*JUBIKOM | Jurnal Aplikasi Bisnis dan Komputer Vol. 2 (1): 14-19, Februari 2022* p**ISSN 2807-5986**
http://www.jubikom.unpak.ac.id **eISSN 2807-1875**

# DESAIN PLATFORM *MONITORING* DAN *OBSERVABILITY* UNTUK *MICROSERVICE* BERBASIS ELASTIC STACK

# DESIGN OF MONITORING AND OBSERVABILITY PLATFORM FOR MICROSERVICES BASED ON ELASTIC STACK

Fahmi Noor Fiqri[1] Irma Anggraeni[2]
[12]Ilmu Komputer, Fakultas Matematika dan Ilmu Pengetahuan Alam, Universitas Pakuan
[1]fahminlb33@gmail.com [2]irmairhamna@unpak.ac.id

## ABSTRAK

*Sistem monitoring dan observability untuk microservice menggunakan Elastic Stack ini merupakan implementasi dashboard dan pelaporan yang bertujuan untuk menghadirkan sistem yang tersentralisasi bagi tim bizops dan tim teknis di Logee Trans untuk memudahkan proses pendeteksian, diagnosis, dan penyelesaian masalah pada sistem yang sedang beroperasi. Sistem ini dibangun menggunakan Elastic Stack yang terdiri atas Elasticsearch, Kibana, dan Logstash. Metode penelitian yang digunakan adalah pendekatan Software Development Life Cycle (SDLC) dan telah berhasil menghasilkan produk berupa dasbor yang dapat memberikan rangkuman aktivitas sistem dan performanya. Setelah dilakukan dua sesi pengukuran untuk mengidentifikasi masalah performa, penggunaan dasbor ini dapat membantu developers untuk meningkatkan performa sistem sebesar 30%.*

Kata kunci: *monitoring, observability, microservices, elastic stack.*

## *ABSTRACT*

*This monitoring and obserbavility for microservices is a dashboard and alerting implementation used to create a centralized system for busines operations and technical team at Logee Trans which enables them to easily detect, diagnose, and troubleshoot problems running in production. This system is built using Elastic Stack which consists of Elasticsearch, Kibana, and Logstash. Software Development Life Cycle (SDLC) are the used methodology in this research paper and have been successfully delivered a dashboard product which can delivers summaries of system activities and performances. This system is then tested two times to identify possible performance problems in the current system and the usage of this dashboard has helped the developers to improve the system performance by 30%.*

*Keywords: monitoring, observability, microservices, elastic stack.*

## PENDAHULUAN

*Monitoring* merupakan salah satu bagian dari tahapan penting dalam *Software Development Life Cycle* (SDLC) yaitu *Maintenance* [1] yang berfungsi untuk mengamati penggunaan sumber daya dan performa sistem komputer [2]. Informasi ini sangat penting dalam suatu ekosistem aplikasi untuk dapat mengamati kondisi internal aplikasi dan menjaga sistem yang stabil dan operasional. Kemampuan suatu sistem untuk dapat merekonstruksi kondisi internalnya berdasarkan outputnya disebut sebagai *observability* [3]. Sifat ini merupakan dasar dari *monitoring*, yaitu dengan mengamati output dari sistem, diharapkan dapat dilakukan diagnostik sistem untuk mengetahui penyebab gangguan dan kemungkinan optimalisasi kinerja sistem.

Properti sistem ini merupakan salah satu properti penting yang harus dimiliki oleh suatu sistem khususnya sistem yang menganut arsitektur *microservice* atau *service oriented architecture*.

*Microservice architecture* merupakan metode rekayasa sistem yang sangat diautomasi, sistem perangkat lunak yang dapat berkembang dan terdiri atas satu atau lebih *microservice*. Sedangkan *microservice* adalah komponen yang dapat di *deploy* pada suatu konteks tertentu yang dapat berkolaborasi melalui komunikasi berbasis pesan (*message-based communication*) [4]. Hal ini menghasilkan sistem yang dapat dikembangkan secara independen dan terisolasi satu dengan lainnya sehingga memudahkan proses pengembangan perangkat lunak [5]. Dengan demikian, selain waktu pengembangan yang lebih efektif, performa, dan biaya juga dapat diminimalisasi.

Di sisi lain, dengan mengimplementasikan arsitektur *microservice*, jumlah komponen di dalam suatu sistem dapat bertambah pesat seiring dengan berkembangnya fitur aplikasi. Hal ini menyebabkan proses *monitoring* menjadi lebih sulit karena jumlah komponen yang banyak. Selain itu, tantangan untuk dapat mengumpulkan, menganalisis, dan mengambil kesimpulan berdasarkan data dari banyak komponen menjadi lebih sulit karena diperlukan suatu sistem yang terdistribusi yang mampu menerima, mengolah, dan menyajikan data secara *realtime*.

Hal-hal tersebut merupakan masalah yang sering dihadapi oleh tim *bizops* dan tim teknis di PT XYZ salah satu anak perusahaan bergerak di bidang logistik. Data yang tersebar dan tidak adanya sistem yang terintegrasi untuk menampilkan rangkuman dan catatan detail sistem menyulitkan *tim bizops* dan tim teknis untuk mendeteksi dan memberikan respons cepat ketika terjadi kesalahan pada aplikasi yang digunakan oleh pelanggan. Untuk mengatasi masalah ini, *Elasticsearch* yang merupakan layanan analisis *realtime* dan terdistribusi hadir untuk menjadi solusi penyimpanan dan eksplorasi data dalam jumlah besar dari sistem dengan banyak komponen [6]. *Elasticsearch* merupakan satu bagian dari *Elastic Stack*, yaitu ekosistem analisis data dari *Elastic*. Pada *Elastic Stack* terdapat banyak layanan untuk melakukan *monitoring* dan analisis seperti *Kibana* dan *Logstash*.

Pemanfaatan *Elastic Stack* ini telah digunakan pada berbagai kasus, misalnya Bagnasco dkk. menggunakan *Elasticsearch, Kibana,* dan *Logstash* untuk memonitor dan mengaudit penggunaan sumber daya layanan *cloud*-nya untuk memproses data pada sistem kolaborasi ALICE di LHC dan kemudian mengatur alokasi sumber daya komputasi secara dinamis menggunakan data yang dianalisis pada Kibana [7]. Selain itu pada penelitian lain *stack* yang sama juga digunakan sebagai pusat komunikasi perangkat IoT [8]. Pada kasus ini Elastic digunakan untuk menyimpan data dan melakukan visualisasi data dari perangkat IoT dan kemudian digunakan untuk menghasilkan *insight* dari data yang direkam.

Penelitian lain menggunakan *Elasticsearch* dan *Kibana* untuk melakukan analisis data dari media sosial [9]. Pada penelitian ini peneliti menggunakan Kibana untuk mengolah data berupa tren politik, iklan, dan kesadaran program kesehatan dari pemerintah secara *realtime*. Dibandingkan dengan pengolahan data secara tradisional yang hanya dapat mengolah data secara *offline*, dengan menggunakan *Elastic Stack* analisis dapat dilakukan secara kontinu dan efektif. Berdasarkan latar belakang tersebut, penulis tertarik untuk mengangkat penelitian yang berjudul *Rancang Bangun Platform Monitoring dan Observability untuk Microservice berbasis Elastic Stack* untuk menghadirkan layanan pencatatan dan dasbor untuk mengamati aktivitas dan performa sistem secara *realtime*.

## METODE PENELITIAN

Metode penelitian yang digunakan pada laporan ini adalah *Software Development Life Cycle (SDLC)* yaitu suatu metodologi dengan proses-proses yang didefinisikan dengan jelas untuk menghasilkan perangkat lunak dengan kualitas tinggi [1].

**Perencanaan**

Pada tahap perencanaan dilakukan untuk mengumpulkan informasi dari sistem yang ada untuk membentuk analisis awal sistem. Metode identifikasi yang dilakukan penulis yaitu observasi sistem yang ada di Logee Trans dan proses yang berjalan pada sistem tersebut, dilanjutkan dengan wawancara teknis mengenai sistem yang ada, teknologi yang digunakan, dan bagaimana proses kerja sistem.

**Analisis**

Pada tahap analisis dilakukan untuk proses analisis masalah pada sistem yang ada untuk memformulasikan solusi dan merancang sistem baru untuk menyelesaikan masalah tersebut. Hasil analisis akan digunakan sebagai basis untuk menyusun spesifikasi sistem yang nantinya dikembangkan. Proses yang dilakukan pada analisis ini adalah memahami dan mengidentifikasi permasalahan yang terjadi dan menarik simpulan dari proses analisis yang telah dilakukan. Pembuatan platform *monitoring* ini berdasarkan ekosistem *microservice* yang ada di Logee Trans dan kebutuhan bisnis untuk dapat menyediakan laporan secara *realtime* dalam bentuk dasbor dan visualisasi.

**Perancangan**

Tahap perancangan sistem dilakukan melalui tiga tahap dimulai dari perancangan basis data dilakukan menggunakan pendekatan pemodelan data semi terstruktur menggunakan *Unified Modelling Language (UML)*, dilanjutkan dengan perancangan sistem secara keseluruhan dilakukan dengan menggunakan diagram konteks, *data flow diagram (DFD)*, *flowchart* sistem, dan *mockup* sistem dengan menggunakan simbol untuk menggambarkan proses yang terjadi dalam program komputer secara sistematis, dan terakhir dilakuan perancangan sistem secara detail yang mencakup perancangan visualisasi dan dasbor yang akan digunakan pada sistem yang dikembangkan.

**Implementasi**

Pada tahap implementasi analisis dan rancangan yang sudah dibangun kemudian direalisasikan menjadi sebuah sistem sesuai dengan target. Sistem akan dijalankan di atas *Docker container* [10] menggunakan *Docker Compose* [11] untuk memudahkan proses peluncuran sistem *Elastic Stack*. Sumber data yang digunakan adalah *microservice* yang ada pada ekosistem Logee Trans yang menjadi subjek APM.

**Uji Coba**

Pada tahap ini akan dilakukan uji coba untuk mengevaluasi apakah output dari sistem sudah sesuai dengan hasil yang diinginkan. Proses uji coba ini dibagi menjadi beberapa bagian, yaitu uji struktural, dilakukan untuk mengetahui apakah sistem telah terstruktur dengan baik yang berdasarkan fitur-fitur aplikasi yang telah memenuhi kebutuhan pengguna; uji fungsional, dilakukan untuk mengetahui proses navigasi dan validasi aplikasi apakah sudah sesuai dengan baik sebagaimana rancangan fungsinya; uji validasi, dilakukan untuk menguji data input dan memvalidasi apakah output yang diberikan sistem sudah sesuai.

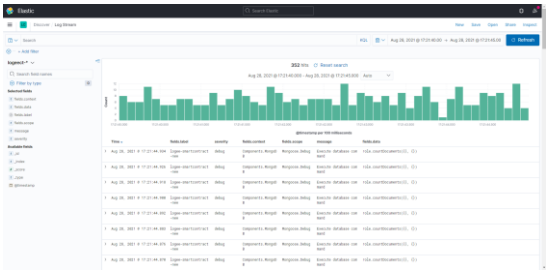
**Penggunaan Sistem**

Tahap penggunaan sistem merupakan target akhir dari proses pengembangan aplikasi, yaitu untuk menciptakan platform *monitoring* ekosistem aplikasi di Logee Trans untuk menghadirkan *insight* melalui visualisasi dan dasbor yang dapat diakses dan mudah dipahami oleh pengguna.

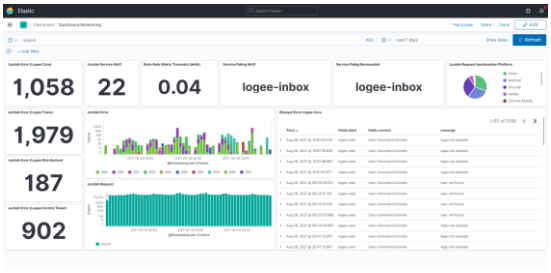
## HASIL DAN PEMBAHASAN

Produk dasbor yang dihasilkan pada penelitian ini terdiri atas empat komponen tampilan, yaitu *log stream*, dasbor, APM, dan *APM transaction*. Masing-masing komponen tersebut secara umum memiliki fungsi yang sama untuk menyediakan akses untuk memonitor kondisi sistem secara *realtime*, tetapi perbedaan antara keempat tampilan tersebut adalah jenis ringkasan informasi yang disajikan. Pada tampilan *log stream*, informasi yang ditampilkan merupakan catatan mentah dari sistem yang dapat digunakan oleh *developers* untuk melakukan diagnosis dan pemecahan masalah pada sistem, sedangkan pada halaman dasbor terdapat agregasi atau rangkuman dari catatan mentah sistem dan performansi yang dikumpulkan melalui agen APM sehingga dapat memberikan gambaran umum mengenai kondisi sistem yang dapat dipahami oleh tim bisnis dan tim teknis.

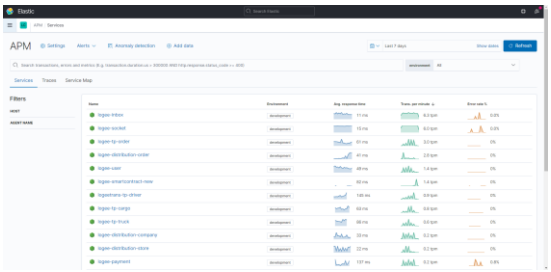
Halaman APM menampilkan rangkuman performa respons untuk semua komponen di dalam sistem dan halaman *APM transaction* menampilkan informasi yang mendetail mengenai performa satu komponen yang ada di dalam sistem. Pada halaman ini tim teknis dapat melihat apa saja komponen atau dependensi yang digunakan oleh sistem untuk membantu mengidentifikasi masalah dan optimisasi performa sistem.



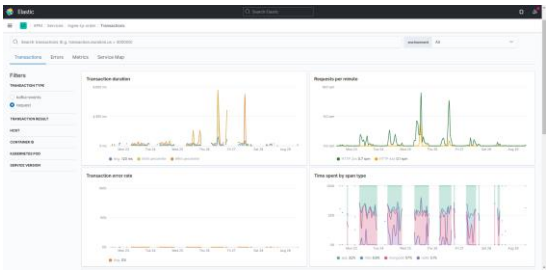
Gambar 1. Tampilan *Log Stream.*




Gambar 2. Tampilan Dasbor.



Gambar 3. Tampilan APM.



Gambar 4. Tampilan *APM Transaction*.

Setelah sistem yang dirancang diintegrasikan dengan keseluruhan sistem yang ada di Logee Trans, dilakukan dua iterasi pengukuran performansi sistem, iterasi pertama dilakukan untuk mengukur performa awal sistem, kemudian dilanjutkan dengan proses identifikasi dan optimisasi sistem berdasarkan informasi yang didapatkan dari sistem *observability*. Setelah potensi optimisasi ditemukan dan diperbaiki, dilakukan pengujian iterasi kedua untuk mengukur seberapa besar perubahan waktu respons sistem (*average response duration*) setelah dilakukan optimisasi.

Proses uji dilakukan melalui teknik *load test* menggunakan Apache JMeter terhadap tujuh API yang tersedia pada komponen *logee-user*. Komponen ini merupakan salah satu komponen inti pada sistem Logee Trans yang melayani permintaan seperti *login*, *logout*¸ dan otentikasi pengguna ke dalam sistem serta menyediakan layanan *Single Sign On (SSO)* dan informasi pengguna pada sistem. Karena fungsinya yang sangat krusial, komponen ini menjadi kandidat pengujian performa aplikasi pada penelitian ini.

Tabel 1. Tabel Perbandingan *Average Transaction Duration*.

| Transaksi | *Avg. transaction duration (ms)* | | % Perubahan |
|---|---|---|---|
| | Sesi 1 | Sesi 2 | |
| *GET /user/v1/tracking* | 133 | 99 | 26% |
| *POST /user/v1/distribution/sales-order* | 478 | 186 | 61% |
| *POST /user/v1/distribution/store* | 402 | 209 | 48% |
| *POST /user/v1/distribution/store-rural* | 259 | 149 | 42% |
| *POST /user/v1/login* | 83 | 77 | 7% |
| *POST /user/v1/logout* | 64 | 56 | 13% |
| *POST /user/v2/login-koja* | 297 | 259 | 13% |
| Rata-Rata Perubahan | | | **30%** |

Pada Tabel 1 di atas secara umum terjadi peningkatan sebesar 30% pada tujuh transaksi API yang masuk ke dalam sistem Logee Trans, dengan optimisasi yang paling tinggi terdapat pada fungsi *distribution/sales-order* yang berhasil mendapatkan peningkatan sebesar 61% dari awalnya membutuhkan waktu respons selama 478 ms menjadi 186 ms. Peningkatan performa ini dapat meningkatkan pengalaman pengguna aplikasi Logee Trans sehingga pengguna tidak perlu menunggu dalam waktu yang lama agar proses transaksinya dapat diselesaikan oleh sistem, sehingga peningkatan performa ini dapat memberikan valuasi bisnis yang lebih tinggi bagi perusahaan.

## KESIMPULAN

Produk dasbor yang dibuat mampu merekam dan menyajikan rangkuman aktivitas dan performansi sistem *microservice* di perusahaan logistik Logee Trans dan memberikan *insight* bagi tim bisnis dan tim teknis. Dasbor yang dibangun menggunakan *Elastic Stack* ini selain mampu merekam dan menyajikan rangkuman data aktivitas dan performa sistem, dasbor ini juga dapat digunakan untuk melakukan diagnostik sistem untuk mengidentifikasi titik kelemahan sistem dan menjadi dasar untuk melakukan optimisasi sistem. Berdasarkan hasil uji coba selama dua iterasi, berhasil dilakukan identifikasi dan optimisasi sistem yang menghasilkan rata-rata peningkatan performa sistem (*average response duration*) sebesar 30%.

## DAFTAR PUSTAKA

[1] A. Altvater, "What Is SDLC? Understand the Software Development Life Cycle," *Stackify*, 2020. https://stackify.com/what-is-sdlc/ (accessed Jun. 21, 2021).
[2] G. Wiesen and H. Bailey, "What Is a System Monitor?," *wiseGEEK*, 2010. https://web.archive.org/web/20101207054610/https://www.wisegeek.com/what-is-a-system-monitor.htm (accessed Jun. 21, 2021).
[3] Y.-Y. Liu, J.-J. Slotine, and A.-L. Barabási, "Observability of complex systems," *Proc. Natl. Acad. Sci.*, vol. 110, no. 7, pp. 2460–2465, Feb. 2013.
[4] I. Nadareishvili, R. Mitra, M. McLarty, and M. Amundsen, *Microservice Architecture: Aligning Principles, Practices, and Culture*. O'Reilly Media, Inc., 2016.
[5] W. Hasselbring and G. Steinacker, "Microservice Architectures for Scalability, Agility and Reliability in E-Commerce," in *2017 IEEE International Conference on Software Architecture Workshops (ICSAW)*, Apr. 2017, pp. 243–246. doi: 10.1109/ICSAW.2017.11.
[6] C. Gormley and Z. Tong, *Elasticsearch: The Definitive Guide: A Distributed Real-Time Search and Analytics Engine*. O'Reilly Media, Inc., 2015.
[7] S. Bagnasco, D. Berzano, A. Guarise, S. Lusso, M. Masera, and S. Vallero, "Monitoring of IaaS and scientific applications on the Cloud using the Elasticsearch ecosystem," *J. Phys. Conf. Ser.*, vol. 608, p. 012016, May 2015, doi: 10.1088/1742-6596/608/1/012016.

[8] M. Bajer, "Building an IoT Data Hub with Elasticsearch, Logstash and Kibana," in *2017 5th International Conference on Future Internet of Things and Cloud Workshops (FiCloudW)*, Aug. 2017, pp. 63–68. doi: 10.1109/FiCloudW.2017.101.
[9] N. Shah, D. Willick, and V. Mago, "A framework for social media data analytics using Elasticsearch and Kibana," *Wirel. Netw.*, Dec. 2018, doi: 10.1007/s11276-018-01896-2.
[10] J. Cito, G. Schermann, J. E. Wittern, P. Leitner, S. Zumberi, and H. C. Gall, "An Empirical Analysis of the Docker Container Ecosystem on GitHub," in *2017 IEEE/ACM 14th International Conference on Mining Software Repositories (MSR)*, May 2017, pp. 323–333. doi: 10.1109/MSR.2017.67.
[11] R. Smith, *Docker Orchestration*, 1st ed. Birmingham: Packt Publishing, 2017.