JURTI, Vol.9 No.2, Juni 2025, ISSN: 2579-8790

Analisis *Model* Arsitektur *Microservices* pada Sistem Informasi Kuliah Kerja Nyata UMM

Muhammad Ibadurrahman Arrasyid Supriyanto*[1], Chalsi Mala Sari[2], Mochamad Gaharu Dida Devedo[3], Ardhan Ismail[4]
[1,2,3,4]Fakultas Teknik,Universitas Mulawarman, Indonesia
e-mail: ***[1]ibadurrahman@ft.unmul.ac.id**, [2]chalsimalasari@ft.unmul.ac.id,
[3]didadevedo@ft.unmul.ac.id, [4]ardhanismail@ft.unmul.ac.id

Abstrak

Berkembangnya Universitas Muhammadiyah Malang menuntut Sistem Informasi KKN beradaptasi terhadap perubahan proses bisnis manajerial. Penelitian ini menganalisis model arsitektur microservices menggunakan pendekatan Domain Driven Design untuk mengidentifikasi Bounded Context dan kandidat microservices. Hasil analisis menghasilkan 13 microservices yang divalidasi berdasarkan prinsip Single Responsibility, Small Size, Independence, dan Data Consistency dengan tingkat keberhasilan 100%. Implementasi menggunakan Kubernetes menunjukkan arsitektur microservices mampu meningkatkan fleksibilitas dan adaptabilitas sistem terhadap perubahan kebutuhan dibandingkan arsitektur monolitik sebelumnya.

Kata kunci*— Microservices, Arsitektur, Service, KKN, Sistem Informasi*

1. PENDAHULUAN

Perkembangan seputar teknologi di bidang bisnis yang semakin cepat membuat beberapa tren baru dalam dunia *Software* Architect. Kebanyakan Sistem informasi yang dibangun saat ini dibangun menggunakan arsitektur monolitik, yaitu suatu aplikasi perangkat lunak yang terbungkus dalam modul yang tidak independen [1]. Menurut peneliti pada [2] monolitik disebut sebagai *dependency hell*, itu disebabkan karena aplikasi tidak konsisten ketika perpusatakaannya diperbarui. Universitas Muhammadiyah Malang (UMM) melalui Direktorat Penelitian dan Pengabdian Kepada Masyarakat memanfaatkan teknologi informasi untuk mendukung tujuan utama Kuliah Kerja Nyata (KKN) yaitu memberikan pengalaman secara langsung baik fisik maupun mental kepada mahasiswa untuk berkontribusi langsung bersama masyarakat secara interdisipliner tanpa terkotak-kotak oleh ilmu masing-masing.

Sistem ini berisikan informasi mengenai laporan kegiatan, informasi kelompok, informasi pembimbing, informasi mengenai lokasi tempat KKN dan kegiatan KKN lainnya untuk meningkatkan pelayanan dalam bidang pengabdian mahasiswa akan lebih cepat. Sistem Informasi KKN ini dikembangkan menggunakan arsitektur monolitik dan memiliki beberapa modul diantaranya modul mahasiswa, modul kelompok KKN, modul TPL (Tim Pembimbing Lapangan), modul DPL (Dosen Pembimbing Lapangan), modul lokasi, dan modul laporan KKN.

Berkembangnya kampus Universitas Muhammadiyah Malang disertai dengan perubahan beberapa proses bisnis di tingkat manajerial. Perubahan proses bisnis ini membuat Sistem Informasi KKN harus bisa beradaptasi terhadap perubahan tersebut. Perubahan tersebut tidak hanya berdampak pada sisi pengelolaan kode program tetapi juga pada arsitektur sistem yang digunakan sehingga penting bagi *developer* untuk mempertimbangkan beberapa macam alternatif arsitektur sistem informasi lain agar pengelolaan sistem informasi bisa lebih adaptif terhadap perubahan kebutuhan (*requirement changes*).

Arsitektur *microservices* merupakan salah satu alternatif arsitektur yang mampu

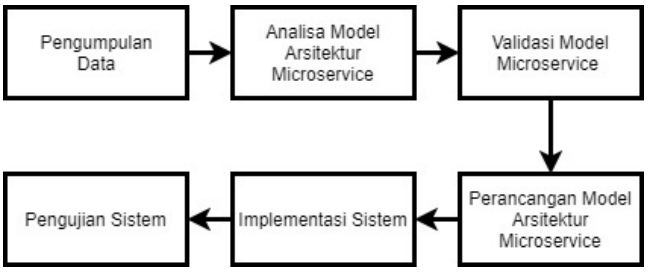
JURTI, Vol.9 No.2, Juni 2025, ISSN: 2579-8790

beradaptasi terhadap perubahan kebutuhan secara terukur dan fleksibel. Menurut peneliti [3]arsitektur *microservices* akan memecah masalah besar menjadi beberapa bagian kecil yang digabungkan ke dalam satu *service*, di mana setiap *service* memiliki tanggung jawabnya sendiri. Pada prinsipnya [4]arsitektur *microservices* membantu manajer proyek dan *developer* untuk menyediakan pedoman dan implementasi aplikasi. Sumber lain menyebutkan arsitektur *microservices* dapat mengurangi biaya infrastruktur sebesar 70% dari anggaran [5].

Domain Driven Design (DDD) dipilih sebagai pendekatan dalam proses identifikasi service pada penelitian ini karena metode tersebut memungkinkan pemisahan domain bisnis secara jelas melalui konsep Bounded Context[6], [7]. Pendekatan DDD terbukti efektif dalam mengidentifikasi batasan antarservice berdasarkan konteks bisnis yang spesifik, sehingga menghasilkan arsitektur *microservices* yang memiliki tanggung jawab tunggal dan bersifat independen [8]. Berdasarkan hal tersebut, penelitian ini berupaya mengidentifikasi model *microservices* pada Sistem Informasi KKN Universitas Muhammadiyah Malang dengan menerapkan pendekatan Domain Driven Design.

2. METODE PENELITIAN

Pada bab ini penulis akan memaparkan metode penelitian yang digunakan dalam penyusunan sebagai berikut :



Gambar 1. Tahapan Penelitian

Gambar 1 memperlihatkan alur tahapan penelitian yang digunakan dalam proses pengembangan model arsitektur *microservices* pada Sistem Informasi Kuliah Kerja Nyata (KKN) Universitas Muhammadiyah Malang. Penelitian ini diawali dengan tahap pengumpulan data, yang bertujuan memperoleh informasi terkait kondisi sistem yang sedang berjalan serta kebutuhan fungsional yang mendasari pengembangan arsitektur baru. Data yang terkumpul kemudian dianalisis pada tahap analisis model arsitektur *microservices* untuk mengidentifikasi batasan domain, komponen utama, serta relasi antar layanan dalam sistem. Hasil analisis tersebut selanjutnya melalui proses validasi model *microservices* guna memastikan kesesuaian rancangan dengan kebutuhan sistem dan prinsip-prinsip arsitektur *microservices*. Setelah model tervalidasi, dilakukan perancangan model arsitektur *microservices* sebagai bentuk penerapan dari hasil analisis dan validasi sebelumnya. Tahap berikutnya yaitu implementasi sistem, di mana rancangan arsitektur diadaptasikan ke dalam lingkungan pengembangan yang sesungguhnya. Proses penelitian diakhiri dengan pengujian sistem untuk menilai kinerja, fungsionalitas, serta keandalan sistem yang telah diimplementasikan.

2.1 Pengumpulan Data

Metode pengumpulan data dalam penelitian ini bertujuan untuk memperoleh informasi yang relevan terkait struktur dan arsitektur Sistem Informasi KKN Universitas Muhammadiyah Malang. Dua metode utama yang digunakan adalah observasi langsung dan dokumentasi. Melalui observasi, peneliti melakukan pengamatan terhadap sistem yang dikelola oleh Divisi Infokom untuk memahami bagaimana sistem berjalan serta mengidentifikasi model arsitektur yang

JURTI, Vol.9 No.2, Juni 2025, ISSN: 2579-8790

digunakan. Hasil observasi menunjukkan bahwa sistem informasi KKN dibangun menggunakan *framework* CodeIgniter dengan penerapan pola arsitektur Model-View-Controller (MVC). Adapun data pendukung lain yang digunakan adalah data yang digunakan berasal dari penelitian [9] dan dokumentasi Sistem Informasi KKN. Secara umum modul sistem terbagi menjadi beberapa bagian seperti terlihat pada Tabel 1.

Tabel 1. Daftar Subsistem dan Modul *Existing*

No	Daftar Subsistem	Modul	Operasi
1	BAA (Biro Administrasi Akademik)	Mahasiswa	Perubahan Data Mahasiswa
		Akademik	Validasi Status Akademik
2	BAU (Biro Administrasi Umum)	Keuangan	Validasi Status Keuangan
3	DPPM (Direktorat Penelitian dan Pengabdian kepada Masyarakat)	Kelompok KKN	Pendaftaran Kelompok
			Perubahan Data Kelompok
		TPL	Kelola Data TPL
		DPL	Kelola Data DPL
		Lokasi	Kelola Data Lokasi
		Penilaian	Penilaian Absensi
			Penilaian Program Kerja
			Penilaian Laporan
		Laporan	Laporan Catatan Kegiatan Harian
			Laporan Akhir
			Download Laporan
			Persetujuan Laporan

Tabel 1 menunjukkan hasil proses pengumpulan data menggunakan Teknik dokumentasi menghasilkan modul dan subsistem yang digunakan oleh sistem *existing*. Modul ini mewakili dari proses bisnis sistem informasi yang telah dibangun. Modul ini terdiri dari 9 buah modul, 15 operasi dan 3 subsistem.

2.2 Analisa Model Arsitektur Microservices

Pembangunan aplikasi berbasis domain memerlukan pemahaman komprehensif terhadap konteks bisnis. Penelitian ini mengadopsi pendekatan *Domain-Driven Design* (DDD) yang dikemukakan oleh Evans [10] untuk memahami kebutuhan dan pemodelan domain melalui model berorientasi objek yang mengintegrasikan perilaku dan data.

2.2.1 Event storming

Event storming merupakan teknik desain kolaboratif yang melibatkan *domain expert* dan *developer*, dengan fokus pada pemodelan proses bisnis[11], [12], [13], [14]. Implementasi *event storming* dilakukan menggunakan tools Miro pada 30 April 2019 dengan partisipan: *domain expert* dari DPPM, *user representative* seorang koordinator KKN 2018/2019, dan *solution architect* sebagai fasilitator. Proses *event storming* terdiri dari tiga tahapan utama menggunakan visual management dengan sticky notes berwarna:

a. Eksplorasi Domain Events (Orange): Identifikasi kejadian bisnis yang merepresentasikan proses yang telah berjalan
b. Pembuatan Commands (Blue): Definisi perintah atau aksi yang menghasilkan domain events, mencakup interaksi user dan proses sistem otomatis
c. Penghubungan Entitas (Light Yellow): Asosiasi entitas data dengan commands dan domain events yang merepresentasikan persistensi data

JURTI, Vol.9 No.2, Juni 2025, ISSN: 2579-8790

Gambar 2. Proses *Event Storming* Sistem Existing Bersama Domain Expert dan Developer

Proses *event storming* pada Gambar 2 menghasilkan identifikasi enam *Bounded Context* utama yang merepresentasikan domain bisnis Sistem Informasi KKN, yaitu: Subsistem BAA, Mahasiswa, Lokasi, DPPM, Kelompok, dan Laporan. Setiap *Bounded Context* memiliki komponen domain *events* (sticky notes orange), *commands* (sticky notes biru), dan *entities* (sticky notes kuning terang) yang saling berelasi membentuk alur proses *sistem existing*.

2.2.2 Penentuan *Bounded Context*

Metode DDD menggunakan konsep *Bounded Context* sebagai batas kontekstual semantik, di mana setiap komponen model software memiliki makna spesifik dalam konteks tertentu. *Bounded Context* memberikan pemahaman eksplisit kepada tim pengembang mengenai komponen yang harus konsisten dan komponen yang dapat berkembang secara independen. Model software dalam *Bounded Context* merefleksikan *Ubiquitous Language*—terminologi standar yang dipahami oleh seluruh stakeholder pengembangan software untuk memastikan konsistensi komunikasi. Tabel 2 merupakan domain existing Sistem Informasi KKN.

Tabel 2. Domain Kuliah Kerja Nyata

Bounded Context	Keterangan
Kelompok	Berisi domain model yang berkaitan dengan daftar kelompok kkn, pengelompokkan otomatis dan no urut kelompok sedangkan pada no urut kelompok terdapat tim pembimbing lapangan, divisi dan dosen pembimbing lapangan kuliah kerja nyata
Laporan	Berisi domain model yang berkaitan dengan nilai kelompok, catatan kegiatan, review, revisi dan nilai catatan kegiatan kuliah kerja nyata
Lokasi	Berisi domain model yang berkaitan dengan lokasi seperti lokasi yang tersedia, sudah diisi dan pin lokasi kuliah kerja nyata
DPPM	Berisi domain model yang berkaitan dengan upload panduan pengumuman, lokasi, kelompok, tim pembimbing lapangan dan laporan kuliah kerja nyata
Mahasiswa	Berisi domain model yang berkaitan dengan biodata , keuangan dan akademik mahasiswa
Dosen Pembimbing Lapangan	Berisi domain model yang berkaitan dengan biodata dan akademik Dosen Pembimbing Lapangan
Keuangan	Eksternal sistem dari Biro Administrasi Keuangan UMM
Administrasi	Eksternal sistem dari Biro Administrasi Akademik UMM
Koordinat	Eksternal sistem dari Google Maps

2.2.3 Context Map

Berdasarkan analisis peta Sistem Informasi KKN, ditemukan enam bagian utama sistem internal (Kelompok, Laporan, Lokasi, DPPM, Mahasiswa, dan Dosen Pembimbing Lapangan) serta tiga sistem eksternal (Keuangan, Administrasi, dan *Google Maps*) yang saling terhubung membentuk satu kesatuan sistem. Bagian Kelompok menjadi pusat utama karena terhubung dengan lima bagian lainnya, berperan penting dalam menjalankan proses bisnis inti yaitu algoritma pembagian kelompok secara otomatis. Hubungan antara Kelompok dan Laporan

JURTI, Vol.9 No.2, Juni 2025, ISSN: 2579-8790

bersifat saling bergantung sehingga membutuhkan sinkronisasi data yang ketat, sementara hubungan DPPM-Kelompok dan Kelompok-Dosen Pembimbing Lapangan berbagi model data yang sama dan memerlukan kesepakatan jelas agar tidak terjadi kesalahan saat ada perubahan.

Untuk koneksi dengan sistem eksternal, digunakan pola hubungan satu arah di mana bagian Mahasiswa mengikuti format data dari Sistem Keuangan dan Administrasi, serta bagian Lokasi mengikuti format dari *Google Maps API*. Pola ini dipilih karena sistem eksternal merupakan sistem lama yang tidak bisa diubah, sehingga sistem internal harus menyesuaikan diri. Untuk melindungi sistem internal, ditambahkan lapisan pelindung *Anti-Corruption Layer* yang berfungsi menerjemahkan bahasa/format data dari sistem lama ke format sistem baru. Hasil analisis menunjukkan bahwa proses migrasi ke arsitektur *microservices* harus dimulai dari membangun koneksi dengan sistem eksternal terlebih dahulu, kemudian dilanjutkan dengan membangun bagian-bagian yang tidak terlalu bergantung pada bagian lain (Laporan, Dosen Pembimbing Lapangan), sedangkan bagian Kelompok yang memiliki banyak ketergantungan memerlukan pengujian menyeluruh dan pendekatan pengembangan berbasis kontrak untuk memastikan sistem tetap berjalan dengan baik.

2.2.4 Identifikasi Aggregate, Entity dan Value Object

Pada tahapan ini, aggregate dikelompokkan berdasarkan business functionality untuk memfasilitasi separation of concerns dan maintainability. *Bounded Context* Mahasiswa menerapkan multi-aggregate pattern dengan membagi aggregate menjadi Keuangan dan Administrasi. Strategi pemisahan ini dipilih karena kedua domain tersebut berasal dari sistem eksternal yang berbeda berdasarkan hasil context mapping, sehingga memerlukan implementasi *Anti-Corruption Layer* (ACL).

Implementasi ACL mengharuskan entitas dan value object dari sistem eksternal untuk tidak digabungkan secara langsung ke dalam satu aggregate tunggal. Hal ini disebabkan oleh perbedaan business functionality dan ownership dari kedua sistem upstream (Keuangan dari BAU dan Administrasi dari BAA). Oleh karena itu, aggregate harus disegregasi berdasarkan functional *boundaries* menjadi aggregate Administrasi dan aggregate Keuangan untuk menjaga isolation dan *independent evolution* dari masing-masing integration point.

Tabel 3. Identifikasi Aggregate, Entity dan Value Object

Domain	Aggregate	Entity	Value Object	Tipe Data
Kelompok	Kelompok	Kelompok	No Urut Kelompok	Numeric
Mahasiswa	Keuangan	Keuangan	Status Tanggungan Pembayaran	Boolean
	Administrasi	Akademik	Jumlah SKS telah ditempuh	Numeric
		Kemahasiswaan	Nomor Induk Mahasiswa	Numeric
Lokasi	Lokasi	Koordinat	Langtitude,Longtitude	Numeric
Dosen Pembimbing Lapangan	Administrasi	Akademik	Jumlah SKS di Ajar	Numeric
		Kepegawaian	Nomor Induk Dosen	Numeric
DPPM	Informasi	Panduan	Panduan	File Extension pdf
		Pengumuman	Pengumuman	Text
	Tim Pembimbing Lapangan	Tim Pembimbing Lapangan	Nomor Induk Pembimbing Lapangan	Numeric
Laporan	Laporan	Laporan	Laporan Kelompok	File Extension pdf
		Catatan Kegiatan	Program Kerja	Text
		Nilai	Nilai Program Kerja	Numeric

JURTI, Vol.9 No.2, Juni 2025, ISSN: 2579-8790

2.2.5 Identifikasi Kandidat Microservices

Identifikasi *microservices* merupakan tahapan transformasi dari *Bounded Context* dan aggregate yang telah divalidasi menjadi *deployable service units* dengan API *endpoints* yang terdefinisi secara eksplisit. Proses identifikasi pada Tabel 4 menghasilkan 13 *microservices*s yang memetakan domain model ke dalam *service-oriented architecture* dengan RESTful API sebagai *communication protocol*.

Tabel 4. Kandidat *Microservices*

Bounded Context	*Microservices*	URL Endpoint	HTTP Methods	Deskripsi Fungsional
Kelompok	Kelompok	api/kelompok	GET, POST, PUT, DELETE	Mengelola informasi detail kelompok KKN termasuk business logic pengelompokan otomatis
Mahasiswa	Mahasiswa	api/mahasiswa	GET, POST, PUT, DELETE	Mengelola biodata mahasiswa sebagai master data
Mahasiswa	Mahasiswa-Keuangan	api/mahasiswa/keuangan	GET, POST, PUT, DELETE	Menyediakan informasi status keuangan mahasiswa dengan integrasi ke sistem BAU
Mahasiswa	Mahasiswa-Akademik	api/mahasiswa/akademik	GET, POST, PUT, DELETE	Menyediakan informasi jumlah SKS yang telah ditempuh mahasiswa
Lokasi	Lokasi	api/lokasi	GET, POST, PUT, DELETE	Mengelola informasi lokasi KKN dengan integrasi Google Maps API sebagai ACL
Dosen Pembimbing Lapangan	Dosen	api/dosen	GET, POST, PUT, DELETE	Mengelola biodata dosen pembimbing lapangan termasuk Nomor Induk Dosen
Dosen Pembimbing Lapangan	Dosen-Akademik	api/dosen/akademik	GET, POST, PUT, DELETE	Menyediakan informasi jumlah SKS yang sedang diampu oleh dosen pembimbing
DPPM	DPPM-Panduan	api/dppm/panduan	GET, POST, PUT, DELETE	Mengelola dokumen panduan KKN dalam format PDF
DPPM	DPPM-Pengumuman	api/dppm/pengumuman	GET, POST, PUT, DELETE	Mengelola dan broadcasting pengumuman KKN kepada mahasiswa
DPPM	DPPM-TPL	api/dppm/tpl	GET, POST, PUT, DELETE	Mengelola data Tim Pembimbing Lapangan
Laporan	Laporan	api/laporan	GET, POST, PUT, DELETE	Mengelola laporan kelompok final KKN dalam format PDF
Laporan	Laporan-Kegiatan	api/laporan/kegiatan	GET, POST, PUT, DELETE	Mengelola catatan kegiatan harian yang dilakukan kelompok selama proses KKN
Laporan	Laporan-Nilai	api/laporan/nilai	GET, POST, PUT, DELETE	Mengelola penilaian kelompok KKN oleh dosen pembimbing dan DPPM

Tabel 4 menampilkan hubungan langsung antara setiap bagian sistem (Bounded Context) dengan *microservicess* (*microservices*) beserta alamat URL-nya yang mengikuti standar RESTful. Seluruh *microservicess* mendukung operasi CRUD secara lengkap (mengambil, menambah, memperbarui, dan menghapus data) untuk memudahkan pengelolaan data dari awal hingga akhir.

3. HASIL DAN PEMBAHASAN

3.1 Validasi Model Microservices

Validasi model *microservices* merupakan tahapan verifikasi untuk memastikan bahwa setiap kandidat *microservices* yang telah diidentifikasi memenuhi prinsip-prinsip fundamental arsitektur *microservices* sebelum dilanjutkan ke tahap implementasi. Proses validasi menggunakan empat kriteria evaluasi yang diadopsi dari *Microsoft Azure Architecture Guidelines* [15] sebagai industry best practices untuk *microservices* design assessment. Empat kriteria validasi yang diterapkan adalah:

JURTI, Vol.9 No.2, Juni 2025, ISSN: 2579-8790

1) *Single Responsibility Principle* (SRP): Setiap *microservices* harus memiliki satu tanggung jawab yang terdefinisi dengan jelas dan terfokus pada satu *business capability*.
2) *Small Size Principle*: Setiap *microservices* harus cukup kecil sehingga dapat dibangun, dipahami, dan dimaintain oleh small development team.
3) *Independence Principle*: Setiap *microservices* harus dapat dikembangkan, di-deploy, dan di-scale secara independen tanpa koordinasi dengan tim lain.
4) *Data Consistency Principle*: Boundaries dari setiap *microservices* tidak boleh menimbulkan masalah dengan konsistensi atau integritas data.

Tabel 5. Validasi Independensi *Service*

Microservices	Dedicated Storage	Independent Deploy	Technology Choice Freedom	API Versioning	Independence Score
Kelompok	(MySQL Schema)	(Kubernetes Pod)	(Node.js/Express)	(Semantic Versioning)	4/4
Mahasiswa	(MySQL Schema)	(Kubernetes Pod)	(Node.js/Express)	(Semantic Versioning)	4/4
Mahasiswa-Akademik	(MySQL Schema)	(Kubernetes Pod)	(Node.js/Express)	(Semantic Versioning)	4/4
Mahasiswa-Keuangan	(MySQL Schema)	(Kubernetes Pod)	(Node.js/Express)	(Semantic Versioning)	4/4
Lokasi	(MySQL Schema)	(Kubernetes Pod)	(Node.js/Express)	(Semantic Versioning)	4/4
Dosen	(MySQL Schema)	(Kubernetes Pod)	(Node.js/Express)	(Semantic Versioning)	4/4
Dosen-Akademik	(MySQL Schema)	(Kubernetes Pod)	(Node.js/Express)	(Semantic Versioning)	4/4
DPPM-Panduan	(MySQL Schema)	(Kubernetes Pod)	(Node.js/Express)	(Semantic Versioning)	4/4
DPPM-Pengumuman	(MySQL Schema)	(Kubernetes Pod)	(Node.js/Express)	(Semantic Versioning)	4/4
DPPM-TPL	(MySQL Schema)	(Kubernetes Pod)	(Node.js/Express)	(Semantic Versioning)	4/4
Laporan	(MySQL Schema)	(Kubernetes Pod)	(Node.js/Express)	(Semantic Versioning)	4/4
Laporan-Kegiatan	(MySQL Schema)	(Kubernetes Pod)	(Node.js/Express)	(Semantic Versioning)	4/4
Laporan-Nilai	(MySQL Schema)	(Kubernetes Pod)	(Node.js/Express)	(Semantic Versioning)	4/4

Tabel 5 menunjukkan bahwa rancangan telah menerapkan empat hal: penyimpanan data tersendiri untuk tiap layanan, sistem berbasis kontainer, kebebasan memilih teknologi, dan pengelolaan versi API secara mandiri. Skor sempurna 4,0 dari 4,0 menandakan bahwa setiap layanan dapat beroperasi secara mandiri tanpa bergantung pada layanan lain.

Tabel 6. Validasi Data Konsistensi

Microservices	Aggregate Alignment	Transaction Boundary	Inter-Service Refs	Consistency Model	Consistency Score
Kelompok	(1:1 mapping)	(Single aggregate)	ID-based	Eventual	4/4
Mahasiswa	(1:1 mapping)	(Single aggregate)	ID-based	Eventual	4/4
Mahasiswa-Akademik	(1:1 mapping)	(Single aggregate)	ID-based	Eventual	4/4
Mahasiswa-Keuangan	(1:1 mapping)	(Single aggregate)	ID-based	Eventual	4/4
Lokasi	(1:1 mapping)	(Single aggregate)	ID-based	Eventual	4/4
Dosen	(1:1 mapping)	(Single aggregate)	ID-based	Eventual	4/4
Dosen-Akademik	(1:1 mapping)	(Single aggregate)	ID-based	Eventual	4/4
DPPM-Panduan	(1:1 mapping)	(Single aggregate)	ID-based	Eventual	4/4
DPPM-Pengumuman	(1:1 mapping)	(Single aggregate)	ID-based	Eventual	4/4
DPPM-TPL	(1:1 mapping)	(Single aggregate)	ID-based	Eventual	4/4
Laporan	(1:1 mapping)	(Single aggregate)	ID-based	Eventual	4/4
Laporan-Kegiatan	(1:1 mapping)	(Single aggregate)	ID-based	Eventual	4/4
Laporan-Nilai	(1:1 mapping)	(Single aggregate)	ID-based	Eventual	4/4

JURTI, Vol.9 No.2, Juni 2025, ISSN: 2579-8790

Tabel 6 menunjukkan bahwa batasan transaksi telah sesuai dengan batasan kumpulan data. Penggunaan nomor ID sebagai penghubung dan mekanisme konsistensi akhir memastikan data tetap utuh meskipun sistem tersebar di berbagai lokasi.

Tabel 7. Hasil Validasi *Microservices*

Microservices	Single Responsibility	Small Size	Independence	Data Consistency	Validation Status
Kelompok	✓	✓	✓	✓	PASSED
Mahasiswa	✓	✓	✓	✓	PASSED
Mahasiswa-Akademik	✓	✓	✓	✓	PASSED
Mahasiswa-Keuangan	✓	✓	✓	✓	PASSED
Lokasi	✓	✓	✓	✓	PASSED
Dosen	✓	✓	✓	✓	PASSED
Dosen-Akademik	✓	✓	✓	✓	PASSED
DPPM-Panduan	✓	✓	✓	✓	PASSED
DPPM-Pengumuman	✓	✓	✓	✓	PASSED
DPPM-TPL	✓	✓	✓	✓	PASSED
Laporan	✓	✓	✓	✓	PASSED
Laporan-Kegiatan	✓	✓	✓	✓	PASSED
Laporan-Nilai	✓	✓	✓	✓	PASSED

Hasil validasi pada Tabel 7 menunjukkan bahwa seluruh 13 kandidat *microservices* memenuhi empat kriteria dasar dengan tingkat keberhasilan 100%. Tabel 5 membuktikan bahwa strategi pemecahan sistem berdasarkan Bounded Context dan batasan agregat dari Domain-Driven Design menghasilkan arsitektur *microservices* yang sejalan dengan proses bisnis. Validasi Prinsip Tanggung Jawab Tunggal menunjukkan bahwa setiap *microservices* memiliki fungsi bisnis yang terfokus tanpa tumpang tindih tanggung jawab. Keselarasan dengan batasan agregat dari desain taktis DDD memastikan logika domain yang padu dengan pemisahan fungsi yang jelas. Validasi Prinsip Ukuran Kecil membuktikan bahwa tingkat kerumitan setiap *microservices* berada dalam rentang yang dapat dikelola oleh tim pengembangan kecil. Ukuran basis kode yang sedang (kurang dari 10.000 baris kode per layanan) mempermudah siklus pengembangan yang cepat.

3.2 Perancangan Model Arsitektur Microservices

Arsitektur penerapan Sistem Informasi KKN diimplementasikan menggunakan Kubernetes pada Node Minikube. Arsitektur ini terdiri dari tiga lapisan utama, yaitu lapisan eksternal yang terhubung dengan internet, lapisan kluster Kubernetes yang mengelola aplikasi, dan lapisan penyimpanan data menggunakan MySQL. Alur komunikasi dimulai dari internet yang terhubung melalui komponen Proksi sebagai pintu masuk ke dalam sistem. Permintaan dari internet diteruskan ke NGINX Ingress Load Balancer yang beroperasi pada Port 80 sebagai titik masuk tunggal yang menerima seluruh lalu lintas eksternal dan mendistribusikannya ke layanan-layanan internal berdasarkan aturan perutean yang telah dikonfigurasi.

Di dalam Kluster Kubernetes terdapat enam pod *microservices* yang diterapkan, yaitu Layanan Mahasiswa Akademik, Layanan Mahasiswa, Layanan DPPM-Panduan, Layanan DPPM-TPL, Layanan Mahasiswa-Keuangan, dan Layanan Dosen Akademik. Setiap pod memiliki struktur hierarki identik dengan empat lapisan bersarang (pod, *ReplicaSet, Container*, dan *Service Logic*) yang terhubung ke komponen Layanan pada Port 80. Ketika NGINX Ingress menerima permintaan, sistem melakukan perutean ke Layanan yang sesuai berdasarkan jalur URL, kemudian meneruskan permintaan ke salah satu pod menggunakan mekanisme penyeimbangan beban internal.

Pada lapisan penyimpanan data, terdapat komponen Layanan MySQL dengan 13 instans basis data terisolasi yang masing-masing didedikasikan untuk satu layanan mikro tertentu, meliputi Mysql-Lokasi, Mysql-Dpl, Mysql-Laporan, Mysql-Laporan-Nilai, Mysql-Laporan-

JURTI, Vol.9 No.2, Juni 2025, ISSN: 2579-8790

Kegiatan, Mysql-Dppm-Tpl, Mysql-Dppm-Pengumuman, Mysql-Dppm-Panduan, Mysql-Dosen-Akademik, Mysql-Dosen, Mysql-Mahasiswa-Keuangan, dan Mysql-Mahasiswa-Akademik. Layanan MySQL terhubung dengan Persistent Volume Claim berkapasitas 1GB yang memastikan data tidak hilang ketika pod dimulai ulang. Koneksi antara pod layanan mikro dengan basis data dilakukan melalui Layanan MySQL Port 3306.

Alur permintaan menyeluruh berjalan sebagai berikut: permintaan dari internet masuk melalui Proksi, diterima NGINX Ingress pada Port 80, dialihkan ke Layanan yang sesuai, diteruskan ke pod, logika bisnis memproses permintaan dan melakukan kueri ke MySQL jika diperlukan, hasil dikembalikan ke kontainer, respons diformat dan dikembalikan hingga sampai ke klien. Seluruh komponen berada dalam Node Minikube untuk pengembangan dan pengujian, dengan pemisahan visual antara lapisan komputasi (*pod microservices*) dan lapisan penyimpanan (Layanan MySQL) yang menunjukkan pemisahan kepentingan antara lapisan aplikasi tanpa status dan lapisan persistensi berkeadaan.

4. KESIMPULAN

Penelitian ini berhasil mengidentifikasi dan memvalidasi 13 *microservices* dari Sistem Informasi KKN UMM menggunakan pendekatan Domain Driven Design. Proses *Event storming* menghasilkan 6 bounded context utama (Kelompok, Laporan, Lokasi, DPPM, Mahasiswa, dan DPL) yang ditransformasi menjadi kandidat *microservices*. Validasi menggunakan empat kriteria Microsoft Azure Architecture Guidelines menunjukkan tingkat keberhasilan 100% untuk Single Responsibility Principle, Small Size Principle, Independence Principle, dan Data Consistency Principle. Implementasi arsitektur menggunakan Kubernetes dengan NGINX Ingress sebagai API Gateway dan 13 database MySQL terisolasi menghasilkan sistem yang lebih fleksibel dan adaptif terhadap perubahan kebutuhan bisnis dibandingkan arsitektur monolitik sebelumnya.

5. SARAN

Penelitian selanjutnya dapat melakukan evaluasi perbandingan kinerja antara arsitektur *microservices* dengan monolitik menggunakan metrik *response time* dan *throughput*. Perlu juga dilakukan analisis mendalam terhadap kompleksitas manajemen data antar *service* dan implementasi distributed transaction untuk memastikan konsistensi data.

DAFTAR PUSTAKA

[1] G. Liu, B. Huang, Z. Liang, M. Qin, H. Zhou, and Z. Li, "*Microservices*: architecture, container, and challenges," in *2020 IEEE 20th International Conference on Software Quality, Reliability and Security Companion (QRS-C)*, IEEE, Dec. 2020, pp. 629–635. doi: 10.1109/QRS-C51114.2020.00107.

[2] V. Velepucha and P. Flores, "A Survey on *Microservices* Architecture: Principles, Patterns and Migration Challenges," *IEEE Access*, vol. 11, pp. 88339–88358, 2023, doi: 10.1109/ACCESS.2023.3305687.

[3] P. Jamshidi, C. Pahl, N. C. Mendonca, J. Lewis, and S. Tilkov, "*Microservices*: The Journey So Far and Challenges Ahead," *IEEE Softw*, vol. 35, no. 3, pp. 24–35, May 2018, doi: 10.1109/MS.2018.2141039.

[4] S. Hassan and R. Bahsoon, "*Microservices* and Their Design Trade-Offs: A Self-Adaptive Roadmap," in *2016 IEEE International Conference on Services Computing (SCC)*, IEEE, Jun. 2016, pp. 813–818. doi: 10.1109/SCC.2016.113.

[5] C. A. Anjum Era, S. T. Alvi, Md. S. Rana, and S. J. Mitu, "A Comprehensive Study of *Microservices* Architecture in Comparison with SOA and Monolithic Models," in *2025 2nd International Conference on Advanced Innovations in Smart Cities (ICAISC)*, IEEE, Feb. 2025, pp. 1–6. doi: 10.1109/ICAISC64594.2025.10959671.

[6] H. Vural and M. Koyuncu, "Does Domain-Driven Design Lead to Finding the Optimal Modularity of a *Microservices*?," *IEEE Access*, vol. 9, pp. 32721–32733, 2021, doi: 10.1109/ACCESS.2021.3060895.

[7] J. Sangabriel-Alarcón, J. O. Ocharán-Hernández, K. Cortés-Verdín, and X. Limón, "Domain-Driven Design for *Microservices* Architecture Systems Development: A Systematic Mapping Study," in *2023 11th International Conference in Software Engineering Research and Innovation (CONISOFT)*, IEEE, Nov. 2023, pp. 25–34. doi: 10.1109/CONISOFT58849.2023.00014.

[8] C. Zhong *et al.*, "Domain-Driven Design for *Microservices*: An Evidence-Based Investigation," *IEEE Transactions on Software Engineering*, vol. 50, no. 6, pp. 1425–1449, Jun. 2024, doi: 10.1109/TSE.2024.3385835.

[9] L. Setiawan, "Sistem Informasi Manajemen Kuliah Kerja Nyata (Kkn) Menggunakan Model Pengembangan Perangkat Lunak Extreme Programming (Xp) (Studi Kasus: Dppm Universitas Muhammadiyah Malang)," Universitas Muhammadiyah Malang, Malang, 2018.

[10] E. Evans, *Domain-Driven Design: Tackling Complexity in the Heart of Software*. Pearson Education, 2003. [Online]. Available: https://books.google.co.id/books?id=hHBf4YxMnWMC

[11] K. Nandi and K. Dey, "Designing Scalable Multi-Agent AI Systems: Leveraging Domain-Driven Design and *Event storming*," *International Journal of Computer Science and Engineering*, vol. 12, no. 3, pp. 10–16, Mar. 2025, doi: 10.14445/23488387/IJCSE-V12I3P102.

[12] N. Zhou *et al.*, "Container orchestration on HPC systems through Kubernetes," *Journal of Cloud Computing*, vol. 10, no. 1, p. 16, 2021, doi: 10.1186/s13677-021-00231-z.

[13] D. M. Le, D.-H. Dang, and V.-H. Nguyen, "Domain-driven design patterns: A metadata-based approach," in *2016 IEEE RIVF International Conference on Computing & Communication Technologies, Research, Innovation, and Vision for the Future (RIVF)*, IEEE, Nov. 2016, pp. 247–252. doi: 10.1109/RIVF.2016.7800302.

[14] C. Praschl, S. Bauernfeind, C. Leitner, and G. A. Zwettler, "Domain-Driven Design as Model Contract in Full-Stack Development," in *2023 3rd International Conference on Electrical, Computer, Communications and Mechatronics Engineering (ICECCME)*, IEEE, Jul. 2023, pp. 1–6. doi: 10.1109/ICECCME57830.2023.10252654.

[15] D. Rendón, B. Hargreaves, and S. Kong, *Azure Architecture Explained: A comprehensive guide to building effective cloud solutions*. Packt Publishing, 2023. [Online]. Available: https://books.google.co.id/books?id=HfzTEAAAQBAJ