Proceeding of the Informatics Conference Vol. 10 No. 21, Guest Editor: Setyawan Widyarto

Implementasi Arsitektur *Microservices* Dalam Pengembangan *Marketplace* Sewa Properti

Brana Pandega[1)], and Setyawan Widyarto[2)]

[1)]*Universitas Budi Luhur, Jl. Ciledug Raya, RT.10/RW.2, Petukangan Utara, Kec. Pesanggrahan, Kota Jakarta Selatan, DKI Jakarta 12260, Indonesia*

[2)]*Centre for Graduate Studies, Berlian Building, 4th Floor, Universiti Selangor, Jalan Zirkon A7/A Seksyen 7, Shah Alam, 40000, Selangor Darul Ehsan, Malaysia*

***Abstract*—Microservices architecture is designed to improve system flexibility, scalability, and maintainability by decomposing applications into small, independent services. This study aims to analyze business and technical requirements, design an appropriate system architecture, and evaluate the performance of individual services in a property rental marketplace platform. The research methodology includes system requirements analysis, microservices architecture design, and service performance testing to ensure that the system meets predefined functional and non-functional requirements. The results indicate that the adoption of microservices architecture enhances development flexibility, enabling system updates and maintenance to be performed without disrupting the entire application. Furthermore, each service can be scaled independently, addressing performance limitations commonly encountered in monolithic systems. However, the implementation of microservices also introduces several challenges, including increased complexity in inter-service communication and distributed data management. Overall, the study concludes that microservices architecture is an effective approach for developing property rental marketplace systems that require high levels of flexibility and scalability.**

***Abstrak*—Arsitektur microservices dikembangkan untuk meningkatkan fleksibilitas, skalabilitas, dan kemudahan pemeliharaan sistem dengan memecah aplikasi menjadi layanan-layanan kecil yang independen. Penelitian ini bertujuan untuk menganalisis kebutuhan bisnis dan teknis, merancang arsitektur sistem yang sesuai, serta mengevaluasi kinerja setiap layanan pada sebuah marketplace sewa properti. Metode yang digunakan meliputi analisis kebutuhan sistem, perancangan arsitektur microservices, serta pengujian kinerja layanan untuk memastikan sistem memenuhi kriteria fungsional dan nonfungsional yang ditetapkan. Hasil penelitian menunjukkan bahwa penerapan arsitektur microservices mampu meningkatkan fleksibilitas pengembangan, sehingga pembaruan dan perbaikan sistem dapat dilakukan tanpa mengganggu keseluruhan aplikasi. Selain itu, setiap layanan dapat diskalakan secara independen untuk mengatasi permasalahan performa yang sering muncul pada arsitektur monolitik. Namun demikian, implementasi microservices juga menghadapi beberapa tantangan, seperti kompleksitas komunikasi antar layanan dan pengelolaan data terdistribusi. Secara keseluruhan, penelitian ini menyimpulkan bahwa arsitektur microservices merupakan pendekatan yang efektif untuk pengembangan marketplace sewa properti yang membutuhkan fleksibilitas dan skalabilitas tinggi.**

***Index Terms*— implementasi microservice, kekurangan, kelebihan, microservices architecture, MSA**

I. PENDAHULUAN

Microservices Architecture (MSA) adalah pendekatan dalam pengembangan perangkat lunak yang merancang aplikasi sebagai sekelompok layanan kecil, independen, dan terpisah yang berfungsi bersama untuk menjalankan aplikasi secara keseluruhan. Setiap layanan pada MSA ini dirancang untuk melakukan tugas-tugas tertentu dan dapat berkomunikasi dengan layanan lainnya melalui protokol yang standar. Dalam era di mana teknologi berkembang dengan sangat cepat, MSA telah menjadi salah satu pendekatan yang paling populer dalam pengembangan perangkat lunak. Dengan memecah aplikasi menjadi berbagai komponen independen yang disebut microservices, arsitektur ini menawarkan fleksibilitas, skalabilitas, dan kemampuan pemeliharaan yang baik.

Dalam literatur berjudul *"Monolithic vs. Microservice Architecture: A Performance and Scalability Evaluation"* [1] yang dipublikasikan tahun 2022, dijelaskan bahwa arsitektur berbasis *microservices* telah mendapatkan popularitas yang luas karena berbagai kelebihannya, seperti peningkatan ketersediaan, toleransi terhadap kesalahan, lebih mudah dalam melakukan perubahan skalabilitas horizontal, serta *agility* yang lebih baik ketika dikaitkan dengan pengembangan perangkat lunak.

Marketplace sewa properti AESIA adalah platform yang memungkinkan pemilik properti untuk menyewakan properti mereka dan penyewa untuk menemukan properti yang sesuai dengan kebutuhan mereka. Dengan semakin meningkatnya permintaan akan sewa properti, kebutuhan akan sistem yang dapat mengelola banyak *request* secara efisien dan *scalable* menjadi sangat penting. Arsitektur monolitik tradisional sering kali menghadapi kendala dalam hal skalabilitas dan

pemeliharaan ketika berhadapan dengan pertumbuhan pengguna dan fitur yang pesat.

Implementasi MSA dalam pengembangan marketplace sewa properti seperti AESIA menawarkan solusi terhadap kendala tersebut. Dengan memecah fungsionalitas aplikasi menjadi layanan-layanan kecil yang independen, setiap layanan dapat dikembangkan, diuji, dan di-deploy secara terpisah, yang pada akhirnya meningkatkan fleksibilitas dan efisiensi pengembangan. Selain itu, MSA memungkinkan skalabilitas yang lebih baik karena setiap layanan dapat diskalakan secara independen sesuai kebutuhan.

Berdasarkan latar belakang tersebut, rumusan masalah dalam penelitian ini adalah sebagai berikut:

1. Bagaimana cara mengimplementasikan arsitektur *microservices* dalam pengembangan marketplace sewa properti.
2. Apa saja tantangan dan keuntungan yang dihadapi dalam penerapan MSA pada marketplace sewa properti.
3. Bagaimana MSA dapat meningkatkan fleksibilitas, skalabilitas, dan kemampuan pemeliharaan sistem marketplace sewa properti.

Penelitian ini bertujuan untuk menjelaskan proses implementasi arsitektur *microservices* dalam pengembangan marketplace sewa properti, mengidentifikasi tantangan dan keuntungan yang muncul dari penerapan MSA pada marketplace sewa properti, serta mengevaluasi dampak penggunaan MSA terhadap fleksibilitas, skalabilitas, dan kemampuan pemeliharaan sistem marketplace sewa properti.

Penelitian ini diharapkan dapat memberikan kontribusi sebagai berikut:

1. Menyediakan panduan praktis dan studi kasus yang detail mengenai implementasi arsitektur *microservices* dalam marketplace sewa properti.
2. Memberikan wawasan tentang keuntungan dan tantangan yang dihadapi dalam penerapan MSA, yang dapat digunakan sebagai referensi oleh pengembang dan manajer proyek perangkat lunak.
3. Menyumbangkan pengetahuan mengenai dampak arsitektur *microservices* terhadap kinerja dan efisiensi sistem marketplace, yang dapat membantu dalam pengambilan keputusan arsitektur pada proyek pengembangan perangkat lunak serupa di masa depan.

Dengan demikian, penelitian ini akan mengkaji secara mendalam bagaimana arsitektur *microservices* dapat diterapkan dalam konteks marketplace sewa properti, serta memberikan pandangan yang komprehensif mengenai manfaat dan tantangan dari pendekatan ini.

II. METODE PENELITIAN

Penelitian ini menggunakan pendekatan studi kasus untuk mengeksplorasi implementasi arsitektur microservices dalam pengembangan marketplace sewa properti AESIA. Studi kasus dipilih karena memungkinkan analisis mendalam terhadap proses implementasi dan evaluasi kinerja sistem yang dihasilkan. Pendekatan ini juga membantu dalam memahami konteks spesifik dan tantangan yang dihadapi selama pengembangan.

Penelitian ini dilakukan melalui beberapa tahapan, yaitu:

1. Analisis kebutuhan, dengan mengidentifikasi kebutuhan bisnis dan teknis dari marketplace sewa properti, termasuk fungsionalitas utama, volume transaksi, dan kebutuhan skalabilitas.
2. Desain arsitektur, yaitu merancang arsitektur microservices yang sesuai dengan kebutuhan yang telah diidentifikasi. Proses ini melibatkan pembuatan diagram arsitektur, identifikasi layanan microservices, dan pemilihan teknologi yang tepat.
3. Implementasi. Membangun marketplace sewa properti berdasarkan desain arsitektur yang telah dibuat. Setiap layanan microservices dikembangkan, diuji, dan di-deploy secara terpisah.
4. Pengujian dan evaluasi. Menguji kinerja dan keandalan sistem yang telah diimplementasikan. Evaluasi dilakukan dengan mengukur metrik-metrik kinerja seperti latensi dan penggunaan sumber daya.

Proses implementasi microservices dilakukan melalui langkah-langkah berikut:

1. Identifikasi Layanan Microservices: Setiap layanan diidentifikasi berdasarkan fungsionalitas bisnis, seperti layanan autentikasi pengguna, layanan pengelolaan properti, dan layanan transaksi pembayaran.
2. Pengembangan Layanan: Setiap layanan dikembangkan secara independen menggunakan bahasa pemrograman dan kerangka kerja yang paling sesuai. API RESTful digunakan untuk komunikasi antar layanan.
3. Penyimpanan Data: Menggunakan database yang terpisah untuk setiap layanan, sesuai dengan kebutuhan spesifiknya. Contoh teknologi yang digunakan meliputi PostgreSQL, MongoDB, dan Redis.

Berikut adalah alat dan teknologi yang digunakan dalam penelitian ini:

1. Framework Pengembangan: CodeIgniter, VueJS dan Node.js
2. Database: MySQL
3. Web Server: NGINX
4. Penyimpanan Objek: MinIO
5. Alat Pengujian: Postman dan Web Browser
6. Platform Deployment: Docker dan Docker Compose

Dengan metodologi tersebut, penelitian diharapkan dapat memberikan pandangan yang mendalam tentang implementasi arsitektur microservices dalam pengembangan marketplace sewa properti, serta mengidentifikasi manfaat dan tantangan

yang dihadapinya.

III. HASIL DAN PEMBAHASAN

A. *Analisis Kebutuhan*

Pada tahap perencanaan pengembangan marketplace sewa properti AESIA, dilakukan analisis kebutuhan untuk mengidentifikasi fitur-fitur utama yang harus disediakan oleh sistem. Beberapa fitur yang diidentifikasi antara lain: Pencarian Properti, Peta Properti, Kontak Pemilik Properti, Manajemen Data Properti, Otentikasi Pengguna dan AI Chatbot. Masing-masing fitur ini memiliki karakteristik dan kebutuhan spesifik yang mempengaruhi desain arsitektur microservices. Berikut adalah analisis kebutuhan dari fitur-fitur tersebut:

1) *Pencarian Properti*

Fitur pencarian properti merupakan salah satu fitur utama yang membutuhkan banyak query read yang kompleks. Pengguna harus dapat melakukan pencarian dengan kriteria yang sangat spesifik, seperti jumlah kamar, lokasi (geospasial), harga, dan lainnya. Selain itu, pencarian harus toleran terhadap kesalahan pengetikan (typo) dan memberikan hasil yang relevan.

Untuk memenuhi kebutuhan ini, service khusus pencarian perlu dibangun terpisah menggunakan Elasticsearch. Elasticsearch dipilih karena kemampuannya dalam menangani pencarian full-text dan query yang kompleks dengan performa tinggi. Elasticsearch juga mendukung pencarian geospasial dan dapat diintegrasikan dengan mudah ke dalam arsitektur microservices.

2) *Kontak Pemilik Properti dan Otentikasi Pengguna*

Fitur kontak pemilik properti memungkinkan pengguna untuk menghubungi pemilik properti melalui form kontak, yang kemudian sistem akan mengirimkan datanya melalui e-mail dan WhatsApp ke pemilik properti.

Fitur otentikasi pengguna mencakup pendaftaran, login, dan otorisasi pengguna. Dalam proses otentikasi, sistem perlu mengirimkan kode verifikasi dan kode OTP ke e-mail dan WhatsApp pengguna.

Mengingat cukup rumitnya sistem E-mail gateway dan WhatsApp gateway apabila digabung dengan fitur-fitur utama, maka untuk memenuhi kebutuhan dua fitur tersebut, service khusus E-mail gateway dan WhatsApp gateway perlu dibangun terpisah.

3) *Manajemen Data Properti*

Fitur manajemen data properti meliputi penambahan, pembaruan, dan penghapusan data properti oleh pemilik properti. Fitur ini memerlukan operasi CRUD (Create, Read, Update, Delete) yang efisien dan cepat. Selain itu, dalam fitur manajemen data properti juga terdapat proses upload dan download foto dan file lampiran yang jumlahnya cukup banyak dan akan terus bertumbuh seiring dengan penambahan jumlah properti yang ditayangkan di marketplace AESIA.

Agar beban kerja dan traffic sistem server utama tidak terbebani dengan aktivitas upload, download dan pemrosesan file, maka perlu dibangun service penyimpanan (storage service) terpisah yang menggunakan protokol Simple Storage Service (S3), yaitu MinIO.

4) *AI Chatbot*

Fitur AI Chatbot memungkinkan pengguna bertanya kepada bot berbasis Artificial Intelligence (AI) mengenai properti dan informasi umum terkait sewa-menyewa properti di marketplace AESIA. Fitur ini memerlukan akses ke API OpenAI atau ChatGPT. Mengingat pemrosesan chat memerlukan sumber daya yang cukup besar dan waktu yang tidak sedikit, apabila service AI Chatbot ini digabungkan dengan layanan utama dikhawatirkan berpotensi menimbulkan bottleneck dan menambah kerumitan. Untuk itu perlu dibangun service chatbot terpisah yang berfungsi memproses chat dan berinteraksi dengan API OpenAI secara independen.

Dengan memecah layanan berdasarkan kebutuhan spesifik ini, arsitektur microservices yang dihasilkan dapat lebih fleksibel, mudah diskalakan, dan responsif terhadap perubahan beban kerja. Pendekatan ini juga memungkinkan pengembangan dan pemeliharaan sistem yang lebih terstruktur dan efisien.

B. *Desain dan Implementasi MSA*

Desain arsitektur microservices untuk marketplace sewa properti ini dirancang untuk memenuhi kebutuhan spesifik dari berbagai fitur yang diidentifikasi pada tahap analisis kebutuhan. Desain arsitektur ini menekankan pada pemisahan tanggung jawab dan skalabilitas, serta penggunaan teknologi yang sesuai untuk setiap layanan.

Arsitektur sistem dibagi menjadi beberapa service lebih kecil yang saling berkomunikasi melalui API REST. Setiap layanan dirancang untuk independen, dengan tanggung jawab spesifik dan dapat diskalakan secara terpisah. Komunikasi antar layanan menggunakan protokol HTTP/HTTPS untuk memastikan keamanan dan kompatibilitas.

1) *Service Manajemen Data Properti*

Layanan ini adalah layanan utama yang bertanggung jawab untuk operasi CRUD (Create, Read, Update, Delete) terkait data properti. Layanan ini dibangun menggunakan Bahasa PHP dengan framework CodeIgniter, dan menggunakan MySQL sebagai database.

2) *Service Pencarian Properti*

Layanan ini menggunakan Elasticsearch untuk menangani query pencarian yang kompleks, termasuk pencarian full-text, geospasial, dan toleransi terhadap typo. Layanan ini memiliki endpoint yang menerima permintaan pencarian dari pengguna dan mengembalikan hasil pencarian yang relevan.

3) *Service E-Mail Gateway*

Layanan ini mengelola pengiriman e-mail untuk berbagai

kebutuhan sistem, misalnya e-mail yang berisi kode verifikasi, kode OTP dan notifikasi. Gateway ini dikembangkan menggunakan bahasa PHP (CodeIgniter), memanfaatkan message broker RabbitMQ untuk mengelola antrian pengiriman e-mail, dan server SMTP untuk mengirimkan e-mail.

4) Service WhatsApp Gateway

Serupa dengan e-mail gateway, layanan WhatsApp Gateway ini bertugas mengelola pengiriman pesan WhatsApp untuk berbagai kebutuhan sistem, misalnya untuk pengiriman kode verifikasi, kode OTP dan notifikasi ke pengguna. Layanan ini dikembangkan menggunakan bahasa Javascript (NodeJS), memanfaatkan message broker RabbitMQ untuk mengelola antrian pengiriman e-mail, dan library whatsapp-web.js untuk mengirimkan pesan.

5) Service Penyimpanan File

Layanan penyimpanan file bertugas melayani proses upload, view/download, penyimpanan dan pemrosesan file. Layanan ini menggunakan MinIO yang berjalan di atas protokol Simple Storage Service (S3), protokol yang sama seperti yang digunakan layanan AmazonS3. Dalam implementasinya untuk layanan penyimpanan file pada marketplace AESIA, MinIO dikombinasikan dengan modul image filter NGINX (ngx_http_image_filter_module), sehingga layanan ini juga dapat melakukan resize dan crop gambar secara instan.

6) Service Chatbot

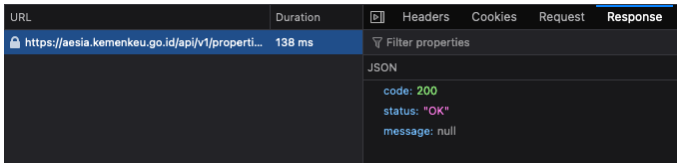
Layanan chatbot bertugas memproses chat, mengirim prompt ke server API OpenAI dan memberikan balasan chat. Layanan ini dikembangkan menggunakan bahasa PHP dengan framework CodeIgniter dan menggunakan database MySQL.

C. Pengujian dan Evaluasi

Pengujian dan evaluasi dilakukan untuk memastikan bahwa setiap layanan dalam arsitektur mikroservis berfungsi sesuai dengan yang diharapkan dan memenuhi kriteria kinerja yang telah ditetapkan. Berikut adalah hasil pengujian untuk masing-masing layanan:

1) Service Manajemen Data Properti

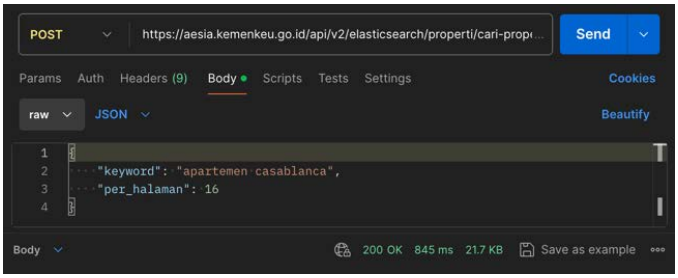
Pengujian dilakukan dengan mengirimkan request untuk memperbarui data properti. Hasil pengujian menunjukkan bahwa fungsi berjalan dengan baik, dengan waktu respons sebesar 138ms sebagaimana terlihat pada Gambar 1. Waktu respons yang cepat menunjukkan bahwa layanan manajemen data properti mampu menangani request dengan efisien.



Gambar 1. Hasil Pengujian Service Data Properti

2) Service Pencarian Properti

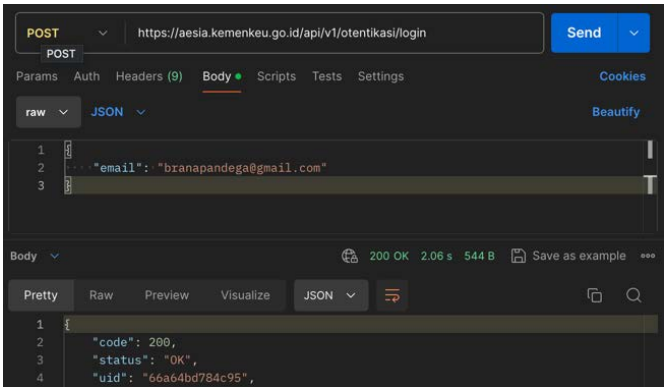
Pengujian layanan pencarian properti dilakukan dengan mengirimkan request pencarian menggunakan keyword yang terdiri dari dua kata, baik yang tidak mengandung typo maupun yang mengandung typo. Hasil pengujian menunjukkan bahwa layanan berfungsi dengan baik, dengan waktu respons sebesar 845ms (Gambar 2) dan cukup stabil di bawah 1000ms ketika dicoba beberapa kali.



Gambar 2. Hasil Pengujian Service Pencarian Properti

3) Service E-mail dan WhatsApp Gateway

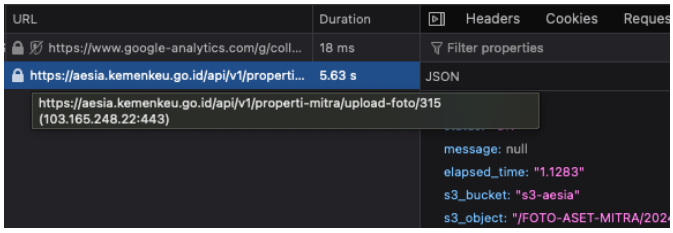
Pengujian layanan e-mail gateway dan WhatsApp gateway dilakukan dengan mengirim request ke endpoint login. Endpoint login mengirimkan request ke kedua gateway secara bersamaan. Hasil pengujian pada Gambar 3 menunjukkan bahwa fungsi berjalan dengan baik, dengan waktu respons sebesar 2,06 detik.



Gambar 3. Hasil Pengujian E-mail dan WhatsApp Gateway

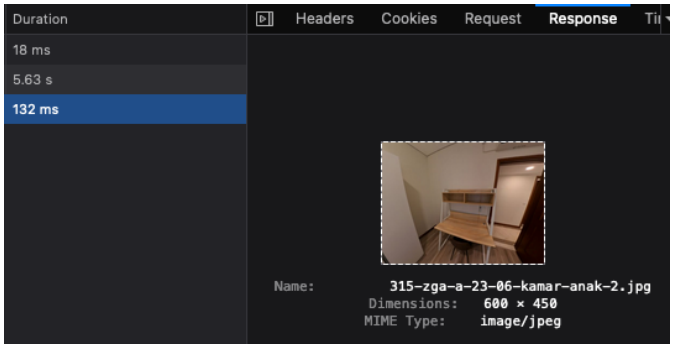
4) Service Penyimpanan File

Pengujian layanan penyimpanan file dilakukan dalam dua tahap, yaitu upload dan download. Pengujian upload dilakukan dengan mengunggah sebuah file gambar berukuran 900KB. Hasil pengujian (Gambar 4) menunjukkan bahwa layanan berfungsi dengan baik, dengan durasi total 5,63 detik dan durasi eksekusi script di sisi server hanya 1,12 detik. Durasi total yang lebih lama disebabkan oleh waktu transfer data dari klien ke server.



Gambar 4. Hasil Pengujian Upload ke Server Penyimpanan File

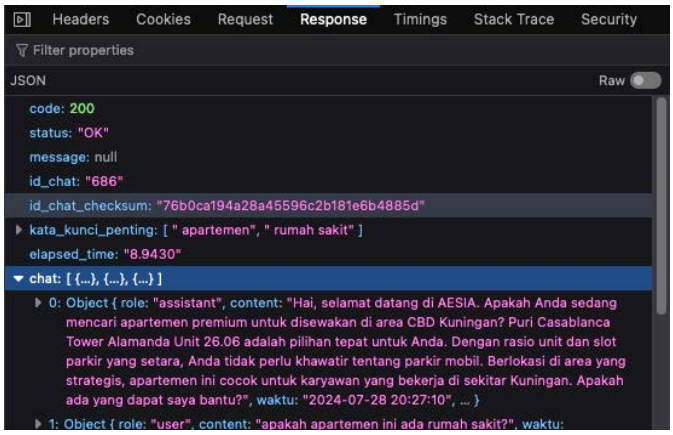
Pengujian download dilakukan dengan mengunduh file gambar properti yang diupload sebelumnya. Server penyimpanan file berhasil melakukan proses resize, crop, dan kompresi dengan baik dan menghasilkan file berresolusi 600x450 pixel, dan berukuran 22,8KB yang selesai diunduh oleh browser dalam waktu 132ms sebagaimana dapat dilihat pada Gambar 5.



Gambar 5. Hasil Pengujian Download File

5) Service Chatbot

Pengujian layanan chatbot dilakukan dengan mengirimkan request yang berisi pertanyaan mengenai sebuah properti. Sebagaimana dapat dilihat pada Gambar 6, hasil pengujian menunjukkan bahwa layanan berfungsi dengan baik, namun server membutuhkan waktu selama 8,94 detik untuk merespon chat tersebut. Lamanya waktu eksekusi ini disebabkan oleh proses pencarian konteks yang relevan ke database dan pengiriman prompt ke server API OpenAI.



Gambar 6. Hasil Pengujian Service Chatbot

IV. DISKUSI

A. *Evaluasi Implementasi Microservices*

Implementasi arsitektur microservices pada marketplace sewa properti AESIA menunjukkan bahwa pendekatan ini berhasil mengatasi beberapa kendala yang dihadapi dalam arsitektur monolitik tradisional. Penggunaan microservices memungkinkan pemisahan tanggung jawab antar layanan, yang menghasilkan peningkatan fleksibilitas, skalabilitas, dan kemudahan pemeliharaan sistem.

1) Fleksibilitas

Setiap layanan yang diimplementasikan pada sistem ini memiliki tanggung jawab dan fungsi yang spesifik. Hal ini memungkinkan tim pengembangan untuk mengembangkan, menguji, dan melakukan deployment layanan secara independen. Fleksibilitas ini terlihat pada layanan seperti manajemen data properti, pencarian properti, dan gateway komunikasi (e-mail dan WhatsApp), yang masing-masing dapat diperbarui atau diperbaiki tanpa mempengaruhi layanan lain.

2) Skalabilitas

Salah satu keunggulan utama dari microservices adalah kemampuannya untuk diskalakan secara independen. Hasil pengujian menunjukkan bahwa layanan pencarian properti, yang memanfaatkan Elasticsearch, mampu menangani query kompleks dengan waktu respons yang stabil di bawah 1000ms. Kemampuan ini sangat penting mengingat volume query pencarian yang tinggi pada marketplace sewa properti. Demikian juga, layanan penyimpanan file dengan MinIO mampu mengelola proses upload dan download file secara efisien, memastikan performa yang optimal meskipun terjadi peningkatan jumlah pengguna dan data.

3) Kemampuan Pemeliharaan

Pemisahan layanan berdasarkan fungsionalitasnya memudahkan dalam pemeliharaan sistem. Setiap layanan dapat diuji dan didebug secara terpisah, yang mengurangi kompleksitas dalam menemukan dan memperbaiki bug. Selain itu, pemisahan ini juga memungkinkan adopsi teknologi yang paling sesuai untuk setiap layanan, seperti penggunaan Node.js pada layanan WhatsApp Gateway dan PHP dengan CodeIgniter pada layanan lainnya.

B. *Tantangan Implementasi Microservices*

Meskipun arsitektur microservices menawarkan banyak keunggulan, implementasinya juga tidak terlepas dari tantangan. Beberapa tantangan utama yang dihadapi dalam penelitian ini antara lain:

1) Kompleksitas Komunikasi Antar Layanan

Komunikasi antar layanan melalui API REST menambah kompleksitas sistem. Setiap layanan harus dirancang untuk dapat berkomunikasi dengan layanan lain secara efisien dan aman. Penggunaan protokol HTTP/HTTPS membantu dalam menjaga keamanan komunikasi, namun juga menambah overhead dalam hal latensi dan manajemen request.

2) Pengelolaan Data yang Terdistribusi

Dengan setiap layanan memiliki database terpisah, konsistensi data menjadi tantangan. Implementasi ini memerlukan strategi yang efektif untuk sinkronisasi dan pengelolaan data terdistribusi, terutama untuk layanan yang saling bergantung, seperti layanan otentikasi pengguna dan manajemen data properti.

3) Pengujian dan Debugging

Pengujian dan debugging pada arsitektur microservices lebih menantang dibandingkan arsitektur monolitik. Setiap layanan harus diuji secara independen serta dalam konteks integrasi dengan layanan lain. Penggunaan alat pengujian seperti Postman dan Web Developer Tools pada browser cukup membantu dalam pengujian fungsionalitas dan performa, namun tetap memerlukan waktu dan usaha lebih untuk memastikan setiap layanan berfungsi dengan baik dalam skenario yang kompleks.

C. Dampak terhadap Kinerja dan Efisiensi

Dari hasil pengujian yang telah dilakukan, dapat disimpulkan bahwa arsitektur microservices memberikan dampak positif terhadap kinerja dan efisiensi sistem marketplace sewa properti AESIA. Waktu respons yang cepat pada layanan manajemen data properti dan layanan pencarian properti menunjukkan bahwa sistem mampu menangani request dengan efisien. Penggunaan message broker RabbitMQ pada layanan e-mail dan WhatsApp Gateway juga memastikan pengiriman pesan yang andal dan tepat waktu.

1) Kinerja Sistem

Pengujian layanan menunjukkan bahwa waktu respons yang diperoleh cukup baik dan stabil. Layanan manajemen data properti memiliki waktu respons sebesar 138ms, layanan pencarian properti sekitar 845ms, dan layanan pengiriman e-mail dan WhatsApp sekitar 2,06 detik. Meskipun layanan chatbot memiliki waktu respons yang lebih lama (8,94 detik), hal ini dapat dimaklumi mengingat proses yang kompleks dalam mengakses API OpenAI.

2) Efisiensi Pengembangan dan Pemeliharaan

Pendekatan microservices memungkinkan pengembangan yang lebih terstruktur dan efisien. Setiap layanan dapat dikembangkan oleh tim yang berbeda dengan fokus pada tanggung jawab spesifiknya, yang mempercepat proses pengembangan secara keseluruhan. Pemeliharaan sistem juga menjadi lebih mudah karena setiap layanan dapat diperbaiki atau ditingkatkan secara independen tanpa harus melakukan perubahan besar pada seluruh sistem.

D. Implikasi dan Rekomendasi

Berdasarkan hasil penelitian ini, beberapa implikasi dan rekomendasi dapat disampaikan:

1) Implementasi di Proyek Serupa

Penerapan arsitektur microservices dapat menjadi solusi efektif untuk proyek pengembangan perangkat lunak serupa, terutama yang memerlukan skalabilitas dan fleksibilitas tinggi. Pengembang dan manajer proyek dapat menggunakan hasil penelitian ini sebagai referensi dalam merancang dan mengimplementasikan microservices pada sistem mereka.

2) Penggunaan Teknologi yang Sesuai

Pemilihan teknologi yang sesuai untuk setiap layanan sangat penting dalam arsitektur microservices. Teknologi seperti Elasticsearch untuk layanan pencarian, MinIO untuk layanan penyimpanan file, dan message broker RabbitMQ untuk pengelolaan antrian pesan telah terbukti efektif dalam penelitian ini. Pengembang perlu mempertimbangkan kebutuhan spesifik setiap layanan dalam memilih teknologi yang akan digunakan.

3) Manajemen Kompleksitas

Manajemen kompleksitas dalam komunikasi antar layanan dan pengelolaan data terdistribusi perlu menjadi fokus dalam implementasi microservices. Penggunaan alat bantu seperti API Gateway dan layanan manajemen konfigurasi dapat membantu dalam mengelola kompleksitas ini. Selain itu, strategi pengujian yang komprehensif sangat diperlukan untuk memastikan setiap layanan berfungsi dengan baik dalam lingkungan yang kompleks.

Dengan demikian, implementasi arsitektur microservices pada marketplace sewa properti AESIA telah memberikan wawasan yang berharga mengenai manfaat dan tantangan pendekatan ini. Penelitian ini diharapkan dapat menjadi panduan praktis bagi pengembang dan manajer proyek dalam mengadopsi microservices untuk sistem mereka, serta mendorong penelitian lebih lanjut dalam bidang ini.

V. KESIMPULAN

Penelitian ini menunjukkan bahwa implementasi arsitektur microservices pada marketplace sewa properti AESIA berhasil meningkatkan fleksibilitas, skalabilitas, dan kemampuan pemeliharaan sistem. Pemisahan layanan berdasarkan fungsionalitas memungkinkan pengembangan, pengujian, dan deployment yang lebih terstruktur dan independen, sehingga mempermudah tim pengembang dalam melakukan pembaruan dan perbaikan tanpa mengganggu keseluruhan sistem. Selain itu, hasil pengujian menunjukkan bahwa layanan seperti pencarian properti dengan Elasticsearch dan penyimpanan file dengan MinIO mampu menangani beban kerja yang kompleks secara efisien.

Meskipun menawarkan banyak keuntungan, implementasi arsitektur microservices juga menghadapi beberapa tantangan. Kompleksitas komunikasi antar layanan, pengelolaan data terdistribusi, serta pengujian dan debugging yang lebih rumit dibandingkan arsitektur monolitik merupakan beberapa kendala yang perlu diatasi. Penggunaan alat pengujian seperti Postman dan Apache JMeter membantu dalam mengelola kompleksitas ini, namun tetap memerlukan strategi yang lebih mendalam untuk memastikan integritas dan kinerja sistem.

Secara keseluruhan, penelitian ini memberikan wawasan berharga tentang manfaat dan tantangan implementasi

arsitektur microservices pada marketplace sewa properti. Temuan ini dapat menjadi panduan praktis bagi pengembang dan manajer proyek dalam merancang dan mengimplementasikan microservices, serta mendorong penelitian lebih lanjut dalam bidang ini untuk mengatasi tantangan yang ada dan meningkatkan efektivitas sistem.

DAFTAR PUSTAKA

[1] G. Blinowski, A. Ojdowska, and A. Przybyłek. (2022). "Monolithic vs. Microservice Architecture: A Performance and Scalability Evaluation." *IEEE Access*. [Online]. vol. 10, pp. 20357-20374. Available: https://doi.org/10.1109/ACCESS.2022.3152803

[2] C. Lai, F. Boi, A. Buschettu and R. Caboni. (2019). "IoT and Microservice Architecture for Multimobility in a Smart City." in *2019 7th International Conference on Future Internet of Things and Cloud (FiCloud)*. [Online]. pp. 238-242. Available: https://doi.org/10.1109/FiCloud.2019.00040

[3] G. Ortiz, J. Boubeta-Puig, J. Criado, D. Corral-Plaza, A. Garcia-de-Prado, I. Medina-Bulo, and L. Iribarne. (2022). "A microservice architecture for real-time IoT data processing: A reusable Web of things approach for smart ports." *Computer Standards & Interfaces*. [Online]. vol. 81, pp. 103604. Available: https://doi.org/10.1016/j.csi.2021.103604

[4] M. Jin et al. (2020). "An Anomaly Detection Algorithm for Microservice Architecture Based on Robust Principal Component Analysis." *IEEE Access*. [Online]. vol. 8, pp. 226397-226408. Available: https://doi.org/10.1109/ACCESS.2020.3044610

[5] M. Nekovee, S. Sharma, N. Uniyal, A. Nag, R. Nejabati and D. Simeonidou. (2020). "Towards AI-enabled Microservice Architecture for Network Function Virtualization." in *2020 IEEE Eighth International Conference on Communications and Networking (ComNet)*. [Online]. pp. 1-8. Available: https://doi.org/10.1109/ComNet47917.2020.9306098

[6] H. Zhang, S. Li, Z. Jia, C. Zhong and C. Zhang. (2019). "Microservice Architecture in Reality: An Industrial Inquiry." in *2019 IEEE International Conference on Software Architecture (ICSA)*. [Online]. pp. 51-60. Available: https://doi.org/10.1109/ICSA.2019.00014

[7] M. E. Gortney et al. (2022). "Visualizing Microservice Architecture in the Dynamic Perspective: A Systematic Mapping Study." *IEEE Access*. [Online]. vol. 10, pp. 119999-120012. Available: https://doi.org/10.1109/ACCESS.2022.3221130

[8] G. Ortiz, J. A. Caravaca, A. García-de-Prado, F. Chavez de la O and J. Boubeta-Puig. (2019). "Real-Time Context-Aware Microservice Architecture for Predictive Analytics and Smart Decision-Making." IEEE Access. [Online]. vol. 7, pp. 183177-183194. Available: https://doi.org/10.1109/ACCESS.2019.2960516

[9] M. L. Fanomezana, A. M. Rapatsalahy, N. R. Razafindrakoto and C. Bădică. (2022). "Proposed Methodology for Designing a Microservice Architecture." *in 2022 23rd International Carpathian Control Conference (ICCC)*. [Online]. pp. 303-308. Available: https://doi.org/10.1109/ICCC54292.2022.9805930

[10] H. Calderón-Gómez et al. (2020). "Telemonitoring System for Infectious Disease Prediction in Elderly People Based on a Novel Microservice Architecture." *IEEE Access*. [Online]. vol. 8, pp. 118340-118354. Available: https://doi.org/10.1109/ACCESS.2020.3005638