, integrasi antar modul, serta dependensi sistem; dan observasi langsung di lingkungan operasional kampus untuk memvalidasi kesesuaian antara proses bisnis terdokumentasi dan praktik lapangan, sekaligus mengidentifikasi kelemahan arsitektur monolitik. Data yang terkumpul dianalisis untuk mengevaluasi kondisi SIKAD saat ini, mengidentifikasi akar masalah dan keterbatasan sistem monolitik sebagai dasar perancangan arsitektur microservices.

2.2.3 Analisis Permasalahan

Tahap analisis permasalahan bertujuan untuk mengevaluasi kondisi SIKAD saat ini yang masih menggunakan arsitektur monolitik seperti yang dapat dilihat pada Gambar 3.

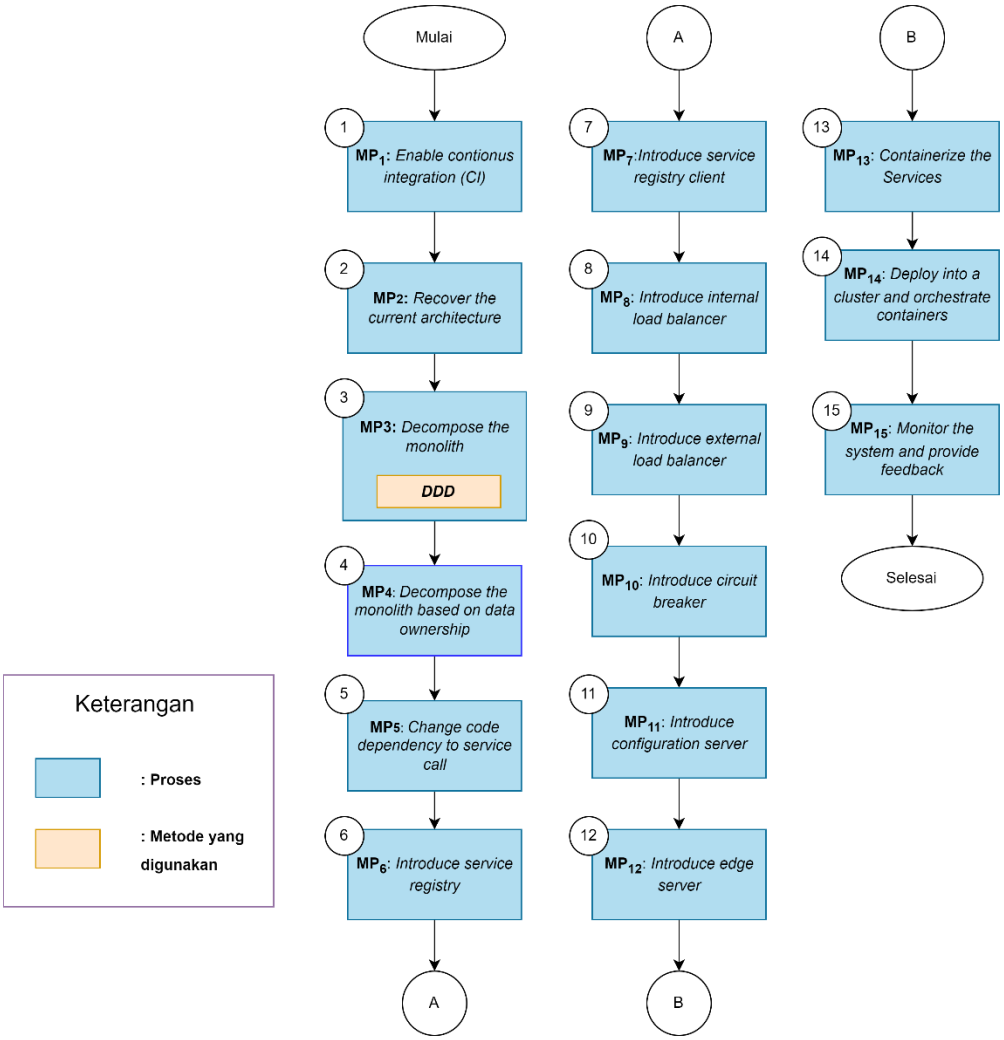


Gambar 3. SIKAD Menggunakan Arsitektur Monolitik

Arsitektur SIKAD masih monolitik dengan tiga lapisan: *presentation layer* berbasis web, *application layer* yang menggabungkan semua logika bisnis, dan *data layer* tunggal. Ketergantungan antar modul sangat tinggi sehingga perubahan kecil memengaruhi seluruh sistem, membatasi skalabilitas, pemeliharaan, dan integrasi layanan baru.

2.2.4 Perancangan Arsitektur *Microservices*

Perancangan arsitektur SIKAD menggunakan Microservices Migration Pattern (Balalaie dkk., 2018) untuk memandu migrasi monolitik ke microservices secara terstruktur dan bertahap, meminimalkan risiko, serta menjaga kompatibilitas dengan sistem lama. Pada penelitian ini, tahapan diadaptasi sebagai design readiness, berupa rancangan konseptual dan spesifikasi teknis tanpa implementasi penuh yang dilihat pada Gambar 4.



Gambar 4. Microservices Migration Pattern

Penjelasan setiap tahapan dari *Microservice Migration Pattern* sebagai berikut:

1) *Enable Continuous Integration* (CI)
   Menyusun lingkungan pengembangan yang mendukung integrasi kode secara berkelanjutan melalui *version control* (Git) dan CI server (misalnya Jenkins atau GitHub Actions). Tujuannya memastikan setiap perubahan kode dapat diuji otomatis sebelum digabungkan ke repositori utama.
2) *Recover Current Architecture*
   Melakukan rekonstruksi arsitektur SIKAD yang ada, termasuk pemetaan modul, dependensi, alur data, dan deployment topology untuk memahami titik kritis dan keterbatasan.
3) *Decompose Monolith*
   Mendekomposisi modul sistem sesuai *bounded context* berdasarkan prinsip *Domain-Driven Design* (DDD).
4) *Decompose Monolith based on Data Ownership*
   Setiap domain memiliki *data ownership* (kepemilikan data) yang jelas untuk menghindari ketergantungan data tersebut satu sama lain.
5) *Change Code Dependency to Service Call*
   Mengganti pemanggilan fungsi langsung antar modul menjadi komunikasi antar layanan melalui REST API, sehingga modul dapat beroperasi secara independen.
6) *Introduce Service Registry*

Menambahkan mekanisme *service discovery* menggunakan *library* Eureka, yang memungkinkan layanan menemukan lokasi layanan lain secara dinamis tanpa konfigurasi manual.

7) *Introduce Service Registry Client*
Melakukan konfigurasi service registry client untuk memfalisitasi lokasi dimanis dari instance setiap service.
8) *Implement Internal Load Balancer*
Melakukan penambahan internal load balancing pada setiap service untuk menyeimbangkan beban antara instance yang tersedia sehingga tidak akan ada server yang mengalami kelebihan beban atau dalam artian lain terhindar dari server yang overload.
9) *Implement External Load Balancer*
Merancang penambahan external load balancing pada setiap layanan untuk menangani lalu lintas dari pengguna eksternal yang terhubung melalui internet public.
10) *Introduce Circuit Breaker*
Merancang penambahan pola *circuit breaker* untuk mencegah kegagalan berantai jika suatu layanan tidak merespons atau mengalami gangguan menggunakan *library* Hystrix.
11) *Introduce Configuration Server*
Merancang konfigurasi terpusat untuk seluruh layanan, sehingga perubahan konfigurasi dapat dilakukan tanpa men-*deploy* ulang setiap layanan menggunakan *library* Spring Cloud Config.
12) *Introduce Edge Server*
Merancang penambahan API gateway sebagai titik masuk tunggal yang mengatur *routing*, *security*, dan *rate limiting* menggunakan *library* Spring Cloud Gateway.
13) *Containerize Services menggunakan Docker*
Merancang pengemasan setiap layanan ke dalam kontainer Docker untuk memudahkan distribusi, scaling, dan pengelolaan dependensi lingkungan.
14) *Deploy into Cluster and Orchestrate Containers*
Merancang microservices *orchestration*, *auto-scaling*, dan *self-healing* dari layanan yang berjalan di dalam kluster menggunakan Kubernetes.
15) *Monitor System and Provide Feedback*
Merancang integrasi *dashboard* pemantauan kinerja layanan dan memberikan umpan balik bagi pengembang menggunakan *library* Grafana dan Prometheus.

Tahapan migrasi microservices dalam penelitian ini direpresentasikan sebagai desain konseptual yang komprehensif, mencakup diagram arsitektur, spesifikasi layanan, batas domain, mekanisme komunikasi, dan rancangan integrasi. Pendekatan ini memastikan design readiness yang memadai untuk implementasi selanjutnya sekaligus memberikan gambaran terstruktur mengenai strategi migrasi.

2.2.5 Pengujian dan Evaluasi

Evaluasi rancangan arsitektur dilakukan dengan pendekatan kuantitatif menggunakan *Microservices Scorecard* yang dilengkapi dengan metode *Analytic Hierarchy Process* (AHP) untuk penentuan bobot kriteria dan sub-kriteria [14]. Tahapan evaluasi dirancang sebagai berikut:

1) Penentuan Bobot Menggunakan AHP
Microservices Scorecard dibagi ke dalam tujuh praktik area seperti *Granularity and Reuse* (GR), *Independence* (IN), *Service Architecture* (SA), *Alignment* (AL), *Ownership* (OW), *Automation* (AU), dan *Governance* (GV) yang mana praktik area tersebut dijadikan sebagai kriteria dan pada masing-masing kriteria terdapat sub-kriteria.
   a. Kriteria dan sub-kriteria disusun dalam bentuk hierarki untuk memudahkan proses pembobotan.
   b. Pembuatan matriks *pairwaise comparison* (perbandungan berpasangan) dilakukan oleh pakar untuk menentukan tingkat kepentingan relatif antar elemen.
   c. Perhitungan bobot dilakukan menggunakan perhitungan *eigenvector*, dengan pengujian konsistensi melalui Consistency Ratio (CR), di mana penilaian dianggap konsisten jika CR ≤ 0,1. Untuk menghitung nilai CR memerlukan perhitungan *Consitency Index* (CI) terlebih dahulu yang dapat dilihat pada persamaan (1).

$$CI = \frac{\lambda_{max} - n}{n} \quad (1)$$

   Di mana, λ_{max} (lambda maksimum) adalah Nilai eigen (eigenvalue) terbesar yang diperoleh dari matriks perbandingan berpasangan setelah dinormalisasi dan dihitung bobotnya, dan n adalah jumlah elemen atau kriteria yang dibandingkan pada matriks perbandingan berpasangan.
   d. Sedangkan *Consistency Ratio* (CR) dihitung menggunakan persaman (2).

$$CR = \frac{CI}{RI} \quad (2)$$

   Di mana, nilai *Random Index* (RI) diperoleh dari tabel statistik. Bobot yang dihasilkan menjadi faktor pengali dalam perhitungan skor akhir setiap kriteria.

2) Perhitungan *Microservice Scorecard*

Penilaian kesiapan desain arsitektur microservices dilakukan berdasarkan skala yang ditetapkan dalam *Microservices Scorecard*. Skala ini digunakan untuk mengukur tingkat pemenuhan setiap kriteria pada masing-masing praktik area. Adapun skala penilaian tersebut adalah sebagai berikut:

- *Not Assessed* (un-scored)
  Aspek desain belum dievaluasi atau tidak ada data yang cukup untuk dilakukan penilaian, yang mana pada kondisi ini belum terdapat informasi atau pengukuran yang memadai untuk menentukan tingkat pencapaian terhadap kriteria yang dimaksud.
- *Fully Ready* (2)
  Desain telah siap sepenuhnya untuk tahap implementasi. Semua spesifikasi, diagram, dan integrasi antar komponen telah terdokumentasi dengan jelas, konsisten, dan dapat langsung digunakan oleh tim pengembang tanpa revisi signifikan.
- *Partially Ready* (1)
  Desain sudah memiliki elemen utama, namun masih terdapat kekurangan seperti detail integrasi yang belum final, dokumentasi yang belum lengkap, atau ketidakjelasan pada sebagian *workflow*, sehingga perlu dilakukan revisi sebelum implementasi.
- *Not Ready* (0)
  Desain belum memenuhi persyaratan untuk implementasi karena banyak aspek yang masih ambigu, tidak terdokumentasi, atau belum diuji kelayakannya.

Nilai akhir setiap sub-kriteria dihitung sebagai rata-rata skor dari dua penilai, dengan mengabaikan nilai *un-scored* agar tidak memengaruhi hasil. Rumus perhitungan rata-rata sub-kriteria ke-i pada area A dengan R penilai dapat dilihat pada persamaan (3).

$$\bar{S}_{A,i} = \frac{\sum_{r=1}^{R} S_{A,i}{}^{(r)}}{n_{A,i}} \quad (3)$$

Keterangan:
$S_{A,i}{}^{(r)}$ = skor penilai ke-r untuk sub-kriteria ke-i
R = jumlah penilai misalnya 2 orang
$n_{A,i}$= jumlah skor valid (0, 1, 2) dan tidak menghitung *un-scored*

Skor suatu praktik area diperoleh dengan menghitung rata-rata nilai semua sub-kriteria yang memiliki setidaknya satu skor valid yang dapat dilihat pada persamaan (4).

$$Score(A) = \frac{1}{m_A} \sum_{i=1}^{m_A} \bar{S}_{A,i} \quad (4)$$

Di mana, m_A adalah jumlah sub-kriteria pada area A yang memiliki skor valid.

Skor keseluruhan desain diperoleh dari rata-rata skor seluruh praktik area. Untuk praktik area (K) = 7, maka dapat menggunakan persamaan (5).

$$Overall\ Score = \frac{1}{K} \sum_{A=1}^{K} Score(A) \quad (5)$$

Jika ingin menggunakan bobot w_A setiap masing-masing area, maka dapat menggunakan persamaan (6).

$$Overall\ Weighted\ Score = \frac{\frac{1}{K} \sum_{A=1}^{K} w_A . Score(A)}{\sum_{A=1}^{K} w_A} \quad (6)$$

Skor dapat dinormalisasi ke dalam bentuk persentase 0–100% dengan membagi skor terhadap nilai maksimum yaitu 2, kemudia mengalikannya dengan 100 seperti pada persamaan (7) dan (8).

$$Persentase\ Area\ A = \frac{Score(A)}{2} x100\% \quad (7)$$

$$Persentase\ Overall\ Score = \frac{Overall\ Score}{2} x100\% \quad (8)$$

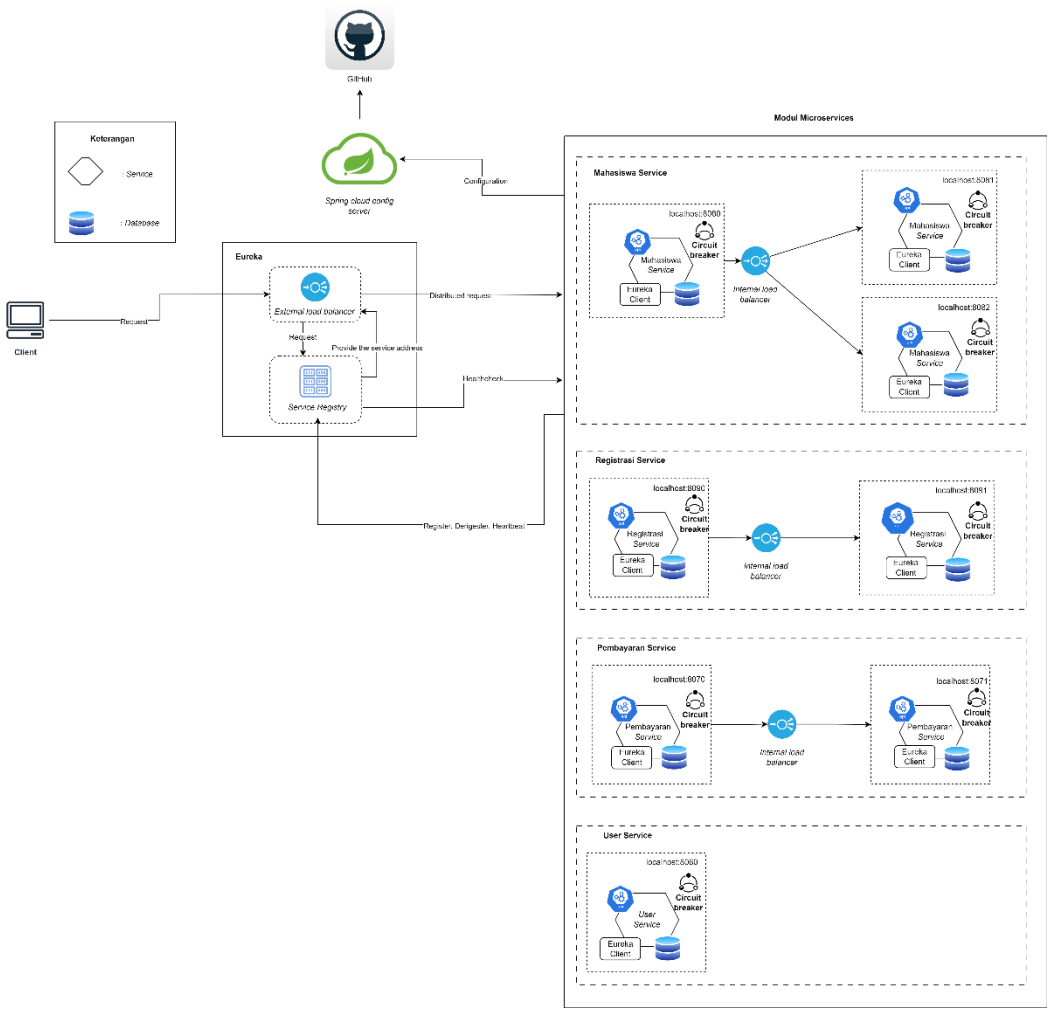
Untuk menginterpretasikan skor tersebut dapat dikategorikan menjadi tiga tingkat *design readiness* yaitu:

- *Low Readiness*: $0{,}00 \leq Persentase\ Overall\ Score \leq 0{,}50$
- *Mediun Readiness*: $0{,}50 < Persentase\ Overall\ Score \leq 0{,}75$
- *High Readiness*: $0{,}75 < Persentase\ Overal\ Score \leq 100$

3. HASIL DAN PEMBAHASAN

3.1 Hasil Rancangan Arsitektur *Microservices*

Untuk memahami bagaimana SIKAD telah didesain ulang menggunakan arsitektur *microservices*, dapat dilihat pada Gambar 5. Gambar ini menunjukkan bagaimana sistem yang sebelumnya berbasis monolitik kini dipecah menjadi layanan-layanan independen yang saling berinteraksi melalui mekanisme terstruktur.



Gambar 5. Rancangan SIKAD Menggunakan Arsitektur Microservices

Dalam tahap desain, SIKAD diposisikan sebagai kumpulan microservices, yang mana setiap layanan dapat dikembangkan, dikelola, dan diintegrasikan secara terpisah. Rancangan arsitektur microservices mencakup beberapa prinsip utama:

1) *Single Entry Point*: emua pengguna berinteraksi melalui API Gateway, yang menangani autentikasi, otorisasi, load balancing, dan pencatatan aktivitas.
2) Pemisahan Domain Layanan: setiap domain fungsional, seperti manajemen data mahasiswa, registrasi, pembayaran, dan akun, menjadi layanan mandiri dengan basis data independen untuk fleksibilitas dan skalabilitas.
3) Mekanisme Penemuan dan Komunikasi Layanan: layanan mendaftar di *service registry* dan berkomunikasi menggunakan HTTP/REST dengan format data standar JSON untuk interoperabilitas.
4) Siklus Pengembangan Terdistribusi: repositori terpisah untuk setiap layanan memungkinkan pengembangan, pengujian, dan deployment paralel, mendukung CI/CD dan meminimalkan risiko gangguan.
5) Kinerja dan Keandalan: layanan kritis dapat diskalakan mandiri; rancangan mencakup *fallback mechanism* dan *circuit breaker* agar kegagalan satu layanan tidak melumpuhkan sistem.
6) Fleksibilitas Teknologi: setiap layanan dapat menggunakan teknologi, *framework*, dan basis data sesuai domainnya, memudahkan evolusi jangka panjang tanpa mengubah seluruh ekosistem.

Pendekatan desain *microservices* menjadikan SIKAD sebagai ekosistem modular yang *scalable* dan *resilient*. Dengan pemisahan layanan per domain, interaksi melalui API Gateway, dan komunikasi terstandar, rancangan ini menyediakan fondasi siap transformasi dan pengembangan sebelum implementasi dimulai.

3.2 Hasil Pembobotan Kriteria Evaluasi Microservices Scorecard Menggunakan AHP

3.2.1 Penentuan Kriteria

Penilaian kualitas rancangan arsitektur *microservices* dilakukan menggunakan tujuh praktik area dari Microservice Scorecard, yang mencakup kualitas desain layanan (*Granularity and Reuse*, *Independence*, *Service Architecture*), kesesuaian strategi dan bisnis (*Alignment*, *Ownership*), serta faktor pendukung implementasi berkelanjutan (*Automation, Governance*). Ketujuh area ini digunakan sebagai kriteria dalam AHP, yang diuraikan pada Tabel 1.

Tabel 1. Kriteria Yang Ditentukan

Kode Kriteria	Nama Kriteria
GR	Granularity and Reuse
IN	Independence
SA	Service Architecture
AL	Alignment
OW	Ownership
AU	Automation
GV	Governance

3.2.2 Pembentukan Matriks *Pairwise Comparison* (Perbandingan Berpasangan)

Setiap praktik area dilakukan pembuatan matriks *pairwise comparison* (perbandingan berpasangan) untuk menilai tingkat kepentingan antar masing-masing kriteria dengan skala Saaty (1-9) seperti yang diuraikan pada Table 2.

Tabel 2. Skala Saaty

Skala	Interpretasi
1	Sama Penting
3	Sedikit Lebih Penting
5	Lebih Penting
7	Sangat Lebih Penting
9	Mutlak Lebih Penting
2, 4, 5, 6, 8	Nilai di antara Skala Utama
Resciprocal Value (1/n)	Nilai kebalikan, jika A lebih penting dari B, maka nilai B adalah 1/n terhadap A

Penilaian perbandingan berpasangan dilakukan oleh dua orang responden ahli yaitu Responden 1 (R1) dan Reponden 2 (R2), yang mana profil dari responden tersebut dapat dilihat pada Tabel 3.

Tabel 3. Profil Responden

Skala	Responden 1 (R1)	Responden 2 (R2)
Jabatan	Devops Engineer	IT Specialist
Perusahaan	Tada Network	PT. Bank Central Asia
Lokasi	Bandung	Jakarta
Pengalaman Kerja	2021 – 2022: Devops Engineer, Part-time di PT. Ihsan Solusi Informatika 2020 – sekarang: Devops Engineer, Full-time, di PT. Tada Network	2023 – sekarang: Backend Developer, pengembangan *microservices* pada fitur transaksional BCA.

Hasil matriks perbandingan berpasangan dari Responden 1 dapat dilihat pada Table 4.

Tabel 4. Matriks Perbandingan Berpasangan Kriteria dari Responden 1

Kriteria	GR	IN	SA	AL	OW	AU	GV
GR	1,00	3,00	3,00	5,00	4,00	3,00	5,00
IN	0,33	1,00	3,00	4,00	4,00	3,00	5,00
SA	0,33	0,33	1,00	5,00	3,00	3,00	5,00
AL	0,20	0,25	0,20	1,00	2,00	0,33	3,00
OW	0,25	0,25	0,33	0,50	1,00	0,33	3,00
AU	0,33	0,33	0,33	3,00	3,00	1,00	3,00
GV	0,20	0,20	0,20	0,33	0,33	0,33	1,00

Hasil matriks perbandingan berpasangan dari Responden 2 dapat dilihat pada Table 5.

Tabel 5. Matriks Perbandingan Berpasangan Kriteria dari Responden 2

Kriteria	GR	IN	SA	AL	OW	AU	GV
GR	1,00	1,00	4,00	4,00	5,00	3,00	9,00
IN	1,00	1,00	2,00	5,00	5,00	4,00	5,00
SA	0,25	0,50	1,00	4,00	3,00	3,00	5,00
AL	0,25	0,20	0,25	1,00	3,00	5,00	4,00
OW	0,20	0,20	0,33	0,33	1,00	0,25	2,00
AU	0,33	0,25	0,33	0,20	4,00	1,00	4,00
GV	0,11	0,20	0,20	0,25	0,50	0,25	1,00

Setelah diperoleh matriks perbandingan berpasangan dari R1 dan R2, langkah selanjutnya adalah menyusun matriks perbandingan berpasangan dengan Geomean menggunakan persamaan (9).

$$GM_{i.j} = \sqrt{a_{ij}{}^{(1)} a_{ij}{}^{(2)}} \quad (9)$$

Di mana, $a_{ij}{}^{(1)}$ merupakan nilai perbandingan antara kriteria i dan j dari R1, sedangkan $a_{ij}{}^{(2)}$ merupakan nilai dari R2. Hasil perhitungan Geomean untuk setiap elemen matriks selanjutnya ditunjukkan pada Tabel 6.

Tabel 6. Matriks Geomean Kriteria dari Responden 1 dan Responden 2

Kriteria	GR	IN	SA	AL	OW	AU	GV
GR	1,00	1,73	3,46	4,47	4,47	3,00	6,71
IN	0,58	1,00	2,45	4,47	4,47	3,46	5,00
SA	0,29	0,41	1,00	4,47	3,00	3,00	5,00
AL	0,22	0,22	0,22	1,00	2,45	1,29	3,46
OW	0,22	0,22	0,33	0,41	1,00	0,29	2,45
AU	0,33	0,29	0,33	0,77	3,46	1,00	3,46
GV	0,15	0,20	0,20	0,29	0,41	0,29	1,00

3.2.3 Penentuan *Eigenvector* (Bobot Prioritas)

Berdasarkan perhitungan *eigenvector*, maka diperoleh bobot prioritas masing-masing kriteria yang dapat dilihat pada Tabel 7.

Tabel 7. Bobot Prioritas Kriteria

Kriteria	Bobot Kriteria	Bobot (%)
GR	0,32	32,00
IN	0,26	26,00
SA	0,17	17,00
AL	0,09	9,00
OW	0,08	8,00
AU	0,05	5,00
GV	0,03	3,00

Nilai *Consistency Ratio* (CR) sebesar 0,06 menunjukkan perbandingan kriteria konsisten dan dapat dijadikan dasar pengambilan keputusan. Bobot tertinggi diperoleh pada *Granularity and Reuse* (32,0%), menandakan modularitas dan kemampuan layanan untuk digunakan kembali menjadi faktor kunci keberhasilan *microservices*, sedangkan *Governance* (0,03) memiliki pengaruh relatif lebih kecil.

3.3 Hasil Perhitungan *Microservice Scorecard*

Setelah arsitektur microservices untuk SIKAD berhasil dirancang, langkah selanjutnya adalah melakukan evaluasi untuk menilai *design readiness* (tingkat kesiapan rancangan). Evaluasi dilakukan dengan menggunakan *Microservices Scorecard*, sebuah kerangka penilaian yang memungkinkan pengukuran kualitas rancangan *microservices* secara terstruktur dan kuantitatif.

Melalui *Microservices Scorecard* setiap aspek penting dari arsitektur *microservices*, seperti keselarasan dengan tujuan bisnis, kemandirian layanan, struktur arsitektur, hingga mekanisme tata kelola, dapat dinilai secara objektif. Dengan pendekatan ini, evaluasi tidak hanya menggambarkan tingkat kesiapan arsitektur secara keseluruhan, tetapi juga memperlihatkan kekuatan dan kelemahan pada tiap aspek. Selanjutnya akan dijabarkan proses perhitungan dan hasil evaluasi untuk masing-masing kriteria dalam *Microservices Scorecard*.

3.3.1 *Granularity and Reuse* (GR)

Aspek *Granularity and Reuse* dievaluasi untuk menilai tingkat ketepatan granularitas layanan serta pemanfaatannya kembali, dengan hasil perhitungan ditunjukkan pada Tabel 8.

Tabel 8. Hasil Scorecard Kriteria GR

Kriteria	Sub-Kriteria	R1	R2	Rata-Rata	Normalisasi	Kontribusi Bobot
GR (0,32)	GR-01 (0,064)	un-scored	1	1,00	0,5	0,032
	GR-02 (0,064)	un-scored	1	1,00	0,5	0,032
	GR-03 (0,064)	un-scored	2	2,00	1,0	0,064
	GR-04 (0,064)	2	2	2,00	1,0	0,064
	GR-05 (0,064)	2	2	2,00	1,0	0,064
Skor Akhir GR = (0,032 + 0,032 + 0,064 + 0,064 + 0,064)						0,256
Presentase GR = (0,256/0,32) x 100%						80%

3.3.2 *Independence* (IN)

Aspek *Independence* dianalisis untuk mengukur sejauh mana layanan dapat dirilis, diganti, diskalakan, dan dibangun secara mandiri, dengan hasil evaluasi disajikan pada Tabel 9.

Tabel 9. Hasil Scorecard Kriteria IN

Kriteria	Sub-Kriteria	R1	R2	Rata-Rata	Normalisasi	Kontribusi Bobot
IN (0,26)	IN-01 (0,065)	2	2	2,00	1,0	0,065
	IN-02 (0,065)	2	2	2,00	1,0	0,065
	IN-03 (0,065)	2	2	2,00	1,0	0,065
	IN-04 (0,065)	1	2	1,50	0,75	0,049
Skor Akhir IN = (0,065 + 0,065 + 0,065 + 0,049)						0,244
Presentase IN = (0,244/0,26) x 100%						93,75%

3.3.3 *Service Architecture* (SA)

Aspek *Service Architecture* dievaluasi untuk menilai kesesuaian desain layanan dalam mendukung fleksibilitas, skalabilitas, serta keteraturan arsitektur, dengan hasil perhitungan ditampilkan pada Tabel 10.

Tabel 10. Hasil Scorecard Kriteria SA

Kriteria	Sub-Kriteria	R1	R2	Rata-Rata	Normalisasi	Kontribusi Bobot
SA (0,17)	SA-01 (0,026)	2	2	2,00	1,0	0,026
	SA-02 (0,018)	2	2	2,00	1,0	0,018
	SA-03 (0,018)	2	1	1,50	0,75	0,014
	SA-04 (0,018)	2	1	1,50	0,75	0,014
	SA-05 (0,018)	2	1	1,50	0,75	0,014
	SA-06 (0,018)	2	1	1,50	0,75	0,014
	SA-07 (0,018)	2	1	1,50	0,75	0,014
	SA-08 (0,018)	2	1	1,50	0,75	0,014
	SA-09 (0,018)	2	1	1,50	0,75	0,014
Skor Akhir SA = (0,026 + 0,018 + 0,014 + 0,014 + 0,014 + 0,014 + 0,014 + 0,014 + 0,014)						0,114
Presentase SA = (0,114/0,17) x 100%						81,47%

3.3.4 *Alignment* (AL)

Aspek *Alignment* dianalisis untuk memastikan kesesuaian rancangan arsitektur microservices dengan kebutuhan bisnis dan tujuan organisasi, sebagaimana ditunjukkan pada Tabel 11.

Tabel 11. Hasil Scorecard Kriteria AL

Kriteria	Sub-Kriteria	R1	R2	Rata-Rata	Normalisasi	Kontribusi Bobot
AL (0,09)	AL-01 (0,03)	2	un-scored	2,00	1,0	0,030
	AL-02 (0,03)	2	2	2,00	1,0	0,030
	AL-03 (0,03)	2	2	2,00	1,0	0,030
Skor Akhir AL = (0,030 + 0,030 + 0,030)						0,090

Kriteria	Sub-Kriteria	R1	R2	Rata-Rata	Normalisasi	Kontribusi Bobot
Presentase AL = (0,090/0,09) x 100%						100%

3.3.5 *Ownership* (OW)

Aspek *Ownership* dievaluasi untuk menilai sejauh mana kepemilikan layanan dapat dikelola secara jelas dan terdistribusi, sehingga setiap layanan memiliki tanggung jawab yang terdefinisi dengan baik, sebagaimana ditunjukkan pada Tabel 12.

Tabel 12. Hasil Scorecard Kriteria OW

Kriteria	Sub-Kriteria	R1	R2	Rata-Rata	Normalisasi	Kontribusi Bobot
OW (0,08)	OW-01 (0,027)	2	0	1	0,5	0,000
	OW-02 (0,027)	2	1	1,5	0,75	0,014
	OW-03 (0,026)	2	1	1,5	0,75	0,013
Skor Akhir OW = (0,000 + 0,014 + 0,013)						0,027
Presentase OW = (0,027/0,08) x 100%						33,11%

3.3.6 *Automation* (AU)

Aspek *Automation* dianalisis untuk menilai tingkat otomatisasi dalam proses pengembangan dan pengelolaan layanan, mencakup otomatisasi build, testing, deployment, hingga monitoring dan pemulihan kegagalan API, sebagaimana ditunjukkan pada Tabel 13.

Tabel 13. Hasil Scorecard Kriteria AU

Kriteria	Sub-Kriteria	R1	R2	Rata-Rata	Normalisasi	Kontribusi Bobot
AU (0,05)	AU-01 (0,008)	2	2	2	1,0	0,008
	AU-02 (0,007)	2	2	2	1.0	0,007
	AU-03 (0,007)	2	0	1	0,5	0,004
	AU-04 (0,007)	0	2	1	0,5	0,004
	AU-05 (0,007)	2	2	2	1,0	0,007
	AU-06 (0,007)	2	un-scored	2	1,0	0,007
	AU-07 (0,007)	0	2	1	0,5	0,004
Skor Akhir AU = (0,008 + 0,007 + 0,004 + 0,004 + 0,007 + 0,007 + 0,004)						0,040
Presentase AU = (0,040/0,05) x 100%						79,00%

3.3.7 *Governance* (GV)

Aspek *Governance* dievaluasi untuk menilai sejauh mana mekanisme tata kelola diterapkan guna menjamin konsistensi, kepatuhan, serta pengendalian dalam pengembangan dan pengoperasian layanan. Hasil penilaian ditunjukkan pada Tabel berikut..

Tabel 14. Hasil Scorecard Kriteria GV

Kriteria	Sub-Kriteria	R1	R2	Rata-Rata	Normalisasi	Kontribusi Bobot
GV (0,03)	GV-01 (0,01)	1	2	1,50	0,75	0,008
	GV-02 (0,01)	un-scored	1	1	0,50	0,005
	GV-03 (0,01)	un-scored	2	2	1,00	0,010
Skor Akhir GV = (0,008 + 0,005 + 0,004 + 0,020)						0,023
Presentase GV = (0,023/0,03) x 100%						75,00%

Setelah dilakukan evaluasi pada masing-masing kriteria dalam rancangan arsitektur microservices, langkah selanjutnya adalah mengintegrasikan seluruh hasil penilaian ke dalam skor akhir. Rekapitulasi ini bertujuan untuk memberikan gambaran menyeluruh mengenai tingkat pemenuhan praktik arsitektur berdasarkan bobot dan kontribusi masing-masing kriteria. Ringkasan skor keseluruhan *Microservices Scorecard* ditunjukkan pada Tabel 15.

Tabel 15. Hasil Scorecard Keseluruhan Kriteria

Praktik Area (Kriteria)	Bobot Kriteria	Kontribusi Bobot	Persentase (%)
GR	0,32	0,256	80
IN	0,26	0,244	93,75
SA	0,17	0,139	81,47
AL	0,09	0,090	100
OW	0,08	0,027	33,11
AU	0,05	0,040	79
GV	0,03	0,023	75
Bobot Keseluruhan	0,817		
Persentase Bobot Keseluruhan	0,817 x 100% = 81,7%		

Evaluasi rancangan arsitektur microservices menunjukkan tingkat pemenuhan 81,7%, termasuk kategori *high readiness*. Skor tertinggi diperoleh pada *Alignment* (100%), diikuti *Independence* (93,75%), *Service Architecture* (81,47%), dan *Granularity and Reuse* (80%). *Automation* (79%) dan *Governance* (75%) cukup baik, sementara *Ownership* (33,11%) perlu perhatian khusus untuk memperkuat kepemilikan layanan.

KESIMPULAN

Transformasi arsitektur monolitik menjadi microservices pada SIKAD Itenas berhasil dirancang menggunakan Microservices Migration Pattern, memecah sistem menjadi layanan independen seperti manajemen mahasiswa, registrasi, dan pembayaran, sehingga mengatasi keterbatasan skalabilitas, fleksibilitas, dan integrasi sistem lama.

Evaluasi menggunakan Microservices Scorecard menunjukkan skor kesiapan 81,7% (high readiness), dengan kontribusi terbesar dari Granularity and Reuse (80%) dan Independence (93,75%), sedangkan Ownership (33,11%) memerlukan perhatian lebih dalam pengelolaan layanan. Pendekatan migrasi bertahap—melalui API Gateway, service discovery, dan load balancing—menunjukkan rancangan yang scalable, resilient, dan maintainable. Penelitian ini menyediakan kerangka terstruktur bagi migrasi microservices yang dapat dijadikan acuan institusi pendidikan tinggi atau organisasi lain. Penelitian selanjutnya disarankan menguji rancangan melalui implementasi nyata untuk mengevaluasi performa, keandalan, dan keamanan.

DAFTAR PUSTAKA

[1] F. Ponce, G. Marquez, and H. Astudillo, "Migrating from monolithic architecture to microservices: A Rapid Review," *Proc. - Int. Conf. Chil. Comput. Sci. Soc. SCCC*, 2019, doi: 10.1109/SCCC49216.2019.8966423.

[2] M. Waseem, P. Liang, M. Shahin, A. Di Salle, and G. Márquez, "Design, monitoring, and testing of microservices systems: The practitioners' perspective," *J. Syst. Softw.*, vol. 182, 2021, doi: 10.1016/j.jss.2021.111061.

[3] N. Dragoni *et al.*, "Microservices : Yesterday , Today , and Tomorrow," pp. 195–196.

[4] A. Balalaie, A. Heydarnoori, P. Jamshidi, D. A. Tamburri, and T. Lynn, "Microservices migration patterns," *Softw. - Pract. Exp.*, vol. 48, no. 11, pp. 2019–2042, 2018, doi: 10.1002/spe.2608.

[5] S. Newman, *Building Microservices_ Designing Fine-Grained Systems-O_Reilly Media (2015)*. O'Reilly Media, 2015.

[6] H. Suryotrisongko, "Arsitektur Microservice untuk Resiliensi Sistem Informasi," *Sisfo*, vol. 06, no. 02, pp. 231–246, 2017, doi: 10.24089/j.sisfo.2017.01.006.

[7] R. Mufrizal and D. Indarti, "Refactoring Arsitektur Microservice Pada Aplikasi Absensi PT. Graha Usaha Teknik," *J. Nas. Teknol. dan Sist. Inf.*, vol. 5, no. 1, pp. 57–68, 2019, doi: 10.25077/teknosi.v5i1.2019.57-68.

[8] U. Syarif and Pizaini, "Penerapan Event-Driven Microservices Pada APlikasi Layanan Penerimaan Peserta Didik Baru," *JIPI (Jurnal Ilm. Penelit. dan Pembelajaran Inform.*, vol. 07, no. September, pp. 745–756, 2022, doi: https://doi.org/10.29100/jipi.v7i3.3067.g1312.

[9] M. Arief and A. U. Chaniago, "Penerapan Arsitektur Microservices Pada Web E-Commerce Menggunakan Metode GRPC," *Cloud J. Netw. Syst. Secur. Syst.*, vol. 2, no. 1, pp. 1–9, 2025, doi: https://doi.org/10.36040/JATI.V6I2.5412.

[10] W. Sismadi, B. Agung Martono, Y. Susanto, and A. Muzaeni, "Implementasi Arsitektur Microservices Pada Web Aplikasi Penerimaan Mahasiswa Baru," *EDUTECH J. Inov. Pendidik. Berbantuan Teknol.*, vol. 4, no. 2, pp. 105–114, 2024, doi: https://doi.org/10.51878/edutech.v4i2.3062.

[11] M. Lecrivain, H. Barry, D. Tamzalit, and H. Sahraoui, "MONO2REST: Identifying and Exposing

Microservices: a Reusable RESTification Approach," *Proc. - 2025 IEEE/ACM 22nd Int. Conf. Softw. Syst. Reuse, ICSR 2025*, pp. 33–43, 2025, doi: 10.1109/ICSR66718.2025.00010.

[12] L. M. Alchuluq and F. Nurzaman, "Analisis Pada Arsitektur Microservice Untuk Layanan Bisnis Toko Online," *Tekinfo J. Bid. Tek. Ind. dan Tek. Inform.*, vol. 22, no. 2, pp. 61–68, 2021, doi: 10.37817/tekinfo.v22i2.1761.

[13] M. G. Amri, T. Raharjo, A. N. Fitriani, and N. R. P. Hutasuhut, "Critical Success Factors of Microservices Architecture Implementation in the Information System Project," *Int. J. Adv. Comput. Sci. Appl.*, vol. 15, no. 10, pp. 623–633, 2024, doi: 10.14569/IJACSA.2024.0151064.

[14] T. Harputlugil, "Analytic Hierarchy Process (AHP) As an Assessment Approach for Architectural Design: Case Study of Architectural Design Studio," *Iconarp Int. J. Archit. Plan.*, vol. 6, no. 2, pp. 217–245, 2018, doi: 10.15320/iconarp.2018.53.