Jurnal Teknologi Informasi dan Ilmu Komputer (JTIIK) DOI: 10.25126/jtiik20241128307
Vol. 11, No. 2, April 2024, hlm. 377-384 p-ISSN: 2355-7699
Akreditasi KEMENRISTEKDIKTI, No. 36/E/KPT/2019 e-ISSN: 2528-6579

PEMANFAATAN ARSITEKTUR MICROSERVICE UNTUK PENINGKATAN PERFORMANSI WEBSITE LOMBA NASIONAL KREATIVITAS MAHASISWA

Sanjaya[*1], Murnawan[2]

[1,2]Universitas Widyatama, Bandung
Email: [1]sanjaya@widyatama.ac.id, [2]murnawan@widyatama.ac.id
[*]Penulis Korespondensi

(Naskah masuk: 16 Desember 2023, diterima untuk diterbitkan: 16 April 2024)

Abstrak

Lomba Nasional Kreativitas Mahasiswa (LO Kreatif) merupakan ajang tahunan yang diadakan oleh APTISI 7 Jatim untuk mahasiswa perguruan tinggi swasta di Indonesia. Meskipun proses pendaftaran lomba telah beralih ke platform digital, penelitian terbaru menunjukkan keterbatasan performa website LO Kreatif saat menghadapi lebih dari 1000 pengguna secara bersamaan. Evaluasi performansi menggunakan metode *load testing* mengidentifikasi dua faktor utama yang berkontribusi pada penurunan performa, yaitu penggunaan *file load* pada halaman web dan penggunaan data gambar, proses *query* yang kompleks, serta penggunaan *library*. Penelitian ini mengusulkan solusi dengan merancang arsitektur *microservice* sebagai alternatif terhadap arsitektur monolitik yang digunakan saat ini. Analisis terhadap penelitian sebelumnya menunjukkan bahwa arsitektur monolitik lebih cocok untuk aplikasi dengan jumlah pengguna kecil, sedangkan arsitektur *microservice* menawarkan skalabilitas dan reliabilitas yang lebih baik. Melalui *Design Science Research Methodology (DSRM)*, penelitian ini melibatkan enam tahapan, mulai dari identifikasi permasalahan hingga komunikasi hasil kepada pemangku kepentingan. Hasil evaluasi menunjukkan bahwa arsitektur *microservice* berhasil meningkatkan *throughput* tanpa mengalami penurunan, meskipun rata-rata *error rate* sebesar 8.52% masih memerlukan perbaikan. Dengan demikian, arsitektur *microservice* dapat menjadi solusi untuk meningkatkan performansi website LO Kreatif, namun perhatian terhadap aspek perangkat keras juga krusial untuk mendapatkan performansi optimal.

Kata kunci: arsitektur *microservices*, *error rate*, *load testing*, performansi, *throughput*

UTILIZATION OF MICROSERVICES ARCHITECTURE FOR IMPROVING THE PERFORMANCE OF NATIONAL STUDENT CREATIVITY COMPETITION WEBSITE

Abstract

The National Student Creativity Competition (LO Kreatif) is an annual event organized by APTISI 7 East Java for students from private higher education institutions in Indonesia. Despite the registration process for the competition transitioning to a digital platform, recent research indicates limitations in the performance of the LO Kreatif website when faced with more than 1000 simultaneous users. Performance evaluation through load testing identified two main contributing factors to the performance decline: the use of load files on web pages and the utilization of image data, complex query processes, and library usage. This study proposes a solution by designing a microservice architecture as an alternative to the currently employed monolithic architecture. Analysis of previous research suggests that a monolithic architecture is more suitable for applications with a small user base, while a microservice architecture offers better scalability and reliability. Utilizing the Design Science Research Methodology (DSRM), this research involves six stages, ranging from problem identification to communicating results to stakeholders. Evaluation results show that the microservice architecture successfully increased throughput without experiencing a decline, although the average error rate of 8.52% still requires improvement. Thus, the microservice architecture can be a solution to enhance the performance of the LO Kreatif website, but attention to hardware aspects is crucial for optimal performance.

Keywords: *microservices architecture, error rate, load testing, performance, throughput*

1. PENDAHULUAN

Lomba Nasional Kreativitas Mahasiswa atau LO Kreatif merupakan lomba yang diadakan setiap tahun oleh APTISI 7 Jatim bersama instansi terkait untuk mahasiswa perguruan tinggi swasta di Indonesia. Saat ini lomba kreativitas mahasiswa sudah menggunakan website untuk membantu proses pendaftaran lomba. Namun, berdasarkan hasil

pengujian performansi menggunakan metode *load testing* terhadap website lomba nasional kreativitas mahasiswa yang dilakukan oleh (Hadi et al., 2022) menyebutkan bahwa website lomba nasional kreativitas mahasiswa hanya dapat melayani 500 pengguna yang aktif secara bersamaan namun, ketika menyentuh angka 1000 pengguna keatas, mulai terjadi penurunan performa yang signifikan dengan rata-rata kesalahan 25-28 persen. Menurut (Hadi et al., 2022), terdapat 2 hal yang menyebabkan penurunan performa pada web lomba nasional kreativitas mahasiswa, yaitu penggunaan *file load* pada halaman web yang mempengaruhi kecepatan dalam merespon permintaan pengguna dan penggunaan data gambar, proses *query* yang kompleks, juga penggunaan *library* yang mempengaruhi kinerja pada web. Diketahui bahwa pada tahun 2020 dan 2021 terdapat 1500 sampai 2400 peserta yang mendaftar pada lomba nasional kreativitas mahasiswa.

Berdasarkan penelitian yang telah dilakukan oleh (Hadi et al., 2022) tersebut dan penelitian yang dilakukan oleh (Al-Debagy & Martinek, 2018), dapat dsimpulkan bahwa saat ini website lomba nasional kreativitas mahasiswa masih menggunakan arsitektur monolitik. Menurut (Oboko, 2016) arsitektur monolitik merupakan aristektur yang menekankan bahwa aplikasi berisi komponen-komponen yang saling bergantung sama lain dan harus dikembangkan, dideploy dan dikelola sebagai sebuah kesatuan dikarenakan semuanya dijalankan sebagai sebuah kesatuan proses sistem operasi.

Penelitian yang dilakukan oleh (Al-Debagy & Martinek, 2018) menyebutkan bahwa penggunaan arsitektur berbasis monolitik lebih cocok digunakan untuk aplikasi dengan jumlah pengguna kecil yang kurang dari 100 pengguna, kesimpulan ini berdasarkan hasil pengamatan terhadap *throughput* yang dihasilkan *Apache JMeter* menggunakan metode *load testing dan concurrency testing* pada penelitian tersebut. Sehingga hal tersebut membuatnya kurang cocok bila dipakai untuk website yang dapat diakses secara nasional yang tentunya membutuhkan reliabilitas yang mumpuni. Selain itu kesimpulan yang didapatkan oleh (Blinowski, Ojdowska & Przybylek, 2022), juga menyimpulkan bahwa arsitektur monolitik merupkan pilihan yang lebih baik untuk aplikasi kecil yang tidak dirancang untuk menangani beban pengguna yang tinggi. Oleh karena itu untuk meningkatkan performansi pada website lomba nasional kreativitas mahasiswa maka, peneliti mengusulkan penggunaan arsitektur *microservice* yang memiliki skalabilitas dan reliabilitas yang lebih baik dibandingkan dengan arsitektur monolitik.

Arsitekur *microservice* merupakan gaya arsitektural dalam pengembangan perangkat lunak. Pada gaya arsitektural ini, aplikasi akan dideploy kedalam sekumpulan *service* kecil yang dapat berdiri secara mandiri namun, juga dapat saling bekerja sebagai sebuah sistem. Menurut (Asri et al., 2022), arsitektur *microservices* memastikan *service* dapat discale secara mandiri dan dikembangkan menggunakan teknologi yang berbeda. Skalabilitas pada perangkat lunak sendiri merupakan kemampuan suatu layanan dalam meningkatkan kapasitas sumber daya saat terjadinya peningkatan beban kerja (Tanuwijaya, Palit & Noertjahyana, 2021). Sedangkan menurut (Insanittaqwa, 2017), reliabilitas pada perangkat lunak merupakan probabilitas kegagalan operasi perangkat lunak dalam periode waktu yang telah ditentukan pada lingkungan tertentu. Skalabilitas dan Reliabilitas merupakan parameter dari sekian banyak parameter yang digunakan untuk memperlihatkan performansi perangkat lunak berbasis web.

Penelitian yang dilakukan oleh (Asrowardi, Putra & Subyantoro, 2019), telah berhasil membuktikan bahwa penggunaan arsitektur *microservice* dapat meningkatkan skalabilitas dan reliabilitas pada *e-commerce* dengan melakukan *stress testing* menggunakan beban sama dengan lebih besar dari 500 terhadap rancangan arsitektur *microservices* untuk *e-commerce* yang telah dikembangkan pada penelitian tersebut. Berdasarkan hasil dari penelitian tersebut, arsitektur *microservice* kemungkinan dapat menjadi solusi untuk meningkatkan performansi pada website lomba nasional kreativitas mahasiswa. Oleh sebab itu maka, penelitian ini bertujuan merancang arsitektur *microservice* untuk membuktikan bahwa penggunaan arsitekur *microservices* dapat meningkatkan performansi pada website lomba nasional kreativitas mahasiswa. Penelitian ini menggunakan metode penelitian *Design Science Research Methodology (DSRM)* yang mana menurut (vom Brocke, Hevner and Maedche, 2020) *Design Science Research Methodology (DSRM)* merupakan metode untuk meningkatkan pengetahuan manusia melalui pembuatan artefak inovatif yang dapat menyelesaikan permasalahan dan meningkatkan lingkungan yang telah diinisiasikan.

2. METODE PENELITIAN

Penggunaan metode penelitian *DSRM* pada penelitian ini adalah untuk merancang arsitektur *microservice* yang sesuai dengan website lomba nasional kreativitas mahasiswa untuk selanjutnya dilakukan evaluasi performansi yang dilakukan untuk mendapatkan hasil pengujian performansi yang dihasilkan oleh arsitektur *microservice* yang dirancang agar dapat memberikan gambaran dari jika arsitektur *microservice* diimplementasikan pada website lomba nasional kreativitas mahasiswa. Metode *DSRM* sendiri memiliki 6 tahapan yaitu 1) Identifikasi Permasalahan dan Motivasi (*Identifying Problem and Motivation*) merupakan tahap dimana peneliti menjelaskan permasalahan penelitian secara spesifik dan menjustifikasi nilai dari solusi. Penjustifikasian nilai solusi dapat memberikan 2

(dua) hal yaitu, memotivasi peneliti dan membantu audiens mengapresiasi pemahaman peneliti terhadap permasalahan yang terjadi. Sumber daya yang diperlukan pada kegiatan ini adalah pengetahuan mengenai permasalahan penelitian dan kepentingan solusinya. 2) Mendefinisikan Tujuan dari Solusi (*Defining Purpose of a Solution* merupakan tahap dimana peneliti menentukan tujuan dari solusi, tujuan dari solusi dapat ditentukan melalui definisi masalah dan pengetahuan mengenai apa yang mungkin dan dapat dilakukan. Tujuan dapat bersifat kuantitatif, contohnya bagaimana solusi yang diinginkan mungkin lebih baik daripada yang sekarang, atau kualitatif, contohnya pendeskripsian bagaimana artefak baru diharapkan dapat menjadi solusi untuk permasalahan yang belum terselesaikan. Tujuan harus disimpulkan secara rasional dari spesifikasi permasalahan. 3) Perancangan dan Pengembangan (*Designing and Development*) merupakan tahap dimana peneliti menciptakan artefak, secara konsep artefak *DSRM* dapat berupa objek apa pun yang dirancang selama terdapat kontribusi penelitian yang tertanam dalam desain tersebut. kegiatan ini termasuk menentukan fungsionalitas artefak yang diinginkan dan arsitekturnya kemudian buat artefak sebenarnya. 4) Demonstrasi (*Demonstration*) merupakan tahap dimana peneliti mendemonstrasikan penggunaan artefak untuk menyelesaikan 1 (satu) atau lebih permasalahan. Kegiatan ini dapat melibatkan penggunaanya dalam eksperimen, simulasi, studi kasus, pembuktian atau kegiatanya lainnya yang sesuai. 5) Evaluasi (*Evaluation*) merupakan tahap dimana peneliti mengukur seberapa baik artefak yang menjadi solusi untuk menyelesaikan permasalahan. Kegiatan ini melibatkan perbandingan tujuan solusi dengan hasil observasi penggunaan artefak sesuai dengan konteks. Kegiatan ini tergantung pada jenis permasalahan dan artefak, evaluasi dapat mengambil banyak bentuk. Di akhir kegiatan peneliti dapat memutuskan apakah mengulang kembali ke tahap ketiga untuk meningkatkan efektivitas artefak atau melanjutkan ke komunikasi dan meninggalkan peningkatan pada proyek berikutnya. 6) Komunikasi (*Communication*) merupakan tahap dimana peneliti mengkomunikasikan semua aspek permasalahan dan artefak kepada pemangku kepentingan yang relevan. Bentuk komunikasi yang sesuai tergantung tujuan penelitian dan audiens. Gambaran metode *DSRM* dapat dilihat pada gambar 1.

Berdasarkan tahapan pada metode *DSRM* tersebut. Berikut ini merupakan tahapan yang dilakukan untuk mendapatkan hasil pada penelitian ini.

2.1 Identifikasi Permasalahan dan Motivasi

Pada tahap ini dilakukan dengan melakukan studi pustaka terhadap beberapa artikel ilmiah berbahasa Indonesia maupun berbahasa Inggris yang membahas mengenai arsitektur *microservice*.

2.2 Mendefinisikan Tujuan Solusi

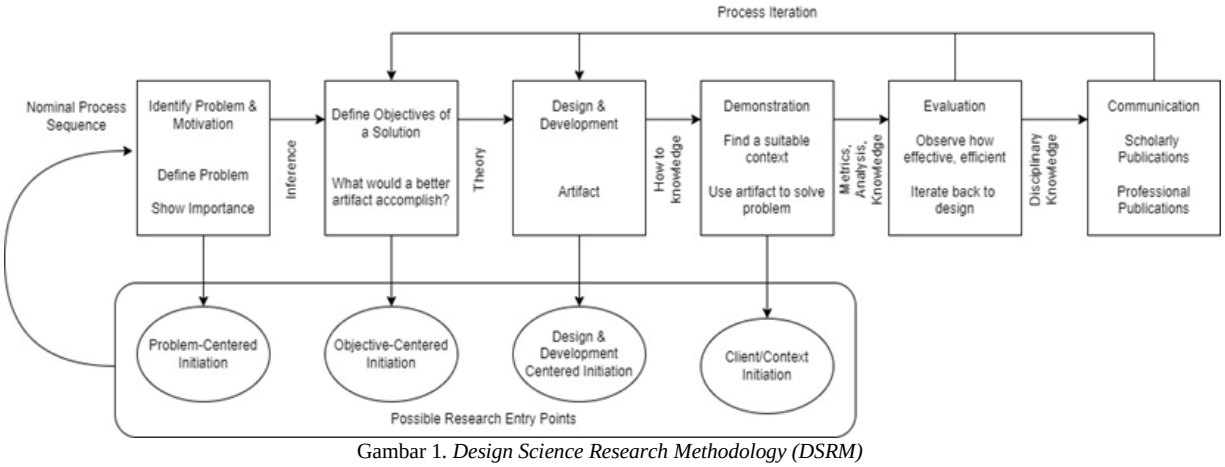
Pada tahap ini dilakukan dengan menentukan tujuan penelitian berdasarkan hasil studi pustaka yang telah dilakukan pada tahap identifikasi permasalahan dan motivasi.

2.3 Perancangan dan Pengembangan

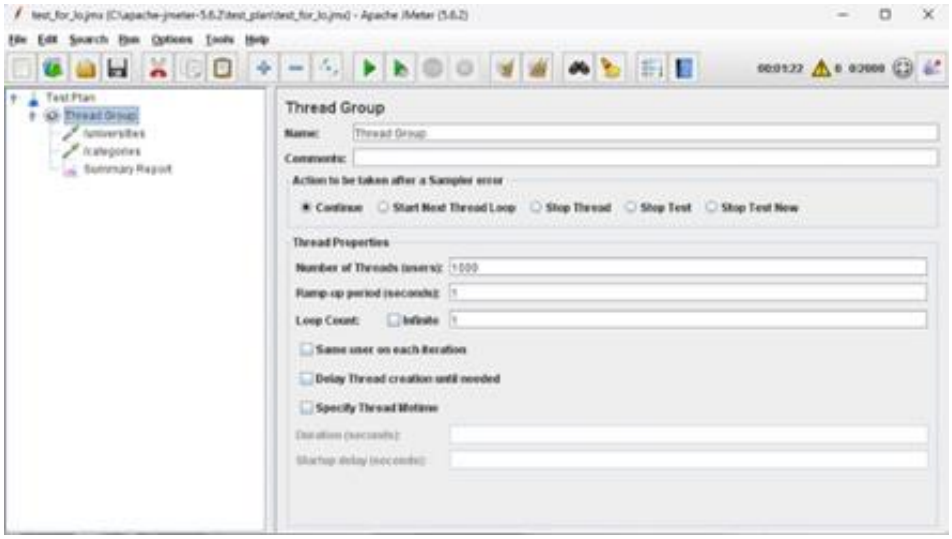
Pada tahap ini dilakukan dengan melakukan perancangan arsitektur *microservice* dengan melakukan observasi terhadap website lomba nasional kreativitas mahasiswa tahun 2023, studi pustaka terhadap buku panduan lomba nasional kreativitas mahaiswa tahun 2023, merumuskan kebutuhan arsitekur *microservice* berdasarkan hasil observasi website lomba nasional kreativitas mahasiswa dan studi pustaka buku panduan lomba nasional kreativitas mahasiswa dan merancang arsitektur *microservice* berdasarkan dengan kebutuhan arsitektur *microservice* yang telah dirumuskan.

2.4 Demonstrasi

Pada tahap ini dilakukan dengan mengembangkan arsitektur *microservice* yang telah dirancang pada tahap perancangan dan pengembangan menggunakan *framework Lumen Laravel*, *Kong API Gateway sebagai API Gateway*, *Mysql* sebagai basis data yang digunakan untuk menyimpan data, dan mengkonfigurasi *Docker Desktop* pada *service microservice*.



Gambar 1. *Design Science Research Methodology (DSRM)*



Gambar 2. Skenario Pengujian

Pengembangan dilakukan secara terbatas dengan hanya mengembangkan *API* yang digunakan untuk kebutuhan komunikasi antar *service* dan pengujian saja. Kemudian dilakukan uji coba *API Application Programming Interfaces)* dari arsitekur *microservice* yang telah dirancang dengan menggunakan aplikasi *Postman*.

2.5 Evaluasi

Pada tahap ini dilakukan dengan melakukan evaluasi performansi terhadap rancangan arsitekur *microservice* yang telah dikembangkan pada tahap demonstrasi. Evaluasi performansi dilakukan dengan melakukan pengujian performansi menggunakan metode *load testing* dengan bantuan aplikasi *Apache JMeter*. *Load testing* pada penelitian ini dimulai dengan menetukan *API* yang digunakan untuk *load testing*, merancang *test plan* pada *Apache JMeter*, setelah itu dilakukan *load testing* dengan mengamati *error rate* dan *throughput* yang dihasilkan terhadap *microservice* yang telah dikembangkan. Menurut (Tangela & Katari, 2022), *Error Rate* pada *Apache JMeter* merupakan jumlah *error* yang ditemukan ketika dilakukannya pengujian performansi dikarenakan *request* yang dikirimkan dan diterima mengalami kegagalan.sedangkan menurut (Alam & Fitriyana, 2022), *throughput* pada *Apache JMeter* merupakan jumlah *request* yang diproses per unit oleh sistem dan secara langsung merepresentasikan kapasitas perangkat lunak dalam mengangkat beban. Untuk mendapatkan *error rate* dan *throughput* yang dihasilkan pada penelitian ini maka, *load testing* pada penelitian ini dilakukan sebanyak 3 kali pengujian dengan memasukan *number of thread (virtual users)* sebagai berikut 1000, 1500 dan 2000 pada *Apache JMeter* dengan *ramp-up* yang dipakai adalah 1 dan *loop count* yang dipakai adalah 1. Berikut ini merupakan skenario pengujian pada peneltian ini.

Tabel 1. *API* Yang Digunakan Untuk *Load Testing*

API	*Method*	*Service*
/universities	*GET*	Universitas *Service*
/categories	*GET*	Pendaftaran *Service*

2.6 Komunikasi

Pada tahap ini dilakukan penulisan artikel ilmiah untuk mengkomunikasikan hasil temuan yang didapatkan melalui penelitian ini.

3. ARSITEKTUR *MICROSERVICE*

3.1 Kebutuhan Arsitektur

Website lomba nasional kreativitas mahasiswa merupakan website yang digunakan sebagai media untuk melakukan pendaftaran lomba nasional kreativitas mahasiswa. Agar arsitektur *microservice* yang dirancang sesuai dengan website lomba nasional kreativitas mahasiswa maka, arsitektur *microservices* yang dirancang harus mendukung fungsionalitas yang terdapat pada website lomba nasional kreativitas mahasiswa. Berikut ini kebutuhan arsitektur *microservice* website lomba nasional kreativitas mahasiswa.

Tabel 2. Kebutuhan Arsitekur *Microservices*

Kebutuhan	Deskripsi
User Service	*User Service* merupakan *service* yang bertanggung jawab mengelola data yang berhubungan dengan pengguna pada website lomba nasional kreativitas mahasiswa
Universitas *Service*	Universitas *Service* merupakan *service* yang bertanggung jawab mengelola data yang berhubungan dengan Universitas pada website lomba nasional kreativitas mahasiswa.
Pendaftaran *Service*	Pendaftaran *Service* merupakan *service* yang bertanggung jawab mengelola data yang berhubungan dengan pendaftaran peserta lomba pada website lomba nasional kreativitas mahasiswa.
Pembayaran *Service*	Pembayaran *Service* merupakan *service* yang bertanggung jawab mengelola data yang berhubungan dengan pembayaran peserta lomba pada website lomba nasional kreativitas mahasiswa.

Kebutuhan	Deskripsi
File Service	*File Service* merupakan *service* yang bertanggung jawab mengelola file yang berhubungan dengan lomba pada website lomba nasional kreativitas mahasiswa.

Sedangkan *API* yang digunakan dalam pengujian adalah sebagai berikut.

Umumnya antar muka atau *UI (User Interface)* pada arsitektur *microservice* akan dibuat menjadi sebuah aplikasi monolitik yang bertugas mengirim dan menerima *request* pengguna menuju *service* maupun sebaliknya menggunakan *API (Application Programming Interfaces)*. Sehingga berikut ini merupakan kebutuhan antar muka website lomba nasional kreativitas mahasiswa.

Tabel 3. Kebutuhan *Microservices UI*

Kebutuhan	Aktor
Pendaftaran Universitas	Peserta Lomba, Admin Universitas
Pendaftaran Peserta	Peserta Lomba
Pendaftaran Admin Universitas	Admin Universitas
Login	Peserta Lomba, Admin Universitas
Profil Pengguna	Peserta Lomba, Admin Universitas
Ubah Password	Peserta Lomba, Admin Universitas
Update Profil Tim	Peserta Lomba
Pembayaran	Peserta Lomba, Admin Universitas
Upload Berkas	Peserta Lomba
History Pembayaran	Peserta Lomba, Admin Universitas
Refund Pembayaran	Peserta Lomba

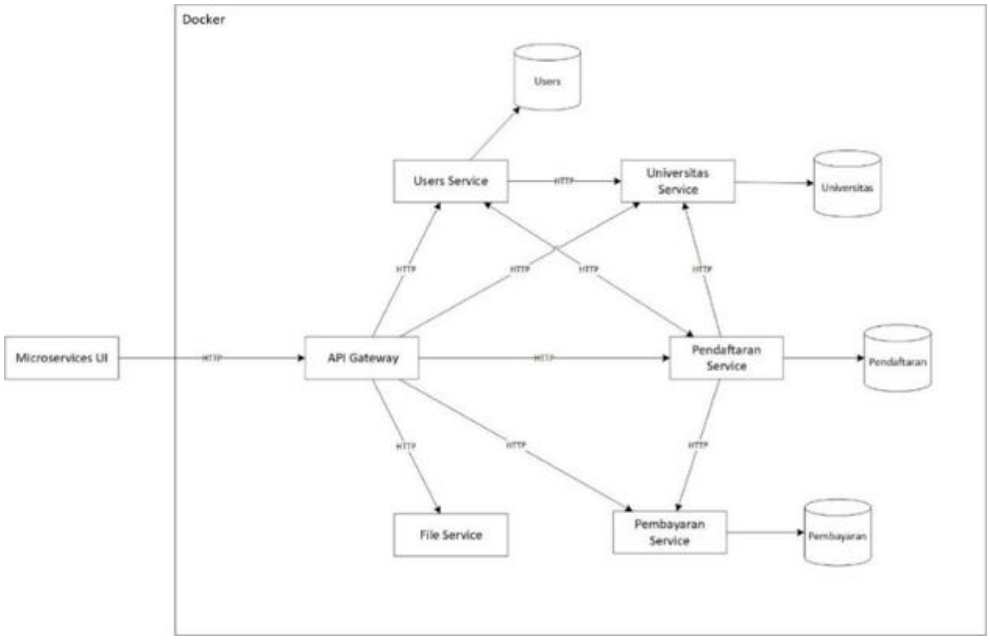
3.2 Rancangan Arsitektur

Berdasarkan kebutuhan arsitektur tersebut maka, rancangan arsitektur *microservice* website lomba nasional kreativitas mahasiswa adalah sebagai berikut.

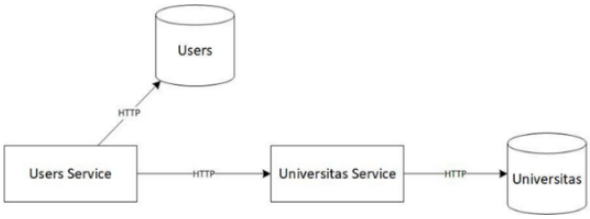
dan Rancangan arsitektur *microservice* tersebut menggunakan teknologi *Container* dan *Docker* sebagai *container management* yang digunakan untuk manampung semua *service* pada rancangan arsitektur *microservice* tersebut. Penggunaan teknologi *container* pada rancangan arsitektur *microservices* tersebut adalah untuk memberikan isolasi pada setiap *service* sehingga *service* dapat memiliki dan menggunakan teknologinya sendiri tanpa bergantung dengan *service* lainnya maupun teknologi yang terdapat pada sistem operasi *host*. Setiap *service* pada rancangan arsitektur *microservice* tersebut saling terhubung dan berkomunikasi satu sama lain secara langsung melalui *REST API*. Namun khusus untuk *File Service* tidak terdapat hubungan antar *service* yang terjadi hal ini dikarenakan *File Service* hanya digunakan sebagai tempat penyimpanan *file* yang dibutuhkan. Komunikasi antar *service* yang terjadi pada rancangan arsitektur tersebut adalah adalah sebagai berikut.

1. Komunikasi *Users* Universitas *Service*

Komunikasi pada Gambar 4. merupakan komunikasi antar *service* pada *Users Service* dan Universitas *Service*. Pada komunikasi *Users Service* dan Universitas *Service*, *method* yang digunakan adalah *POST* dan *GET*. Penggunaan *method POST* pada komunikasi ini adalah untuk membuat dan menyimpan data universitas dari admin pada basis data admin yang terdapat di Universitas *Service* pada saat pembuatan akun admin universitas yang digunakan untuk pembayaran kolektif untuk tim dari admin universitas. Sedangkan akun *user* admin yang digunakan untuk *login* akan disimpan pada basis data *Users Service*. Kemudian *method GET* pada komunikasi ini digunakan untuk mengambil data universitas dari admin yang terdapat pada basis data Universitas *Service*.

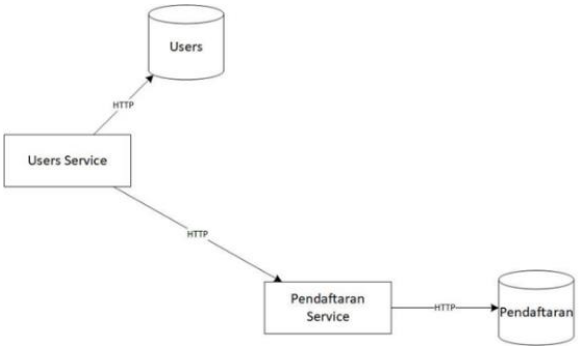


Gambar 3. Rancangan Arsitektur *Microservice*



Gambar 4. Komunikasi *Users* dan Universitas *Service*

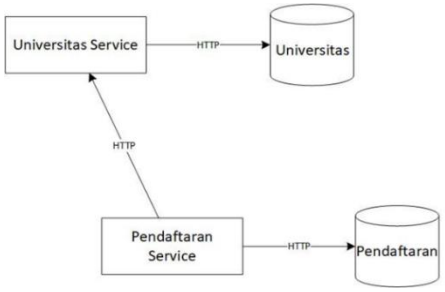
2. Komunikasi *Users* dan Pendaftaran *Service*



Gambar 5. Komunikasi *Users* dan Pendaftaran *Service*

Komunikasi antar *service* pada Gambar 5. merupakan komunikasi yang terjadi pada *Users Service* dan Pendaftaran *Service*. Pada komunikasi *Users Service* dan Universitas *Service*, *method* yang digunakan adalah *POST* dan *GET*. Penggunaan method *POST* pada komunikasi ini adalah untuk membuat dan menyimpan data peserta yang mengikuti lomba pada saat membuat akun peserta lomba sedangkan akun yang digunakan untuk *login* disimpan pada basis data pada *Users Service*. Kemudian penggunaan *method GET* pada komunikasi ini digunakan untuk mengambil data peserta yang berada pada basis data Pendaftaran *Service*.

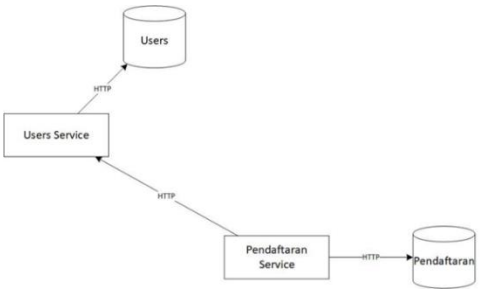
3. Komunikasi Pendaftaran dan Universitas *Service*



Gambar 6. Komunikasi Pendaftaran dan Universitas *Service*

Komunikasi antar *service* pada Gambar 6. merupakan komunikasi yang terjadi pada Pendaftaran *Service* dan Universitas *Service*. Pada komunikasi Pendaftaran *Service* dan Universitas *Service*, *method* yang digunakan adalah *GET*. Penggunaan *method GET* pada komunikasi ini adalah untuk mendapatkan data universitas dari peserta lomba yang berasal dari basis data Universitas *Service*.

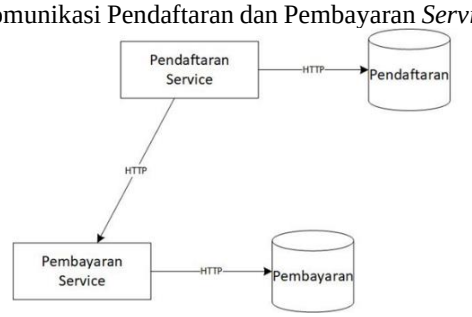
4. Komunikasi Pendaftaran dan *Users Service*



Gambar 7. Komunikasi Pendaftaran dan *Users Service*

Komunikasi antar *service* pada Gambar 7. merupakan komunikasi yang terjadi pada Pendaftaran *Service* dan *Users Service*. Pada komunikasi Pendaftaran *Service* dan *Users Service*, *method* yang digunakan adalah *GET*. Penggunaan method *GET* pada komunikasi ini adalah untuk mendapatkan data peserta lomba yang menjadi ketua tim yang berasal dari basis data *Users Service*.

5. Komunikasi Pendaftaran dan Pembayaran *Service*



Gambar 8. Komunikasi Pendaftaran dan Pembayaran *Service*

Komunikasi antar *service* pada Gambar 8. merupakan komunikasi yang terjadi pada Pendaftaran *Service* dan Pembayaran *Service*. Pada komunikasi Pendaftaran *Service* dan Pembayaran *Service*, *method* yang digunakan adalah *GET*. Penggunaan *method GET* pada komunikasi ini adalah untuk mendapatkan data pembayaran peserta lomba yang berasal dari basis data Pembayaran *Service*.

6. Komunikasi *Microservices UI* dan *Service*

Sama seperti *service*, *User Interface (UI)* atau antar muka pada rancangan arsitektur *microservices* tersebut juga tidak saling bergantung dengan salah satu *service* dan berkomunikasi dengan *service* pada *Docker* dengan menggunakan *REST API* yang didaftarkan pada *API Gateway*. Penggunaan *API Gateway* pada rancangan arstektur *microservices* bukan hanya untuk mendaftarkan *API* yang digunakan oleh *Microservices UI* namun juga digunakan untuk mengamankan semua *service* pada

rancangan arsitektur *microservices*. Berikut ini merupakan service yang dibutuhkan oleh *Microservices UI* yang perlu didaftarkan pada *API Gateway*.

Tabel 4. *Service* Yang Digunakan *Microservices UI*

Service	*Method*	Deskripsi
User Service	*POST*	Digunakan untuk melakukan *login*, registrasi peserta lomba dan registrasi admin universitas.
	GET	Digunakan untuk mendapatkan data pengguna yang aktif,
Universitas *Service*	*GET*	Digunakan untuk mendapatkan data universitas yang terdaftar.
	POST	Digunakan untuk registrasi universitas yang belum terdaftar.
Pendaftaran *Service*	*PUT*	Digunakan untuk melakukan pendaftaran lomba dengan cara memilih kategori lomba dan mengupdate data anggota tim.
Pembayaran Service	*POST*	Digunakan untuk melakukan pembayaran mengikuti lomba, *refund* pembayaran dan menampilkan *history* pembayaran.
	PUT	Digunakan untuk melakukan update status pembayaran peserta lomba yang berasal dari *payment gateway*.
File Service	*POST*	Digunakan untuk *upload* berkas dan karya peserta, juga untuk mendownload berkas yang dibutuhkan untuk proses pendaftaran.

4. HASIL DAN PEMBAHASAN

4.1 Hasil Evaluasi Performansi

Error rate dan *Throughput* yang dihasilkan dari penguijan dengan menggunakan metode *load testing* pada *Apache JMeter* yang dilakukan sebanyak 3 kali pengujian dengan memasukan *number of thread (virtual users)* sebagai berikut 1000, 1500 dan 2000 dengan *ramp-up* yang dipakai adalah 1 dan *loop count* yang dipakai adalah 1 adalah sebagai berikut.

Tabel 5. Hasil *Load Testing*

Load	*Error Rate %*	*Throughput / Sec*
1000	1.10%	41.7/*Sec*
1500	7.00%	46.9/*Sec*
2000	17.47%	49.0/*Sec*

Sedangkan untuk penggunaan sumber daya yang *Docker Desktop* yang dihasilkan pada saat pengujian dilakukan dengan *number of thread (virtual users)* sebagai berikut 1000, 1500 dan 2000 pada penelitian ini adalah sebagai berikut.

Tabel 6. Penggunaan Sumber Daya Pada *Docker Desktop*

Container CPU Usage	*Container Memory Usage*
390.75% / 400% (*4 core allocated*)	1.02*GB* / 3.61*GB*

4.2 Pembahasan Hasil

1. *Error Rate*

Berdasarkan hasil pengujian *load testing* yang dihasilkan *Apache JMeter* maka, rata-rata *error rate* yang dihasilkan adalah sebagai berikut.

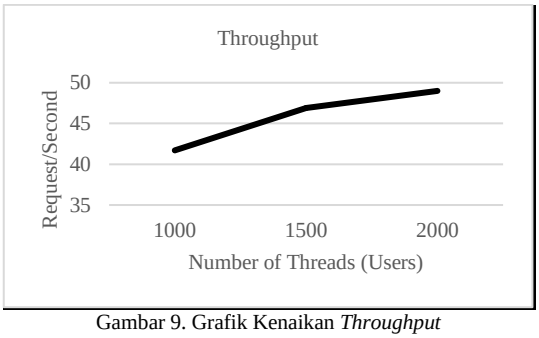
Tabel 7. Rata-Rata *Error Rate*

Load	*Error Rate %*
1000	1.10%
1500	7.00%
2000	17.47 %
Rata-Rata	8.52%

Seperti yang terlihat pada Tabel 7. rata-rata *error rate* dari hasil *load testing* berada di angka 8.52% untuk aplikasi yang menggunakan arsitektur *microservice* yang mana pada penelitian yang dilakukan oleh (Hadi et al., 2022) rata-rata *error rate* yang dihasilkan website lomba nasional kreativitas mahasiswa pada tahun 2022 berada dikisaran angka 25% atau 28% saat dilakukan pengujian dengan jumlah beban sama dengan atau lebih besar dari 1000. Berdasarkan rata-rata *error rate* yang didapatkan *error* yang terjadi pada saat pengujian performansi berada dibawah rata-rata *error* yang dihasilkan website lomba nasional kreativitas mahasiswa pada tahun 2022 sehingga dapat dikatakan bahwa secara performansi arsitektur *microservice* lebih baik dari arsitekur monolitik yang diwakilkan melalui rata-rata *error rate* yang didapatkan melalui penelitian yang dilakukan oleh (Hadi et al., 2022), Namun rancangan arsitektur *microservice* yang dirancang pada penelitian ini tidak bisa mendapatkan rata-rata *error rate* agar dapat dikatakan memiliki performansi yang baik pada *load testing* yaitu 0%. Hal ini dapat terjadi dikarenakan perangkat keras yang digunakan untuk melakukan pengujian pada penelitian ini adalah Intel core i3-1115G4 yang merupakan prosesor berjenis *dual core* dengan 4 *thread* tidak mampu memproses banyaknya *request* pengguna yang dihasilkan oleh *Apache JMeter*, dimana pada saat *request* aplikasi terjadi, aplikasi akan menggunakan sumber daya yang tersedia pada *CPU*, sehingga pada saat pengujian dengan jumlah beban sama dengan atau lebih besar dari 1000 dilakukan maka, *CPU* akan mematikan salah satu proses *request* yang berjalan pada aplikasi agar penggunaan *CPU* tidak mencapai 400% sesuai dengan data pada Tabel 6. yang menyebabkan aplikasi menghasilkan *error* yang kemudian dikalkulasikan oleh *Apache JMeter* dalam bentuk persentase.

2. *Throughput*

Berdasarkan hasil pengujian *load testing* yang dihasilkan, terlihat bahwa *throughput* yang dihasilkan mengalami kenaikan saat dilakukan pengujian dengan setiap beban pada penelitian ini. Untuk lebih jelas kenaikan *throughput* pada penelitian ini dapat dilihat pada Gambar 9. berikut ini.



Gambar 9. Grafik Kenaikan *Throughput*

Kenaikan *throughput* yang awalnya berada pada angka 41.7/*Sec* saat dilakukan pengujian dengan beban 1000 terus meningkat menjadi 46.9/*Sec* setelah pengujian dengan beban 1500 dan kembali meningkat menjadi 49.0/*Sec* saat dilakukan pengujian dengan beban 2000. Kenaikan *throughput* yang terjadi merupakan hal yang bagus dikarenakan menandakan bahwa aplikasi dapat menyesuaikan jumlah beban yang diberikan. Hal ini dapat terjadi dikarenakan secara teori *throughput* diukur menggunakan *Request Per Second (RPS)* yang mana ketika beban yang diberikan bertambah, maka akan menyebabkan *throughput* ikut bertambah.

5. KESIMPULAN

Berdasarkan pembahasan hasil evaluasi maka, dapat disimpulkan bahwa arsitektur *microservice* yang dirancang dapat meningkatkan performansi website lomba nasional kreativitas mahasiswa dimana rata-rata *error rate* yang dihasilkan berada pada angka 8.52%. Sedangkan *throughput* yang dihasilkan pada penelitian ini terus mengalami peningkatan tanpa adanya penurunan saat diberikan beban yang berbeda. Hal ini dapat terjadi dikarenakan setiap fungsi pada arsitektur *microservice* termasuk antar mukanya diproses secara terpisah bukan diproses sebagai satu kesatuan pada sistem operasi seperti arsitektur monolitik sehingga hal ini yang membuatnya dapat meningkatkan peformansi pada website lomba nasional kreativitas mahasiswa.

Namun arsitektur *microservice* yang dirancang pada penelitian ini belum bisa dikatakan memiliki performansi yang baik dikarenakan *error rate* yang dihasilkan arsitektur *microservice* yang dirancang pada penelitian ini belum bisa mencapai angka agar dapat dikatakan memiliki performansi yang baik yaitu 0%. Hal ini dapat terjadi dikarenakan perangkat keras yang digunakan untuk melakukan pengujian, sehingga dapat disimpulkan juga bahwa sama seperti arsitektur monolitik, perangkat keras yang digunakan pada arsitektur *microservices* juga berpengaruh terhadap performansi yang dihasilkan.

DAFTAR PUSTAKA

ALAM, E.N. dan FITRIYANA, D., 2022. Performance Testing Analysis of Bandungtanginas Application with Jmeter. *International Journal of Innovation in Enterprise System*, 6(02), pp.146-155.

AL-DEBAGY, O. dan MARTINEK, P., 2018. A Comparative Review of Microservices and Monolithic Architectures. *18th IEEE International Symposium on Computational Intelligence and Informatics, CINTI 2018 - Proceedings*. pp.149-153.

ASRI, S.A., ASTAWA, I.N.G.A., SUNAYA, I.G.A.M., NUGROHO, I.M.R.A. DAN SETIAWAN, W., 2022. Implementation of Asynchronous Microservices Architecture on Smart Village Application. *International Journal on Advanced Science, Engineering and Information Technology*, 12(3), pp.1236-1243.

ASROWARDI, I., PUTRA, S.D. dan SUBYANTORO, E., 2019. Designing microservice architectures for scalability and reliability in e-commerce. *Journal of Physics: Conference Series*. pp.1-7.

BLINOWSKI, G., OJDOWSKA, A. dan PRZYBYLEK, A., 2022. Monolithic vs. Microservice Architecture: A Performance and Scalability Evaluation. *IEEE Access*. pp. 20357-20374.

HADI, H.N., ADITYA, A., PURWIANTONO, F.E. dan LISTIO, S.W.P., 2022. Pengujian Performa Pada Website Lomba Nasional Kreativitas Mahasiswa. *Jurnal Informatika*, 22(1), pp.100-110.

INSANITTAQWA, V.F., 2017. *Prediksi Reliabilitas Perangkat Lunak Menggunakan Support Vector Regression dan Model Mining*. Institut Teknologi Sepuluh Nopember.

OBOKO, R., 2016. *Development of a Scalable Microservice Architecture for Web Services using OS-level Virtualization*. University of Nairobi.

TANGELA, A. dan KATARI, P., 2022. *Testing Lifestyle Store Website Using JMeter in AWS and GCP*.

TANUWIJAYA, A., PALIT, H.N. dan NOERTJAHYANA, A., 2021. Penerapan Microservices dan Amazon Elastic Container Service untuk Mendukung Scalability. *Jurnal Infra*, 9(2).

VOM BROCKE, J., HEVNER, A. dan MAEDCHE, A., 2020. Introduction to Design Science Research. pp.1–13.