ISSN 2085-4552

Kinerja Arsitektur Interoperabilitas Menggunakan *Government Service Bus* (GSB) dan *Peer to Peer* (P2P)

Eko Wiyatnanto[1], Arief Indriarto Haris[2]
[1,2]Pusat Teknologi Informasi dan Komunikasi Penerbangan dan Antariksa, Lembaga Penerbangan dan Antariksa Nasional (LAPAN), Jakarta, Indonesia
[1]eko.wiyatnanto@lapan.go.id, [2]arief.indriarto@lapan.go.id

Diterima 05 November 2020
Disetujui 24 Mei 2021

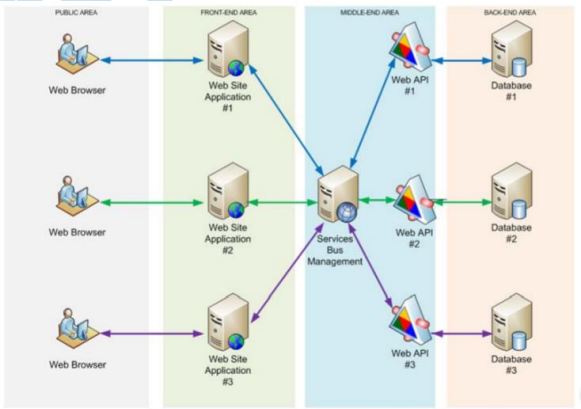
***Abstract* — The role of interoperability architecture is one of the solutions to data redundancy problems and data differences that cause the level of data accuracy to be low. However, the performance of the interoperability architecture needs to be evaluated as an effort to improve the quality of the application itself. This study aims to evaluate the interoperability architecture between architectures using the Government Service Bus (GSB) and Peer to Peer (P2P), through several tests, namely load testing and stress testing. Load testing and stress testing aim to measure the speed and resilience of an application by sending requests and measuring the response of the application. The difference is that in load testing, testing is carried out with certain load conditions, while in stress testing, testing is carried out under extreme conditions. Testing load testing, determining the load condition with a multiple of the number of users 100 to 500, with the JMeter tools. In stress testing, testing is carried out with a total of 1000 users, with the Loader.io tools. The results of the load testing show that the average access time for GSB is smaller than P2P, with an average of 2ms to 309ms and a request error rate of 0%. The results of stress testing show that the GSB architecture is faster than the P2P architecture with an average difference in access time of 2957ms, a difference in throughput of 1.5 request/second, but GSB has a higher request error rate than P2P with a difference of 16.05%. In general conditions with certain loads, the GSB architecture has superior performance, this can be seen from the access time and request errors. Meanwhile, in extreme conditions the GSB architecture experiences a decline in performance, this can be seen from a larger request error rate when handling very large requests.**

***Index Terms*—Government Service Bus; Interoperability Architecture; MANTRA**

I. PENDAHULUAN

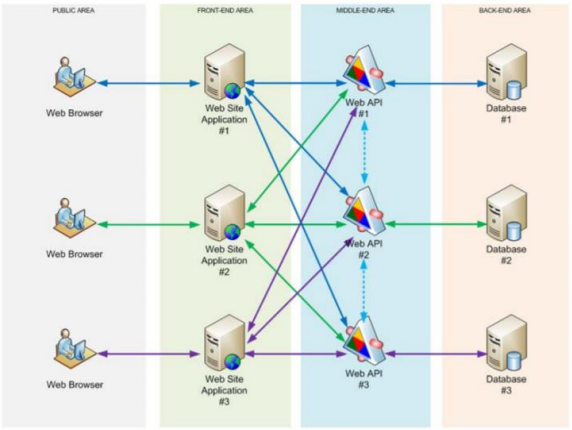
Interoperabilitas *data* aplikasi di lingkungan Lembaga Penerbangan dan Antariksa Nasional (LAPAN) sangat dibutuhkan untuk mendorong efesiensi serta penghematan biaya pengembangan aplikasi. LAPAN memiliki aplikasi dengan berbagai bahasa pemrograman dan framework aplikasi yang berbeda-beda dan hampir setiap aplikasi dikembangkan menggunakan DBMS (*Database Management System*). Pembangunan aplikasi terus dilakukan tanpa interkoneksi data mengakibatkan terjadi redudansi *data* dan perbedaan *data* sehingga keakuratan *data* menjadi rendah. Hal ini mengakibatkan ketersediaan *data* tidak optimal. Oleh karena itu perlu dibangun sistem yang mampu membuat interkoneksi sebagai media pertukaran *data* dan informasi antar sistem.

Secara sederhana interoperabilitas dapat dikatakan sebagai suatu sistem yang memiliki kemampuan berkomunikasi dan bekerja sama dengan sistem lain tanpa adanya batasan terhadap akses informasi [1]. *Government Service Bus* (GSB) merupakan suatu sistem yang mengelola integrasi informasi dan membentuk interkoneksi sebagai jembatan pertukaran data antar sistem instansi pemerintah menggunakan *Application Programming Interface* (API) [2].

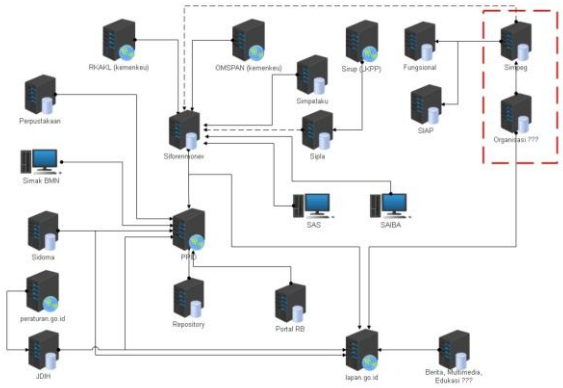


Gambar 1. Topologi Layanan API/Webservice melalui GSB [2]

Pemaanfaatan akses layanan berbagi pakai menggunakan API *Webservice Point to Point* (P2P) atar *server* membentuk koneksi komunikasi sebagai media lalu lintas *data* [2].

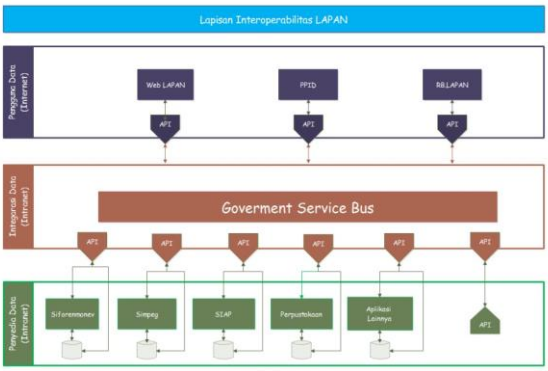


Gambar 2. Topologi Layanan API/*Webservice Point To Point* (P2P) [2]



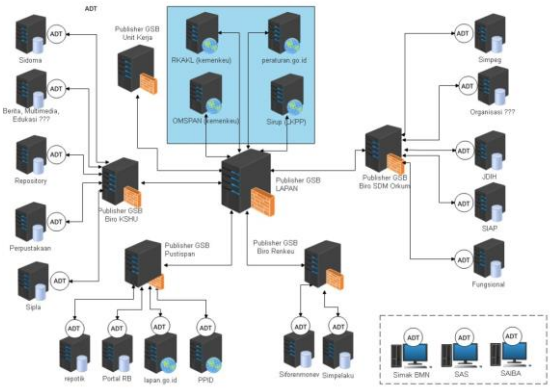
Gambar 3. Arsitektur Interoperabilitas *Peer to Peer* (P2P) [3]

Pada arsitektur *Peer to Peer* (P2P) (Gambar 3), pemberian ijin akses satu aplikasi ke aplikasi lainnya disimpan di masing-masing aplikasi [3].



Gambar 4. Layer/Lapisan Interoperabilitas [3]

Terdapat tiga *layer*/lapisan data pada arsitektur interoperabilitas menggunakan GSB yaitu *layer* penyedia data, *layer* integrasi *data* dan *layer* pengguna data (Gambar 4). Arsitektur interoperabilitas menggunakan GSB memanfaatkan GSB sebagai layer integrasi *data* yang menghubungkan antara *layer* penyedia *data* ke *layer* pengguna *data* [3].



Gambar 5. *Layer*/Lapisan Interoperabilitas [3]

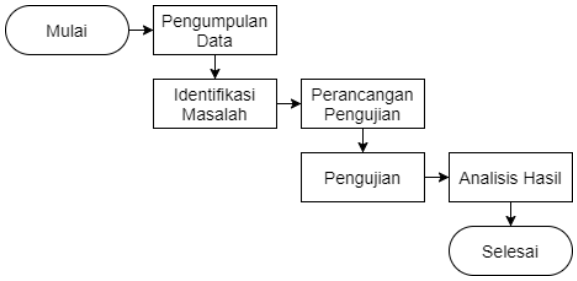
Interkoneksi pada arsitektur interoperabilitas dapat menggambarkan keterkaitan *data* pada setiap aplikasi sehingga dapat meminimalisir redudansi baik data maupun aplikasi (Gambar 5). Dengan arsitektur interoperabilitas setiap aplikasi dapat memanfaatkan dan berbagi *data* dengan aplikasi lain melalui interkoneksi *Rest* API. Dengan demikian data pada setiap aplikasi merupakan *data* tunggal yang dihasilkan oleh produsen *data*/sumber *data* [3].

Pada penelitian sebelumnya, dilakukan perancangan sistem informasi terkait penanggulangan gawat darurat terpadu, yang dimana pengembangan interoperabilitas pada sistem tersebut bertujuan agar dapat melakukan sinkronisasi *data* secara otomatis dengan sistem informasi lainnya [4]. Adapun pada penelitian sebelumnya yang dilakukan di lingkunagan LAPAN, aplikasi MANTRA dan metode *Representational State Transfer* (REST) dapat diterapkan interkoneksi untuk pertukaran *data* dan informasi di antara sistem-sitem yang ada di lingkungan LAPAN [3]. Namun, dalam implementasinya kinerja kedua metode tersebut belum optimal dan belum dilakukan pengujian, diantaranya adalah kemampuan aplikasi pada sisi kecepatan penanganan beban dengan jumlah tertentu dan ketahanan aplikasi dalam kondisi beban yang sangat tinggi.

Oleh karena itu, penelitian ini bertujuan untuk mengevaluasi kinerja dari interoperabilitas *data* aplikasi menggunakan GSB dan P2P. Pengukuran kinerja penting dilakukan karena selama ini kinerja kecepatan dan ketahanan arsitektur kedua metode tersebut memang sama sekali belum pernah dilakukan evaluasi, disamping itu juga sebagai bahan pertimbangan dalam penerapan suatu arsitektur aplikasi interoperabilitas di lingkungan LAPAN. Dengan mengetahui tingkat interoperabilitas suatu layanan, maka tingkat efektifitas layanan yang dihasilkan suatu organisasi dapat diketahui dengan lebih cepat, sehingga memungkinkan untuk dilakukan percepatan peningkatan kualitas layanan di masa mendatang [5].

II. METODOLOGI PENELITIAN

Pada penelitian ini, terdapat serangkaian tahapan yang dilakukan (Gambar 5). Pada tahap pengumpulan data, penulis melakukan observasi langsung terkait kebutuhan sistem informasi di lingkungan LAPAN, studi literatur, dan wawancara terhadap penyedia layanan dan beberapa *stakeholder*. Setelah dilakukan pengumpulan data, penulis melakukan identifikasi masalah terkait kinerja arsitektur interoperabilitas menggunakan GSB dan P2P.



Gambar 6. Tahapan Penelitian

Pada tahap perancangan pengujian, penulis melakukan perancangan terkait metode pengujian, skenario pengujian, hingga *tools* dan *parameter* yang akan digunakan, serta penyiapan instrumen pendukung lainnya seperti penyiapan *endpoint* sebagai target data. Pada tahap pengujian untuk mengukur kinerja GSB dan P2P, penulis menggunakan metode *load testing* dan *stress testing*. *Load testing* dan *stress testing*, keduanya memiliki kesamaan tujuan yaitu untuk mengukur kecepatan dan ketahanan suatu aplikasi dengan cara mengirimkan *request* dan mengukur respon dari aplikasi tersebut. Perbedaannya pada *load testing* dilakukan pengujian dengan *load condition* tertentu, sedangkan pada *stress testing* dilakukan pengujian pada kondisi ekstrem dengan mensimulasikan sejumlah besar *user* mengakses dan mengirimkan *request* ke sistem, sehingga memberikan beban ke sistem dalam memproses banyak *data* [6] [7]. Dengan melakukan pengujian *load testing* dan *stress testing*, akan diperoleh data terkait dengan kehandalan dan performansi dari aplikasi tersebut, untuk selanjutnya dapat dilakukan analisis agar diperoleh solusi untuk peningkatan aplikasi di masa mendatang [8].

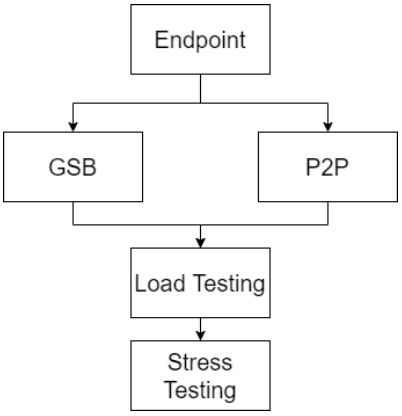
Adapun dalam pengujian *load testing* terdapat metode-metode turunan yang dapat digunakan, dalam penelitian ini penulis menggunakan metode *Black Box Testing*. *Black box testing* akan menguji aplikasi secara fungsional tanpa mengetahui struktur dari program (*no knowledge*) [9]. Sedangkan pada pengujian *stress testing*, penulis menggunakan metode *Gorilla Testing*, dimana modul program diuji berulang kali untuk memastikan ketahanan dan kinerja dari modul tersebut dapat bekerja sesuai dengan fungsinya [9].

Tools yang penulis pergunakan untuk pengujian *load testing* dan *stress testing* adalah Loader.io dan JMeter. Loader.io adalah sebuah *tool* terbuka yang dikembangkan oleh SendGrid. Loader.io memiliki fitur pengujian *Clients per test*, *Clients per second*, dan *Maintain client load tests* dan menyajikan laporan dalam bentuk *visual* [10]. Paket gratis di Loader.io memungkinkan throughput hingga 10.000 permintaan per detik untuk menjalankan tes beban dalam durasi waktu 1 menit [11]. Apache *JMeter* adalah sebuah *load and performance testing tool* terbuka yang dibuat menggunakan bahasa pemrograman Java. JMeter bisa digunakan untuk melakukan uji beban pada API dengan *protocol* HTTP. JMeter juga dapat digunakan untuk aplikasi yang menggunakan server/protocol FTP, SMTP, LDAP, Java *Database Connectiviy* (JDBC), dan *generic* TCP *connections* [12].

Penulis mengevaluasi arsitektur interoperabilitas melalui pengujian dengan membandingkan lamanya waktu eksekusi dan tingkat *stress* pada arsitektur interoperabilitas. Perbandingan kecepatan akses dan ketahanan sistem menjadi tolak ukur performa antara arsitektur interoperabilitas GSB dengan P2P. Oleh karena itu, perlu dilakukan pengujian load test dan stress test untuk mengukur kecepatan akses dan ketahanan sistem. Pengujian *load test* arsitektur interoperabilitas *data* menggunakan GSB dan P2P dapat dilakukan pada *website* PPID LAPAN menggunakan JMeter. Pengujian dilakukan menggunakan kelipatan jumlah pengguna 100, 200, 300, 400, dan 500. Pengguna melakukan akses secara bersamaan dalam 1 detik sehingga dalam 1 detik terdapat 100 user bersamaan melakukan akses dengan menggunakan *protocol REST* dengan *method GET*. Untuk pengujian *stress test*, alat bantu yang digunakan adalah Loader.io. Pada Loader.io dilakukan pengujian 1000 pengguna yang melakukan permintaan secara bersamaan dalam 1 detik selama 1 menit dengan menggunakan *protocol REST* dengan *method GET*.

III. HASIL DAN ANALISIS PENELITIAN

Pada arsitektur interoperabilitas data aplikasi menggunakan GSB dan P2P, integrasi data aplikasi dapat terjadi melalui interkoneksi antara halaman website sebagai *REST API client* dengan sumber data sebagai *enpoint REST API*. Pada arsitektur GSB *REST API client* melakukan permintaan data ke *endpoint* melalui GSB, dimana GSB berperan sebagai proxy untuk *endpoint*, sedangkan pada P2P *REST API Client* melakukan permintaan data langsung ke *endpoint* tanpa melalui GSB. Pengujian pada arsitektur GSB, dilakukan dengan mensimulasikan pengguna sebagai *Rest API client* yang mengakses *endpoint* melalui GSB, sedangkan pada arsitektur P2P mensimulasikan pengguna sebagai *Rest API client* mengakses langsung ke *endpoint*.



Gambar 7. Skema Pengujian

Penulis menyiapkan satu *endpoint* sebagai *target data* untuk kedua arsitektur, dimana *endpoint* tersebut digunakan sebagai target pengujian *load testing* dan *stress testing* pada arsitektur GSB maupun P2P. Pengujian dilakukan dengan *tools* JMeter untuk *load testing* dan Loader.io untuk *stress testing*. Penulis membuat dua *thread* pengujian yaitu:

1. *Thread Load Testing*

Pada proses pengujian ini dilakukan simulasi performa dari sisi kecepatan dan ketahanan beban ketika pengguna mengakses *target data*. Pengujian kecepatan akses dilakukan dengan melakukan percobaan dengan kelipatan jumlah pengguna 100, 200, 300, 400, dan 500. Pengguna melakukan akses secara bersamaan dalam 1 detik sehingga dalam 1 detik terdapat 100 user bersamaan melakukan akses dengan menggunakan protocol *REST* dengan *method GET* ke *endpoint* melalui GSB dan P2P. Setiap kelipatan pengujian dilakukan perekaman hasil kecepatan rata-rata, waktu *minimal* akses, waktu maksimal akses, dan *error request*. Pada *thread* ini dilakukan konfigurasi *thread* pada JMeter dengan memasukkan alamat *endpoint* pada kolom http *request*, menetapkan nilai *Number of Threads (users)* dengan kelipatan jumlah pengguna dan *Ramp-up Periods (seconds)* dengan nilai 1 (detik).

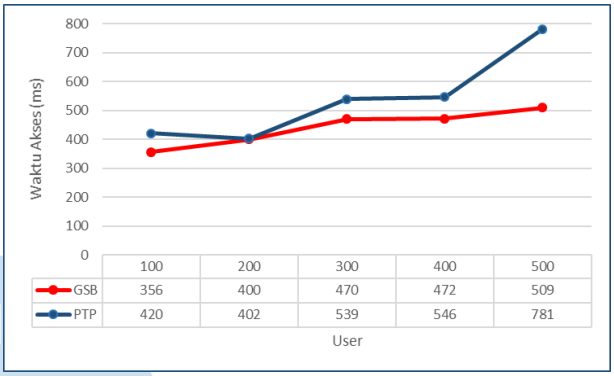
Berdasarkan pengujian dari jumlah *user* 100 hingga 500 dengan kelipatan 100 *user* setiap kali pengujian didapat hasil kecepatan rata-rata akses, waktu minimal akses, waktu maksimal akses, dan *error request* dari setiap percobaan dengan yang dilakukan sebagaimana disajikan pada Tabel 1 dan 2.

Tabel 1. Hasil Pengujian *Load Testing* pada Arsitektur Interoperabilitas GSB

No.	*User*	Rata-rata (ms)	Waktu Akses (ms)		Error Request (%)
			Minimal	Maksimal	
1	100	356	335	577	0
2	200	400	345	1419	0
3	300	470	339	5063	0
4	400	472	350	1302	0
5	500	509	331	1588	0

Tabel 2. Hasil Pengujian *Load Testing* pada Arsitektur Interoperabilitas P2P

No.	*User*	Rata-rata (ms)	Waktu Akses (ms)		Error Request (%)
			Minimal	Maksimal	
1	100	420	337	1365	0
2	200	402	332	1889	0
3	300	539	346	2020	0
4	400	546	330	4749	0
5	500	781	428	5675	0



Gambar 8. Perbandingan Hasil Pengujian Interoperabilitas

Berdasarkan perbandingan hasil pengujian interoperabilitas pada Gambar 8, kecepatan akses arsitektur interoperabilitas data aplikasi menggunakan GSB lebih cepat dibanding dengan arsitektur P2P dilihat dari waktu rata-rata akses arsitektur menggunakan GSB lebih kecil dibanding P2P, dengan rata-rata 2ms hingga 309ms. Pada kondisi dengan jumlah beban tertentu (100 sampai 500 user) arsitektur GSB dan P2P dapat melayani *request* dengan tingkat keberhasilkan 100%, yang mana menggambarkan kedua arsitektur tersebut memiliki tingkat ketahanan yang sama. Sedangkan dari sisi kecepatan, arsitektur GSB lebih unggul dibanding arsitektur P2P.

2. *Thread Stress Testing*

Pada proses pengujian ini dilakukan simulasi performa dari sisi kecepatan dan ketahanan beban dengan kondisi eksrem ketika *user* mengakses *target data*. Pengujian *stress testing* dilakukan menggunakan Loader.io dengan akses secara bersamaan sebesar 1000 *user* dalam 1 detik selama 1 menit dengan menggunakan *protocol REST* dengan *method GET* ke *endpoint* melalui GSB dan P2P. Setiap kelipatan pengujian dilakukan perekaman hasil kecepatan rata-rata, waktu minimal akses, dan waktu maksimal akses.

Pada *thread* ini, penulis melakukan konfigurasi dasar pada Loader.io yaitu dengan menambahkan dan verifikasi *target host* ppid.lapan.go.id. Selanjutnya penulis menetapkan nilai pada kolom pengujian dengan tipe *Client per seconds*, *Users* sebesar 1000, durasi sebesar 1 menit dan *method* GET pada Loader.io.

Tabel 3. Hasil Perbandingan Pengujian *Stress Testing*

	GSB	P2P
Rata-rata (ms)	11198	14155
Min (ms)	306	229
Max (ms)	55353	83263
Error Request (%)	20,99	4,94
Throughput (request/second)	6,5	5,0

Berdasarkan hasil pengujian *stress testing* arsitektur interoperabilitas data, aplikasi menggunakan GSB lebih cepat dibanding dengan arsitektur P2P dengan selisih waktu rata-rata akses 2957ms, selisih *throughput* sebesar 1.5 request per detik, dan tingkat error pada arsitektur interoperabilitas data menggunakan arsitektur GSB lebih tinggi dari arsitektur P2P dengan selisih tingkat error 16,05%. Pada kondisi ekstrem, arsitektur GSB memiliki tingkat kecepatan dan *throughput* yang lebih tinggi, namun memiliki tingkat *error* yang lebih tinggi dibanding arsitektur P2P.

IV. SIMPULAN

Berdasarkan hasil pengujian *load testing* dan *stress testing*, pada kondisi umum dengan beban tertentu, arsitektur GSB memiliki performansi yang lebih unggul, hal ini terlihat dari waktu akses dan *error request*. Sedangkan pada kondisi ekstrem arsitektur GSB mengalami penurunan performansi, hal ini terlihat dari tingkat error request yang lebih besar ketika menangani request dalam jumlah sangat besar.

Pengembangan yang dapat dilakukan pada penelitian selanjutnya adalah melakukan pengujian pada aplikasi lainnya yang menerapkan arsitektur interoperabilitas GSB, P2P, dan arsitektur interoperabilitas alternatif lainnya, yang memiliki keberagaman fitur, kompleksitas, dan dengan memperhatikan perhitungan dan pengalokasian *resource* yang tepat.

UCAPAN TERIMA KASIH

Penulis mengucapkan terima kasih kepada Lembaga Penerbangan dan Antariksa Nasional (LAPAN), khususnya satuan kerja Pusat Teknologi Informasi dan Informasi Penerbangan dan Antariksa yang telah mendukung dan memfasilitasi kegiatan penelitian ini.

DAFTAR PUSTAKA

[1] NCOIC, "What is Interoperability?," *NCOIC.org*, 2018. [Online]. Available: http://www.ncoic.org/what-is-interoperability.

[2] A. Sasongko, H. A. Aziz, P. Setyantana, D. Sukyadi, and A. Basuki, *MANTRA - Manajemen Integrasi Informasi dan Pertukaran Data (Government Service Bus)*. Jakarta: DIREKTORAT E-GOVERNMENT DIREKTORAT JENDERAL APLIKASI INFORMATIKA KEMENTERIAN KOMUNIKASI DAN INFORMATIKA, 2017.

[3] A. Haryanto, E. Wiyatnanto, and R. A. Ferianda, "Perancangan Arsitektur Interoperabilitas Data Aplikasi Menggunakan Government Service Bus (GSB)," in *Kajian dan Penerapan Teknologi Informasi dan Komunikasi*, I. G. Prihanto, Suwardi, H. Gunawan, and F. Alusi, Eds. Jawa Tengah: Penerbit Nugra Media, 2018, pp. 43–52.

[4] D. B. Santoso, A. E. Pramono, and A. G. Persada, "Pengembangan Interoperabilitas Sistem Penanggulangan Gawat Darurat Terpadu (SPGDT) Kabupaten Kebumen," *J. Manaj. Inf. Kesehat. Indones.*, vol. 7, no. 1, p. 43, 2019.

[5] N. Winastan and R. Ferdiana, "Metode Pengukuran Interoperabilitas untuk Mengukur Kinerja Pelayanan Pemerintah," in *Seminar Nasional Aplikasi Teknologi Informasi (SNATI)*, 2014, pp. 13–18.

[6] D. Gea, "Pengujian Kualitas Website Ditinjau dari Perspektif Accessibility, Experience, Marketing dan Technology," *ComTech Comput. Math. Eng. Appl.*, vol. 5, no. 1, p. 35, 2014.

[7] N. M, K. K. S, A. Saha, and D. Chahar, "Comparative Study on Performance Testing with JMeter," *IJARCCE Int. J. Adv. Res. Comput. Commun. Eng.*, vol. 5, no. 2, pp. 70–76, 2016.

[8] E. N. Delta and Asmunin, "Performance Test dan Stress Website Menggunakan Open Source Tools," *J. Manaj. Inform.*, vol. 6, no. 1, pp. 208–215, 2016.

[9] D. I. Permatasari *et al.*, "Pengujian Aplikasi menggunakan Metode Load Testing dengan Apache JMeter pada Sistem Informasi Pertanian," *J. Sist. dan Teknol. Inf.*, vol. 8, no. 1, p. 135, 2020.

[10] Loader.io, "Test Types," *support.loader.io*, 2015. [Online]. Available: http://support.loader.io/article/16-test-types.

[11] Kamarudin, Kusrini, and A. Sunyoto, "Uji Kinerja Sistem Web Service Pembayaran Mahasiswa Menggunakan Apache JMeter (Studi Kasus: Universitas AMIKOM Yogyakarta)," *Teknol. Inf.*, vol. XIII, no. 1, pp. 44–52, 2018.

[12] T. A. S. Foundation, "Apache JMeter," *jmeter.apache.org*, 2018. [Online]. Available: http://jmeter.apache.org/.