Jurnal Ilmiah KURSOR
Menuju Solusi Teknologi Informasi
Vol. 11, No. 4, Desember 2022

ISSN 0216 – 0544
e-ISSN 2301– 6914

DESIGN AND DEVELOPMENT OF BACKEND APPLICATION FOR THESIS MANAGEMENT SYSTEM USING MICROSERVICE ARCHITECTURE AND RESTFUL API

[a]Ach. Khozaimi, [b]Yoga Dwitya Pramudita, [c]Firdaus Solihin, [d]Ahmad Khairi Ramadan

[a, b, c, d] Teknik Informatika, Trunojoyo University of Madura 69162
E-mail: khozaimi@trunojoyo.ac.id, yoga@trunojoyo.ac.id, fsolihin@trunojoyo.ac.id

Abstract

A thesis is a scientific work completed by students with the aim of developing the knowledge gained during the lecture period. Students at Universitas Trunojoyo Madura (UTM), Faculty of Engineering, particularly Informatics Engineering, carry out their theses manually and on paper. Thesis Management System (TMS) is software designed to help with the thesis execution process by reducing paper usage and increasing time efficiency. Monolithic system development can disrupt the service process if improvements are being made to the system. Therefore, in this research, a Thesis Management System (TMS) will be built using a microservice approach to make it easier to maintain and develop the system, for example, system scalability. As a means of communication between services, TMS is designed and developed using the REST API. TMS has undergone system performance testing to verify that it performs well under certain conditions. The results show that the number of requests increases the performance response time, CPU usage, and memory consumption, with an average resource usage of each service based on a response time of 61.64 ms, CPU usage of 8.64%, and memory usage of 89.47 Mb. As the number of requests on the service increases, so does resource usage in each service, but this has no effect on device performance because the increase is so low.

Key words: Thesis management, RESTful API, Microservice, Monolithic, TMS-UTM.

INTRODUCTION

Thesis is a scientific work that contains a result explanation of research problem in a particular field of science. The thesis must also be scientifically accountable and carried out based on predetermine rules or procedures[1]. The purpose of thesis is that students are able to write scientific papers in accordance with their fields and are assisted by supervisors who are suitable for that field [2]. All this time, the Faculty of Engineering students, especially the Informatics Engineering Study Program, carry out the process of making thesis manually or paper-based, both in implementation and in managing thesis data/files. The process of executing the thesis manually is very time-consuming, laborious, and requiring a lot of paper, both for the thesis purposes, proposal printing and the thesis itself. In addition, during the thesis process, students need to meet with some of parties such as coordinators, supervisors and examiners to be able to complete the administration of the thesis file.

Several previous researchers have succeeded building a system, the aim is to minimize problems that occur in the implementation of the thesis. In 2010 Akh. Khozaimi, Firdaus Solihin and Achmad Jauhari have successfully developed a Final Project Information System or Sistem Informasi Tugas

Akhir (SIMTAK) [1]. The focus of research is system development that can manage, monitor and record the thesis process also provide information related to the thesis. This system is redeveloped in 2019 by Ach. Khozaimi, Sigit Susanto Putro and Mujibur Rohman [3]. In this study, a computational method in the form of cosine similarity was applied to reduce the degree of similarity in the title and the method used by students in the implementation of the thesis. The system that has been built in previous research uses a monolithic architecture, where every service in the system has a high linkage so that when a change occurs in a service it can affect other services. Therefore, it is necessary to build a system that implements a microservice architecture where each service will work independently, so that it can facilitate further development in system improvement. It is enough to fix the system for each problematic service without interfering of another services performance [4].

Monolithic architecture development is a way of developing a system where all the components in the system are packaged together [4]. The development of a thesis management system with a monolithic architecture has several weaknesses, such as the difficulty of repairing because every component or service in a monolithic system has a very high linkage. Another problem arises when making technological changes that are quite difficult to do because the developer has to change the entire system. Unlike the microservice architecture, in a microservice architecture, each service in a system will be independence. So that it can make it easier to repairs to a service, it will not affect other services [4]. The adoption of a microservice architecture is also very advantageous in terms of scalability, because it can only scale the required service components without the need to scale the entire system as in a monolithic architecture[5]. In implementing a microservice architecture, where each service is separate, an intermediary web service is needed [6].

Web services are divided into SOAP snd REST[7]. SOAP is an XML-based mark-up language used to exchange object data on a network. Sending data using SOAP is not optimal for large-scale data [6]. In addition, the use of XML-based also has a larger format than JSON in REST communication applications. REST is a very common architectural style used today [8][9] where for the communication process using JSON. APIs that follow the REST style are called RESTful APIs. REST is also a practical approach in web application development [9]. REST is very suitable to be used to communicate between systems that work independently such as microservices. REST is simpler and lighter than SOAP [10][11].

Thesis Management System is a system that was built with the aim of simplifying the thesis implementation process and changing the thesis implementation process from Base on Paper to Paperless so that paper usage can be minimized. The development of the thesis management system will be carried out by implementing a microservice architecture with RESTful API communication so that further improvements can be carried out more easily. The results of the development of this thesis management system will be tested functionally to ensure the function of each service runs well and performance testing to ensure the system can work well under certain conditions. [12].

MATERIAL AND METHODS

System Overview

Thesis Management System (TMS) is a system built to simplify the thesis execution process and change the thesis implementation process from Base on Paper to Paperless, so that paper usage can be minimized and at the same time can save time. The development of the thesis management system is carried out by implementing a microservice architecture with RESTful API communication, that makes further improvements and developments can be carried out more easily[13]. In the board sense, the description of the system architecture that will be applied in this study is based on the microservices architecture, which is shown in the following figure.

This system is divided into 4 small services, namely user services, discussion services, scheduling services and thesis services. The distribution of services in this system is based on the required features and to facilitate the scaling of system components. The user service will be in charge of managing user data including lecturer and student data, the discussion service will handle the discussion process that is accessed when the guidance process and trial discussions occur, the scheduling service will solve problems related

to scheduling from registration to the schedule has been arranged, and the last service is thesis service, this service is in charge of overcoming the thesis process, ranging from topic delivery, lecturer guidance to thesis assessment. Each service will manage its own data with a different database. The thesis service will be connected to the thesis database.

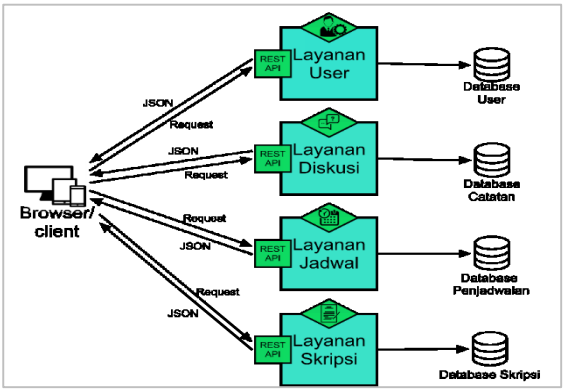

Fig 1. System architecture.

Microservice

Microservice is a new approach in implementing service-oriented systems [14]. Microservices divide a complex system into a collection of small independent services [9]. By microservice architecture approach, it is possible to develop each service separately, this can speed up system development. In the application of microservice architecture also allows the use of different technologies in each microservice [10]. In this software architecture development, the emphasis is on separating system services into several services according to the scope of needs. Apart from microservices, there are also other services such as monolithic. Monolithic architecture is a way of developing software where all the components of the business reside in a system that is packaged together [3].

The characteristics and advantages of microservice architecture is that, it can use various technologies and can easily change the technology used, resilience, scalability, and deployment of targeted independent organizations [10]. For applications that require high concurrency and capacity, a microservice architecture is a good choice. Microservices architecture provides an effective solution in solving problems such as large, complex, and time-consuming team projects in software update iterations, and difficult maintenance on large application integrations and releases [3].

The communication pattern in the microservices architecture is divided into 2 parts. Application communication and backend services are as shown in Figure 2. Backend services here have their respective functions and duties. Apps are consumers calling backend services such as mobile apps and web service clients.

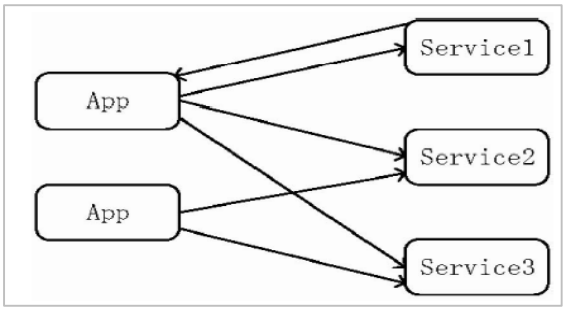

Fig 2. Application and service communication backend.

Connecting directly to the backend service is a very flexible way. However, this allows the application to call various backend services for requests which can cause delays for remote calls. Backend services are usually based on a microservices architecture. Service sharing is affected by business development. This causes all types of applications need to be improved to be able to adapt to service changes. Another communication pattern in the microservice architecture is communication using an API gateway which can be seen in Figure 3.

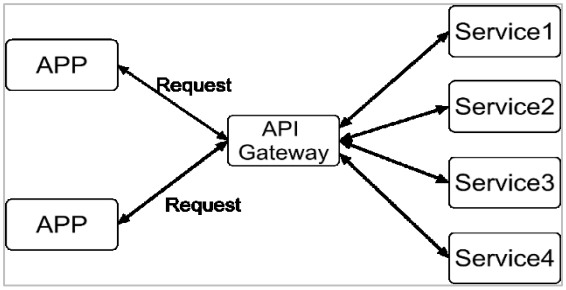

Fig 3. API gateway communication.

In this communication pattern, the application and each service need to access the API gateway to communicate. Each backend service will be registered to the API gateway. The API Gateway will accept requests from applications and direct them to the service in question [15]. Every request must go through

an API gateway, so that every application cannot communicate directly with every service.

RESTful API

API (Application Program Interface) is an interface programming application that provides a way for applications to relate and integrate from one application to another. [16]. The API here has a role as an intermediary between different applications. REST (Representational State Transfer) is a software architecture model that is very commonly used for distributed systems that focuses on scalability and interaction between components in the system and generalization of interfaces [8]. REST is a practical approach to developing web applications where the developer's system needs to be upgraded or required a simpler way to communicate with independent systems. REST is stateless and data oriented. All requests are independent and the server does not store any request status [9].

Application Program Interface (API) that follows the REST style is called a RESTful API which uses a Uniform Resource Identifier (URI) to represent data [9]. RESTful API is also a distributed system that focuses on scalability and interactions between components documented by the API method, it makes easier for developers to learn about the system that is running and can be used as a guide to use the features provided [17]. RESTful APIs use URIs (Uniform Resource Identifiers) which are used to identify resources. The methods used in REST include [18] [9]:

GET to get resources

POST to create a new source

PUT to update resource based on resource id

DELETE is used to delete a resource or set of resources.

RESULT AND DISCUSSION

The Thesis Management System test scenario is a performance test of the system that has been built. Performance testing is carried out on the system to ensure the system can be used under the expected workload. Performance testing on the client side is to get the response time of the request. Meanwhile, from the server side, it is used to determine the amount of CPU and RAM usage when executing a request. The parameters to be tested in the performance test can be seen in the following table.

Table 1. Performance test parameters.

No	Parameter	Unit	Description
1	Response Time	Milisecon (ms)	Client time to wait for response to requests made
2	CPU Usage	Percent(%)	CPU usage while performing a process
3	RAM Usage	Bytes	Amount of RAM (memory) usage when executing requests

The performance test of the Thesis Management System is carried out with 2 tools, namely apache jmeter [19] and performance monitoring and testing is carried out in the test environment listed in the following table.

Table 2. Testing scope

No	Device	Type
1	Operating System	Windows 10
2	CPU	Intel® Core¬TM i7-9700 CPU @ 3.20 GHz (12 CPUs), ~3.2 GHz
3	Memory	8192 MB

Table 3. API

No	Server	Endpoint
1	Thesis	Assesment
2	Thesis	Thesis
3	Scheduling	Manage_schedule
4	Scheduling	Validation

Performance testing is performed on 2 services with 5 request cases ranging from 200, 400, 600, 800 to 1000 requests. From each service, 2 APIs are used as test materials and each API is tested 5 times for each request case. The API used in this test plan can be seen in the following table.

Thesis Service Test Results

In the Thesis service, there are 2 APIs tested, namely the thesis API and the assessment API. The following is a graph of the results of testing the response time, CPU usage and memory usage on the API thesis and the API assessment on the thesis service.

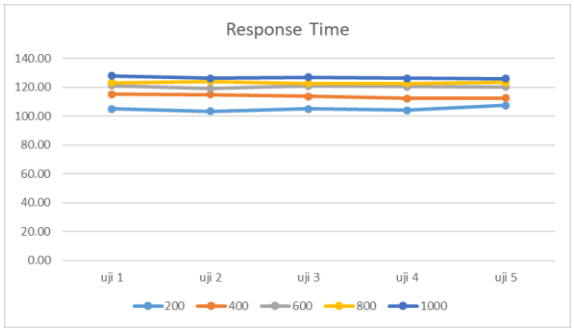

Fig 4. Graphics of thesis API response time

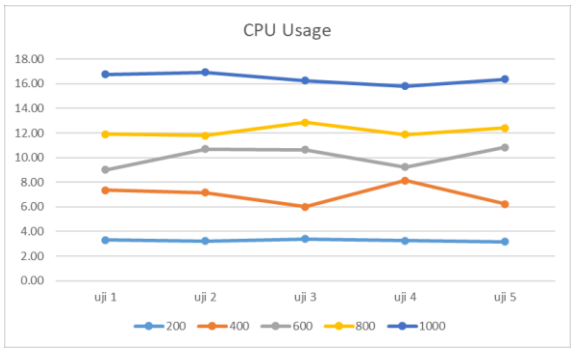

Fig 5. Graphics thesis API CPU usage

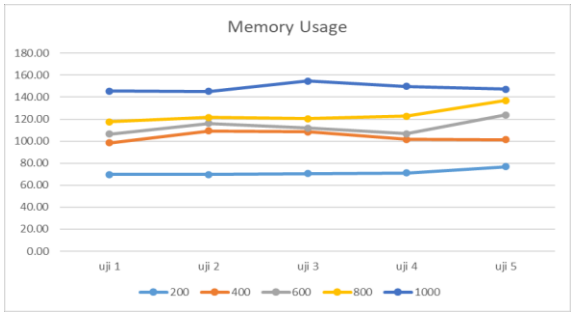

Fig 6. Graphics of thesis API memory usage

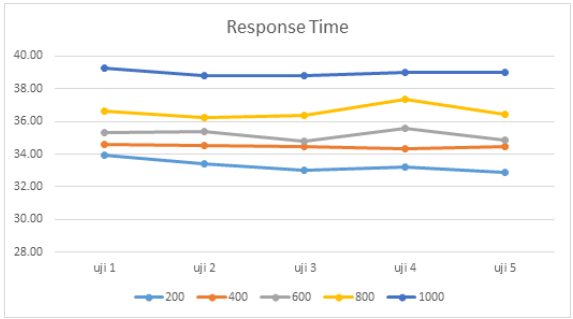

Fig 7. Graphics of assessment API response time

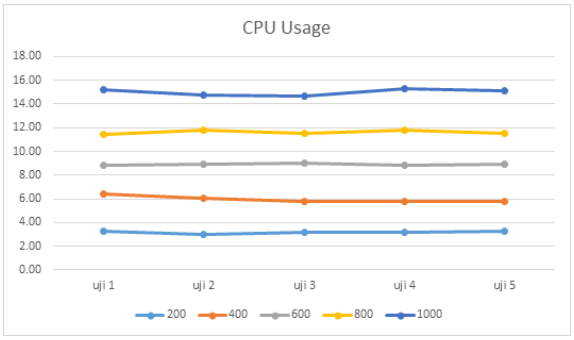

Fig 8. Graphics of assessment API CPU usage

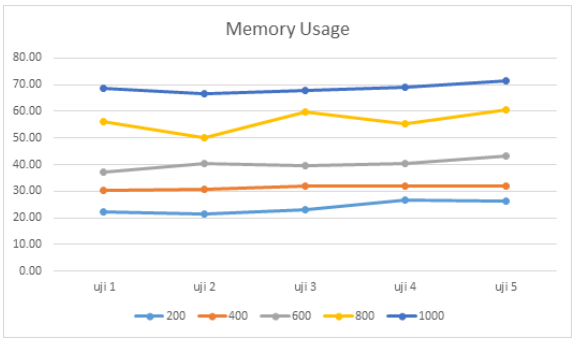

Fig 9. Graphics of assessment API memory usage

Scheduling Service Test Results

In the Scheduling service, there are 2 APIs tested, namely the API manage schedule and the API validation. The following is a graph of the results of testing response time, CPU usage and memory usage on the API manage schedule and API validation on the scheduling service.

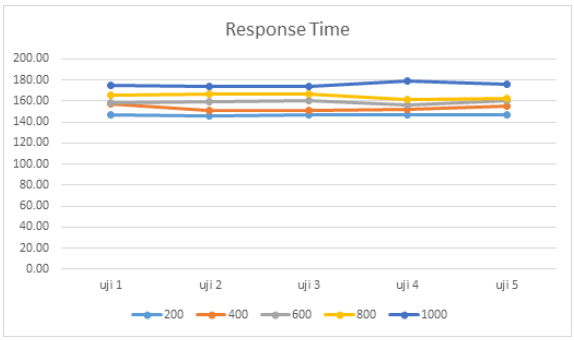

Fig 10. Graphics of manage schedule API response time

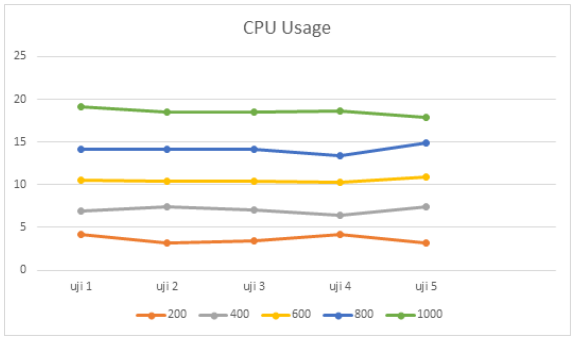

Fig 11. Graphics of manage schedule API CPU usage

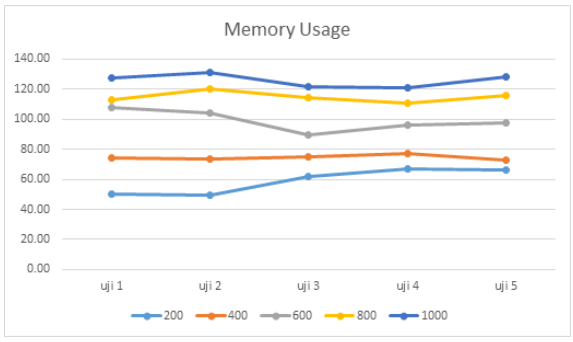

Fig 12. Graphics of manage schedule API memory usage

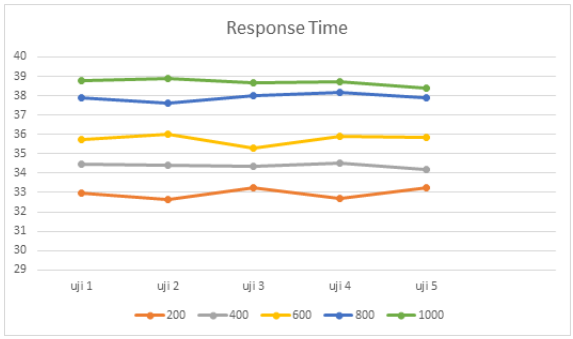

Fig 13. Graphics of validation API response time

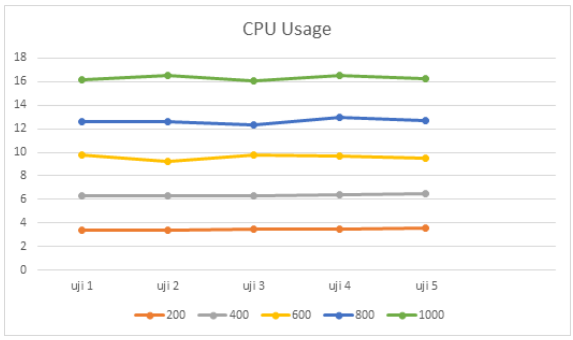

Fig 14. Graphics of validation API CPU usage graph

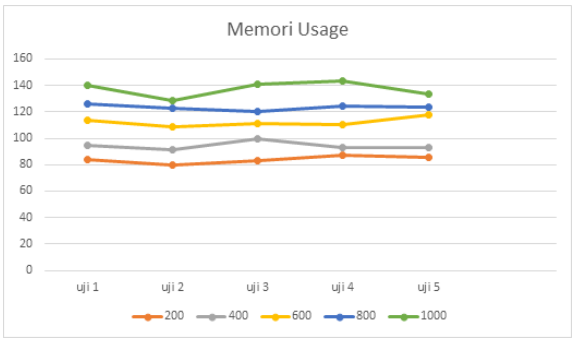

Fig 15. Graphics of validation API memory usage

Based on the results of the tests that have been carried out on the thesis service, the average data for each test indicator is obtained. For thesis API testing on thesis services, the average response time is 117.83 ms with an average response time increase of 4.83%. The average CPU usage is 9.78% with an average CPU usage increase of 53.43% and an average memory usage of 112.14 Mb with an average memory usage increase of 20.83%. As for testing scheduling services, the average of each indicator for each API tested is obtained. In testing the Manage schedule API, the average response time is 159.78 ms with an average increase in response time of 4.62%. The average CPU usage is 10.77% with an average CPU usage increase of 52.38% and an average memory usage of 94.70 Mb with an average memory usage increase of 21.14%. While the API Validation the average response time is 35.94 ms, while the average increase in response time is 4.10%. The average CPU usage in API validation is 9.67% with an average increase in CPU usage of 49.02% and an average memory usage of 110.22 Mb with an average increase in memory usage of 13.07%.

CONCLUSION

Based on the overall test results, it is known that the number of requests causes an increase in performance both in response time, CPU usage and memory usage with an average resource usage of each service based on response time of 61.64 ms, CPU usage of 8.64% and memory usage of 89 ,47 Mb. However, the increase in resource usage on client and server devices that occurs does not affect device performance because the increase is not too high.

REFERENCES

[1] A. Khozaimi, Ach.; Solihin, Firduas; Jauhari, "PERANCANGAN DAN PEMBUATAN SISTEM INFORMASI TUGAS AKHIR (SIMTAK)," J. Simantec, vol. 1, no. 3, pp. 203–211, 2010, [Online]. Available: https://simantectrunojoyo.files.wordpress.com/2014/04/6-khozaimi-firdaus-jauhari-perancangan-dan-pembuatan-sistem-informasi-tugas-akhir.pdf.

[2] Y. R. Pradana, A. A. Supianto, and Y. T. Mursityo, "Prediksi Bidang Penelitian dan Rekomendasi Dosen Pembimbing Skripsi Berdasarkan Konten Latar Belakang pada Naskah Proposal Menggunakan Metode Multi-Class Support Vector Machine dan Weighted Product," J. Teknol. Inf. dan Ilmu Komput., vol. 8, no. 2, p. 403, 2021, doi: 10.25126/jtiik.2021824511.

[3] A. Khozaimi, S. S. Putro, and M. Rohman, "Pengembangan Aplikasi Managemen Tugas Skripsi (Studi Kasus : Program Studi Teknik Informatika Universitas Trunojoyo Madura)," MATRIK J. Manajemen, Tek. Inform. dan Rekayasa Komput., vol. 18, no. 2, pp. 237–245, 2019, doi: 10.30812/matrik.v18i2.392.

[4] Z. Ren et al., "Migrating web applications from monolithic structure to microservices architecture," ACM Int. Conf. Proceeding Ser., 2018, doi: 10.1145/3275219.3275230.

[5] Y. Chandra, T. Putra, T. Adi, P. Sidi, and J. E. Samodra, "Implementasi Arsitektur Microservice pada Aplikasi Web Pengajaran Agama Islam Home Pesantren," J. Inform. Atma Jogja, vol. 1, no. November, pp. 88–97, 2020.

[6] F. Arifien and M. Riastuti, "Model Interoperabilitas Web Service Feeder PDDIKTI Menggunakan Enterprise Javabeans (EJB) dan REST-API," vol. 3, 2019.

[7] D. Rathod, "Performance Evaluation of Restful Web Services and Soap / Wsdl Web Services," Int. J. Adv. Res. Comput. Sci., vol. 8, no. 7, pp. 415–420, 2017, doi: 10.26483/ijarcs.v8i7.4349.

[8] A. Belkhir, M. Abdellatif, R. Tighilt, N. Moha, Y. G. Gueheneuc, and E. Beaudry, "An observational study on the state of REST API uses in android mobile applications," in Proceedings - 2019 IEEE/ACM 6th International Conference on Mobile Software Engineering and Systems, MOBILESoft 2019, 2019, pp. 66–75, doi: 10.1109/MOBILESoft.2019.00020.

[9] B. M. Adam, A. Rachmat Anom Besari, and M. M. Bachtiar, "Backend Server System Design Based on REST API for Cashless Payment System on Retail Community," IES 2019 - Int. Electron. Symp. Role Techno-Intelligence Creat. an Open Energy Syst. Towar. Energy Democr. Proc., pp. 208–213, 2019, doi: 10.1109/ELECSYM.2019.8901668.

[10] F. Halili and E. Ramadani, "Web Services: A Comparison of Soap and Rest Services," Mod. Appl. Sci., vol. 12, no. 3, p. 175, 2018, doi: 10.5539/mas.v12n3p175.

[11] R. Choirudin and A. Adil, "Implementasi Rest Api Web Service dalam Membangun Aplikasi Multiplatform untuk Usaha Jasa," MATRIK J. Manajemen, Tek. Inform. dan Rekayasa Komput., vol. 18, no. 2, pp. 284–293, 2019, doi: 10.30812/matrik.v18i2.407.

[12] D. I. Permatasari, "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, doi: 10.26418/justin.v8i1.34452.

[13] A. A. Mulyawan, "Sistem Pengelolaan Target Perusahaan dengan Microservices Architecture untuk Membantu Peningkatan Kinerja Perusahaan," JATISI (Jurnal Tek. Inform. dan Sist. Informasi), vol. 9, no. 1, pp. 12–22, 2022, doi: 10.35957/jatisi.v9i1.1423.

[14] M. A. F. N. Liqoo, "Analisis Pada Arsitektur Microservice Untuk Layanan Bisnis Toko Online," TEKINFO, vol. 22,

no. 2, pp. 61–68, 2021, [Online]. Available: https://journals.upi-yai.ac.id/index.php/TEKINFO/article/download/1761/1463.

[15] M. Song, C. Zhang, and E. Haihong, "An Auto Scaling System for API Gateway Based on Kubernetes," 2018 IEEE 9th Int. Conf. Softw. Eng. Serv. Sci., pp. 109–112, 2018.

[16] R. S. Saputra, I. R. Munadi, and D. D. Sanjoyo, "Implementasi Dan Analisis Performansi Platform As a Service Untuk Api Gateway Menggunakan Kong," in e-Proceeding of Engineering, 2018, vol. 5, no. 3, pp. 4973–4979, [Online]. Available: https://libraryeproceeding.telkomuniversity.ac.id/index.php/engineering/article/viewFile/7883/7776.

[17] O. Sahin and B. Akay, "A Discrete Dynamic Artificial Bee Colony with Hyper-Scout for RESTful web service API test suite generation," Appl. Soft Comput. J., vol. 104, p. 107246, 2021, doi: 10.1016/j.asoc.2021.107246.

[18] R. Gunawan and A. Rahmatulloh, "JSON Web Token (JWT) untuk Authentication pada Interoperabilitas Arsitektur berbasis RESTful Web Service," J. Edukasi dan Penelit. Inform., vol. 5, no. 1, p. 74, 2019, doi: 10.26418/jp.v5i1.27232.

[19] E. H. Halili, Apache JMeter. 2008.

.