Jurnal Teknik Mesin, Industri, Elektro dan Informatika
Volume 4, Nomor 3, September 2025
E-ISSN: 2963-7805; P-ISSN: 2963-8208, Hal. 37-47
DOI: https://doi.org/10.55606/jtmei.v4i3.5825
Tersedia: https://ejurnal.politeknikpratama.ac.id/index.php/jtmei

Penerapan Arsitektur Microservices pada Sistem Informasi Penjualan Online untuk Meningkatkan Skalabilitas dan Kinerja Sistem

Rosyid Fadillah Nurrokhim[1*], Fahrul Febriansyah Siregar[2], Ferdy Al Firdaus[3], Rizki Aditiya[4], Faizal Muhammad Yahya[5], Lutfiandra Pohan [6], Ryan Satria Pratama[7]
[1-7]Universitas Esa Unggul, Indonesia
Email: rosyidfadillah01@student.esaunggul.ac.id[1], fahrulsiregar41@student.esaunggul.ac.id[2], yahya.hafiz10@student.esaunggul.ac.id[3], lutfiandrapohann@gmail.com[4], adit02888@student.esaunggul.ac.id[5], satriaryan433@gmail.com[6], ferdy.firdausferdy@gmail.com[7]*
[*]Penulis korespondensi: rosyidfadillah01@gmail.com[1]

Abstract. *This study discusses the application of microservices architecture in online sales information systems as a solution to the limitations of monolithic architecture in terms of scalability and performance. Using the Software Development Life Cycle (SDLC) method of the Waterfall model, the system is built by dividing applications into small, independent services and communicating with each other through a REST API in JSON format. The implementation is carried out using Node.js, ReactJS, and Docker technologies to support more efficient management and isolation of the system. The results of the study show that the application of microservices architecture is able to improve data processing efficiency and system development flexibility. However, the complexity of managing communication between services through API Gateway is a challenge that needs to be considered in implementation. Overall, this study confirms that microservices architecture is a relevant and effective approach in the development of online sales information systems, especially to support the scalability, performance, and sustainability needs of future application development.*

Keywords: *Microservices; Online Sales; REST API; Scalability; SDLC Waterfall*

Abstrak. Penelitian ini membahas penerapan arsitektur microservices pada sistem informasi penjualan online sebagai solusi atas keterbatasan arsitektur monolitik dalam aspek skalabilitas dan kinerja. Dengan menggunakan metode Software Development Life Cycle (SDLC) model Waterfall, sistem dibangun dengan membagi aplikasi menjadi layanan-layanan kecil yang independen dan saling berkomunikasi melalui REST API dengan format JSON. Implementasi dilakukan menggunakan teknologi Node.js, ReactJS, dan Docker untuk mendukung pengelolaan serta isolasi sistem yang lebih efisien. Hasil penelitian menunjukkan bahwa penerapan arsitektur microservices mampu meningkatkan efisiensi pengolahan data dan fleksibilitas pengembangan sistem. Namun, kompleksitas pengelolaan komunikasi antar layanan melalui API Gateway menjadi tantangan yang perlu diperhatikan dalam implementasi. Secara keseluruhan, penelitian ini menegaskan bahwa arsitektur microservices merupakan pendekatan yang relevan dan efektif dalam pengembangan sistem informasi penjualan online, terutama untuk mendukung kebutuhan skalabilitas, kinerja, dan keberlanjutan pengembangan aplikasi di masa depan.

Kata kunci: Layanan Mikro; Penjualan Online; REST API; SDLC Waterfall; Skalabilitas

1. LATAR BELAKANG

Perkembangan teknologi informasi yang pesat saat ini telah membawa dampak besar pada dunia bisnis, khususnya pada sistem informasi penjualan online. Di sektor e-commerce, sistem yang digunakan harus mampu mengelola transaksi dalam jumlah besar dengan efisiensi tinggi dan keandalan yang terjaga. (Asrowardi et al., 2020) Meskipun banyak platform menggunakan arsitektur monolitik, sistem ini seringkali menghadapi bottleneck dalam pengolahan data dan transaksi, terutama ketika skalanya meningkat. Arsitektur monolitik, yang menggabungkan seluruh aplikasi dalam satu kesatuan, cenderung menghambat pengembangan,

Naskah Masuk: 19 Juli 2025; Revisi: 18 Agustus 2025; Diterima: 28 September 2025; Terbit: 30 September 2025

pembaruan, dan pemeliharaan sistem secara efektif, serta mengonsumsi banyak sumber daya server. (Belluano et al., 2020)

Dalam beberapa tahun terakhir, arsitektur microservices mulai dipertimbangkan sebagai solusi utama untuk mengatasi keterbatasan yang ada pada sistem monolitik. (Anwar & Kautsar, 2024) Dengan membagi aplikasi menjadi layanan-layanan kecil yang independen, microservices memungkinkan setiap bagian sistem berkembang secara mandiri, tanpa mempengaruhi keseluruhan aplikasi. Layanan-layanan ini dapat di-deploy dan dikembangkan secara terpisah, yang memberikan fleksibilitas dan kemudahan pemeliharaan. Keuntungan utama dari arsitektur ini adalah kemampuannya untuk meningkatkan skalabilitas dan kinerja sistem secara keseluruhan. (Mulyawan et al., 2022, #)

Arsitektur microservices memungkinkan aplikasi untuk lebih modular, di mana setiap layanan dapat menangani satu fungsi spesifik dalam sistem. Misalnya, dalam sistem informasi penjualan online, layanan untuk pengelolaan produk, keranjang belanja, dan pemrosesan transaksi dapat dipecah menjadi unit-unit terpisah yang berkomunikasi melalui API. (Atmojo et al., 2022) Dengan begitu, beban kerja dapat dibagi secara merata antar layanan, meningkatkan efisiensi dan responsivitas sistem. Hal ini tidak hanya mengurangi ketergantungan antar layanan tetapi juga mempercepat proses pengembangan dan pengujian setiap layanan secara terpisah. (Mulyawan et al., 2022)

Namun, penerapan arsitektur microservices juga menghadapi tantangan tersendiri. Salah satunya adalah kompleksitas dalam pengelolaan komunikasi antar layanan yang berbeda. Setiap layanan membutuhkan API Gateway untuk menangani permintaan yang datang, dan pengelolaan autentikasi serta otorisasi menjadi aspek penting yang harus diperhatikan. (Dahri et al., 2022)Meskipun demikian, penggunaan microservices dalam sistem informasi penjualan online memberikan kemudahan skalabilitas, di mana perusahaan dapat menambah atau mengurangi layanan sesuai kebutuhan tanpa mengganggu kinerja sistem secara keseluruhan. (Atmojo et al., 2022)

Sebagaimana diungkapkan dalam berbagai studi sebelumnya, implementasi microservices pada platform e-commerce dapat memberikan dampak positif dalam hal kecepatan pengolahan data, kemudahan pemeliharaan, dan efisiensi sumber daya. (Alchuluq & Nurzaman, 2021) Oleh karena itu, adopsi microservices pada sistem informasi penjualan online menjadi pilihan yang strategis bagi perusahaan untuk meningkatkan kinerja mereka. Dengan terus berkembangnya cloud computing dan containerization technologies seperti Docker, penerapan arsitektur microservices menjadi semakin lebih mudah dan efisien untuk dijalankan,

memberikan keuntungan jangka panjang bagi sistem yang diimplementasikan. (Asrowardi et al., 2020)

2. KAJIAN TEORITIS

Arsitektur Monolitik

Arsitektur monolitik merupakan pendekatan tradisional dalam pengembangan aplikasi, di mana semua komponen aplikasi digabungkan menjadi satu kesatuan besar. Dalam arsitektur ini, setiap perubahan pada satu bagian aplikasi bisa mempengaruhi bagian lainnya, sehingga mempersulit pengembangan, pemeliharaan, dan pengujian. (Anwar & Kautsar, 2024) Sistem berbasis monolitik biasanya memiliki ketergantungan antara komponen-komponennya, yang menyulitkan pengelolaan dan pengembangan lebih lanjut. Selain itu, arsitektur monolitik sering kali memakan banyak sumber daya, terutama pada server, karena seluruh aplikasi dijalankan dalam satu proses yang sama. (Atmojo et al., 2022)

Arsitektur Microservice

Arsitektur microservice adalah pendekatan yang membagi aplikasi besar menjadi layanan-layanan kecil yang dapat berjalan secara independen. (Belluano et al., 2020) Setiap layanan memiliki fungsi spesifik dan berkomunikasi dengan layanan lain melalui antarmuka yang telah ditentukan, seperti API. Salah satu keunggulan utama dari arsitektur microservice adalah fleksibilitas dalam pengembangan dan skala yang lebih baik, memungkinkan aplikasi untuk beradaptasi dengan kebutuhan baru tanpa mengganggu bagian lain dari sistem. Setiap layanan dalam microservices dapat dikembangkan, diuji, dan di-deploy secara terpisah, yang membuat pemeliharaan sistem menjadi lebih efisien dan mudah. (Dinova & Utomo, 2023)

REST API

REST (Representational State Transfer) adalah gaya arsitektur yang banyak digunakan dalam desain web service. REST mengandalkan HTTP untuk komunikasi antara klien dan server, dengan resources yang dapat diakses menggunakan URL yang unik. Dalam konteks microservices, REST API memungkinkan layanan-layanan yang terpisah untuk saling berkomunikasi melalui protokol HTTP. Metode HTTP yang umum digunakan dalam REST API antara lain adalah GET, POST, PUT, dan DELETE. REST sangat ringan, kompatibel dengan berbagai platform, dan mudah untuk diimplementasikan. (Iqbal et al., 2023)

API

Application Programming Interface (API) adalah antarmuka yang memungkinkan dua sistem perangkat lunak untuk berkomunikasi satu sama lain. (Mulyawan et al., 2022) API memungkinkan pengembang untuk mengakses fungsi atau layanan yang ada di dalam suatu

aplikasi atau sistem tanpa perlu mengetahui implementasi internalnya. Dalam konteks arsitektur microservices, API memainkan peran penting dalam memungkinkan komunikasi antar layanan yang terdistribusi, memungkinkan sistem yang lebih besar untuk saling terhubung dan berinteraksi secara terorganisir. Keunggulan API adalah kemampuannya untuk menghubungkan aplikasi yang berbeda platform dan bahasa pemrograman. (Sinambela et al., 2021)

JSON

JSON (JavaScript Object Notation) adalah format pertukaran data yang ringan, mudah dibaca dan ditulis oleh manusia, serta mudah diproses oleh komputer. JSON sering digunakan dalam komunikasi antar layanan yang menggunakan REST API, karena formatnya yang sederhana dan tidak bergantung pada bahasa pemrograman tertentu. (Dinova & Utomo, 2023) Dalam konteks microservices, JSON memfasilitasi pertukaran data antar layanan dengan mengurangi beban pada server dan mempercepat proses komunikasi antar komponen. Keuntungan utama dari penggunaan JSON adalah kemampuannya untuk mendukung integrasi antar aplikasi yang menggunakan teknologi yang berbeda. (Belluano et al., 2020)

Marketplace

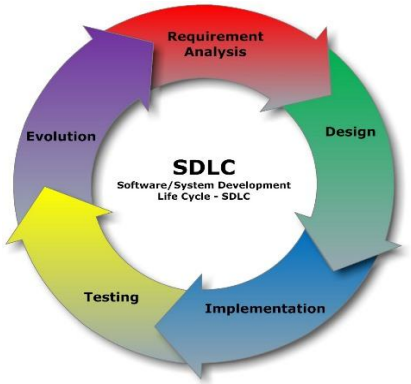
Marketplace adalah sebuah platform online yang memungkinkan penjual dan pembeli untuk berinteraksi, berbagi informasi tentang produk, dan melakukan transaksi. (Atmojo et al., 2022) Dalam konteks e-commerce, marketplace bertindak sebagai perantara antara berbagai pihak, yang memungkinkan pembeli untuk membandingkan harga dan produk dari berbagai penjual. Penggunaan arsitektur microservices dalam marketplace memberikan keuntungan dalam pengelolaan dan pengembangan fitur-fitur baru tanpa mengganggu operasional yang sudah berjalan. Microservices juga memungkinkan skalabilitas yang lebih baik, sehingga marketplace dapat mengelola lebih banyak transaksi dengan efisien. (Anwar & Kautsar, 2024)

Node.js

Node.js adalah sebuah platform pengembangan aplikasi yang menggunakan JavaScript untuk membuat aplikasi server-side. Node.js dikenal dengan kemampuan I/O asinkron dan event-driven, yang memungkinkan aplikasi untuk menangani banyak permintaan secara bersamaan tanpa mengganggu performa. (Alchuluq & Nurzaman, 2021) Dalam pengembangan microservices, Node.js sering digunakan untuk membangun REST API yang menghubungkan layanan-layanan kecil dalam sistem. Keunggulan Node.js adalah kemampuannya untuk menangani trafik tinggi dan koneksi simultan dengan efisien, membuatnya sangat cocok untuk aplikasi berbasis microservices yang membutuhkan komunikasi cepat antar layanan.

3. METODE PENELITIAN

Penelitian ini menggunakan metode SDLC Waterfall (Software Development Life Cycle) dalam pengembangan sistem informasi berbasis microservices. SDLC Waterfall dikenal dengan pendekatannya yang sistematis dan berurutan, di mana setiap tahapan dalam pengembangan harus diselesaikan sebelum melanjutkan ke tahap berikutnya. Metode ini dipilih untuk menjaga agar proses pengembangan berjalan dengan terorganisir dan dapat dipantau secara efektif. (Alchuluq & Nurzaman, 2021)



Gambar 1. Alur SDLC.

Analisis Kebutuhan Sistem

Pada tahap ini, dilakukan identifikasi secara menyeluruh terhadap kebutuhan sistem yang akan dibangun. Tim pengembang bekerja sama dengan pemangku kepentingan untuk mendefinisikan kebutuhan fungsional dan non-fungsional sistem, seperti keamanan, performa, dan aksesibilitas. Selama tahap ini, berbagai data dikumpulkan melalui wawancara dengan pengguna, pengamatan langsung, serta dokumentasi yang ada. Hasil dari tahap analisis ini adalah spesifikasi kebutuhan yang jelas, yang akan menjadi dasar bagi desain dan pengembangan sistem. Selain itu, dilakukan penentuan batasan sistem, baik dari sisi teknologi maupun kebijakan yang berlaku. Spesifikasi ini nantinya akan digunakan untuk merancang arsitektur sistem yang tepat. (Anwar & Kautsar, 2024)

Desain Sistem

Setelah kebutuhan sistem dianalisis, langkah berikutnya adalah perancangan arsitektur sistem. Pada tahap ini, desain sistem menggunakan arsitektur microservices, dimana aplikasi dibagi menjadi layanan-layanan kecil yang dapat beroperasi secara independen. Setiap layanan memiliki fungsi spesifik dan terisolasi dari layanan lain, yang membuat pengembangan dan pemeliharaan lebih mudah dilakukan. (Atmojo et al., 2022) Desain ini juga mencakup aspek teknis seperti REST API untuk komunikasi antar layanan, penggunaan JWT (JSON Web Token) untuk autentikasi, dan MySQL untuk basis data. Selain itu, UI/UX juga didesain agar

mudah digunakan oleh pengguna akhir. Desain sistem ini bertujuan untuk memastikan bahwa aplikasi dapat diskalakan dan mudah untuk diperbarui di masa depan. (Asrowardi et al., 2020)

Pengembangan Sistem

Pada tahap pengembangan, tim pengembang akan mulai menulis kode program berdasarkan desain sistem yang telah disetujui. Bahasa pemrograman yang digunakan dalam pengembangan sistem ini adalah PHP dan JavaScript, sementara untuk antarmuka pengguna (UI) menggunakan ReactJS. Setiap layanan dalam arsitektur microservices dikembangkan secara terpisah, dengan mengacu pada desain API dan basis data yang telah ditentukan. (Belluano et al., 2020) Penggunaan container Docker membantu dalam mengelola layanan secara lebih efisien, memungkinkan pengembang untuk mengisolasi setiap bagian dari sistem. Selama pengembangan, dilakukan integrasi antar layanan untuk memastikan bahwa semua bagian aplikasi dapat berkomunikasi dengan lancar menggunakan REST API dan JSON sebagai format pertukaran data. (Atmojo et al., 2022)

Pengujian Sistem

Setelah sistem dikembangkan, tahap selanjutnya adalah pengujian untuk memastikan bahwa semua fitur bekerja sesuai dengan yang diharapkan. Pengujian dilakukan dengan menggunakan metode black-box, dimana sistem diuji berdasarkan fungsionalitas yang diberikan, tanpa memeriksa struktur internalnya. (Dahri et al., 2022) Setiap layanan diuji secara terpisah untuk memastikan bahwa response time, keandalan, dan komunikasi antar layanan berjalan dengan baik. Pengujian dilakukan untuk mengidentifikasi dan mengatasi bug atau masalah yang muncul selama pengembangan. Selain itu, dilakukan juga pengujian integrasi untuk memastikan bahwa seluruh sistem dapat bekerja sebagai satu kesatuan yang utuh dan saling terhubung. (Belluano et al., 2020)

Implementasi Sistem

Setelah sistem berhasil melewati tahap pengujian, tahap selanjutnya adalah implementasi atau deploy sistem dalam lingkungan produksi. Pada tahap ini, aplikasi yang sudah diuji di development environment dipindahkan ke production environment untuk digunakan oleh pengguna akhir. (Dinova & Utomo, 2023) Implementasi dilakukan secara bertahap untuk meminimalisir gangguan pada pengguna. Selama implementasi, dilakukan juga monitoring terhadap kinerja sistem untuk memastikan bahwa aplikasi dapat berfungsi dengan baik di dunia nyata. Proses implementasi ini juga mencakup training pengguna dan dokumentasi sistem agar pengguna dapat mengoperasikan aplikasi dengan optimal. (Mulyawan et al., 2022)

Pemeliharaan Sistem

Setelah sistem diterapkan dan digunakan, tahap terakhir adalah pemeliharaan. Pada tahap ini, sistem akan terus dipantau untuk memastikan kinerjanya tetap stabil dan efisien. Setiap perubahan kebutuhan atau perbaikan bug akan dikelola melalui pembaruan sistem. Pemeliharaan dilakukan untuk memastikan bahwa aplikasi tetap berfungsi dengan baik seiring waktu dan dapat beradaptasi dengan kebutuhan baru. Proses pemeliharaan ini juga melibatkan optimasi sistem secara berkelanjutan untuk menjaga agar layanan dapat berjalan dengan skala besar dan efisien dalam jangka panjang. (Sinambela et al., 2021)

3. HASIL DAN PEMBAHASAN

Pada tahap ini, kita mengambil contoh dari API Shopee v2.product.get_category yang dapat ditemukan di dokumentasi resmi Shopee untuk mengakses kategori produk dalam sistem e-commerce berbasis arsitektur microservices.

Desain Microservice

Pada arsitektur microservices, API v2.product.get_category dapat diintegrasikan ke dalam Product Service atau Category Service. Dalam hal ini, service ini bertanggung jawab untuk berkomunikasi dengan API Shopee untuk mengambil data kategori produk yang ada pada platform Shopee dan mengelola data tersebut di dalam sistem internal.

Contoh penerapannya:

a. Product Management Microservice: Service ini dapat menggunakan API v2.product.get_category untuk secara dinamis mengambil kategori produk dari Shopee. Setelah data kategori diterima, service ini bisa menyimpannya dalam cache atau database internal dan kemudian menyediakan API bagi service lain atau aplikasi front-end untuk mengaksesnya.
b. Category Service: Service khusus yang menangani data kategori, yang dapat terhubung langsung ke API Shopee untuk melakukan sinkronisasi data kategori secara berkala. Service ini kemudian bisa mengelola kategori yang diterima dari Shopee dan menyesuaikannya dengan kebutuhan platform e-commerce internal.

Komunikasi Antar Microservices

Di dalam lingkungan microservices, setiap service berkomunikasi satu sama lain melalui REST API. Sebagai contoh, Category Service yang berinteraksi dengan API Shopee untuk mengambil data kategori produk akan mengirimkan data tersebut ke service lain, seperti Product Service, yang memerlukan kategori tersebut untuk proses pengelolaan produk. Dengan

demikian, API ini tidak hanya menyediakan data kategori tetapi juga memungkinkan berbagai microservices di dalam ekosistem e-commerce berkomunikasi secara terpisah dan efisien.

Keamanan dan Autentikasi

Penggunaan access_token dan sign dalam setiap permintaan API memastikan bahwa setiap permintaan yang dilakukan antara microservices dan Shopee adalah sah dan terotentikasi. Sebagai contoh, sebuah API Gateway dapat bertugas untuk menangani autentikasi dan validasi parameter seperti access_token sebelum meneruskan permintaan ke service terkait.

Dalam implementasi microservices, API Gateway ini akan menjadi titik masuk untuk semua permintaan eksternal dan memastikan bahwa semua komunikasi yang terjadi antara layanan dan API eksternal, seperti Shopee, aman dan terotentikasi.

Skalabilitas

Salah satu keuntungan utama dari arsitektur microservices adalah kemampuannya untuk melakukan skalabilitas secara independen pada tiap service. Misalnya, Category Service yang mengakses API Shopee secara rutin mungkin memerlukan lebih banyak sumber daya jika permintaan data kategori meningkat. Dengan arsitektur microservices, kita bisa mengalokasikan lebih banyak resource untuk Category Service tanpa mempengaruhi layanan lainnya dalam ekosistem.

Penanganan Error dan Kegagalan

Dalam implementasi microservices, penanganan error menjadi hal yang sangat penting. Jika terjadi kesalahan seperti invalid access_token atau error_auth, microservices dapat menangani error tersebut dengan cara yang terisolasi. Sebagai contoh, Category Service dapat memicu retry mechanism atau fallback service untuk mencoba kembali menghubungi API Shopee jika kegagalan terjadi, memastikan bahwa data kategori tetap tersedia meskipun terdapat masalah sementara.

Selain itu, jika API Shopee tidak dapat diakses, Category Service dapat menyediakan data kategori yang disalin (cached) dari request sebelumnya untuk memastikan bahwa pengguna atau sistem lain tetap dapat mengakses data tersebut tanpa adanya gangguan.

API v2.product.get_category dari Shopee memainkan peran penting dalam integrasi data kategori produk dalam sistem e-commerce berbasis arsitektur microservices. Dengan membagi aplikasi menjadi layanan-layanan kecil yang saling terhubung melalui API, platform e-commerce dapat meningkatkan skalabilitas, fleksibilitas, dan efisiensi sistem secara keseluruhan. Selain itu, penerapan microservices juga memastikan bahwa setiap layanan dapat

dikelola, dikembangkan, dan di-deploy secara terpisah, serta dapat beradaptasi dengan kebutuhan baru tanpa memengaruhi keseluruhan sistem.

4. KESIMPULAN

Berdasarkan hasil penelitian dan pembahasan yang telah dilakukan, dapat disimpulkan bahwa penerapan arsitektur microservices pada sistem informasi penjualan online dapat meningkatkan skalabilitas dan kinerja sistem secara signifikan. Melalui pemecahan aplikasi menjadi layanan-layanan kecil yang mandiri, setiap bagian dari sistem dapat dikembangkan, diuji, dan di-deploy secara terpisah, tanpa mengganggu keseluruhan aplikasi. Salah satu keuntungan utama dari arsitektur microservices ini adalah kemampuannya untuk menangani transaksi dalam jumlah besar dengan efisiensi tinggi dan keandalan yang terjaga.

Penggunaan API Shopee, seperti v2.product.get_category, dalam ekosistem microservices memungkinkan data kategori produk yang ada di Shopee untuk diakses dan dikelola dengan cara yang efisien oleh platform e-commerce. Integrasi API ini memungkinkan layanan seperti Product Management Service dan Category Service untuk saling berkomunikasi dengan lancar dan menyediakan data yang dibutuhkan oleh sistem lain dalam arsitektur.

Selain itu, dengan menggunakan teknologi cloud computing dan containerization, penerapan microservices menjadi lebih mudah dan efisien. Hal ini memastikan bahwa sistem dapat beradaptasi dengan kebutuhan baru dan meminimalkan masalah skalabilitas yang biasanya dihadapi oleh sistem berbasis monolitik.

Namun, meskipun terdapat banyak keuntungan, penerapan microservices juga membawa tantangan, terutama terkait dengan komunikasi antar layanan, pengelolaan autentikasi, dan penanganan error yang lebih kompleks. Oleh karena itu, penggunaan API Gateway dan strategi fallback serta retry mechanism menjadi penting untuk memastikan sistem tetap berfungsi dengan baik meskipun ada gangguan.

DAFTAR REFERENSI

Alchuluq, L. M., & Nurzaman, F. (2021). Analisis pada arsitektur microservice untuk layanan bisnis toko online. *TEKINFO, 22*(2), 61–68. http://ejournal.unib.ac.id/index.php/rekursif

Anwar, M. D., & Kautsar, I. A. (2024). Arsitektur perangkat lunak berbasis layanan mikro pada sistem manajemen informasi kantin. *Physical Sciences, Life Science and Engineering, I*(2), 1–13. https://doi.org/10.47134/pslse.v1i2.196

Asrowardi, I., Putra, S. D., & Subyantoro, E. (2020). Designing microservice architectures for scalability and reliability in e-commerce. *Journal of Physics: Conference Series, 1450*(1), 012077. https://doi.org/10.1088/1742-6596/1450/1/012077

Atmojo, S., Utami, R., Dewi, S., & Widhiyanta, N. (2022). Implementasi sistem informasi desa berbasis arsitektur microservices. *SMATIKA: STIKI Informatika Journal, 12*(1), 55–66. https://doi.org/10.32664/smatika.v12i01.658

Belluano, P. L. L., Purnawansyah, Panggabean, B. L. E., & Herman. (2020). Sistem informasi program kreativitas mahasiswa berbasis web service dan microservice. *ILKOM Jurnal Ilmiah, 12*(1), 8–16. https://doi.org/10.33096/ilkom.v12i1.492.8-16

Dahri, F., Elhanafi, A. M., Handoko, D., & Wulan, N. (2022). Implementation of microservices architecture in learning management system e-course using web service method. *Sinkron: Jurnal dan Penelitian Teknik Informatika, 6*(1), 76–82. https://doi.org/10.33395/sinkron.v7i1.11229

Dinova, C. A., & Utomo, I. C. (2023). Pengembangan arsitektur microservice pada learning management system e-learning menggunakan metode web service. *Jurnal Ilmu Komputer dan Informatika (JIKI), 3*(2), 125–141. https://doi.org/10.54082/jiki.102

Iqbal, M., Handoko, W., & Syahputra, A. K. (2023). Optimalisasi e-learning melalui implementasi microservice untuk peningkatan skalabilitas dan efisiensi pembelajaran online. *JURDIMAS (Jurnal Pengabdian Kepada Masyarakat), 6*(3), 439–444. https://doi.org/10.33330/jurdimas.v6i3.2496

Mulyawan, A. A., Kusumasari, T. F., & Alama, E. N. (2022). Sistem pengelolaan target perusahaan dengan microservices architecture untuk membantu peningkatan kinerja perusahaan. *JATISI (Jurnal Teknik Informatika dan Sistem Informasi), 9*(1), 12–22.

Sinambela, A., Ernawati, & Coastera, F. F. (2021). Implementasi arsitektur microservices pada rancang bangun aplikasi marketplace berbasis web. *Jurnal Rekursif, 9*(1), 1–13. https://doi.org/10.32664/rekursif.v9i1.658