PENGEMBANGAN RESTFUL API UNTUK PRODUK E-COMMERCE RITEL DENGAN PENDEKATAN MICROSERVICES DAN SPRING BOOT

Nicholas Wicaksono Tutuko, Felix David
Teknik Informatika, Universitas Kristen Satya Wacana
Jalan Diponegoro No. 52-60, Salatiga, Jawa Tengah 50711, Indonesia
nicholasnicho148@gmail.com

ABSTRAK

E-commerce ritel mengalami pertumbuhan pesat seiring meningkatnya penggunaan teknologi digital, namun sistem backend yang digunakan sering kali belum mendukung skalabilitas dan modularitas yang dibutuhkan dalam skenario produksi. Arsitektur monolitik yang umum digunakan sulit dikembangkan secara fleksibel dan sering menimbulkan kendala saat menghadapi beban transaksi tinggi atau integrasi sistem baru. Penelitian ini bertujuan untuk mengembangkan sistem backend berbasis RESTful API menggunakan pendekatan arsitektur microservices guna meningkatkan skalabilitas, fleksibilitas, dan efisiensi sistem e-commerce ritel. Pendekatan yang digunakan adalah Research and Development (R&D) dengan tahapan identifikasi masalah, perancangan sistem, implementasi tiga layanan utama (produk, pesanan, pembayaran), serta integrasi melalui API Gateway. Teknologi yang digunakan meliputi MongoDB, PostgreSQL, dan MySQL untuk masing-masing layanan, serta Spring Boot dan Spring Cloud Gateway sebagai fondasi sistem. Hasil pengujian menggunakan Postman dan JMeter menunjukkan bahwa sistem mampu merespons permintaan secara stabil hingga 500 transaksi, dengan throughput tinggi dan waktu respons optimal. Penelitian ini memberikan kontribusi nyata terhadap pengembangan sistem backend e-commerce yang modular, terukur, dan siap digunakan dalam lingkungan produksi berskala besar.

Kata kunci : *API, Microservices, Spring Boot, E-commerce, Backend, API Gateway*

1. PENDAHULUAN

Industri e-commerce ritel telah mengalami pertumbuhan yang pesat dalam dekade terakhir, terutama didorong oleh meningkatnya adopsi internet, penetrasi perangkat mobile, serta perubahan perilaku konsumen yang semakin nyaman dengan transaksi daring[1]. Transformasi digital ini mendorong perusahaan untuk menyediakan sistem informasi yang tidak hanya cepat, tetapi juga fleksibel dan mampu menyesuaikan diri dengan dinamika pasar yang terus berubah. Dalam konteks ini, sistem backend memiliki peran penting sebagai fondasi dari layanan e-commerce yang berkelanjutan. Meskipun banyak platform e-commerce telah beroperasi secara daring, sebagian besar masih menggunakan arsitektur monolitik[2]. Arsitektur ini memiliki keterbatasan dari sisi fleksibilitas, pemeliharaan, dan skalabilitas. Ketika suatu sistem monolitik tumbuh secara kompleks, pembaruan satu fitur dapat berdampak terhadap keseluruhan sistem. Hal ini menghambat pengembangan berkelanjutan dan menyulitkan integrasi dengan layanan lain, terutama dalam menangani beban trafik yang tinggi secara efisien. Permasalahan seperti ini menjadi tantangan utama bagi pengembang dan pengelola sistem e-commerce modern.

Penelitian ini bertujuan untuk mengembangkan RESTful API backend e-commerce ritel yang modular, terukur, dan efisien menggunakan pendekatan arsitektur microservices[3]. Pendekatan ini diharapkan dapat mengatasi keterbatasan arsitektur monolitik dengan cara memisahkan fungsionalitas sistem ke dalam layanan-layanan kecil yang dapat dikembangkan dan dikelola secara independen. Dengan demikian, sistem akan menjadi lebih mudah diatur, diuji, di-deploy, dan diskalakan sesuai kebutuhan bisnis.

Sebagai langkah penyelesaian masalah, penelitian ini menggunakan pendekatan Research and Development (R&D), yang mencakup tahapan identifikasi masalah, studi literatur, perancangan sistem, implementasi, serta pengujian. Sistem dibangun menggunakan framework Spring Boot dengan arsitektur microservices yang terdiri dari tiga layanan utama: Product Service menggunakan MongoDB, Order Service menggunakan PostgreSQL, dan Payment Service menggunakan MySQL. Ketiga layanan ini diintegrasikan melalui API Gateway berbasis Spring Cloud Gateway yang memungkinkan komunikasi terpisah antar layanan tetap terkontrol dan efisien. Pengujian dilakukan menggunakan Postman untuk verifikasi fungsionalitas dan JMeter untuk pengujian performa pada berbagai tingkat beban transaksi.

2. TINJAUAN PUSTAKA

2.1. Microservices Architecture

Arsitektur *microservices* adalah pendekatan dalam pengembangan perangkat lunak yang membagi aplikasi menjadi layanan-layanan kecil yang berdiri sendiri. Setiap layanan memiliki tanggung jawab spesifik dan dapat dikembangkan, diuji, serta di-*deploy* secara independen. *Microservices* meningkatkan fleksibilitas, skalabilitas, dan

pemeliharaan sistem. Namun, tantangan utama dari pendekatan ini adalah kompleksitas dalam pengelolaan layanan, komunikasi antar-*microservices*, dan keamanan.

2.2. Restful API

RESTful API adalah implementasi dari arsitektur Representational *State Transfer (REST)* yang digunakan untuk komunikasi antara klien dan *server* melalui protokol *HTTP*. *RESTful API* memiliki karakteristik seperti stateless, resource-oriented, dan mendukung berbagai format respons seperti *JSON* dan *XML*. Dalam konteks *e-commerce*, *RESTful API* memungkinkan layanan produk, pesanan, dan pembayaran untuk saling berinteraksi secara efisien.

2.3. Java Spring Boot

Spring Boot adalah *framework Java* untuk membangun aplikasi berbasis *microservices* dengan konfigurasi otomatis, *dependency injection*, dan integrasi *database*. Keunggulannya meliputi:

a. Dukungan *Microservices*: Memiliki fitur bawaan untuk pengelolaan layanan.
b. Integrasi Mudah: Mendukung database seperti *MongoDB, PostgreSQL*.
c. Konfigurasi Otomatis: Mempermudah proyek**.**

2.4. Teknologi Database

Dalam penelitian ini, digunakan tiga jenis database untuk masing-masing layanan.

a. *MongoDB*: Digunakan untuk menyimpan data produk karena fleksibilitasnya dalam menangani data semi-terstruktur**.**
b. *PostgreSQL*: Digunakan untuk layanan pemesanan.
c. *MySQL*: Digunakan untuk layanan pembayaran karena kecepatan dan efisiensinya dalam menangani data transaksi.

2.5. E-Commerce

Industri *e-commerce* telah mengalami pertumbuhan yang pesat dalam beberapa tahun terakhir, didorong oleh peningkatan akses internet dan perubahan perilaku konsumen . *E-commerce* ritel adalah salah satu bentuk penerapan *e-commerce* yang fokus pada penjualan langsung kepada konsumen akhir. Pengembangan *RESTful API* yang efisien dan skalabel sangat penting untuk mendukung pertumbuhan bisnis *e-commerce*, termasuk *ecommerce* ritel. API ini memungkinkan integrasi dengan berbagai sistem dan aplikasi, seperti aplikasi mobile, sistem pembayaran, dan layanan logistik [4]

2.6. Penelitian Terkait

Penelitian oleh A. R. Guntakandla dalam artikelnya berjudul *"Microservices and Modular Architecture: Revolutionizing E-Commerce Scalability"* membahas bagaimana arsitektur microservices dapat meningkatkan skalabilitas platform e-commerce secara signifikan. Studi ini menekankan pentingnya modularisasi dan pemisahan layanan dalam mendukung sistem yang tahan terhadap lonjakan trafik pengguna serta mudah diadaptasi untuk perubahan bisnis. Temuan ini memperkuat landasan pendekatan microservices dalam pengembangan sistem backend e-commerce ritel yang dilakukan dalam penelitian ini [5].

Buku "*Microservices Best Practices for Java"* oleh Eberhard Wolff, yang diterbitkan pada tahun 2018, membahas strategi membangun arsitektur *microservices* dengan *Java Spring Boot* serta pengelolaan layanan menggunakan Docker dan Kubernetes. Buku ini sangat relevan untuk meningkatkan skalabilitas, fleksibilitas, dan efisiensi *RESTful API* yang akan diterapkan pada sistem e-*commerce* ritel.[6]

Penelitian oleh M. Shaikh dan P. Devgun dalam artikel berjudul *"Building scalable and secure microservices with Spring Boot"* menjelaskan strategi pengembangan sistem microservices yang scalable dan aman menggunakan framework Spring Boot. Penelitian ini membahas praktik-praktik pengembangan dan konfigurasi sistem backend modern berbasis Java, serta menyoroti pentingnya arsitektur modular dalam mendukung performa dan keamanan aplikasi skala besar. Temuan ini memperkuat pendekatan dalam penelitian ini untuk membangun sistem backend yang efisien dan terukur dengan Spring Boot [7].

R. Tsyganok dalam penelitiannya yang berjudul *"Methodology for building scalable microservice architectures on Go for high-load e-commerce platforms"* membahas pendekatan arsitektural dalam membangun sistem e-commerce berbasis microservices yang mampu menangani beban tinggi secara efisien. Penelitian ini menekankan pentingnya modularisasi dan service independence dalam mengoptimalkan throughput sistem serta mendukung integrasi berkelanjutan. Temuan ini relevan sebagai landasan untuk membangun sistem e-commerce ritel yang adaptif dan tahan terhadap lonjakan trafik pengguna [8].

Penelitian oleh Sanjaya dan Murnawan berjudul "Pemanfaatan Arsitektur *Microservice* untuk Peningkatan Performansi *Website* Lomba Nasional Kreativitas Mahasiswa", yang diterbitkan dalam Jurnal Teknologi Informasi dan Ilmu Komputer (JTIIK) pada tahun 2024, membahas penggunaan microservices untuk meningkatkan performa website dengan metode *Design Science Research Methodology (DSRM)*. Hasil penelitian menunjukkan peningkatan throughput sistem tanpa penurunan performa yang signifikan. Penelitian ini menjadi referensi dalam membangun sistem *e-commerce* yang mampu menangani transaksi produk dalam jumlah besar dengan performa yang stabil.[9]

Penelitian Andrzej Barczak dan Michał Barczak berjudul "*Performance Comparison of Monolith and Microservices Based Applications"*, membahas perbandingan performa aplikasi monolitik dan

microservices menggunakan *Azure Module* dan *JMeter*. Hasil penelitian menunjukkan bahwa *microservices* unggul dalam menangani beban kerja besar dibandingkan aplikasi *monolitik.* Temuan ini memperkuat keputusan penggunaan arsitektur *microservices* pada sistem *e-commerce* untuk meningkatkan skalabilitas dan efisiensi sistem[10]

3. METODE PENELITIAN

Penelitian ini menggunakan pendekatan *Research and Development (R&D)* . Pendekatan ini bertujuan untuk mengembangkan produk berupa sistem *backend e-commerce* berbasis *RESTful API* dengan arsitektur *microservices*. *R&D* dipilih karena cocok untuk menghasilkan produk teknologi berupa perangkat lunak, bukan hanya membuktikan hipotesis

3.1. Tahapan Penelitian

Tahapan penelitian mengikuti model *R&D* yang telah disesuaikan dengan konteks pengembangan sistem:

a. Identifikasi Masalah dan Studi Literatur
Menganalisis kendala sistem monolitik dalam pengembangan sistem e-commerce dan mencari solusi melalui studi literatur mengenai *microservices, RESTful API, dan Spring Boot.*
b. Perancangan
Mendesain arsitektur sistem backend dengan layanan terpisah: *Product Service, Order Service,* dan *Payment Service.* Pemilihan teknologi dilakukan berdasarkan kebutuhan setiap layanan.
c. Implementasi
Sistem dikembangkan menggunakan framework *Spring Boot,* dengan *MongoDB* untuk produk, *PostgreSQL* untuk pesanan, dan *MySQL* untuk pembayaran.
d. Pengujian
Pengujian dilakukan dengan dua pendekatan: pengujian fungsionalitas menggunakan *Postman,* dan pengujian kinerja menggunakan *JMeter.*
e. Evaluasi
Evaluasi dilakukan dengan menganalisis hasil pengujian terhadap tujuan awal, yakni skalabilitas dan efisiensi sistem.

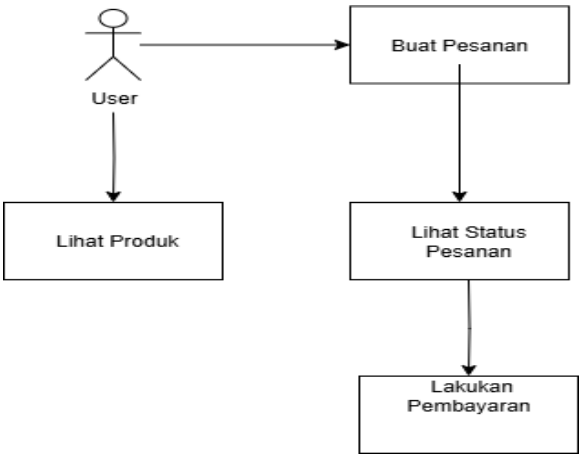
3.2. Alat dan Bahan

Penelitian ini akan menggunakan beberapa tech stack antara lain :

a. Perangkat Keras: Komputer atau laptop dengan spesifikasi yang memadai.
b. Perangkat Lunak: Sistem Operasi: *Windows,* Bahasa Pemrograman: *Java Development Kit* (*JDK), Framework: Spring Boot, Database: MongoDB, PostgreSQL, MySQL., IDE: IntelliJ IDEA.*
c. Alat Pengujian: *Postman , JMeter*.

3.3. Interaksi Pengguna dan Sistem

Bagian ini menggambarkan hubungan antara pengguna dan fitur utama yang tersedia pada sistem backend e-commerce. Pengguna dapat melakukan beberapa aktivitas, seperti melihat produk, membuat pesanan, memantau status pemesanan, serta melakukan pembayaran.



Gambar 1. Interaksi pengguna dan layanan sistem

3.4. Kriteria Keberhasilan

Kriteria Keberhasilan penelitian ini antara lain :

a. Fungsionalitas: API mampu menjalankan operasi CRUD, pencarian, dan filter produk.
b. Kinerja: API mampu menangani beban kerja tinggi dengan waktu respons yang optimal.

4. HASIL DAN PEMBAHASAN

4.1. Implementasi Sistem

Pada bab ini, dijelaskan implementasi sistem backend e-commerce ritel berbasis microservices menggunakan *Spring Boot.* Arsitektur yang digunakan terdiri dari beberapa komponen utama, yaitu API *Gateway, Product Service, Order Service, Payment Service,* dan database yang terpisah untuk setiap layanan [11]

4.2. Arsitektur Sistem

Gambar 2. Gambar Diagram Arsitektur Sistem

Penjelasan Komponen Arsitektur :

a. *Client : User.*
b. *API Gateway*: Mengarahkan *request* dari client ke layanan yang sesuai.

c. *Product Service*: Mengelola data produk dengan MongoDB.
d. *Order Service*: Menangani transaksi pemesanan dengan *PostgreSQL*.
e. *Payment Service*: Mengelola pembayaran dengan *MySQL*.

4.3. Implementasi Setiap Layanan

Pada bagian ini dijelaskan bagaimana setiap layanan diimplementasikan, termasuk struktur *database* dan *endpoint API* yang digunakan. Yang pertama adalah *API Gateway* ini akan dikembangkan menggunakan *Spring Cloud Gateway*. Komponen ini bertugas menerima permintaan dari *client*, kemudian meneruskan permintaan tersebut ke layanan *microservices* yang sesuai berdasarkan *URI* yang digunakan. Konfigurasi routing dilakukan melalui file application.yml, di mana setiap route diarahkan ke layanan *product-service, order-service,* dan *payment-service*. Sistem ini menggunakan *Eureka Server* untuk *service discovery* sehingga setiap *service* dapat terdaftar dan dikenali secara otomatis oleh *API Gateway.*

Kemudian *Product Service* bertanggung jawab dalam pengelolaan data produk. Data produk disimpan di dalam *MongoDB* karena struktur datanya yang fleksibel dan cocok untuk penyimpanan dokumen. *Model Product* memuat atribut seperti *id, name, description, dan price.*
Layanan ini mengimplementasikan dua *endpoint* utama, yaitu:

a. *GET /products*: untuk mengambil seluruh data produk
b. *POST /products*: untuk menambahkan produk baru ke sistem

Kedua *endpoint* ini dikendalikan melalui layer controller dan disambungkan ke service logic yang bertugas mengakses *database*.

Selanjutnya adalah *Order Service*, ini digunakan untuk menangani transaksi pemesanan produk. Layanan ini menyimpan data pesanan ke dalam *PostgreSQL*, karena kebutuhan sistem yang menuntut konsistensi transaksi dan integritas data.

Model Order terdiri dari atribut id, productName, quantity, dan status. Layanan ini menyediakan dua endpoint utama:

a. *POST /orders:* untuk membuat pesanan baru
b. *GET /orders/status/{status}:* untuk melihat daftar pesanan berdasarkan status (misalnya: *PENDING, COMPLETED*)

Semua data pesanan dikelola secara modular, terpisah dari layanan lainnya.

Dan yang terakhir adalah *Payment Service*, ini bertugas memproses transaksi pembayaran untuk setiap pesanan yang telah dibuat. Data pembayaran disimpan di dalam *MySQL* karena sistem membutuhkan kecepatan dan efisiensi dalam pemrosesan transaksi keuangan.

a. *Model Payment* memiliki atribut id, *orderId, amount*, dan *paymentStatus.* Endpoint utama yang tersedia adalah:
b. *POST /payments:* menerima data pembayaran dan menyimpannya dalam system.
c. Layanan ini juga dibangun terpisah dari Order dan *Product*, menjaga prinsip *loose coupling* dalam arsitektur *microservices.*

4.4. Pengujian Sistem

Proses pengujian sistem dilakukan untuk memastikan bahwa fungsionalitas dan kinerja sistem berjalan sesuai dengan kebutuhan yang telah dirancang. Pengujian ini bertujuan untuk mendeteksi kesalahan, memastikan setiap layanan API bekerja dengan benar, serta mengukur performa sistem saat menerima beban tinggi. Pengujian dilakukan melalui dua tahap utama sebagai berikut :

a. Pengujian Fungsionalitas dengan Postman
   Bertujuan untuk memverifikasi bahwa setiap *endpoint API* dapat merespons dengan benar sesuai dengan permintaan yang diberikan.
   - *Product Service* meliputi *GET /products* untuk mendapatkan semua produk. dan *POST /products* untuk enambahkan produk baru.
   - *Order Service* meliputi P*OST /orders* untuk embuat pesanan baru dan *GET/orders/status/{status}*untuk mendapatkan pesanan berdasarkan status.
   - *Payment Service* hanya dengan menggunakan method *POST /payments* digunakan untuk memproses pembayaran.
b. Pengujian Kinerja dengan *JMeter*
   Digunakan untuk mengukur waktu respons dan kemampuan sistem dalam menangani Melakukan Pengujian kinerja dilakukan untuk mengukur kecepatan respon, stabilitas, dan kemampuan sistem dalam menangani beban yang berbeda-beda. dilakukan uji beban pada *endpoint* utama, seperti */products*, */orders*, dan */payments*, dengan jumlah data yang bervariasi, yaitu:
   - 1 data untuk memastikan sistem dapat merespons permintaan dasar.
   - 10 data untuk menguji kinerja pada jumlah permintaan yang sedikit.
   - 100 data untuk mengukur stabilitas sistem pada beban menengah.
   - 500 data untuk menguji kemampuansistem dalam menangani beban tinggi.

Pengujian ini dilakukan menggunakan *JMeter* untuk menganalisis: Waktu respons dan *throughput.* Dengan variasi jumlah data ini, diharapkan dapat mengetahui sejauh mana performa sistem
mampu menangani lonjakan permintaan secara efisien.

4.5. Hasil Pengujian fungsionalitas

Pengujian fungsionalitas dilakukan untuk memastikan bahwa setiap *endpoint* dari layanan *microservices* memberikan hasil yang sesuai dengan fungsinya. Pengujian dilakukan menggunakan Postman dengan hasil sebagai berikut:

Tabel 1. Tabel hasil pengujian fungsionalitas

Nama Layanan	Endpoint	Metode	Tujuan	Status Pengujian
Product Service	/products	GET	Menampilkan daftar produk	Berhasil /response 200
Product Service	/products	POST	Menambahkan produk baru	Berhasil /response 200
Order Service	/orders	POST	Membuat pesanan baru	Berhasil /response 200
Order Service	/orders/status/{status}	GET	Melihat pesanan berdasarkan status	Berhasil /response 200

Dari table di atas bisa dilihat bahwa pengujian dilakukan dengan mengirim *request* menggunakan metode *HTTP* yang sesuai (*GET* atau *POST*) dan mengevaluasi *response code* yang diterima. Hasil pengujian menunjukkan bahwa seluruh endpoint merespons dengan status kode 200 OK, yang menandakan bahwa layanan berfungsi dengan baik sesuai ekspektasi. Ini membuktikan bahwa setiap layanan baik untuk pengelolaan produk, pemesanan, maupun pembayaran telah berhasil diimplementasikan secara fungsional dan dapat saling berkomunikasi melalui arsitektur *microservices* yang dibangun.

4.6. Hasil Pengujian Kinerja

Pengujian dilakukan pada *endpoint* utama sistem *e-commerce* dengan jumlah data sampel berbeda: 1, 10, 100, dan 500. Pengujian ini bertujuan untuk mengukur waktu *respons, throughput,* dan kestabilan sistem saat beban meningkat, berikut ini hasilnya :

Tabel 2. Tabel hasi uji kinerja

Jumlah Pengguna	GET /products (ms)	POST /products (ms)	POST /orders (ms)	POST /payments (ms)	Throughput
1	240	13	90	19	12.7 req/sec
10	87	37	187	33	11.9 req/sec
100	17	10	14	17	35.1 req/sec
500	31	6	15	11	82.3 req/sec

Dari table di atas bis akita ketahui bahwa :

a. *Respon Time*
   - *GET /products* menunjukkan peningkatan efisiensi signifikan, dari 240 ms menjadi 31 ms saat diuji dari 1 hingga 500 pengguna. Hal ini kemungkinan disebabkan oleh *caching* atau optimasi *query*.
   - *POST /products* juga mengalami penurunan waktu respon dari 13 ms menjadi 6 ms, menunjukkan efisiensi dalam penyimpanan data produk.
   - *POST /orders* menunjukkan waktu respon yang relatif stabil di sekitar 15 ms, meskipun terdapat sedikit fluktuasi.
   - *POST /payments* konsisten di kisaran 11 ms bahkan saat beban tinggi, menunjukkan bahwa proses pembayaran berjalan stabil.

b. *Throughput*
   Throughput meningkat dari 12.7 *req/*sec (1 user) menjadi *82.3 req/sec (500 user),* menunjukkan bahwa sistem dapat menangani permintaan dalam jumlah besar secara efisien tanpa terjadi *bottleneck.*

.

5. KESIMPULAN DAN SARAN

Penelitian ini berhasil mengembangkan *RESTful API* untuk sistem *e-commerce* ritel menggunakan pendekatan m*icroservices* berbasis *Spring Boot*, dengan tiga layanan utama yaitu *Product Service, Order Service,* dan *Payment Service* yang diintegrasikan melalui *API Gateway.* Seluruh layanan telah diuji secara fungsional dan menunjukkan hasil yang sesuai dengan tujuan sistem, serta mampu menangani beban pengguna dalam jumlah besar dengan waktu respon dan throughput yang optimal. Hal ini membuktikan bahwa arsitektur *microservices* efektif dalam membangun sistem *backend* yang skalabel dan efisien. Untuk pengembangan lebih lanjut, sistem dapat dilengkapi dengan fitur keamanan seperti autentikasi dan otorisasi guna meningkatkan keandalan dalam lingkungan produksi.

DAFTAR PUSTAKA

[1] International Trade Administration, "Impact of COVID Pandemic on E-Commerce," *trade.gov*, 2021. [Online]. Available: https://www.trade.gov/impact-covid-pandemic-e-commerce. [Accessed: Jul. 10, 2025].

[2] Rumah Coding, "Membangun Aplikasi Berbasis Microservice dengan Spring Boot dan Kubernetes," *rumahcoding.co.id*, 2023. [Online]. Available: https://www.rumahcoding.co.id/artikel/membangun-aplikasi-berbasis-microservice-dengan-spring-boot-dan-kubernetes. [Accessed: Jul. 10, 2025].

[3] L. Goel, "Microservices architecture with Spring Boot for financial services," *Int. J. Creative Research Thoughts (IJCRT)*, vol. 12, no. 6, pp. 234–245, 2024.

[4] V. Shyrobokov, “Using Spring Boot to simplify the development of multi-module services,”

Univ. Library Eng. Technol. Res., vol. 2, no. 1, pp. 31–36, 2025.

[5] A. R. Guntakandla, "Microservices and modular architecture: Revolutionizing e-commerce scalability," *J. Comput. Sci. Technol. Stud.*, vol. 7, no. 4, pp. 133–139, May 2025.

[6] L. Rahmi and S. Syah, "E-Commerce information system design using Spring Boot framework and PostgreSQL," *J. Sainstek*, vol. 10, no. 2, pp. n/a , 2024.

[7] M. Shaikh and P. Devgun, "Building scalable and secure microservices with Spring Boot," *Int. Res. J. Modern. Eng. Technol. Sci.*, vol. 7, no. 5, May 2025.

[8] R. Tsyganok, "Methodology for building scalable microservice architectures on Go for high-load e-commerce platforms," *Univ. Library Eng. Technol. Res.*, vol. 1, no. 2, pp. 42–46, 2024.

[9] I. A. K. P. Paramitha, D. M. Wiharta, and I. M. A. Suyadnya, "Perancangan dan implementasi RESTful API pada sistem informasi manajemen dosen Universitas Udayana," *J. SPEKTRUM*, vol. 9, no. 3, pp. 15–25, 2022.

[10] Sanjaya and Murnawan, "Pemanfaatan arsitektur microservice untuk peningkatan performansi website Lomba Nasional Kreativitas Mahasiswa," *J. Teknol. Inf. dan Ilmu Komputer (JTIIK)*, vol. 11, no. 1, pp. n/a , 2024.

[11] A. Barczak and M. Barczak, "Performance comparison of monolith and microservices based applications," in *Proc. IEEE Int. Conf. Cloud Engineering (IC2E)*, 2020, pp. 120–125.