INTEGRASI DUA APLIKASI *WEB* MENGGUNAKAN ARSITEKTUR *MICROSERVICES* (STUDI KASUS: APLIKASI PENGUKURAN CPL DAN TUGAS AKHIR)

FACHRIZAL TAUFIQ GOE
H071191008

PROGRAM STUDI SISTEM INFORMASI
DEPARTEMEN MATEMATIKA
FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM
UNIVERSITAS HASANUDDIN
MAKASSAR
2024

INTEGRASI DUA APLIKASI *WEB* MENGGUNAKAN ARSITEKTUR *MICROSERVICES* (STUDI KASUS: APLIKASI PENGUKURAN CPL DAN TUGAS AKHIR)

FACHRIZAL TAUFIQ GOE
H071191008

Skripsi

Diajukan sebagai salah satu syarat untuk mencapai gelar sarjana

Program Studi Sistem Informasi

Pada

PROGRAM STUDI SISTEM INFORMASI
FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM
UNIVERSITAS HASANUDDIN
MAKASSAR
2024

SKRIPSI

INTEGRASI DUA APLIKASI WEB MENGGUNAKAN ARSITEKTUR MICROSERVICES (STUDI KASUS: APLIKASI PENGUKURAN CPL DAN TUGAS AKHIR)

FACHRIZAL TAUFIQ GOE
H071191008

Skripsi,

Telah dipertahankan di depan Panitia Ujian Sarjana Sistem Informasi pada tanggal 20 Agustus 2024
dan dinyatakan telah memenuhi syarat kelulusan

Pada

Program Studi Sistem Informasi
Department Matematika
Fakultas Matematika dan Ilmu Pengetahuan Alam
Universitas Hasanuddin
Makassar
2024

Mengesahkan:	Mengetahui:
Pembimbing tugas akhir,	Ketua Program Studi,
Muhammad Sadno, S. Si., M. Si	Prof. Drs. Jeffry Kusuma, Ph. D.
NIP. 19900816202204300	NIP. 196411121987031002

PERNYATAAN KEASLIAN SKRIPSI DAN PELIMPAHAN HAK CIPTA

Dengan ini saya menyatakan bahwa, skripsi berjudul "INTEGRASI DUA APLIKASI WEB MENGGUNAKAN ARSITEKTUR MICROSERVICES (STUDI KASUS: APLIKASI PENGUKURAN CPL DAN TUGAS AKHIR)" adalah benar karya saya dengan arahan dari pembimbing (Muhammad Sadno, S.Si., M.Si. sebagai Pembimbing Utama). Karya ilmiah ini belum diajukan dan tidak sedang diajukan dalam bentuk apa pun kepada perguruan tinggi mana pun. Sumber informasi yang berasal atau dikutip dari karya yang diterbitkan maupun tidak diterbitkan dari penulis lain telah disebutkan dalam teks dan dicantumkan dalam Daftar Pustaka skripsi ini. Apabila di kemudian hari terbukti atau dapat dibuktikan bahwa sebagian atau keseluruhan skripsi ini adalah karya orang lain, maka saya bersedia menerima sanksi atas perbuatan tersebut berdasarkan aturan yang berlaku.

Dengan ini saya melimpahkan hak cipta (hak ekonomis) dari karya tulis saya berupa skripsi ini kepada Universitas Hasanuddin.

Makassar, 20 Agustus 2024

Fachrizal Taufiq Goe
H071191008

KATA PENGANTAR

Puji Syukur penulis panjatkan ke hadirat Tuhan yang Maha Esa atas segala limpahan Rahmat dan karunia-Nya, yang telah memberikan kesempatan dan kelancaran bagi penulis dalam penyelesaian tugas akhir yang berjudul "INTEGRASI DUA APLIKASI WEB MENGGUNAKAN ARSITEKTUR MICROSERVICES (STUDI KASUS: APLIKASI PENGUKURAN CPL DAN TUGAS AKHIR)" ini. Dengan berbagai rintangan yang dihadapi saat menyelesaikan tugas ini, tidak lupa untuk penulis mengucapkan terima kasih atas kontribusi dan bantuannya kepada:

1. Pembimbing utama, **Muhammad Sadno, S.Si., M.Si.**
2. Kedua dosen penguji, **Dr. Muhammad Hasbi, M.Sc.** dan **Jeriko Gormantara, S.Si., M.Si.**
3. Bapak/Ibu **Dosen Program Studi Sistem Informasi** beserta seluruh tenaga pendidik juga seluruh staf dan pegawai **Departemen Matematika**.
4. Kedua orang tua penulis, Bapak **Eron Hon Goe** dan ibu **Frilda Hunowu**.
5. Teman teman **Sistem Informasi Ang. 2019**

Penulis menyadari bahwa skripsi ini masih jauh dari sempurna dikarenakan terbatasnya pengalaman dan pengetahuan yang dimiliki penulis. Oleh karena itu, penulis mengharapkan segala bentuk saran serta masukan bahkan kritik yang membangun dari berbagai pihak. Semoga tulisan ini memberikan manfaat kepada semua pihak yang membutuhkan dan terutama untuk penulis.

Makassar, 20 Agustus 2024

Fachrizal Taufiq Goe
H071191008

ABSTRAK

FACHRIZAL TAUFIQ GOE. **Integrasi dua aplikasi web menggunakan arsitektur microservices (studi kasus: aplikasi pengukuran CPL dan tugas akhir)** (dibimbing oleh Muhammad Sadno, S. Si., M. Si).

Aplikasi web memainkan peran krusial dalam kehidupan sehari-hari, mencakup berbagai keperluan seperti bisnis, komunikasi, hiburan, dan pendidikan. Seiring dengan meningkatnya kompleksitas aplikasi web, kebutuhan akan skalabilitas, keamanan, dan performa optimal juga semakin penting. Arsitektur microservices merupakan pendekatan yang efektif untuk mengatasi tantangan ini, memungkinkan aplikasi web dibangun secara modular dengan masing-masing microservices dapat berjalan mandiri. Skripsi ini mengkaji penerapan arsitektur microservices dalam proyek Sistem Informasi Farmasi (SIFA) di Fakultas Farmasi Universitas Hasanuddin. SIFA merupakan hasil integrasi dari dua aplikasi terpisah—Capaian Pembelajaran (CPL) dan Tugas Akhir (TA)—yang sebelumnya beroperasi secara independen namun memiliki data dan fitur serupa. Integrasi ini menghadapi masalah terkait kemiripan data dan fitur autentikasi yang dapat meningkatkan kompleksitas sistem serta pemeliharaan. Penelitian ini mengusulkan solusi berbasis microservices untuk mengelola fitur yang redundan, mengoptimalkan pengiriman email dan notifikasi, serta memperbaiki performa dan keamanan sistem secara keseluruhan. Dengan pendekatan ini, diharapkan integrasi CPL dan TA menjadi lebih efisien, meningkatkan penggunaan sumber daya, dan memperbaiki kinerja aplikasi secara menyeluruh.

Kata kunci: microservices architecture, application programming interface, api gateway, database, integrasi aplikasi web, tes integrasi

ABSTRACT

FACHRIZAL TAUFIQ GOE. **Integration of Two Web Applications Using Microservices Architecture (Case Study: CPL Measurement Application and Final Project)** (supervised by Muhammad Sadno, S. Si., M. Si).

Web applications play a crucial role in daily life, serving various needs such as business, communication, entertainment, and education. As web applications become more complex, the demands for scalability, security, and optimal performance grow increasingly important. Microservices architecture is an effective approach to address these challenges, allowing web applications to be built in a modular fashion with each microservice able to operate independently. This thesis explores the application of microservices architecture in the Pharmacy Information System (SIFA) project at the Faculty of Pharmacy, Hasanuddin University. SIFA is the result of integrating two separate applications—Learning Achievement (CPL) and Final Project (TA)—which previously operated independently but shared similar data and features. The integration faces issues related to data similarity and authentication features, which can increase system complexity and maintenance. This research proposes a microservices-based solution to manage redundant features, optimize email and notification delivery, and enhance overall system performance and security. With this approach, it is anticipated that the integration of CPL and TA will become more efficient, improve resource utilization, and enhance the overall performance of the application.

Keywords: microservices architecture, application programming interface, api gateway, basis data, web app integration, integration test

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

BAB I
PENDAHULUAN

1.1 Latar Belakang

Aplikasi *web* telah menjadi bagian penting dari kehidupan sehari-hari. Aplikasi *web* digunakan untuk berbagai keperluan, mulai dari keperluan bisnis hingga keperluan pribadi. Dalam perkembangannya, aplikasi *web* menjadi semakin kompleks dan membutuhkan skalabilitas yang tinggi. Penggunaannya mencakup transaksi keuangan, komunikasi, hiburan, hingga pendidikan. Dengan adanya perkembangan teknologi, keamanan dan performa aplikasi *web* menjadi fokus utama para pengembang untuk memastikan pengalaman pengguna yang optimal.

Salah satu pendekatan untuk membangun aplikasi *web* yang kompleks dan skalabilitas tinggi adalah dengan menggunakan arsitektur *microservices*. Aplikasi dengan arsitektur microservices adalah aplikasi terdistribusi yaitu semua modul atau elemennya adalah *microservices* dan dapat dijalankan secara mandiri (Velepucha & Flores, 2023). Dengan menggunakan arsitektur *microservices*, aplikasi *web* dapat diimplementasikan secara modular. Hal ini memudahkan pengembangan, pemeliharaan, dan skalabilitas aplikasi *web*. Dalam beberapa kasus, diperlukan untuk mengintegrasikan dua aplikasi *web* yang sudah ada. Integrasi dua aplikasi *web* dapat dilakukan dengan menggunakan arsitektur *microservices*.

Proyek SIFA, singkatan dari Sistem Informasi Farmasi, merupakan suatu solusi inovatif yang digunakan di Fakultas Farmasi Universitas Hasanuddin. Aplikasi ini secara unik terbentuk melalui integrasi dua aplikasi besar sebelumnya, yaitu aplikasi pengukuran Capaian Pembelajaran (CPL) dan Tugas Akhir (TA). Kedua aplikasi tersebut awalnya beroperasi secara terpisah. Namun, kedua aplikasi memiliki fitur yang sama dan bergantung pada data-data yang mirip, sehingga integrasi kedua aplikasi diperlukan untuk mempertahankan konsistensi data yang digunakan oleh kedua aplikasi.

Permasalahan utama yang dihadapi oleh kedua aplikasi ini adalah kemiripan data yang tersimpan pada masing masing aplikasi, sehingga apabila data pada satu aplikasi berubah maka data pada aplikasi lainnya juga harus berubah. Permasalahan lainnya adalah kesamaan fitur yang digunakan oleh masing masing aplikasi seperti fitur autentikasi atau mekanisme login yang diterapkan oleh masing masing aplikasi sehingga kredensial dari satu aplikasi dapat berbeda dengan aplikasi lainnya. Keberadaan fitur yang mirip pada kedua aplikasi dapat mengakibatkan peningkatan kompleksitas sistem dan pemeliharaan yang sulit. Selain itu dapat menimbulkan ketidakseimbangan dalam pengelolaan sumber daya dan mengakibatkan potensi kesalahan dalam pengiriman informasi kepada pengguna.

Sebagai solusi inovatif terhadap permasalahan permasalahan ini, dalam skripsi ini, saya mengusulkan pendekatan integrasi menggunakan arsitektur *microservices*. Langkah ini diambil dengan tujuan memberikan solusi yang efektif,

mengatasi masalah autentikasi, serta meningkatkan kinerja dan keamanan sistem secara menyeluruh. Dengan mengadopsi pendekatan *microservices*, diharapkan integrasi antara CPL dan TA dapat dilakukan dengan lebih efisien dan efektif.

Dalam penelitian ini, solusi yang diusulkan adalah dengan memanfaatkan pendekatan *microservices* untuk mengelola fitur-fitur yang redundan tersebut. Dengan mendekomposisi fungsionalitas ke dalam *microservices* yang independen, pengiriman *email* dan notifikasi dapat dioptimalkan dan diintegrasikan lebih baik. Pendekatan ini memungkinkan pengelolaan fitur secara terpusat dan menghindari duplikasi yang tidak perlu. Sebagai hasilnya, integrasi antara CPL dan TA dapat mencapai efisiensi yang lebih tinggi, mengoptimalkan penggunaan sumber daya, dan meningkatkan keseluruhan kinerja aplikasi.

1.2 Rumusan Masalah

Berdasarkan latar belakang yang telah diuraikan, maka rumusan masalah dalam penelitian ini adalah sebagai berikut:

1. Bagaimana mengintegrasikan aplikasi manajemen capaian pembelajaran dan tugas akhir menggunakan arsitektur *microservices*?
2. Bagaimana pengujian integrasi aplikasi manajemen capaian pembelajaran dan tugas akhir menggunakan arsitektur *microservices*?

1.3 Batasan Masalah

Berdasarkan rumusan masalah yang telah diuraikan, maka batasan masalah dalam penelitian ini adalah sebagai berikut:

1. Jenis aplikasi *web*: Penelitian ini hanya membahas integrasi aplikasi manajemen capaian pembelajaran dan tugas akhir yang menggunakan arsitektur *microservices*.
2. Aspek yang dikaji: Penelitian ini hanya menganalisis tantangan-tantangan, solusi, dan efektivitas arsitektur *microservices* untuk mengintegrasikan aplikasi manajemen capaian pembelajaran dan tugas akhir.

1.4 Tujuan Penelitian

Berdasarkan rumusan masalah yang telah diuraikan, maka tujuan penelitian ini adalah sebagai berikut:

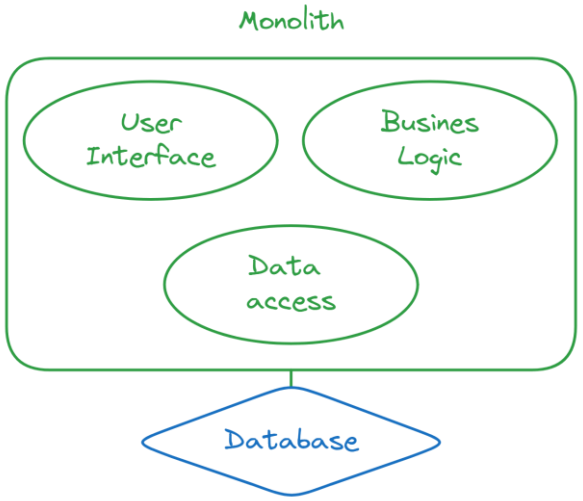
1. Mengintegrasikan aplikasi manajemen capaian pembelajaran dan tugas akhir menggunakan arsitektur *microservices*.
2. Menguji integrasi aplikasi manajemen capaian pembelajaran dan tugas akhir menggunakan arsitektur *microservices*.

1.5 Landasan Teori

1.5.1 *Application Programming Interface*

Application Programming Interface (API) adalah aturan dan protokol yang memungkinkan berbagai perangkat lunak untuk berinteraksi satu sama lain. API menyediakan serangkaian fungsi dan metode yang dapat diakses oleh pengembang untuk memudahkan integrasi dan komunikasi antar aplikasi. Dengan menggunakan API, pengembang dapat mengambil keuntungan dari fitur atau layanan yang disediakan oleh suatu platform atau sistem tanpa perlu mengetahui implementasi internalnya. API memfasilitasi pengembangan aplikasi yang lebih efisien dan interoperabilitas antar perangkat lunak, membantu mempercepat proses pengembangan, dan mendukung ekosistem yang lebih luas dalam dunia pemrograman.

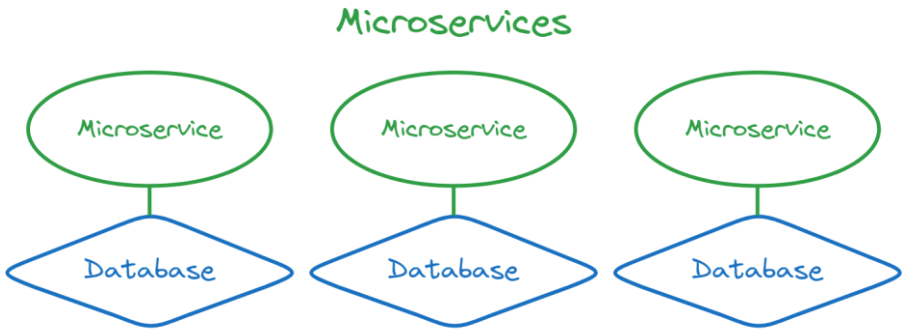
1.5.2 Arsitektur Monolith



Gambar 1. Arsitektur aplikasi *monolith*

Arsitektur *monolith* adalah arsitektur aplikasi yang menggunakan *single codebase* (basis kode tunggal) untuk melayani berbagai layanan dan antarmuka berbeda (Kalske, 2017). Dalam arsitektur ini, semua komponen aplikasi, termasuk *user interface* (antarmuka pengguna), *business logic* (logika bisnis), dan akses data, digabungkan menjadi satu unit atau aplikasi besar seperti pada Gambar 1. Hal ini berarti bahwa semua fitur dan fungsionalitas aplikasi berada dalam satu proyek yang sama dan saling bergantung satu sama lain.

1.5.3 Arsitektur *Microservices*



Gambar 2. Arsitektur aplikasi *microservices*

Arsitektur microservices, sebagai evolusi dari pola arsitektur monolitik, mulai mendapatkan perhatian signifikan sejak awal 2010-an. Arsitektur *microservices* adalah pendekatan dalam pengembangan perangkat lunak oleh aplikasi yang dibangun sebagai serangkaian layanan kecil yang berjalan secara independen seperti pada Gambar 2. Setiap layanan (*microservices*) berfokus pada tugas tertentu dan dapat berkomunikasi dengan layanan lain melalui antarmuka yang didefinisikan dengan baik. Arsitektur ini memberikan sejumlah keuntungan, termasuk skalabilitas yang lebih baik, perubahan dan pemeliharaan yang lebih mudah, serta kemampuan untuk menggunakan teknologi yang berbeda untuk setiap *microservices*.

Dengan *microservices*, setiap layanan berfokus pada satu area fungsional, memiliki basis kode dan penyimpanan data sendiri, dan dapat dikembangkan, diuji, dan di-*deploy* secara terpisah. Setiap *microservice* dapat menggunakan teknologi dan bahasa pemrograman yang paling sesuai dengan tugasnya, dan tim pengembang dapat bekerja secara independen pada berbagai layanan, mempercepat pengembangan dan pemeliharaan.

Salah satu keuntungan signifikan dari arsitektur *microservices* adalah kemampuannya untuk menangani skalabilitas dengan lebih baik. Dalam lingkungan monolitik, skalabilitas sering kali berarti menambah sumber daya untuk seluruh aplikasi, bahkan jika hanya bagian tertentu yang membutuhkan peningkatan. Dengan *microservices*, hanya layanan yang memerlukan skalabilitas yang ditingkatkan, yang memungkinkan penggunaan sumber daya yang lebih efisien dan pengurangan biaya operasional. Selain itu, setiap *microservice* dapat dioptimalkan dan diperbarui secara terpisah, memungkinkan perbaikan dan pembaruan dilakukan tanpa mempengaruhi keseluruhan sistem.

Berbagai penelitian akademik juga menyertakan kasus studi implementasi microservices di industri. Analisis ini memberikan wawasan tentang penerapan nyata dari arsitektur microservices, tantangan yang dihadapi, dan solusi yang diterapkan oleh organisasi. Kasus studi ini sering mencakup perusahaan teknologi besar seperti

Netflix, Amazon, dan eBay yang telah mengadopsi microservices untuk meningkatkan skalabilitas dan fleksibilitas aplikasi mereka.

1.5.4 API *Gateway*

API Gateway adalah layanan manajemen dan kontrol yang berorientasi pada API, terpusat secara serial, dan kuat yang muncul di system boundary. (Zhao, 2018). API *Gateway* adalah komponen yang bertindak sebagai gerbang masuk untuk *microservices* dalam arsitektur *microservices*. Tugas utamanya adalah menyediakan titik akses tunggal untuk *client* agar dapat berinteraksi dengan seluruh *microservices* yang ada. API *Gateway* menangani tugas-tugas seperti autentikasi, otorisasi, dan routing, serta memberikan antarmuka yang bersih untuk *client* eksternal.

1.5.5 *Database*

Dalam konteks arsitektur *microservices*, basis data sering kali dikelola oleh setiap *microservices* secara terpisah. Pendekatan ini dikenal sebagai basis data terdistribusi. *Database* dapat berupa relasional atau non-relasional, tergantung pada kebutuhan aplikasi. Penggunaan basis data terdistribusi memungkinkan setiap *microservices* untuk mempertahankan otonomi dan independensi dalam pengelolaan data.

1.5.6 Komunikasi Antara *microservices*

1.5.6.1 Komunikasi *Synchronous*

Komunikasi ini melibatkan pertukaran pesan secara langsung antara *microservices*. Sebuah *microservices* membuat permintaan ke *microservices* lain dan menunggu *microservices* tersebut memproses hasilnya dan mengirimkan respons kembali (Shafabakhsh, 2020). Keuntungan dari komunikasi ini adalah sederhana dan mudah dipahami, namun perlu diingat bahwa ketergantungan pada layanan eksternal dapat menyebabkan masalah performa jika tidak dikelola dengan baik.

1.5.6.2 Komunikasi *Asynchronous*

Bentuk komunikasi *asynchronous* dapat diimplementasikan dalam *microservices* ketika layanan bertukar pesan satu sama lain melalui *message broker* (Shafabakhsh, 2020). Komunikasi ini melibatkan pertukaran pesan tanpa menunggu respons. Metode ini menggunakan sistem antrian pesan atau layanan pesan yang mendukung komunikasi *asynchronous*. Keuntungan utamanya adalah peningkatan kinerja dan skala, karena setiap *microservices* dapat menjalankan tugasnya tanpa harus menunggu respons langsung.

1.5.7 *Integration Testing*

Pengujian integrasi dalam konteks arsitektur *microservices* sangat penting untuk memastikan bahwa semua *microservices* dapat berinteraksi dengan baik dan memberikan fungsionalitas yang diharapkan. Pengujian integrasi melibatkan uji coba keseluruhan sistem dari semua komponen yang diintegrasikan. Ini dapat mencakup pengujian fungsionalitas, kinerja, dan keandalan antarmuka antarmuka antar *microservices*.

1.5.8 *Containers*

Containers adalah sebuah teknologi virtualisasi yang memungkinkan pengemasan dan pengiriman aplikasi bersama dengan semua dependensinya, termasuk perpustakaan, variabel lingkungan, dan kode eksekusi, dalam unit yang dapat dijalankan secara konsisten di berbagai lingkungan. Kontainer membantu dalam memastikan bahwa aplikasi dapat dijalankan dengan benar di berbagai platform, dari lingkungan pengembangan hingga produksi. Teknologi kontainer populer seperti Docker telah menjadi bagian integral dari implementasi arsitektur *microservices* karena memungkinkan isolasi aplikasi dan penerapannya dengan cepat.

1.5.9 *Message Broker*

Message broker adalah perangkat lunak yang bertindak sebagai perantara dalam pertukaran pesan antara komponen atau layanan yang berbeda dalam arsitektur *microservices*. Penggunaan *message broker* memungkinkan komunikasi asinkron antara *microservices*, memfasilitasi integrasi yang longgar dan meningkatkan ketahanan sistem. Beberapa *message broker* populer termasuk RabbitMQ, Apache Kafka, dan ActiveMQ.

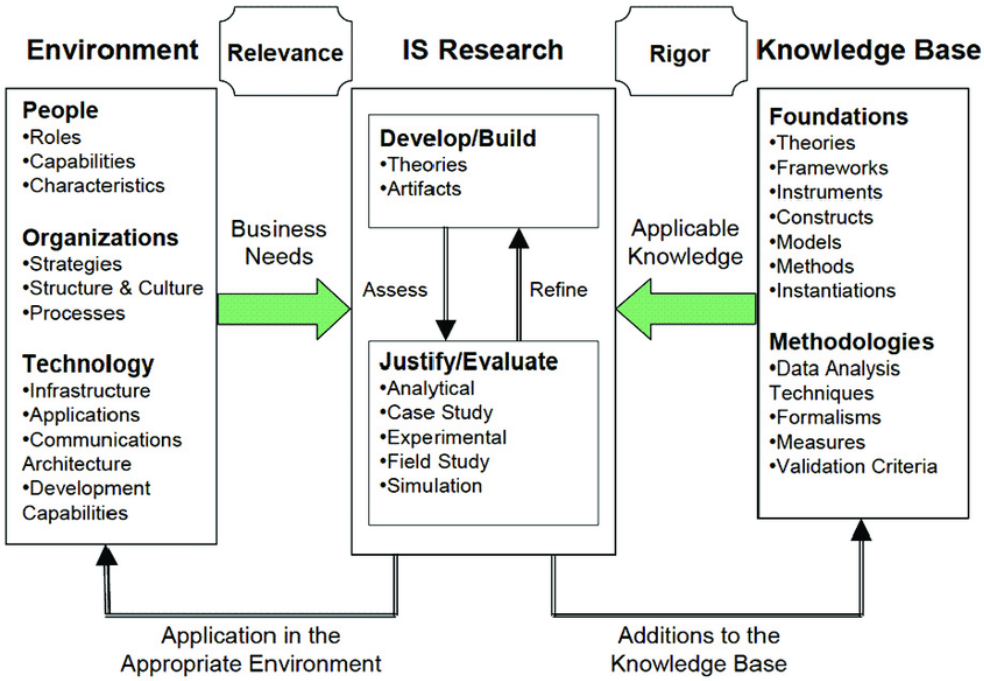
Kafka adalah platform *open-source* yang digunakan untuk mengelola aliran data secara *real-time*. Dikembangkan oleh LinkedIn dan didonasikan ke Apache Software Foundation, Kafka dirancang untuk menangani aliran data yang besar dalam skala *horizontal*. Ini sangat populer di antara perusahaan-perusahaan teknologi yang membutuhkan sistem pengiriman pesan yang cepat, andal, dan skalabel.

1.5.10 *Hypertext Transfer Protocol*

HTTP, atau *Hypertext Transfer Protocol*, merupakan protokol komunikasi yang digunakan untuk pertukaran informasi di dalam *World Wide Web* (WWW). Protokol ini memungkinkan transfer data antara *client* dan *server* melalui jaringan komputer. HTTP adalah bagian integral dari arsitektur *web* dan digunakan untuk menginisiasi permintaan dari *client* dan meresponnya dengan mengirimkan data atau sumber daya yang diminta. Dengan kata lain, HTTP berfungsi sebagai sarana utama dalam proses pengambilan dan penyampaian informasi di lingkungan *web*.

1.5.11 Lingkungan Sistem Informasi

Gambar 3 menyajikan *Framework* Hevner atau kerangka konseptual Hevner yaitu kerangka konseptual untuk memahami, melaksanakan, dan mengevaluasi penelitian *Information System* (IS) yang menggabungkan paradigma ilmu perilaku dan ilmu desain (Hevner, 2004).



Gambar 3. Kerangka konseptual penelitian sistem informasi.

source: Hevner (2004)

Lingkungan sistem informasi dari penelitian ini seperti pada Gambar 4 berikut:

Gambar 4. Konsep penelitian sistem informasi.

Berdasarkan Gambar 4, Penelitian ini berfokus pada penerapan arsitektur *microservices* dalam pengembangan aplikasi *web* di lingkungan perguruan tinggi. mahasiswa, dosen, dan admin merupakan pengguna (*people*) utama aplikasi ini, sedangkan universitas (*organization*) menyediakan infrastruktur dan kebijakan yang mendukung penggunaan teknologi ini. Teori *microservices* menjadi dasar dalam perancangan aplikasi. Tes integrasi (*methodologies* / *validation criteria*) dilakukan untuk memastikan bahwa komponen-komponen *microservices* dapat bekerja sama dengan baik. Melalui dua *case study*, yaitu pengembangan aplikasi manajemen capaian pembelajaran (CPL) dan aplikasi tugas akhir (TA). Hasil akhir penelitian ini merupakan aplikasi dalam bentuk *artifacts* berupa *container images*.

1.6 Tinjauan Pustaka

Penelitian terdahulu terkait permasalahan ini dirangkum dalam Tabel 1 berikut:

Tabel 1. Penelitian terkait

No.	Peneliti dan Judul penelitian	Pembahasan	Persamaan	Perbedaan
1	Yodihamzah, 2023. Rancang Bangun *Backend Microservices* Aplikasi MeetingYuk Berbasis Bahasa Pemrograman Go	Membehas tentang pengembangan ulang aplikasi MeetingYuk menjadi sepenuhnya mengimplementasikan arsitektur *microservices* karena keterbatasan dalam hal skalabilitas, responsivitas, dan efisiensi penggunaan sumber daya pada aplikasi sebelumnya.	Penelitian berfokus pada berfokus pada proses development aplikasi *microservices* dan bentuk komunikasi antara *microservices*.	Pengujian tidak berfokus pada integrasi melainkan pada *latency*, *response per second*, *response time*, dan konsumsi memori.
2	Risnanto, 2022. Rancang Bangun Sistem Informasi Layanan Mandiri Perpustakaan Berbasis Arsitektur *Microservice* (studi kasus: Pusat Perpustakaan UIN Syarif	Merancang aplikasi sistem informasi perpustakaan berbasis arsitektur *microservices* yang akan digunakan pada perpustakaan pusat pada UIN Hidayatullah Jakarta.	Studi kasus dalam penelitian ini adalah aplikasi yang terkait dengan akademik dari Universitas.	Penelitian lebih kepada skalabilitas dan ketersediaan dari aplikasi, kesesuaian *user interface*, perancangan *database* dan perancangan *deployment* aplikasi.

No.	Peneliti dan Judul penelitian	Pembahasan	Persamaan	Perbedaan
	Hidayatullah Jakarta)			
3	AL LATHIEF, 2019. Rancang Bangun Aplikasi Ujian Tugas Akhir Berbasis *Outcome Based Assessment* Dari Sisi Perancangan *Back-End* dan *Database*	Merancang aplikasi berbasis *website* yang dapat menangani proses bisnis ujian tugas akhir pada DTETI UGM. Ujian tugas akhir berbasis *Outcome-Based Assessment* (OBA)	Studi kasus dalam penelitian ini adalah aplikasi yang terkait dengan akademik dari Universitas. Membahas tentang *backend development*, pembuatan API dan Implementasi *Token-Based Authentication* yang juga diterapkan pada penelitian penulis.	Penelitian ini berfokus pada bentuk jadi aplikasi. Seperti desain *User Interface*, desain *Frontend* dan *Backend* dan desain *database* yang rinci. Skripsi ini tidak mengimplementasikan *microservices* secara langsung. Skema pengujian API pada penelitian ini menggunakan metode *blackbox* yang mempunyai kesamaan pada *integration test* pada penelitian penulis.
4	Belluano, 2020. Sistem Informasi Program Kreativitas Mahasiswa berbasis *Web Service* dan *Microservice*.	Penelitian ini mengusulkan pembangunan infrastruktur *microservice* untuk memelihara dan menyesuaikan alur proses bisnis dalam sistem informasi. Metode penelitian	Persamaan dengan penelitian Saya adalah penggunaan arsitektur *microservice* dan teknologi terkini seperti *web service*, UML, serta pengelolaan aplikasi dengan GORM, Go	Perbedaannya terletak pada fokus penelitian, yaitu penelitian Saya lebih menekankan integrasi aplikasi *eksisting* dengan *microservice* untuk mengatasi masalah

No.	Peneliti dan Judul penelitian	Pembahasan	Persamaan	Perbedaan
		mencakup studi lapangan dan kepustakaan terkait PKM, Sistem Informasi, *Web Service*, dan *Microservice* dengan paradigma Convention Over Configuration. Perancangan sistem menggunakan prototyping dan UML, dengan pengelolaan aplikasi DBMS menggunakan GORM, Go Validator, dan Gorilla Mux. Implementasi dilakukan dengan AMQP, menghasilkan sistem *web service* berbasis *microservice* untuk pelaporan PKM dengan transaksi cepat.	Validator, dan Gorilla Mux.	autentikasi dan redundansi fitur, sementara penelitian ini lebih umum dalam konteks pengembangan infrastruktur *microservice* untuk pemeliharaan alur bisnis dalam sistem informasi.
5	Suryotrisongko, 2017. Arsitektur *microservice* untuk *resiliensi* sistem informasi.	Pengembangan *software Open Source* untuk manajemen asosiasi/keanggotaan dengan fokus pada	Penelitian ini adalah keduanya memiliki fokus pada pengembangan sistem informasi yang *resilient*, mampu menangani	Perbedaan utama terletak pada konteks dan skala implementasi. Penelitian Saya lebih berfokus pada integrasi aplikasi *eksisting*

No.	Peneliti dan Judul penelitian	Pembahasan	Persamaan	Perbedaan
		meningkatkan kualitas resiliensi sistem. Software tersebut telah diimplementasikan pada sistem manajemen anggota AISINDO dan dipublikasikan sebagai plugin WordPress untuk digunakan secara internasional. Evaluasi menunjukkan kebutuhan akan peningkatan kinerja sistem untuk mengatasi jumlah member yang terus meningkat. Penelitian ini melibatkan penyusunan model dan pembuatan *proof of concept* dari modifikasi arsitektur *software* yang terdistribusi, berbasis *microservice*, dan *Docker-container* untuk menciptakan *Resilient Information Systems* pada AISINDO, sehingga mendapatkan data empiris untuk evaluasi dan	perubahan dan gangguan di luar skenario yang telah ditetapkan sebelumnya. Keduanya juga mencakup penggunaan teknologi terbaru seperti *microservices* dan *Docker* untuk meningkatkan kinerja dan ketahanan sistem.	dengan *microservices* untuk mengatasi masalah autentikasi dan redundansi fitur, sementara penelitian ini lebih terkait dengan pengembangan dari awal untuk meningkatkan *resiliensi* sistem manajemen anggota pada skala asosiasi profesi. Meskipun demikian, kedua penelitian memiliki tujuan yang sejalan dalam menghadapi kompleksitas dan perubahan dalam pengembangan sistem informasi modern.

No.	Peneliti dan Judul penelitian	Pembahasan	Persamaan	Perbedaan
		pengembangan model lebih lanjut.		

BAB II
METODE PENELITIAN

2.1 Jadwal Penelitian

Penelitian akan dilaksanakan secara online pada bulan Oktober sampai November 2023 dengan jadwal seperti pada Tabel 2.

Tabel 2. Jadwal penelitian

No	Tahapan penelitian	Oktober				November			
		1	2	3	4	1	2	3	4
1	Perancangan aplikasi								
2	Pengembangan aplikasi								
3	Pengujian aplikasi								

2.2 Tahapan Penelitian

Tahapan penelitian adalah tahapan tahapan yang diikuti demi kelancaran dalam menjalankan sebuah penelitian. Terdapat tiga tahapan dalam penelitian ini adalah sebagai berikut:

1. Tahapan yang pertama adalah tahapan perancangan aplikasi. Tahapan perancangan aplikasi mencakup perancangan arsitektur sistem dan menyiapkan dokumentasi rinci untuk memandu pengembangan dan memastikan semua aspek desain dipahami dengan jelas.
2. Tahap yang kedua adalah tahapan pengembangan aplikasi. Pada tahap ini, kode ditulis dan berbagai komponen aplikasi diintegrasikan sesuai desain. Proses ini memastikan bahwa semua bagian aplikasi berfungsi dengan baik dan terhubung secara efektif.
3. Tahap yang ketiga adalah tahapan pengujian aplikasi. Tahapan pengujian melibatkan pengujian tes integrasi aplikasi untuk memastikan fitur-fitur berfungsi dengan baik dan memenuhi kriteria yang telah ditentukan.

2.3 Instrumen Penelitian

1. Perangkat Keras:

   Perangkat keras yang akan digunakan adalah laptop dengan spesifikasi sebagai berikut:

   - 4 *Core* CPU

- 16GB RAM
- 256GB SSD

2. Perangkat Lunak:
   - *Integrated Development Environment* (IDE)
   - *Version Control System* (VCS)
   - *Containerization Tools*
   - API *Testing Tools*
   - API *Gateway*
   - *Message Broker*
   - *Web Browser*

2.4 Teknik Pengumpulan Data

Data terkait implementasi *microservices* dan integrasi dikumpulkan melalui tes integrasi. Tes integrasi dilakukan menggunakan Postman untuk menguji API.

2.5 Rancangan Tes Integrasi

Pada Tabel 3 berikut adalah rangkaian skenario-skenario tes integrasi antara setiap *microservice*:

Tabel 3. Rangkaian skenario-skenario tes integrasi

No	Deskripsi	Harapan Status
1	*Login*	Sukses
2	Mengakses CPL sebelum *login*	Gagal
3	Membuat CPL	Sukses
4	Membaca CPL	Sukses
5	Memperbarui CPL	Sukses
6	Menghapus CPL	Sukses
7	Mengakses TA sebelum *login*	Gagal
8	Membuat TA	Sukses
9	Membaca TA	Sukses
10	Memperbarui TA	Sukses
11	Menghapus TA	Sukses

12	Mengakses *User* sebelum *login*	Gagal
13	Membuat *User*	Sukses
14	Membaca *User*	Sukses
15	Memperbarui *User*	Sukses
16	Menghapus *User*	Sukses
17	Mengakses *Notification* sebelum login	Gagal
18	Membaca *Notification*	Sukses