Terakreditasi SINTA Peringkat 2
Surat Keputusan Direktur Jenderal Pendidikan Tinggi, Riset, dan Teknologi, Nomor: 158/E/KPT/2021
masa berlaku mulai Volume 5 Nomor 2 Tahun 2021 sampai Volume 10 Nomor 1 Tahun 2026

Terbit online pada laman web jurnal: **http://jurnal.iaii.or.id**

JURNAL RESTI
(Rekayasa Sistem dan Teknologi Informasi)
Vol. 6 No. x (2022) x - x ISSN Media Elektronik: 2580-0760

Implementasi dan Pengujian Arsitektur Microservice pada Platform E-Commerce Menggunakan Docker dan Kubernetes

Indra Jayadi[1], Dwi Ely Kurniawan[2]
[1]Teknik Informatika, Politeknik Negeri Batam
[2]Teknologi Rekayasan Multimedia, Politeknik Negeri Batam
[1]indrajayadi264@gmail.com, [2]dwialikhs@polibatam.ac.id

Abstract

E-commerce platforms continue to grow rapidly, requiring systems that are scalable, reliable, and able to adapt to sudden increases in user demand. To meet these requirements, this study aims to implement and evaluate a microservices-based e-commerce platform using Docker and Kubernetes. The research method follows four stages: identification, planning, implementation, and evaluation. The system design consists of three independent services—User Service, Product Service, and Payment Service—integrated through REST APIs and deployed in a Kubernetes cluster with containerization provided by Docker. Through functional and performance testing, the study is expected to demonstrate that the proposed microservices architecture can provide low latency, improved scalability, and high system reliability to support e-commerce operations..

Keywords: microservices, Docker, Kubernetes, e-commerce, REST API

Abstrak

Platform e-commerce mengalami pertumbuhan pesat yang menuntut sistem berskala besar agar mampu menangani lonjakan pengguna secara efisien, stabil, dan andal. Untuk memenuhi kebutuhan tersebut, penelitian ini bertujuan untuk mengimplementasikan dan menguji arsitektur microservices pada platform e-commerce dengan memanfaatkan Docker dan Kubernetes. Metode penelitian menggunakan empat tahapan utama, yaitu identifikasi, perencanaan, pelaksanaan, dan tindakan. Rancangan sistem terdiri dari tiga layanan independen—User Service, Product Service, dan Payment Service—yang saling berkomunikasi melalui REST API dan dijalankan pada kluster Kubernetes dengan dukungan kontainerisasi Docker. Melalui pengujian fungsional dan performa, penelitian ini diharapkan dapat menunjukkan bahwa arsitektur microservices mampu memberikan latensi rendah, skalabilitas yang baik, serta keandalan sistem untuk mendukung operasional e-commerce.

Kata kunci: *microservices, Docker, Kubernetes, e-commerce, REST API*

1. Pendahuluan

Perkembangan e-commerce dalam satu dekade terakhir menunjukkan peningkatan yang sangat signifikan. Berbagai penelitian terkini melaporkan bahwa e-commerce telah mengalami pertumbuhan pesat[1], baik dari sisi nilai transaksi[2], pengguna internet[3], maupun UMKM yang berjualan online[4]. Perubahan perilaku masyarakat yang semakin bergantung pada layanan digital mendorong lonjakan jumlah pengguna dan transaksi e-commerce[5].

Kondisi tersebut menuntut platform e-commerce untuk memiliki kinerja dengan waktu respons yang cepat serta stabil meskipun menghadapi beban tinggi, serta mampu menyesuaikan diri terhadap lonjakan permintaan secara tiba-tiba. Namun, sebagian besar platform e-commerce saat ini masih menerapkan arsitektur monolitik[6]. Arsitektur monolitik memiliki keterbatasan dalam hal skalabilitas dan ketahanan sistem ketika menghadapi lonjakan beban secara tiba-tiba. Berdasarkan hasil penelitian Guntakandla (2025) keterbatasan ini dapat menyebabkan waktu respons melambat hingga 4,2 detik serta menurunkan performa sistem hingga 45% pada kondisi beban tinggi[7].Untuk menjawab permasalahan tersebut arsitektur microservice menjadi pilihan utama dan populer, platform e-commerce yang memanfaatkan arsitektur microservices menunjukkan peningkatan sebesar 45% lebih cepat dibandingkan sistem monolitik dengan waktu pemrosesan permintaan rata-rata berkurang dari 2,3 detik menjadi 1,2 detik[7].

Namun, penerapan microservice tidak bisa lepas dari kebutuhan akan teknologi pendukung yang memastikan setiap layanan dapat berjalan secara konsisten, terisolasi dan mudah dikelola. Pada titik inilah docker dan kubernetes memegang peran penting. Sejumlah

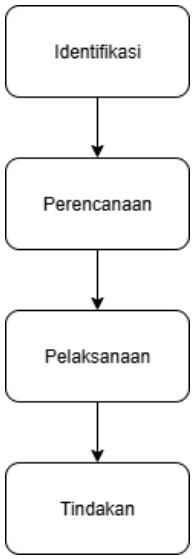
Diterima Redaksi: xx-xx-xxxx | Selesai Revisi: xx-xx-xxxx | Diterbitkan Online: xx-xx-xxxx

penelitian menunjukkan bahwa docker berperan penting dalam mengemas layanan microservice menjadi kontainer yang portable dan ringan, sedangkan kubernetes banyak diadopsi sebagai platform orkestrasi dengan kemampuan deployment otomatis, penskalaan horizontal, load balancing, dan pemulihan layanan[8][9][10][11].

Oleh karena itu, penelitian ini difokuskan pada implementasi arsitektur microservice pada platform e-commerce dengan dukungan Docker dan Kubernetes, serta pengujian kinerjanya secara langsung. Pengujian dilakukan menggunakan Postman untuk memastikan konektivitas antar layanan, serta Apache JMeter untuk melakukan uji beban dan menganalisis waktu respon sistem pada berbagai skenario penggunaan. Dengan pendekatan ini, penelitian diharapkan dapat menunjukkan performa arsitektur microservice yang diimplementasikan menggunakan Docker dan Kubernetes, serta mengukur kinerjanya melalui pengujian konektivitas dan uji beban pada platform e-commerce.

2. Metode Penelitian

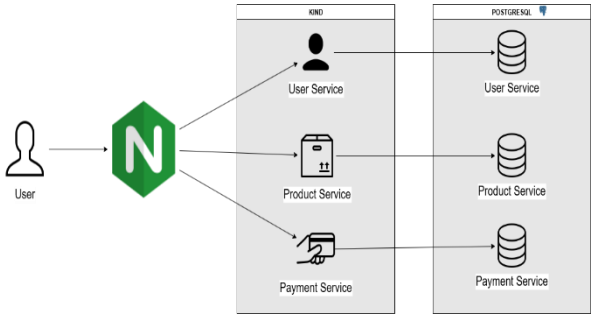
Penelitian ini menggunakan methode research and development karna menghasilkan prototipe sistem e-commerce berbasis microservice yang diimplementasikan dan diuji secara nyata. Metode ini relevan karena tidak hanya menjelaskan konsep, tetapi juga mengembangkan dan mengevaluasi produk sistem yang berfungsi penuh. Pendekatan ini mengacu pada tahapan penelitian (Suthendra & Pakereng, 2020)[12]. Dengan demikian, research method relevan digunakan untuk menilai sejauh mana arsitektur microservices yang dikembangkan dengan docker dan Kubernetes mampu mengatasi berbagai keterbatasan sistem monolitik, baik dari sisi skalabilitas, fleksibilitas, maupun efisiensi pengelolaan layanan. Tahapan penelitian ini disusun secara sistematis dalam empat langkah utama, yaitu identifikasi, perencanaan, pelaksanaan, dan tindakan, yang masing-masing saling berkesinambungan dalam menghasilkan sistem yang optimal.



Gambar 1. Research Method

Tahap identifikasi dilakukan untuk merumuskan fokus penelitian yang berkaitan dengan performa sistem e-commerce berbasis microservices. Dalam konteks ini, aspek utama yang menjadi perhatian adalah kecepatan waktu respons dan stabilitas sistem ketika menerima beban pengguna yang tinggi. Berdasarkan studi literatur, kedua aspek tersebut merupakan indikator penting dalam menilai kualitas layanan e-commerce, karena waktu respons yang lambat maupun sistem yang tidak stabil dapat menurunkan pengalaman pengguna. Oleh karena itu, penelitian ini diarahkan untuk membangun arsitektur microservices dengan docker dan kubernetes, serta menguji sejauh mana sistem yang diimplementasikan mampu menjaga waktu respons tetap rendah dan stabil ketika menerima beban hingga ribuan permintaan.

Tahap berikutnya adalah perencanaan, yang difokuskan pada penyusunan rancangan arsitektur sistem microservices. Layanan utama dibagi menjadi tiga, yaitu User Service, Product Service, dan Payment Service, yang masing-masing berjalan secara independen namun saling berkomunikasi melalui REST API[13].



Gambar 2. Gambaran Umum

Pada tahap perencanaan, setiap layanan dan endpoint tidak hanya dibuat untuk diuji, tetapi dirancang untuk menyelesaikan permasalahan utama pada sistem e-

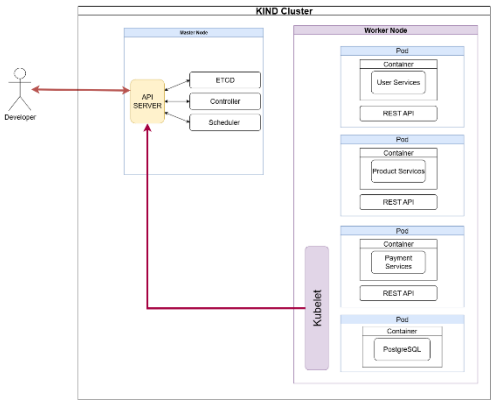
DOI: https://doi.org/10.29207/resti.v6iX.xxx

commerce berbasis monolitik, seperti keterikatan antar modul dan rendahnya skalabilitas. Layanan User Service dikembangkan untuk mengisolasi proses autentikasi pengguna agar tidak mengganggu layanan lain, Product Service untuk mengelola data produk secara mandiri tanpa memengaruhi performa sistem, dan Payment Service untuk menangani transaksi serta integrasi dengan Midtrans secara terpisah. Dengan demikian, desain tiga layanan ini bertujuan sebagai solusi arsitektural terhadap masalah skalabilitas dan keandalan sistem, bukan sekadar untuk pengujian teknis.

Pada sisi frontend, sistem menggunakan React, Vite, sebagai bundler dan Tailwind CSS untuk Styling. React dipilih karena mendukung antarmuka yang interaktif terutama untuk aplikasi berskala besar melalui mekanisme virtual DOM[14]. Vite digunakan karena proses pengembangan yang lebih cepat dan konfigurasi vite lebih ringkas dan lebih sedikit langkah pengaturan dibandingkan dengan menggunakan webpack[15]. Tailwind CSS digunakan untuk mempercepat proses styling frontend dan fleksibilitas yang lebih besar serta kemampuan untuk melakukan kostumisasi gaya yang mudah, menjadikannya cocok untuk pembuatan website microservice ini[16].

Arsitektur microservice pada sistem ini terdiri dari tiga layanan utama. User Service menyediakan fitur registrasi akun, seperti email, password, dan username, serta fungsi login. Product Service memungkinkan pengguna untuk melihat daftar produk, menampilkan detail produk, menambahkan produk ke keranjang, hingga melakukan checkout. Selain itu, admin dapat menambahkan, mengubah, maupun menghapus produk, termasuk mengunggah gambar produk. Sementara itu, Payment Service berfungsi untuk mengelola data diri dan alamat pengiriman saat checkout, melakukan inisiasi pembayaran melalui Midtrans, serta memantau status transaksi dan riwayat pembelian. Admin juga memiliki akses untuk memantau seluruh transaksi, melihat alamat pengguna, serta mengubah status pembayaran dari pending menjadi paid atau canceled.

Arsitektur sistem e-commerce dirancang menggunakan pendekatan microservice yang dikontainerisasi dengan Docker dan dikelola oleh Kubernetes pada klaster KIND. Tujuan perancangan ini adalah memastikan sistem dapat diskalakan, andal, serta mudah diuji[16].



Gambar 3. *Rancangan Arsitektur Microservice*

Gambar no 3 menunjukkan bahwa rancangan arsitektur microservice yang dibuat terdiri dari master node menjalankan komponen pengendali yaitu API Server sebagai pintu masuk utama semua permintaan kluster, baik dari kubectl maupun komponen internal,ETCD sebagai basis data key-value yang menyimpan seluruh state dan konfigurasi kluster,Scheduler sebagai menempatkan pod baru ke worker node yang sesuai berdasarkan ketersediaan sumber daya,Controller manager sebagai komponen yang mengawasi kondisi sistem dan membetulkannya bila ada perbedaan dengan konfigurasi yang sudah ditentukan[17].

Worker node merupakan tempat dimana container aplikasi dieksekusi dan dikelola agar tetap aktif, terhubung, serta siap menerima permintaan pengguna dan didalam nya ada kubelet yaitu merupakan agen yang memastikan pod berjalan sesuai intruksi dari API server[17].

Sistem terdiri atas tiga layanan utama, yaitu User Service, Product Service, dan Payment Service, yang masing-masing diimplementasikan sebagai REST API. Seluruh layanan berjalan pada pod terpisah dan saling terhubung melalui service internal kubernetes. PostgreSQL digunakan sebagai basis data terpusat.

Akses dari luar klaster dilakukan melalui nginx pada port 30080, yang meneruskan permintaan ke layanan sesuai jalur API. Untuk kebutuhan administratif, koneksi langsung ke PostgreSQL dapat dilakukan melalui port-forward ke port 5432.

Tahap pelaksanaan, pada tahap ini peneliti mulai mengimplementasikan sistem berdasarkan rencana yang telah disusun. Layanan-layanan yang telah dirancang dikembangkan sebagai aplikasi independen dan diimplementasikan dalam kontainer Docker. Selanjutnya, Kubernetes digunakan untuk mengelola dan mengorkestrasi kontainer tersebut dalam kluster menggunakan Kubernetes in Docker (KIND). Setelah implementasi selesai, pengujian dilakukan untuk mengevaluasi konektivitas antar layanan dan waktu respons API, guna memastikan bahwa semua layanan dapat berkomunikasi dengan baik dan waktu respons sesuai dengan ekspektasi.

DOI: https://doi.org/10.29207/resti.v6iX.xxx

Pengujian dilakukan pada perangkat laptop dengan spesifikasi

Tabel 1. *Spesifikasi laptop untuk pengujian*

OS	Windows 11
Processor	AMD Ryzen 5 5600H
RAM	16 GB Ram
Core CPU	6 Core
GPU	GeForce RTX 3050

Spesifikasi perangkat ini menjadi acuan dalam interpretasi hasil pengujian, mengingat kapasitas sumber daya berpengaruh terhadap response time, throughput, serta konsumsi CPU dan memori.

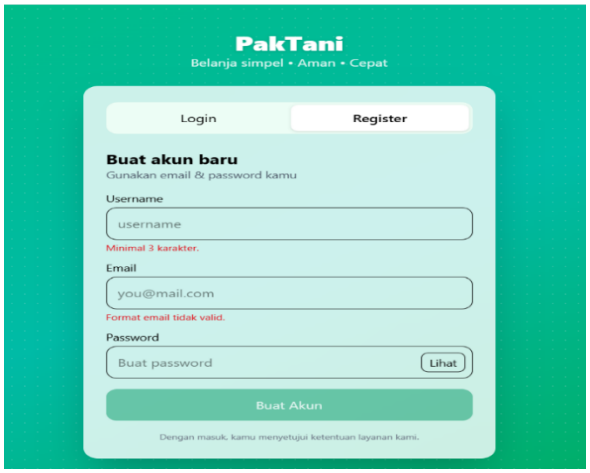
Tahap terakhir adalah tindakan dilakukan untuk mengevaluasi hasil implementasi arsitektur microservices yang telah dibangun. Evaluasi ini mencakup dua jenis pengujian. Pertama, pengujian fungsional dilakukan menggunakan Postman dengan tujuan memastikan bahwa setiap endpoint pada masing-masing layanan, seperti User Service, Product Service, dan Payment Service, dapat berjalan sesuai dengan fungsinya serta saling terintegrasi tanpa menimbulkan error. Pengujian ini penting untuk memverifikasi bahwa sistem telah sesuai dengan rancangan yang dibuat pada tahap perencanaan.

Kedua, dilakukan pengujian performa menggunakan Apache JMeter dengan skenario beban hingga 1.000 pengguna simultan. Pengujian ini difokuskan pada tiga metrik utama, yaitu response time, throughput, dan error rate. Response time digunakan untuk mengukur kecepatan sistem dalam merespons permintaan, throughput digunakan untuk mengetahui kemampuan sistem dalam memproses sejumlah permintaan per satuan waktu, sedangkan error rate digunakan untuk mengukur tingkat kegagalan yang terjadi selama pengujian. Ketiga metrik tersebut umum digunakan dalam evaluasi performa aplikasi e-commerce, seperti yang dilakukan oleh Patil & Khot (2023) dalam penelitian mereka menggunakan Apache JMeter[18].

Hasil pengujian dianalisis untuk menilai tingkat kestabilan dan keandalan sistem. Apabila ditemukan kendala seperti waktu respons yang tinggi, throughput rendah, atau error rate melebihi ambang batas toleransi, maka dilakukan tindakan perbaikan berupa optimasi konfigurasi, penyesuaian parameter pada Kubernetes, maupun refaktorisasi kode pada layanan tertentu[19]. Dengan cara ini, tahap tindakan tidak hanya berfungsi sebagai evaluasi, tetapi juga sebagai dasar perbaikan agar sistem microservices yang dibangun benar-benar memenuhi kebutuhan performa yang diharapkan.

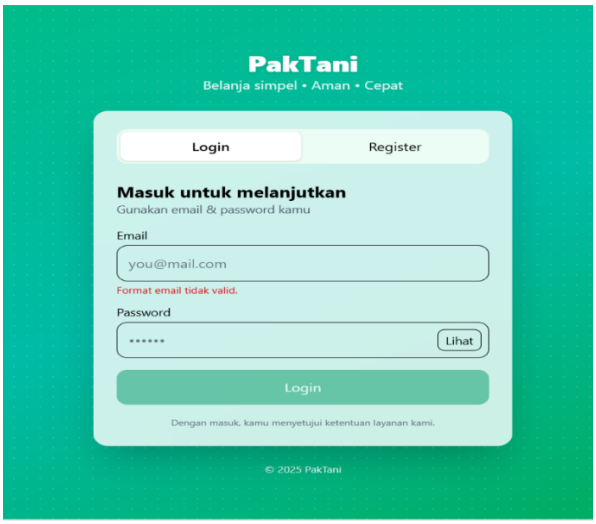
3. Hasil dan Pembahasan

Implementasi sistem e-commerce berbasis arsitektur microservices berhasil dilakukan dengan membangun tiga layanan utama, yaitu User Service, Product Service, dan Payment Service. Ketiga layanan tersebut dikembangkan secara independen, dikontainerisasi menggunakan Docker, serta dijalankan dalam kluster Kubernetes dengan KIND sebagai platform orkestrasi.



Gambar 4. *Tampilan halaman Register*

Gambar 4 Halaman register berfungsi sebagai pintu masuk awal bagi pengguna untuk membuat akun baru pada sistem, dengan form yang terdiri atas tiga komponen utama, yaitu username, email, dan password. Pada tahap ini, sistem menerapkan validasi minimal tiga karakter untuk username agar lebih terstruktur, mewajibkan penggunaan alamat email yang sesuai format standar agar dapat diverifikasi, sedangkan password tidak dibatasi oleh kriteria kompleksitas tertentu sehingga pengguna memiliki keleluasaan dalam menentukan kombinasi sandinya.

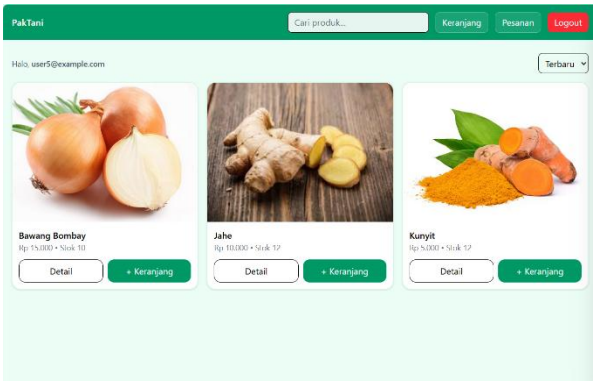


Gambar 5. *Tampilan halaman Login*

Gambar 5 menampilkan halaman login yang digunakan pengguna untuk masuk ke dalam sistem setelah terlebih dahulu melakukan proses registrasi. Pada halaman ini, pengguna diminta untuk mengisi alamat email dan password yang telah didaftarkan agar dapat diverifikasi oleh sistem. Setelah proses autentikasi berhasil, pengguna akan diarahkan ke halaman sesuai dengan

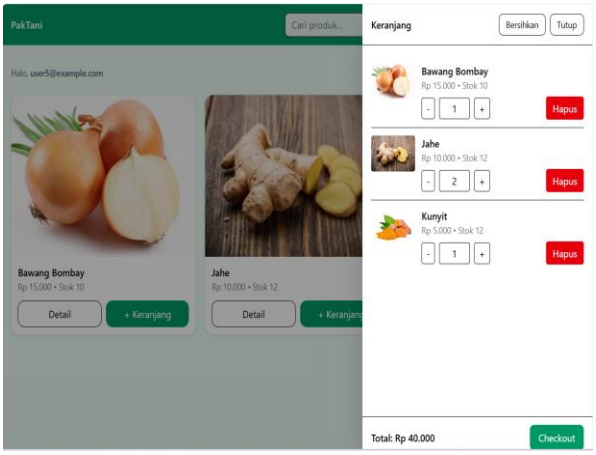
DOI: https://doi.org/10.29207/resti.v6iX.xxx

perannya, yaitu sebagai user dengan akses ke fitur belanja atau sebagai admin dengan akses ke fungsi manajemen.



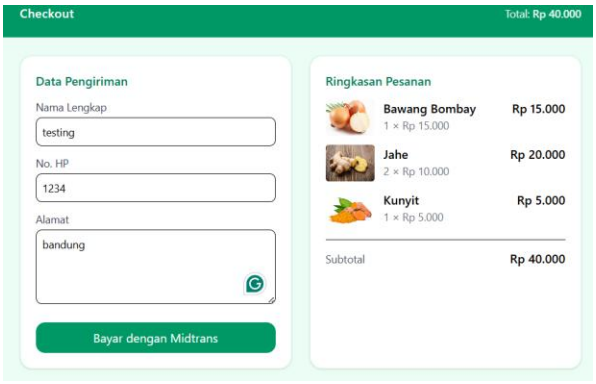
Gambar 6. *Tampilan halaman user*

Gambar 6 menampilkan halaman utama yang dapat diakses oleh pengguna setelah berhasil login dengan role user. Pada halaman ini, pengguna diberikan akses untuk melihat daftar produk yang tersedia serta menggunakan fitur sesuai dengan kebutuhan, seperti menampilkan detail produk, menambahkan produk ke keranjang, dan melanjutkan ke proses checkout. Tampilan ini berfungsi sebagai antarmuka utama bagi pengguna dalam berinteraksi dengan sistem e-commerce.



Gambar 7. *Tampilan halaman keranjang*

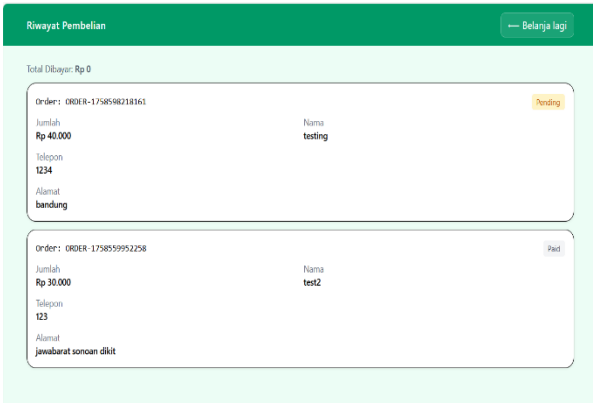
Gambar 7 menampilkan halaman keranjang belanja yang digunakan oleh pengguna untuk menampung produk yang telah dipilih sebelum melakukan checkout. Pada halaman ini, pengguna dapat melihat daftar produk yang dimasukkan ke dalam keranjang beserta informasi detailnya, seperti nama produk, jumlah, dan harga. Fitur keranjang ini memudahkan pengguna untuk meninjau kembali pilihan produk, menambah atau mengurangi jumlah, serta menghapus produk sebelum melanjutkan ke tahap pembayaran.



Gambar 8. *Tampilan halaman checkout*

Gambar 8 menampilkan halaman checkout yang digunakan pengguna untuk menyelesaikan proses pembelian. Pada halaman ini ditampilkan ringkasan produk yang akan dibeli beserta total harga. Selain itu, pengguna juga diminta mengisi informasi penting seperti alamat pengiriman dan metode pembayaran sebelum melanjutkan transaksi.

Sistem pembayaran pada platform ini terintegrasi dengan Midtrans sebagai payment gateway. Setelah pengguna menekan tombol konfirmasi, data transaksi akan dikirimkan ke Midtrans untuk diproses. Integrasi ini memungkinkan berbagai metode pembayaran seperti transfer bank, e-wallet, maupun kartu kredit, serta memastikan keamanan transaksi. Status pembayaran yang dikirimkan kembali oleh Midtrans secara otomatis dicatat oleh sistem, dan admin dapat untuk memperbarui status pesanan, sehingga pengguna mengetahui apakah transaksi berhasil (paid), masih menunggu (pending), atau gagal (canceled).

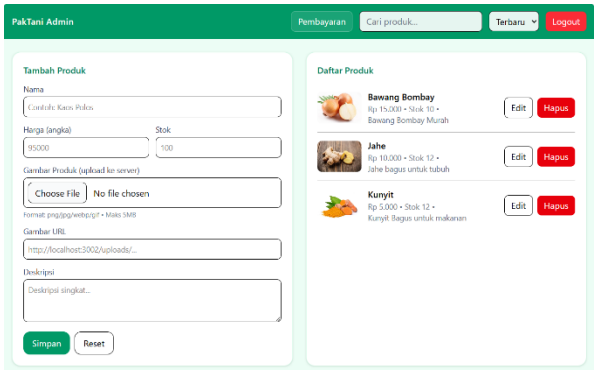


Gambar 9. *Tampilan halaman riwayat pembelian*

Gambar 9 menampilkan halaman riwayat pembelian yang berfungsi untuk menampilkan daftar transaksi yang telah dilakukan oleh pengguna. Pada halaman ini ditampilkan informasi penting seperti nama produk, jumlah, total harga, tanggal transaksi, serta status pembayaran. Fitur riwayat pembelian ini memberikan transparansi kepada pengguna karena setiap transaksi

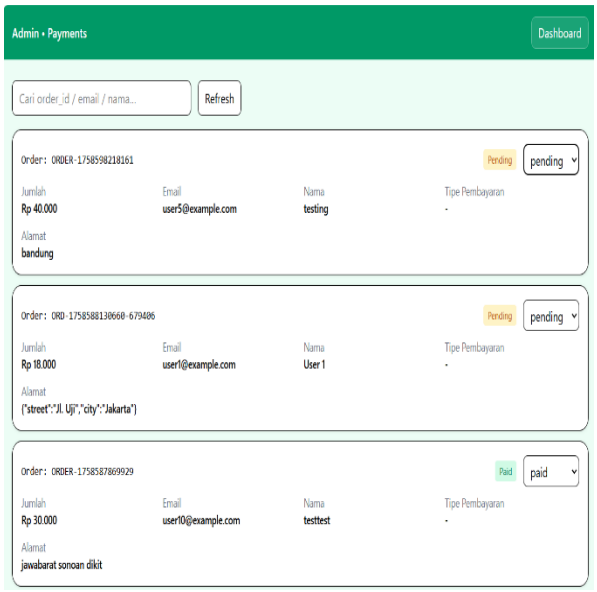
DOI: https://doi.org/10.29207/resti.v6iX.xxx

terdokumentasi dengan jelas. Selain itu, pengguna dapat memantau status pesanan apakah sudah selesai (paid), masih menunggu proses (pending), atau dibatalkan (canceled). Dengan adanya riwayat pembelian, pengalaman pengguna menjadi lebih informatif dan akuntabel, sekaligus meningkatkan tingkat kepercayaan terhadap platform e-commerce.



Gambar 10. *Halaman admin*

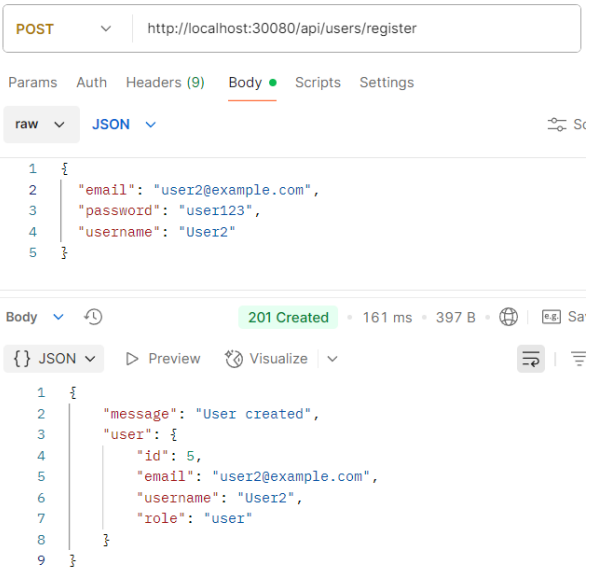
Gambar 10 menampilkan halaman admin yang digunakan untuk mengelola seluruh aktivitas pada platform e-commerce. Melalui halaman ini, admin memiliki hak akses penuh untuk melakukan manajemen produk, seperti menambahkan produk baru, mengubah detail produk, mengunggah gambar, maupun menghapus produk yang tidak relevan.



Gambar 11. *Tampilan halaman pembayaran admin*

Gambar 11 menampilkan halaman pembayaran pada sisi admin yang berfungsi untuk memantau sekaligus mengelola transaksi pengguna. Pada halaman ini, admin dapat melihat daftar pesanan yang berisi informasi seperti nama pengguna, detail produk, total harga, serta status pembayaran. Tidak hanya sebatas memantau, admin juga memiliki kewenangan untuk mengubah status pembayaran sesuai kondisi transaksi. Misalnya, status dapat diubah dari pending menjadi paid apabila pembayaran berhasil, atau menjadi canceled apabila transaksi dibatalkan. Dengan adanya fitur ini, sistem e-commerce mampu menjaga konsistensi data transaksi, meningkatkan transparansi, serta memberikan fleksibilitas bagi admin dalam memastikan setiap pembayaran tercatat sesuai keadaan sebenarnya.

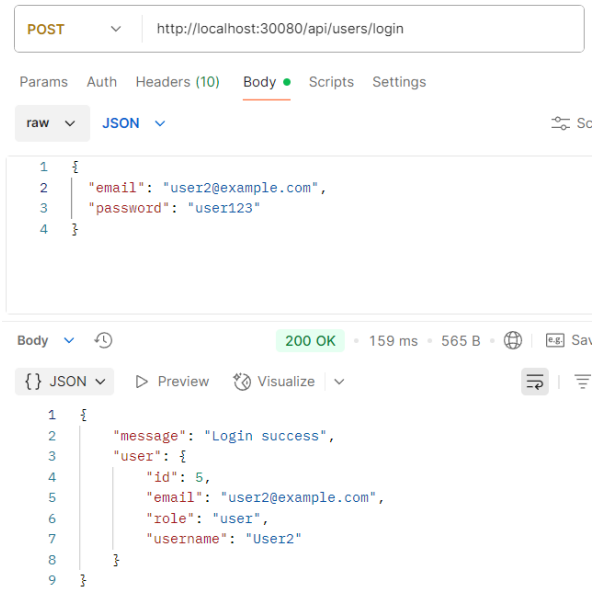
Pengujian fungsional dilakukan menggunakan Postman untuk memastikan setiap layanan microservices berjalan sesuai rancangan. Pengujian mencakup berbagai endpoint, seperti registrasi dan login pada User Service, pengelolaan produk pada Product Service, serta proses checkout pada Payment Service. Hasil pengujian menunjukkan seluruh endpoint dapat merespons permintaan dengan benar sesuai spesifikasi tanpa menimbulkan error, sehingga membuktikan integrasi antar layanan telah berjalan dengan baik.



Gambar 12. *Api request ke endpoint POST /users/register*

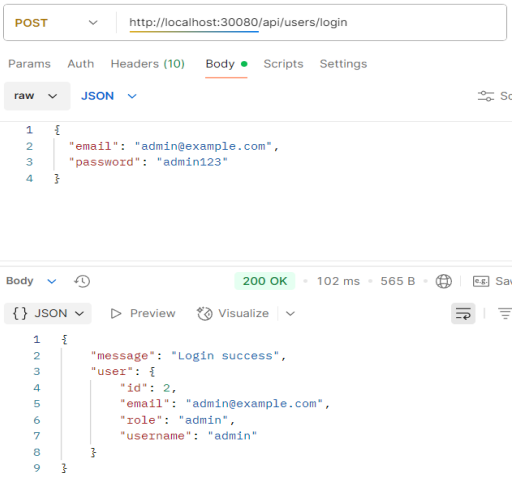
Gambar 12 menunjukkan hasil pengujian pada layanan user service untuk fitur registrasi. Endpoint /users/register menerima input berupa email, password, dan username. Respons yang dihasilkan adalah kode status HTTP 201 yang menandakan bahwa proses pendaftaran berhasil dilakukan. Hasil ini membuktikan layanan registrasi berfungsi sesuai rancangan dan data pengguna baru berhasil disimpan di basis data.

DOI: https://doi.org/10.29207/resti.v6iX.xxx



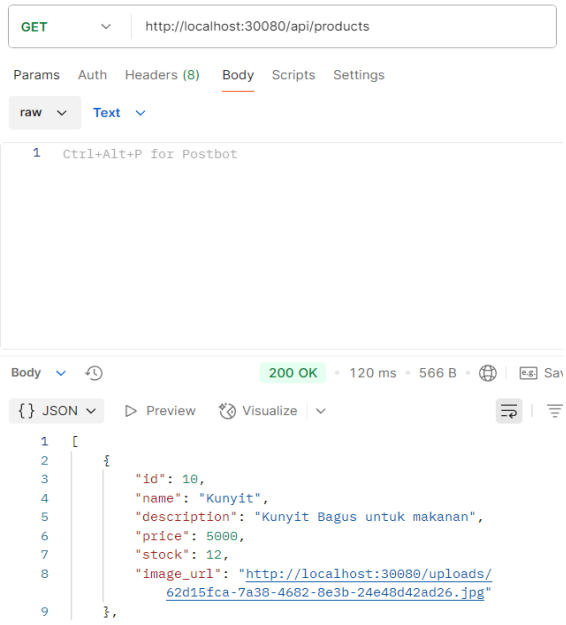
Gambar 13. *Api request ke endpoint POST /users/login (user login)*

Gambar 13 menunjukkan fitur login untuk pengguna biasa. Respons HTTP 200 yang diterima membuktikan bahwa sistem dapat memverifikasi kredensial dengan benar. Hal ini memastikan bahwa layanan autentikasi pengguna berjalan stabil dan dapat digunakan untuk mengakses fitur selanjutnya, seperti melihat produk atau melakukan transaksi.



Gambar 14. *Api request ke endpoint POST /users/login (admin login)*

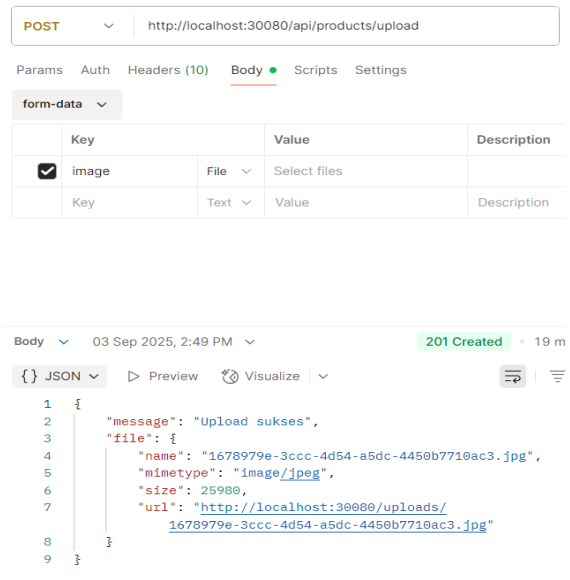
Pada gambar 14 hasil uji ini menampilkan proses login admin dengan respons HTTP 200 yang menunjukkan autentikasi berhasil dilakukan. Perbedaan utama antara login admin dan user terletak pada hak akses (privilege) yang dimiliki. Berhasilnya login admin menegaskan bahwa sistem telah mengimplementasikan manajemen otorisasi dengan benar, sehingga admin dapat melakukan fungsi manajemen produk maupun monitoring transaksi. Keberhasilan ini penting karena otorisasi berlapis merupakan salah satu aspek fundamental dalam keamanan sistem informasi. Dalam konteks e-commerce, mekanisme autentikasi dan otorisasi yang tepat tidak hanya melindungi data sensitif, tetapi juga memastikan perbedaan peran (role-based access control) berjalan sesuai desain arsitektur microservices[20].



Gambar 15. *Api request ke endpoint GET api/product (list product)*

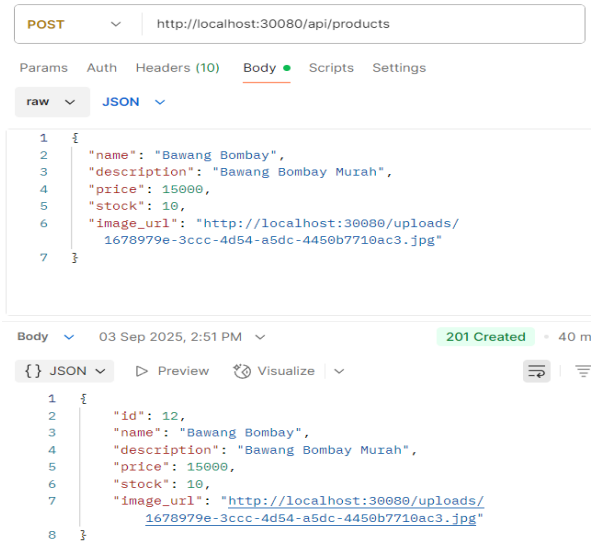
Pada gambar 15 pengujian ini dilakukan untuk memastikan bahwa Product Service mampu menampilkan daftar produk yang tersedia kepada pengguna. Respons yang diterima berupa array JSON berisi data produk, yang menunjukkan bahwa endpoint berhasil mengambil data dari basis data dan menampilkannya dengan benar. Hasil ini membuktikan bahwa layanan katalog produk telah berfungsi sesuai rancangan, karena keberhasilan dalam menampilkan informasi produk merupakan inti dari fitur product catalog pada platform e-commerce. Keandalan Product Service menjadi penting mengingat katalog produk adalah komponen utama yang memengaruhi pengalaman pengguna, kepuasan pelanggan, dan keberhasilan transaksi[21].

DOI: https://doi.org/10.29207/resti.v6iX.xxx



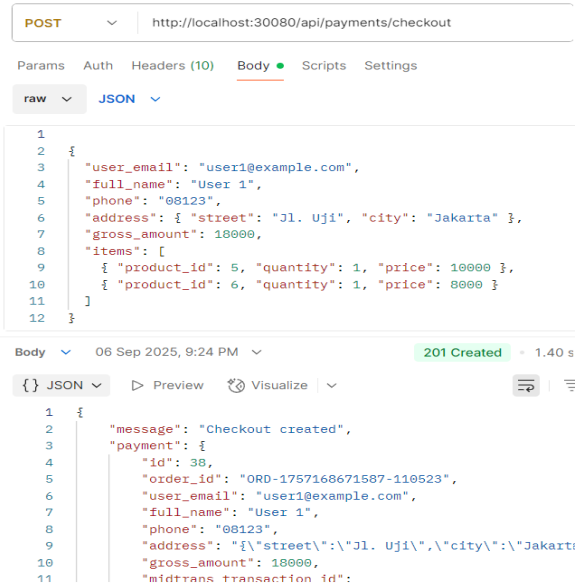
Gambar 16. *Api request ke endpoint POST api/product (upload gambar)*

Pada gambar 16 uji coba ini berfokus pada kemampuan sistem dalam menerima unggahan file gambar produk. Respons yang diterima menunjukkan keberhasilan integrasi antara backend dan storage sehingga proses unggah dapat berjalan dengan lancar. Fitur ini memiliki peran krusial karena tampilan visual produk secara langsung memengaruhi pengalaman pengguna, tingkat kepercayaan konsumen, serta keputusan pembelian dalam platform e-commerce. Penelitian sebelumnya juga menegaskan bahwa kualitas representasi visual produk berkontribusi signifikan terhadap user engagement dan tingkat konversi transaksi[22].



Gambar 17. *Api request ke endpoint POST api/product (Create Product)*

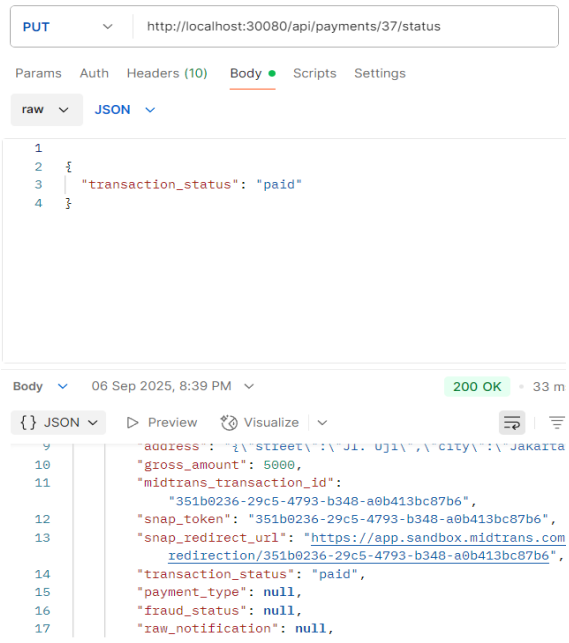
Pada gambar 17 Pengujian ini melengkapi validasi bahwa Product Service mendukung pembuatan produk baru dengan atribut lengkap, seperti nama, deskripsi, harga, dan gambar. Respons sukses memperkuat hasil uji sebelumnya dengan menunjukkan bahwa layanan mampu mengelola data produk secara konsisten sesuai kebutuhan bisnis. Keberhasilan pengujian ini menegaskan bahwa sistem telah menerapkan fungsi manajemen produk berbasis CRUD secara efektif, yang merupakan inti dari proses operasional e-commerce. Dalam praktiknya, keandalan manajemen produk sangat berpengaruh terhadap efisiensi rantai pasok digital, ketersediaan informasi produk yang akurat, serta pengalaman pengguna yang lebih baik.



Gambar 18. *Api request ke endpoint POST api/payment (Checkout)*

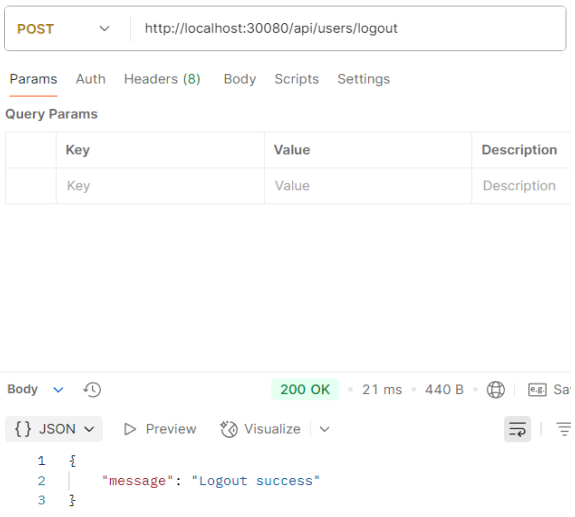
Pada gambar 18 memperlihatkan hasil uji proses checkout, di mana endpoint menerima data alamat, metode pembayaran, dan ringkasan transaksi. Respons HTTP 200 menunjukkan bahwa transaksi berhasil diinisiasi, sehingga menegaskan integrasi Payment Service dengan penyedia pihak ketiga, yaitu Midtrans, berjalan dengan baik. Keberhasilan pengujian ini sangat penting karena proses checkout merupakan tahap kritis dalam siklus transaksi e-commerce. Stabilitas dan keandalan integrasi pembayaran tidak hanya memastikan kelancaran transaksi, tetapi juga meningkatkan kepercayaan pengguna terhadap platform[23].

DOI: https://doi.org/10.29207/resti.v6iX.xxx



Gambar 19. *Api Api request ke endpoint PUT /payment (Change Status)*

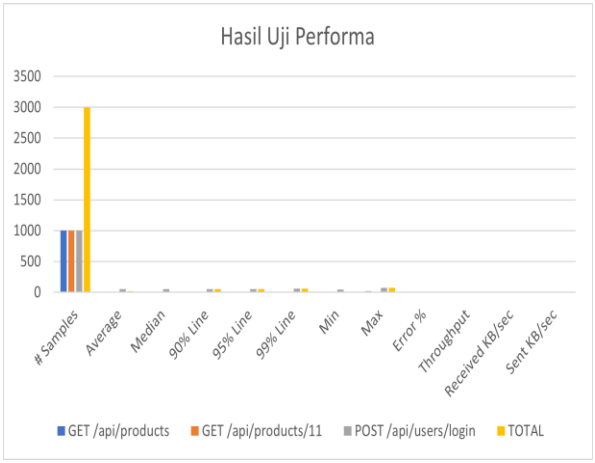
Pada gambar 19 menunjukkan kemampuan admin untuk mengubah status transaksi, misalnya dari pending menjadi paid. Respons sukses membuktikan bahwa sistem mendukung siklus penuh transaksi, mulai dari inisiasi pembayaran hingga penyelesaian. Hal ini penting karena keberhasilan pengelolaan status transaksi merupakan indikator keandalan alur bisnis dalam e-commerce.



Gambar 20. *Api Api request ke endpoint POST /api/users/logout*

Pada gambar 20 menunjukkan status logout berhasil dengan code HTTP 200 yang menegaskan fungsi telah berjalan sesuai rancangan.

Pengujian performa dilakukan menggunakan Apache JMeter dengan skenario beban hingga 1.000 pengguna simultan pada endpoint utama. Alat ini dipilih karena mendukung simulasi concurrent users dalam jumlah besar, sehingga mampu merepresentasikan kondisi nyata ketika platform e-commerce menghadapi lonjakan trafik. Melalui pendekatan ini, metrik utama seperti waktu respon, throughput, dan error rate dapat dianalisis secara terukur untuk menilai stabilitas serta skalabilitas sistem. Penggunaan JMeter sebagai load testing tool telah banyak diterapkan dalam penelitian microservices untuk mengevaluasi keandalan layanan berbasis Kubernetes.



Gambar 21. *Hasil Uji Performa*

Gambar 21 menampilkan hasil pengujian performa pada beberapa endpoint dengan 1.000 pengguna simultan. Hasilnya menunjukkan waktu respon rata-rata hanya 20 ms, throughput 9,98 request/detik, dan error rate 0%. Endpoint dengan beban terberat adalah /api/users/login dengan waktu respon 53 ms, Hasil pengujian menunjukkan bahwa rata-rata waktu respons masih berada jauh di bawah ambang batas 2 detik yang direkomendasikan untuk pengalaman interaktif, meskipun throughput relatif rendah. Temuan ini sejalan dengan hasil dari Adaptive load balancing and fault-tolerant microservices architecture for high-availability web systems using Docker and Spring Cloud (2025)[24], di mana sistem mampu menangani hingga 1.000 thread simultan dengan kestabilan yang tinggi meskipun throughput tidak selalu maksimal.

Tabel 2. *Hasil Uji Performa*

Endpoint	Rata-rata (ms)	Min (ms)	Max (ms)	Throughput (req/det)	Error Rate
GET /api/products	3	2	22	3,34	0 %
GET /api/products/11	3	2	13	3,34	0%
POST /api/users/login	53	49	73	3,33	0%
TOTAL	20	2	73	9,98	0%

Tabel 2 menunjukkan bahwa rata-rata waktu respon berada pada kisaran 2–53 ms, dengan nilai rata-rata

DOI: https://doi.org/10.29207/resti.v6iX.xxx

keseluruhan 20 ms. Endpoint /api/users/login memiliki waktu respon paling tinggi (53 ms), sementara GET /api/products dan GET /api/products/11 relatif rendah, yaitu sekitar 3 ms. Throughput sistem secara keseluruhan tercatat 9,98 request/detik dengan error rate 0% pada seluruh pengujian. Temuan ini menegaskan bahwa meskipun throughput relatif rendah, seluruh request berhasil diproses tanpa error, sehingga reliabilitas sistem tetap terjaga. Stabilitas performa dengan tingkat kesalahan nol merupakan indikator penting bagi keandalan arsitektur microservices, karena sistem yang mampu menjaga integritas data dan konsistensi layanan di bawah beban tinggi akan lebih siap menghadapi skenario penggunaan berskala besar.

4. Kesimpulan

Berdasarkan hasil implementasi dan pengujian, dapat disimpulkan bahwa penerapan arsitektur microservices berbasis Docker dan Kubernetes mampu menyelesaikan permasalahan yang ditemukan pada sistem monolitik. seperti ketergantungan antar modul dan penurunan performa saat beban meningkat. Pemisahan layanan menjadi tiga microservices independen terbukti meningkatkan modularitas, menjaga stabilitas sistem, serta memudahkan proses pengelolaan dan pengujian setiap komponen secara terpisah. Seluruh layanan seperti User Service, Product Service, dan Payment Service berjalan sesuai rancangan, serta mampu menjaga integritas data dan komunikasi antar layanan.

Dari sisi kinerja, pengujian menunjukkan bahwa sistem microservices memiliki response time yang stabil, error rate 0%, serta mampu melayani skenario beban hingga 1.000 pengguna simultan tanpa kegagalan request. Meskipun throughput yang diperoleh masih relatif rendah, kestabilan dan reliabilitas sistem menjadi nilai utama yang ditawarkan.

Kekuatan utama dari pendekatan ini terletak pada kemampuan orkestrasi Kubernetes yang mendukung skalabilitas, fault tolerance, dan automatisasi deployment, serta Docker yang menjamin portabilitas dan isolasi setiap layanan. Dengan demikian, arsitektur microservices ini sangat sesuai untuk platform e-commerce modern yang membutuhkan fleksibilitas, ketahanan, dan skalabilitas dinamis dalam menghadapi lonjakan beban pengguna maupun transaksi.

Daftar Rujukan

[1] O. Wijaya, "E-Commerce: Perkembangan, Tren, dan Peraturan Perundang-Undangan," *E-Bisnis J. Ilm. Ekon. dan Bisnis*, vol. 16, no. 1, pp. 41–47, 2023, doi: 10.51903/e-bisnis.v16i1.1083.

[2] I. Q. A'yun, L. Anggraini, G. D. Asmara, and R. M. Khoirunnisa, "Analysis of the Development of E-Commerce Transactions in the 6 Highest Transaction Countries in Southeast Asia," *J. Econ. Res. Soc. Sci.*, vol. 8, no. 2, pp. 207–221, 2024, doi: 10.18196/jerss.v8i2.22033.

[3] N. Maisaroh, A. Habibi, and M. Iqbal, "Growth Trends of Internet Users and E-commerce in Increasing Economic Growth in Indonesia," *J. Ekon. Pembang.*, vol. 13, no. 3, pp. 41–50, 2024.

[4] G. Memarista, E. T. Gunawan, and N. Kristina, "E-Commerce Usage and Indonesian Msme'S Performance," *JMBI UNSRAT (Jurnal Ilm. Manaj. Bisnis dan Inov. Univ. Sam Ratulangi)*., vol. 10, no. 2, pp. 846–860, 2023, doi: 10.35794/jmbi.v10i2.48062.

[5] J. D. S. Amory, M. Mudo, and R. J, "Transformasi Ekonomi Digital dan Evolusi Pola Konsumsi: Tinjauan Literatur tentang Perubahan Perilaku Belanja di Era Internet," *J. Minfo Polgan*, vol. 14, no. 1, pp. 28–37, 2025, doi: 10.33395/jmp.v14i1.14608.

[6] S. Daeli, K. J. D. Lase, and P. Sumihar, "Implementation of Microservices Architecture in a Retail Web Application Using Apache Kafka as a Message Broker," *Eng. Math. Comput. Sci. J.*, vol. 7, no. 2, pp. 215–224, 2025, doi: 10.21512/emacsjournal.v6i.

[7] Anusha Reddy Guntakandla, "Microservices and Modular Architecture: Revolutionizing E-Commerce Scalability," pp. 133–137, 2025, doi: 10.32996/jcsts.2025.7.4.15.

[8] A. Iurchenko, "Optimization of Microservices Architecture Performance in High-Load Systems," *Am. J. Eng. Technol.*, vol. 07, no. 05, pp. 123–132, 2025, doi: 10.37547/tajet/volume07issue05-10.

[9] K. Vishnivetskii, "Dynamic Scaling and Performance Optimization for Microservices using Kubernetes," *Asian J. Res. Comput. Sci.*, vol. 18, no. 3, pp. 213–220, 2025, doi: 10.9734/ajrcos/2025/v18i3587.

[10] E. Wansart, M. Goffart, J. Iurman, and B. Donnet, "MSTG: A Flexible and Scalable Microservices Infrastructure Generator," *TMA 2024 - Proc. 8th Netw. Traffic Meas. Anal. Conf.*, 2024, doi: 10.23919/TMA62044.2024.10559148.

[11] M. H. BHATTACHARYA and H. K. MITTAL, "Exploring the Performance of Container Runtimes within Kubernetes Clusters," *Int. J. Comput.*, vol. 22, no. 4, pp. 509–514, 2023, doi: 10.47839/ijc.22.4.3359.

[12] J. A. Suthendra and M. A. I. Pakereng, "Implementation of Microservices Architecture on E-Commerce Web Service," *ComTech Comput. Math. Eng. Appl.*, vol. 11, no. 2, pp. 89–95, 2020, doi: 10.21512/comtech.v11i2.6453.

[13] S. Chauhan *et al.*, "LLM-Generated Microservice Implementations from RESTful API Definitions," pp. 161–173, 2025, doi: 10.5220/0013391000003928.

[14] V. Veeri, "Performance Optimization Techniques in React Applications: A Comprehensive Analysis," *Int. J. Res. Comput. Appl. Inf. Technol.*, vol. 7, no. 2, pp. 1165–1177, 2024, doi: 10.5281/zenodo.14146734.

[15] T. A. Nguyen, "A comparative analysis of Webpack and Vite as build tools for Ja-vaScript," 2024.

[16] N. S., U. Sree R., and P. Mohan, "Comparison of Utility-First CSS Framework," *J. Innov. Technol.*, vol. 2024, no. 1, 2024, doi: 10.61453/joit.v2024no32.

[17] S. K. Mondal, Z. Zheng, and Y. Cheng, "On the Optimization of Kubernetes toward the Enhancement of Cloud Computing," *Mathematics*, vol. 12, no. 16, 2024, doi: 10.3390/math12162476.

[18] S. Pradeep and Y. K. Sharma, "A Pragmatic Evaluation of Stress and Performance Testing Technologies for Web Based Applications," *Proc. - 2019 Amity Int. Conf. Artif. Intell. AICAI 2019*, no. February 2019, pp. 399–403, 2019, doi: 10.1109/AICAI.2019.8701327.

[19] V. Cortellessa, D. Di Pompeo, R. Eramo, and M. Tucci, "A

DOI: https://doi.org/10.29207/resti.v6iX.xxx

model-driven approach for continuous performance engineering in microservice-based systems," *J. Syst. Softw.*, vol. 183, 2022, doi: 10.1016/j.jss.2021.111084.

[20] Arun Kumar Akuthota, "Role-Based Access Control (RBAC) in Modern Cloud Security Governance: An In-depth Analysis," *Int. J. Sci. Res. Comput. Sci. Eng. Inf. Technol.*, vol. 11, no. 2, pp. 3297–3311, 2025, doi: 10.32628/cseit25112793.

[21] H. Alnuhait, W. Alzyadat, A. Althunibat, H. Kahtan, B. Zaqaibeh, and H. A. Al-Khawaja, "Web application performance assessment: A study of responsiveness, throughput, and scalability," *Int. J. Adv. Appl. Sci.*, vol. 11, no. 9, pp. 214–226, 2024, doi: 10.21833/ijaas.2024.09.023.

[22] M. S. Pereira, A. Cardoso, C. Fernandes, S. Rodrigues, and F. D'Orey, "The Influence of the Image and Photography of E- Commerce Products on the Purchase Decision of Online Consumers," *Lect. Notes Inst. Comput. Sci. Soc. Telecommun. Eng. LNICST*, vol. 458 LNICST, no. February, pp. 39–51, 2023, doi: 10.1007/978-3-031-25222-8_4.

[23] S. Supriyati and E. Nurfiqo, "Effectiveness of Payment Gateway in E-Commerce," no. January, 2019, doi: 10.4108/eai.18-7-2019.2287932.

[24] P. Zhang, L. Xiang, Z. Song, and Y. Yang, "Adaptive load balancing and fault-tolerant microservices architecture for high-availability web systems using docker and spring cloud," *Discov. Appl. Sci.*, vol. 7, no. 7, 2025, doi: 10.1007/s42452-025-07320-7.

DOI: https://doi.org/10.29207/resti.v6iX.xxx