

OPTIMASI ARSITEKTUR MONOLITIK KE *MICROSERVICE*  
DENGAN *EVENT-DRIVEN* PADA APLIKASI *LEARNING*  
*MANAGEMENT SYSTEM (LMS)*

Diajukan Sebagai Syarat Untuk Menyelesaikan  
Pendidikan Program Strata-1 Pada  
Jurusan Teknik Informatika



Oleh :

Harisatul Aulia  
NIM : 09021282025062

**Jurusan Teknik Informatika**  
**FAKULTAS ILMU KOMPUTER UNIVERSITAS SRIWIJAYA**  
**2024**

LEMBAR PENGESAHAN SKRIPSI

OPTIMASI ARSITEKTUR MONOLITIK KE *MICROSERVICE* DENGAN  
*EVENT-DRIVEN* PADA APLIKASI *LEARNING*  
*MANAGEMENT SYSTEM (LMS)*

Oleh :

Harisatul Aulia  
NIM : 09021282025062

Palembang, 30 Juli 2024

Mengetahui  
Ketua Jurusan Teknik Informatika



Dr. Muhammad Fachrullozqi, S.Si., M.T.  
NIP. 198005222008121002

Pembimbing

Mastura Diana Marieska, M.T  
NIP. 198603212018032001

## TANDA LULUS UJIAN KOMPREHENSIF SKRIPSI

Pada hari Jumat tanggal 26 Juli 2024 telah dilaksanakan ujian komprehensif Skripsi oleh jurusan Teknik Informatika Fakultas Ilmu Komputer Universitas Sriwijaya.

Nama : Harisatul Aulia  
NIM : 09021282025062  
Judul : Optimasi Arsitektur Monolitik ke *Microservice* dengan *Event-Driven* pada Aplikasi *Learning Mangement System (LMS)*

dan dinyatakan LULUS.


1. Ketua Penguji

Desty Rodiah, S.Kom., M.T.  
NIP. 198912212020122011



2. Penguji I

Osvari Arsalan, S.Kom., M.T.  
NIP. 198806282018031001



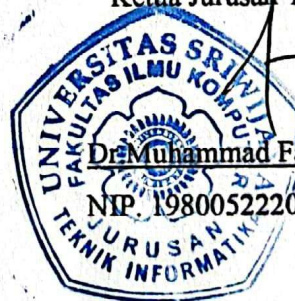
3. Pembimbing I

Mastura Diana Marieska, M.T.  
NIP. 198603212018032001



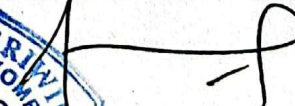
Mengetahui,

Ketua Jurusan Teknik Informatika



Dr. Muhammad Fachrurrozi, S.Si., M.T.

NIP. 198005222008121002



## HALAMAN PERNYATAAN BEBAS PLAGIAT

Yang bertanda tangan di bawah ini:

Nama : Harisatul Aulia  
NIM : 09021282025062  
Judul : Optimasi Arsitektur Monolitik ke *Microservice* dengan  
*Event-Driven* pada Aplikasi *Learning Management System (LMS)*  
Hasil Pengecekan *iThenticate/Turnitin* : 8%

Menyatakan bahwa laporan skripsi saya merupakan hasil karya sendiri dan bukan hasil penjiplakan/plagiat. Apabila ditemukan unsur penjiplakan/plagiat dalam laporan skripsi ini, maka saya bersedia menerima sanksi akademik dari Universitas Sriwijaya sesuai dengan ketentuan yang berlaku.

Demikian, pernyataan ini saya buat dengan sebenarnya dan tidak ada paksaan oleh siapapun.



Palembang, 30 Juli 2024



Harisatul Aulia

NIM 09021282025062

## **MOTTO DAN PERSEMBAHAN**

Motto :

- Beat Yesterday
- People come and go
- If something to good to be true. it probably is
- Fiat sapientia praevallet

Kupersembahkan karya tulis ini kepada :

- Civitas Akademik Indonesia
- Universitas Sriwijaya
- Fakultas Ilmu Komputer
- Dosen Pembimbing  
Akademik dan Skripsi
- Orang tua dan keluarga
- Sahabat dan teman-teman

## ABSTRACT

Monolithic is a classic application architecture that lacks modularity as an important factor of its development. All components of a monolithic application run in a single process. High traffic on one part of the application can cause the entire system to slow down or even fail. Then, application performance can degrade significantly as load increases. In the microservices approach, the application is broken down into a series of separate small services, which can be developed, tested, and deployed independently. This allows the developer to adopt a more modular and responsive approach to change application requirements. Performance improvement is also an aspect that can be gained when adopting a microservice architecture. This research will specifically evaluate the migration process from monolithic to microservice architecture with an event driven approach on Civil Servant Warrior's existing startup LMS application. The LMS features that required for high performance are enrollment, payment and online exam. Theoretically, microservices is able to improve the performance of the application. However, to find out how much performance optimization is produced, the performance of the two systems was examined. In this research, there are two scenarios to consider: low load and high load. Low load is defined as 800 rps and high load is defined as 8000 rps. The use of asynchronous and the ability to distribute workloads more evenly makes microservices have better performance. The increase in response time is 52.24% at low load and an increase in rps of 31.27% at high load.

**Keywords:** Monolithic, Microservice, Event-Driven-Architecture, Event Carried State Transfer, Learning Management System

Palembang, July 30<sup>th</sup> 2024

Approved  
Head of Informatics Engineering Department



*[Signature]*  
Dr. Muhammad Fachrullozki, S.Si., M.T.  
NIP. 198005222008121002

Supervisor

Mastura Diana Marieska, M.T  
NIP. 198603212018032001

## ABSTRAK


Monolitik adalah arsitektur aplikasi klasik yang kurang memperhatikan modularitas sebagai faktor penting dalam pengembangannya. Semua komponen aplikasi pada monolitik berjalan dalam satu proses tunggal. Tingginya *traffic* pada satu bagian aplikasi dapat menyebabkan keseluruhan sistem menjadi lambat atau bahkan mengalami kegagalan. Akhirnya, performa aplikasi dapat menurun secara signifikan saat beban aplikasi meningkat. Pada pendekatan *microservices*, aplikasi dipecah menjadi serangkaian layanan kecil terpisah, yang dapat dikembangkan, diuji, dan dijalankan secara independen. Hal ini memungkinkan tim pengembang untuk mengadopsi pendekatan yang lebih modular dan responsif dalam menghadapi perubahan kebutuhan aplikasi. Peningkatan performa juga menjadi aspek yang bisa didapatkan ketika mengadopsi arsitektur *microservice*. Penelitian ini akan secara khusus mengevaluasi proses migrasi dari arsitektur monolitik ke *microservice* dengan pendekatan *event driven* pada aplikasi LMS milik startup Civil Servant Warrior yang sudah ada sebelumnya. Fitur LMS yang menjadi fokus adalah yang membutuhkan performa tinggi yaitu pendaftaran, pembayaran dan pelaksanaan ujian online. *Microservice* secara teoritis dapat meningkatkan performa pada aplikasi. Namun untuk mengetahui seberapa besar optimasi performa yang dihasilkan, maka dilakukan pengujian performa kedua sistem. Pada penelitian ini, terdapat dua skenario yang perlu dipertimbangkan: *low load* dan *high load*. *Low load* didefinisikan sebanyak 800 rps dan *high load* didefinisikan sebanyak 8000 rps. Penggunaan *asynchronous* dan kemampuan untuk mendistribusikan beban kerja dengan lebih merata membuat *microservice* memiliki performa yang lebih unggul. Peningkatan response time sebesar 52.24% pada *low load* dan peningkatan rps sebesar 31.27% pada *high load*.

**Kata kunci:** Monolitik, *Microservice*, *Event-Driven-Architecture*, *Event Carried State Transfer*, *Learning Management System*

Palembang, 30 Juli 2024

Mengetahui  
Ketua Jurusan Teknik Informatika



  
Muhammad Fachrurrozzi, S.Si., M.T.  
NIP. 198005222008121002

Pembimbing

  
Mastura Diana Marieska, M.T  
NIP. 198603212018032001

## KATA PENGANTAR

Puji syukur kepada Allah Subanahu Wa Ta'ala atas Rahmat dan karunia Nya, penyusunan tugas akhir dengan judul “Optimasi Arsitektur Monolitik ke *Microservice* dengan *Event-Driven* pada Aplikasi *Learning Management System (LMS)*” dapat diselesaikan dengan baik. Tugas akhir ini disusun sebagai syarat dalam menyelesaikan studi Strata-1 program studi Teknik Informatika Fakultas Ilmu Komputer Universitas Sriwijaya.

Penulis ingin mengucapkan terima kasih kepada semua pihak yang telah membantu dan memberikan dukungan, bimbingan, kerjasama dan motivasi kepada penulis dalam menyelesaikan tugas akhir ini. Adapun penulis ingin mengucapkan terima kasih kepada:

1. Kedua orang tua serta keluarga penulis, yang telah memberikan dukungan dan doa.
2. Bapak Prof. Dr. Erwin, S.Si., M.Si., selaku Dekan Fakultas Ilmu Komputer Universitas Sriwijaya.
3. Bapak Dr. Muhammad Fachrurrozi, S.Si., M.T, selaku Ketua Jurusan Strata-1 Teknik Informatika Fakultas Ilmu Komputer Universitas Sriwijaya.
4. Ibu Mastura Diana Marieska, M.T., selaku Dosen Pembimbing skripsi atas segala teladan, bimbingan dan dukungan yang di berikan kepada peneliti.
5. Bapak Dr. Abdiansah, S.Kom., M.Cs., Bapak Osvari Arsalan, M.T., Ibu Desty Rodiah, S.Kom.,M.T., serta Bapak/Ibu dosen Teknik Informatika lainnya yang telah memberikan ilmu, teladan, dan pembelajaran yang berharga selama peneliti berkuliah di Teknik Informatika.
6. Kak Kemas M Husein Alviansyah, selaku mentor yang membantu menyediakan dan membiayai semua *cost cloud server* untuk penelitian ini.
7. Arya Yunanta, selaku sahabat penulis yang sering berdiskusi bersama mengenai topik sistem terdistribusi.
8. Ibnu Nalaprana, selaku teman seperbimbingan penulis yang sering menemani bimbingan dan mengurus berkas.
9. Kak Faisal Morensya, sebagai alumni dan mentor yang telah menjadi contoh, memotivasi penulis dan berkontribusi besar terhadap *growth* penulis menjadi *software engineer* dengan mempercayai penulis untuk terlibat ke dalam *project* nya serta menjadi teman berdiskusi.



10. Teman-teman magang tim backend dikti yang telah berbagi moment yang sangat berkesan. kalian awesome guys.
11. Teman Teman Damri enjoyer yang menjadi saksi pp plg-indralaya, teman teman reg-b yang menemani perkuliahan selama ini, serta teman-teman di Teknik informatika yang tidak dapat disebut satu-persatu.
12. Pegawai Fakultas Ilmu Komputer Universitas Sriwijaya yang telah membantu penulis selama proses pengerjaan skripsi berlangsung.

Penulis menyadari dalam penyusunan tugas akhir ini jauh dari sempurna baik dari segi penyusunan, bahasan, atapun penulisannya. Oleh karena itu, penulis mengharapkan kritik dan saran yang sifatnya membantu. Semoga tugas akhir ini bermanfaat bagi pengembangan wawasan dan peningkatan ilmu pengetahuan di kalangan masyarakat luas serta dapat dijadikan kajian untuk penelitian lainnya.

Palembang, 30 Juli 2024



Harisatul Aulia

NIM 09021282025062

## DAFTAR ISI

	Halaman
HALAMAN JUDUL.....	i
HALAMAN PENGESAHAN .....	ii
HALAMAN PERSETUJUAN KOMISI PENGUJI .....	iii
HALAMAN PERNYATAAN BEBAS PLAGIAT .....	iv
HALAMAN MOTTO DAN PERSEMBAHAN.....	v
ABSTRACT.....	vi
ABSTRAKSI .....	vii
KATA PENGANTAR .....	viii
DAFTAR ISI.....	x
DAFTAR TABEL.....	xiii
DAFTAR GAMBAR .....	xiv
DAFTAR LAMPIRAN.....	xvi
BAB I PENDAHULUAN.....	I-1
1.1 Pendahuluan.....	I-1
1.2 Latar Belakang.....	I-1
1.3 Rumusan Masalah.....	I-4
1.4 Tujuan Penelitian.....	I-4
1.5 Manfaat Penelitian.....	I-5
1.6 Batasan Masalah .....	I-5
1.7 Sistematika Penulisan .....	I-6
1.8 Kesimpulan.....	I-7
BAB II KAJIAN LITERATUR .....	II-1
2.1 Pendahuluan.....	II-1
2.2 Landasan Teori .....	II-1
2.2.1 Arsitektur Perangkat Lunak.....	II-1
2.2.1.1 Monolitik .....	II-2
2.2.1.2 <i>Microservice</i> .....	II-4
2.2.2 <i>Containerization</i> .....	II-7
2.2.3 Message Broker .....	II-8
2.2.4 Load Testing .....	II-9
2.2.5 Load Testing Metrics.....	II-10
2.2.5.1 Response Time .....	II-10
2.2.5.2 Error Rate .....	II-10
2.2.6 Golang.....	II-11

2.2.7 Rational Unified Process .....	II-11
2.3 Penelitian Terkait.....	II-13
2.4 Kesimpulan .....	II-17
BAB III METODOLOGI PENELITIAN.....	III-1
3.1 Pendahuluan.....	III-1
3.2 Pengumpulan Data.....	III-1
3.2.1 Jenis dan Sumber Data .....	III-1
3.2.2 Metode pengumpulan data .....	III-1
3.3 Tahapan Penelitian.....	III-2
3.3.1 Kerangka Kerja.....	III-2
3.3.2 Kriteria Pengujian.....	III-3
3.3.3 Format Data Pengujian .....	III-4
3.3.4 Alat Bantu Penelitian.....	III-5
3.3.5 Pengujian Penelitian .....	III-5
3.3.6 Analisis Hasil Pengujian dan Kesimpulan .....	III-7
3.4 Metode Pengembangan Perangkat Lunak .....	III-7
3.4.1 Fase Insepsi .....	III-7
3.4.2 Fase Elaborasi.....	III-8
3.4.3 Fase Kontruksi.....	III-8
3.4.4 Fase Transisi.....	III-8
3.5 Manajemen Proyek Penelitian .....	III-9
3.6 Kesimpulan .....	III-16
BAB IV PERANCANGAN PERANGKAT LUNAK.....	IV-1
4.1 Pendahuluan.....	IV-1
4.2 Fase Insepsi.....	IV-1
4.2.1 Pemodelan Bisnis .....	IV-1
4.2.2 Kebutuhan Sistem.....	IV-1
4.2.3 Analisis dan Desain .....	IV-2
4.3 Fase Elaborasi.....	IV-15
4.3.1 Pemodelan Bisnis .....	IV-15
4.3.2 Perancangan <i>service</i> .....	IV-16
4.3.3 Perancangan basis data .....	IV-19
4.3.5 Perancangan antar muka.....	IV-26
4.3.6 Kebutuhan Sistem .....	IV-32
4.3.7 Analisis dan Perancangan.....	IV-32
4.4 Fase Konstruksi .....	IV-48
4.4.1 Kebutuhan Sistem.....	IV-48
4.5 Fase Transisi.....	IV-58
4.5.1 Pemodelan Bisnis .....	IV-58
4.5.2 Rencana Pengujian .....	IV-59
4.5.3 Implementasi .....	IV-59
4.6 Kesimpulan.....	IV-60

BAB V HASIL & PEMBAHASAN .....	V-1
5.1 Pendahuluan.....	V-1
5.2 Service yang diuji .....	V-1
5.3 Konfigurasi Pengujian .....	V-1
5.4 Data Hasil Pengujian monolitik dan <i>microservice</i> .....	V-2
5.4.1 Monolitik .....	V-2
5.4.2 <i>Microservice</i> .....	V-4
5.5 Analisa Hasil Penelitian.....	V-6
5.6 Kesimpulan .....	V-11
BAB VI KESIMPULAN DAN SARAN .....	VI-1
6.1 Pendahuluan.....	VI-1
6.2 Kesimpulan .....	VI-1
6.3 Saran .....	VI-2
DAFTAR PUSTAKA .....	xvii
LAMPIRAN.....	xxi

## DAFTAR TABEL

Halaman

<b>Tabel III-1.</b> Format Pengujian Monolitik .....	III-4
<b>Tabel III-2.</b> Format Pengujian Microservice .....	III-4
<b>Tabel III-3.</b> Tabel Analisis Hasil .....	III-7
<b>Tabel III-4.</b> Tabel WBS Penelitian .....	III-10
<b>Tabel III-5.</b> Tabel Gantt Chart penelitian .....	III-14
<b>Tabel IV-1.</b> Tabel Kebutuhan Fungsional .....	IV-2
<b>Tabel IV-2.</b> Tabel Kebutuhan Non Fungsional .....	IV-2
<b>Tabel IV-3.</b> Tabel Definisi Aktor .....	IV-4
<b>Tabel IV-4.</b> Tabel Skenario Use Case Registrasi platform ujian .....	IV-5
<b>Tabel IV-5.</b> Tabel Skenario Use Case Registrasi Platform Ujian .....	IV-6
<b>Tabel IV-6.</b> Tabel Skenario Use Case Login Platform Ujian .....	IV-7
<b>Tabel IV-7.</b> Tabel Skenario Use Case Pilih Ujian .....	IV-9
<b>Tabel IV- 8.</b> Tabel Skenario Use Case Mengikuti Ujian .....	IV-12
<b>Tabel IV- 9.</b> Tabel Skenario Use Case Mengikuti Ujian .....	IV-14
<b>Tabel IV-10.</b> Tabel Implementasi Kelas Microservices.....	IV-51
<b>Tabel IV-11.</b> Tabel Implementasi Kelas Monolitik .....	IV-52
<b>Tabel IV-12.</b> Tabel Rencana Pengujian .....	IV-59
<b>Tabel IV-13.</b> Implementasi Pengujian .....	IV-59
<b>Tabel V-1.</b> Hasil Pengujian Low Load Monolitik.....	V-3
<b>Tabel V- 2.</b> Hasil Pengujian High Load pada Monolitik.....	V-4
<b>Tabel V-3.</b> Hasil Pengujian Low Load Microservice.....	V-5
<b>Tabel V-4.</b> Hasil Pengujian High Load Microservice .....	V-6
<b>Tabel V-5.</b> Hasil Analisa Hasil Penelitian.....	V-7

## DAFTAR GAMBAR

Halaman

<b>Gambar II- 1.</b> Arsitektur Monolitik.....	II-2
<b>Gambar II- 2.</b> Arsitektur Microservice.....	II-5
<b>Gambar II-3.</b> Contoh Perhitungan Error Rate.....	II-11
<b>Gambar II-4.</b> Rational Unified Process.....	II-12
<b>Gambar III-1.</b> Tahapan Penelitian.....	III-2
<b>Gambar III-2.</b> Kerangka Kerja.....	III-3
<b>Gambar IV-1.</b> Use Case Diagram.....	IV-3
<b>Gambar IV-2.</b> Usulan Arsitektur Monolitik.....	IV-16
<b>Gambar IV- 3.</b> Usulan Arsitektur Microservice.....	IV-18
<b>Gambar IV-4.</b> Model Data Monolitik .....	IV-20
<b>Gambar IV-5.</b> Model data Microservice Try Out Service.....	IV-21
<b>Gambar IV- 6.</b> Model Data Microservice Order Service .....	IV-22
<b>Gambar IV-7.</b> Model Data Microservice Payment Service .....	IV-22
<b>Gambar IV-8.</b> Model Data Microservice Exam Service .....	IV-23
<b>Gambar IV-9.</b> Model Data Microservice Grade Service.....	IV-24
<b>Gambar IV- 10.</b> Model Data Microservice Leaderboard Service .....	IV-25
<b>Gambar IV-11.</b> Model Data Microservice User Service.....	IV-25
<b>Gambar IV-12.</b> Desain Halaman Landing Page.....	IV-26
<b>Gambar IV-13.</b> Desain Halaman Mendaftar Akun .....	IV-27
<b>Gambar IV-14.</b> Desain Halaman Login Akun.....	IV-27
<b>Gambar IV-15.</b> Desain Halaman Dashboard.....	IV-28
<b>Gambar IV-16.</b> Desain Halaman Mendaftar Ujian .....	IV-28
<b>Gambar IV-17.</b> Desain Halaman Checkout.....	IV-29
<b>Gambar IV-18.</b> Desain Halaman Pembayaran .....	IV-29
<b>Gambar IV-19.</b> Desain Halaman Mengikuti Ujian .....	IV-30
<b>Gambar IV-20.</b> Desain Halaman Soal Ujian .....	IV-30
<b>Gambar IV-21.</b> Desain Halaman Telah Mengerjakan Ujian.....	IV-31
<b>Gambar IV-22.</b> Desain Halaman Melihat Leaderboard .....	IV-31
<b>Gambar IV-23.</b> Activity Diagram Mendaftar Akun.....	IV-33
<b>Gambar IV-24.</b> Activity Diagram Login Akun.....	IV-34
<b>Gambar IV-25.</b> Activity Diagram Mendaftar Ujian.....	IV-35
<b>Gambar IV-26.</b> Activity Diagram Mengikuti Ujian.....	IV-36
<b>Gambar IV-27.</b> Activity Diagram Melihat Leaderboard.....	IV-37
<b>Gambar IV-28.</b> Sequence Diagram Mendaftar Akun Monolitik.....	IV-38

<b>Gambar IV-29.</b>	Sequence Diagram Login Akun Monolitik.....	IV-39
<b>Gambar IV-30.</b>	Sequence Diagram Mendaftar Ujian Monolitik.....	IV-40
<b>Gambar IV-31.</b>	Sequence Diagram Mengikuti Ujian Monolitik.....	IV-41
<b>Gambar IV-32.</b>	Sequence Diagram Melihat Leaderboard Monolitik.....	IV-42
<b>Gambar IV-33.</b>	Sequence Diagram Mendaftar Akun Microservice.....	IV-43
<b>Gambar IV-34.</b>	Sequence Diagram Login Akun Microservice.....	IV-44
<b>Gambar IV-35.</b>	Sequence Diagram Mendaftar Ujian Microservice.....	IV-45
<b>Gambar IV-36.</b>	Sequence Diagram Mengikuti Ujian Microservice .....	IV-46
<b>Gambar IV-37.</b>	Sequence Diagram Melihat Leaderbord Microservice .....	IV-47
<b>Gambar IV-38.</b>	Class Diagram Monolitik.....	IV-49
<b>Gambar IV-39.</b>	Class Diagram Microservice.....	IV-50
<b>Gambar IV-40.</b>	Implementasi Halaman Dahsboard.....	IV-53
<b>Gambar IV-41.</b>	Implementasi Halaman Mendaftar Akun.....	IV-53
<b>Gambar IV-42.</b>	Implementasi Halaman Login Akun.....	IV-54
<b>Gambar IV-43.</b>	Implementasi Halaman dashboard.....	IV-54
<b>Gambar IV-44.</b>	Implementasi Halaman Mendaftar Ujian.....	IV-55
<b>Gambar IV-45.</b>	Implementasi Halaman Checkout.....	IV-55
<b>Gambar IV-46.</b>	Implementasi Halaman Pembayaran .....	IV-56
<b>Gambar IV-47.</b>	Implementasi Halaman Mengikuti Ujian.....	IV-56
<b>Gambar IV-48.</b>	Implementasi Halaman soal.....	IV-57
<b>Gambar IV-49.</b>	Implementasi Halaman Telah Mengerjakan Soal.....	IV-57
<b>Gambar IV-50.</b>	Implementasi Halaman Melihat Leaderboard.....	IV-58

## DAFTAR LAMPIRAN

1. Kode Program.....	xxi
2. Dataset Program .....	xxi
3. Surat Persetujuan Publikasi .....	xxii
4. Surat Apresiasi .....	xxiv



# **BAB I**

## **PENDAHULUAN**

### **1.1 Pendahuluan**

Bab pendahuluan akan membahas latar belakang masalah, rumusan masalah, tujuan penelitian, manfaat penelitian, batasan masalah penelitian, dan sistematika penulisan. Bab ini juga memuat penjelasan mengenai gambaran umum dari keseluruhan kegiatan penelitian yang dilakukan.

Pendahuluan dimulai dengan membahas mengenai *microservice* serta penelitian yang berkaitan dengan *Event Driven Architecture*.

### **1.2 Latar Belakang**

Perkembangan teknologi informasi telah mendorong perusahaan untuk terus mengevaluasi dan memperbarui infrastruktur IT mereka. Istilah "monolitik" digunakan untuk menjelaskan gaya arsitektur aplikasi klasik yang kurang memperhatikan modularitas sebagai faktor penting dalam pengembangannya (Megargel et al., 2020). Karena semua komponen aplikasi berjalan dalam satu proses tunggal, meningkatnya permintaan pada satu bagian aplikasi dapat menyebabkan keseluruhan sistem menjadi lambat atau bahkan gagal. Akhirnya, performa aplikasi dapat menurun secara signifikan saat beban aplikasi meningkat, karena keterbatasan dalam isolasi beban kerja antar komponen (Vainio, 2021) . Arsitektur monolitik, yang pernah menjadi standar juga kini seringkali dianggap tidak efisien untuk kebutuhan bisnis modern yang dinamis (Ponce et al., 2019).

Sebagai contoh, perusahaan global seperti Netflix telah mentransformasi ini dengan mengadopsi arsitektur berbasis *microservice*, yang memungkinkan mereka meningkatkan performa (Newman, 2021). Peningkatan performa terjadi juga pada salah satu penelitian yang melakukan migrasi serupa, hasilnya didapat waktu *response-time* rata-rata yang diukur di titik pencatatan *service logging point* meningkat dari 200ms (monolitik) menjadi 40ms (*microservice*) (Megargel et al., 2020).

*Microservices* memecah aplikasi menjadi serangkaian layanan kecil yang terpisah, yang dapat dikembangkan, diuji, dan dikerahkan secara independen. Hal ini memungkinkan tim pengembangan untuk mengadopsi pendekatan yang lebih modular dan responsif dalam menghadapi perubahan kebutuhan pasar (Lewis & Fowler, 2014). Banyak pendekatan implementasi arsitektur *microservice*. Salah satunya adalah implementasi *event-driven*. *Event-Driven* adalah sebuah kerangka dan pendekatan dalam perancangan aplikasi perangkat lunak. Sistem yang menggunakan *event-driven* dibuat untuk menangkap, menyampaikan, dan memproses *event* yang terjadi di antara layanan yang terpisah. Hal ini mengindikasikan bahwa sistem dapat beroperasi secara asinkron sambil tetap dapat berbagi data dan menyelesaikan tugas (RedHat, 2019).

Terdapat beberapa studi yang telah menangani isu-isu terkait, namun masih ada perbedaan substansial dengan penelitian ini. Misalnya, penelitian yang dilakukan oleh (Megargel et al., 2020) menyoroti pentingnya arsitektur aplikasi yang tepat dalam konteks *cloud-computing* sebagai bagian dari strategi transformasi digital organisasi. Fokus utamanya adalah migrasi arsitektur monolitik menjadi

*microservice*, namun tidak secara khusus mengintegrasikan *event-driven* dan mengambil studi kasus yang berbeda. Pada penelitian ini, studi kasus yang diangkat adalah sebuah sistem *learning management system* pada startup Civil Servant Warrior (CSW).

Startup CSW saat ini menggunakan arsitektur monolitik untuk pengembangan dan pengelolaan aplikasi mereka. Dalam arsitektur ini, seluruh fungsionalitas aplikasi dibangun dan dijalankan sebagai satu kesatuan. Semua modul dan komponen yang diperlukan oleh aplikasi, seperti autentikasi pengguna, pengelolaan data, dan antarmuka pengguna, terintegrasi dalam satu basis kode tunggal. Skalabilitas yang lebih baik dapat dicapai dengan mengatur layanan secara independen, memungkinkan startup CSW untuk meningkatkan atau mengurangi sumber daya sesuai kebutuhan spesifik. Sifat *microservice* yang terisolasi memastikan bahwa kegagalan pada satu layanan tidak mengganggu keseluruhan sistem, meningkatkan keandalan aplikasi (Aljawawdeh et al., 2023). Dalam hal performa, arsitektur monolitik seringkali menghadapi tantangan dalam hal responsivitas. Karena semua komponen aplikasi berjalan dalam satu proses yang sama, peningkatan beban pada satu bagian aplikasi dapat mempengaruhi performa keseluruhan.

Penelitian ini secara khusus menganalisis proses migrasi dari arsitektur monolitik ke *microservice* dengan pendekatan *event driven* pada aplikasi LMS yang sudah ada sebelumnya. Aplikasi LMS di implementasikan dengan arsitektur monolitik untuk dimigrasi ke arsitektur *microservice* kemudian dianalisis performanya. Metodologi penelitian mencakup analisis kuantitatif untuk mengukur

performa. Studi ini diharapkan memberikan rekomendasi terutama bagi startup CSW dan organisasi yang mempertimbangkan transisi serupa.

### 1.3 Rumusan Masalah

Terdapat beberapa *research question* (RQ) dalam penelitian ini yaitu sebagai berikut :

1. Bagaimana perbandingan performa arsitektur monolitik dan *microservice* aplikasi *Learning Management System* (LMS)?
2. Bagaimana perbedaan *behaviour* monolitik dan *microservice* pada *low load* dan *high load*?

### 1.4 Tujuan Penelitian

Berdasarkan rumusan masalah yang telah disebutkan, berikut adalah tujuan penelitian ini:

1. Melakukan penerapan migrasi arsitektur dari monolitik ke *microservice* dengan *event-driven* terhadap aplikasi *Learning Management System* (LMS).
2. Mengevaluasi dampak penerapan migrasi arsitektur dari monolitik ke *microservice* dengan *event-driven* terhadap peningkatan performa pada aplikasi *Learning Management System* (LMS).
3. Mengevaluasi *behaviour* arsitektur monolitik dan *microservice* pada aplikasi *Learning Management System*(LMS).

### 1.5 Manfaat Penelitian

Manfaat penelitian ini dapat dijelaskan sebagai berikut:

1. Menyajikan pemahaman tentang bagaimana penerapan transformasi arsitektur dari monolitik ke *microservice* dengan *event-driven* dapat meningkatkan performa pada aplikasi Sistem Manajemen Pembelajaran (LMS).
2. Memberikan rekomendasi bagi organisasi yang mempertimbangkan transisi serupa.
3. Menjadi referensi buat penelitian selanjutnya dibidang sistem terdistribusi khususnya pada topik *microservice* dan *distributed system*.

### 1.6 Batasan Masalah

1. Infrastruktur *deploy* dan *testing* akan diterapkan di *cloud* AWS .
2. Protokol API yang dibahas hanya berfokus pada Restful API over HTTP (TCP: 80).
3. Fitur yang akan di migrasi berfokus kepada ujian online (pendaftaran, pembayaran dan pelaksanaan ujian)
4. Pola *event driven* yang digunakan adalah *event carried state transfer*.
5. *Framework http* yang digunakan adalah *standard http libary* dari golang.
6. Pengujian *load test* menggunakan Grafana k6.

## 1.7 Sistematika Penulisan

Adapun sistematika penulisan pada penelitian ini adalah sebagai berikut:

### **BAB I. PENDAHULUAN**

Bab ini menguraikan mengenai latar belakang, rumusan masalah, tujuan penelitian, manfaat penelitian, batasan masalah, dan sistematika penulisan pada penelitian ini.

### **BAB II. KAJIAN LITERATUR**

Bab ini membahas mengenai dasar-dasar teori mengenai *microservice*, *monolitik*, *event sourcing*, *Event Driven Architecture*, serta metrik-metrik yang berkaitan dengan analisis performa pada sistem ini. Bab ini juga menguraikan penelitian-penelitian terdahulu yang terkait dengan penelitian ini.

### **BAB III. METODOLOGI PENELITIAN**

Bab ini berisi pembahasan mengenai metodologi dan tahapan perancangan penelitian seperti pengumpulan data, metode pengembangan perangkat lunak, dan manajemen proyek penelitian.

### **BAB IV. PENGEMBANGAN PERANGKAT LUNAK**

Pada bab ini akan membahas tentang RUP sebagai metode yang digunakan dalam pengembangan perangkat lunak, yang terdiri

dari 4 fase yaitu: fase insepri, fase elaborasi, fase konstruksi, dan fase transisi.

## **BAB V. KESIMPULAN DAN HASIL**

Pada bab ini akan membahas tentang proses uji coba yang terdiri dari konfigurasi percobaan, dan analisis hasil penelitian.

### **1.8 Kesimpulan**

Berdasarkan latar belakang penelitian yang telah diuraikan maka penelitian ini akan melakukan perubahan arsitektur serta menguji pengaruh arsitektur *microservice* pada aplikasi *Learning Management System* (LMS) untuk peningkatan performa

## DAFTAR PUSTAKA

- Aljawawdeh, H., Sabri, M., & Maghrabi, L. (2023). Toward Serverless and Microservices Architecture: Literature, Methods, and Best Practices. In *Studies in Computational Intelligence* (Vol. 1113). Springer Nature Switzerland. [https://doi.org/10.1007/978-3-031-43300-9\\_47](https://doi.org/10.1007/978-3-031-43300-9_47)
- Chabbi, M., & Ramanathan, M. K. (2022). A study of real-world data races in Golang. In *Proceedings of the ACM SIGPLAN Conference on Programming Language Design and Implementation (PLDI)* (Vol. 1, Issue 1). Association for Computing Machinery. <https://doi.org/10.1145/3519939.3523720>
- Gos, K., & Zabierowski, W. (2020). The Comparison of Microservice and Monolithic Architecture. *International Conference on Perspective Technologies and Methods in MEMS Design*, 150–153. <https://doi.org/10.1109/MEMSTECH49584.2020.9109514>
- Grafana. (2022). *K6 Learn*. Github.Com. <https://github.com/grafana/k6-learn/blob/main/Modules/II-k6-Foundations/03-Understanding-k6-results.md>
- Hakim, Z., & Rizky, R. (2018). *Analisis Perancangan Sistem Informasi Pembuatan Paspor Di Kantor Imigrasi Bumi Serpong Damai Tangerang Banten Menggunakan Metode Rational Unified Process*. 6(2), 103–112. <https://doi.org/https://doi.org/10.33592/jutis.Vol6.Iss2.135>
- Kumari, S., & Rath, S. K. (2015). Performance comparison of SOAP and REST based Web Services for Enterprise Application Integration. *2015 International Conference on Advances in Computing, Communications and Informatics*,



- ICACCI 2015*, 1656–1660. <https://doi.org/10.1109/ICACCI.2015.7275851>
- Lewis, J., & Fowler, M. (2014). *Microservice*. Martinowler.Com.  
<https://martinowler.com/articles/microservices.html>
- Mageed, A. (2023). *Event-Carried State Transfer Integration in Microservices*.  
Medium.Com. <https://levelup.gitconnected.com/event-carried-state-transfer-integration-in-microservices-486bebaae83b>
- Magnoni, L. (2015). Modern messaging for distributed systems. *Journal of Physics: Conference Series*, 608(1). <https://doi.org/10.1088/1742-6596/608/1/012038>
- Megargel, A., Shankararaman, V., & Walker, D. K. (2020). Migrating from Monoliths to Cloud-Based Microservices: A Banking Industry Example. *IOSR Journal of Computer Engineering Ver. IV*, 33(6), 110–114.
- Molyneaux, I. (2007). *The Art of Application Performance Testing* (1st ed.). O'Reilly Media.
- Montemagno, J. (2022). *What is Docker?* Microsoft.Com.  
<https://learn.microsoft.com/en-us/dotnet/architecture/microservices/container-docker-introduction/docker-defined>
- Newman, S. (2021). *Building microservices: Designing fine-grained systems (second edition)*. O'Reilly Media.
- Nickoloff, J. (2019). *Docker in Action*. Mannings Publications.
- Pangestu, G. (2022). *Asynchronous vs Synchronous Programming*. Binus.Ac.Id.  
<https://binus.ac.id/malang/2022/05/asynchronous-vs-synchronous-programming/>

- Ponce, F., Marquez, G., & Astudillo, H. (2019). Migrating from monolithic architecture to microservices: A Rapid Review. *Proceedings - International Conference of the Chilean Computer Science Society, SCCC, 2019-Novem.* <https://doi.org/10.1109/SCCC49216.2019.8966423>
- RedHat. (2019). *What is event-driven architecture?* Redhat.Com. <https://www.redhat.com/en/topics/integration/what-is-event-driven-architecture>
- Richards, M., & Ford, N. (2020). *Fundamentals of Software Architecture An Engineering Approach.* O'Reilly Media.
- Rizal, D. H. F. (2022). *Performance Testing Metrics.* Medium.Com. <https://medium.com/the-legend/performance-testing-metrics-2a973ff42cd5>
- Rouse, M. (2018). *Software Architecture.* <https://www.techopedia.com/definition/24596/software-architecture>
- Seetharamugn. (2023). *The Complete Guide to Event-Driven Architecture.* Medium.Com. <https://medium.com/@seetharamugn/the-complete-guide-to-event-driven-architecture-b25226594227>
- Sheps, A. (2023). *Containerization 101: Basic Concepts, challenges & solutions.* Aquasec.Com. <https://www.aquasec.com/cloud-native-academy/docker-container/containerization-101/>
- Stephens, R. (2015). *Beginning Software Engineering.* John Wiley & Sons, Inc.
- Susnjara, S., & Smalley, I. (2024). *What is containerization?* Ibm.Com. <https://www.ibm.com/topics/containerization>
- Vainio, M. (2021). *The benefits and challenges in migrating from a monolithic*

*architecture into microservice architecture* [University of Helsinki].

<https://helda.helsinki.fi/handle/10138/328296>

Yussan. (2020). *K6.io Load Testing Gampang Untuk Para Developer Web dan API*.

Maucoding.Com. [https://maucoding.com/post/K6io-Load-Testing-Gampang-](https://maucoding.com/post/K6io-Load-Testing-Gampang-Untuk-Para-Developer-Web-dan-API-5eca21e4740dc435c7abe998)

[Untuk-Para-Developer-Web-dan-API-5eca21e4740dc435c7abe998](https://maucoding.com/post/K6io-Load-Testing-Gampang-Untuk-Para-Developer-Web-dan-API-5eca21e4740dc435c7abe998)

Zubkov, S. (2023). *Error Rate*. Adapty.Io. <https://adapty.io/glossary/error-rate/>