p-ISSN: 2252-3006
e-ISSN: 2685-2411

Development Of Service-Oriented Architecture-Based Microservices Management As A Data Integration Service (Case Study: Udayana University)

I Gede Nyoman Agung Jayarana[a1], Kadek Yota Ernanda Aryanto[a2], I Made Gede Sunarya[a3]
[a]Computer Science Study Program, Postgraduate Program, Universitas Pendidikan Ganesha
email: [1]mankde57@gmail.com, [2]yota.ernanda@undiksha.ac.id, [3]sunarya@undiksha.ac.id

Abstrak

Untuk mengakomodasi segala bentuk kegiatan akademik maupun non akademik Universitas Udayana mengembangkan berbagai sistem informasi. Dalam jangka panjang kualitas data yang dihasilkan semakin tidak akurat karena proses pengambilan data dari sistem lain masih menggunakan metode tradisional dengan skema join antar database. Terdapat juga beberapa sistem yang sudah membuat webservice sederhana yang tidak mengikuti aturan penulisan Restful API. Berdasarkan permasalahan tersebut maka dilakukan perubahan proses integrasi data dengan Microservices Management menggunakan metode Service Oriented Architecture untuk proses integrasi dan OAuth 2.0 sebagai protokol keamanan. Hasil yang didapat bahwa sistem microservices management sangat efektif dan efisien untuk meningkatkan proses kinerja, memiliki tingkat resiko yang rendah, memiliki validasi uji kontrol yang baik sebesar 93,33% dan usability dengan grade B dimana sistem dinyatakan baik dan layak digunakan.

.
***Kata kunci**: Service Oriented Architecture, Microservices, Sistem Terintegrasi, OAuth 2.0*

Abstract

Udayana University has developed numerous information systems to accommodate all types of academic and non-academic activities. Because the process of retrieving data from other systems still employs the traditional method with a joint scheme between databases, the resulting data quality becomes increasingly inaccurate over time. Some systems have also established simple web services that do not adhere to the Restful API authoring rules. Based on these issues, modifications were made to the data integration process using Microservices Management, Service Oriented Architecture, and OAuth 2.0 as the security protocol. The results indicate that the microservices management system is highly effective and efficient for improving process performance, has a low risk level, a good control test validation of 93,33 %, and a usability grade of B, indicating that the system is deemed good and usable.

***Keywords**: Service Oriented Architecture, Microservices, Integrated System, OAuth 2.0*

1. Introduction

Since 2015, Udayana University has utilized IMISSU (Integrated Management Information System, the Strategic of UNUD), a Single Sign-On (SSO)-based information system [1]. All of Udayana University's developed information systems still employ the Monolithic Architecture model. Monolithic architecture is a conventional information system development model in which all functionality from web servers, databases, and business processes is encapsulated within a single application, making it impossible to execute each module independently [2].

Long-term issues arise when the system already has a large amount of data and is experiencing rapid data growth, and the resulting data must be able to support other information systems. As depicted in Figure 1, the data integration process between information systems at Udayana University continues to rely on the traditional method of joining tables between databases and making numerous connections to the database directly in the program, thereby reducing the performance of the information system. The addition of server capacity is the solution that must be chosen to improve the performance of the information system in light of its performance decline.

p-ISSN: 2252-3006
e-ISSN: 2685-2411

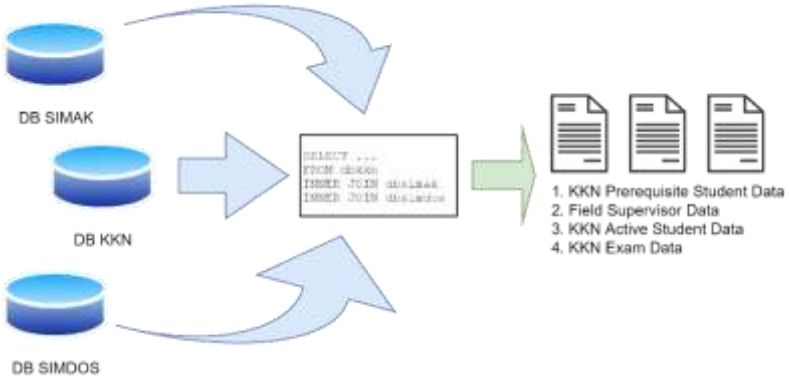

Figure 1. Cross Database Query of Kuliah Kerja Nyata (KKN) Data Collection

Service Oriented Architecture (SOA) is a method of data integration. SOA is an architectural approach capable of managing distributed applications in heterogeneous environments with rapid development and low cost [3]. According to Larrucea [4], microservices are small applications with a single purpose that can be independently implemented, scaled, and tested.

Using the SOA methodology, changes to the data integration process are implemented based on the preceding problem description by creating a small service on each information system. The created services are then stored in microservices management, which is converted into REST API writing format to become one-stop web services. By implementing a web services architecture, it is possible to address the monolithic architecture's weaknesses [5].

2. Research Method / Proposed Method

This research employs development research methods, also known as Research & Development, by looking for information to be used as reference material to develop information that has been received to increase technology, theory, and information as necessary.

2.1 Data Source

In this study, the primary data source is collected directly in the field by the authors [6]; these data are collected as primary data. The available data sources include existing web services about academic systems, personnel, finances, and planning. Table 1 demonstrates some examples of web services.

Table 1. Data Source in the form of Web Service URL

No	URL
1	https://sianita.unud.ac.id/curl/datapegawai/cari/{id_user}
2	https://api-bios.unud.ac.id/keuangan/pengeluaran?TanggalUpdate={tanggal}
3	https://kuisioner-ng.unud.ac.id/api/statusPerKuisioner/{id_user}/{id_periode}?withIdDosen=1
4	https://siluna.unud.ac.id/api/pok_realisasi?id_unit={id_unit}&tahun={tahun}
5	https://sianita.unud.ac.id/curl/datapegawai/list_pegawai?id_unit={id_unit}

2.2 Implementation Plan

In general, the SOA implementation steps in a system are divided into 3 phases, namely the Initiation Phase, the Develop Road Map Phase, the Implementation and Testing Phases [7]. As shown in Figure 2, the initiation phase identifies current conditions and reviews the literature on the work to be achieved. The Develop Roadmap stage shows SOA requirements with designs on the backend and frontend. The implementation and testing phase explains how to implement services and test each service according to the needs and is suitable for use by users who use or client services.

p-ISSN: 2252-3006
e-ISSN: 2685-2411

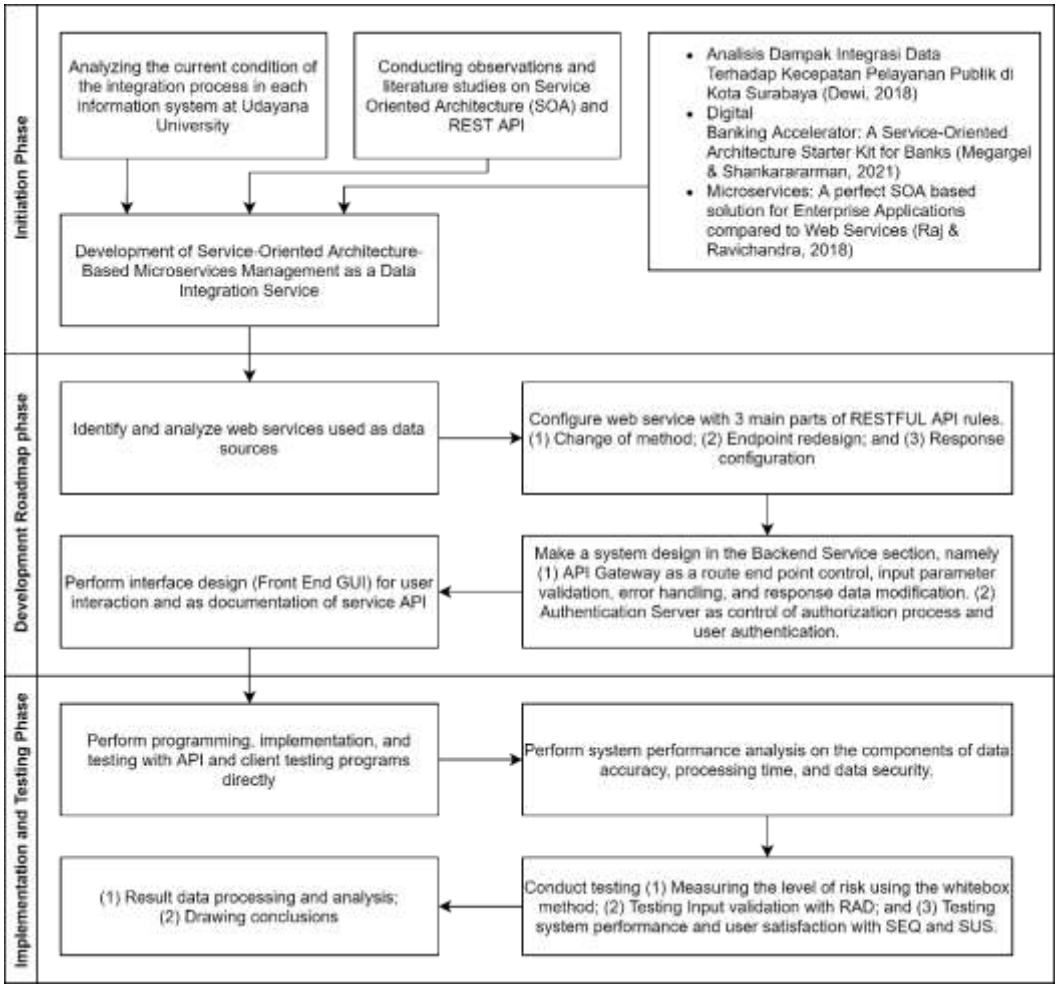

Figure 2. Step of Implementation

This microservices management will be constructed using the Service Oriented Architecture (SOA) concept, thereby transforming small services into a Microservices Management System. This system will eventually interact with web services on other information systems as an intermediary between service providers and service consumers. With this communication design, the Microservices Management System will meet the service client's needs.

As depicted in Figure 3, the Microservices Management Information System comprises three (three) entities: web services at each data provider, clients as data requestors, and Information Systems as intermediaries between web services and clients.

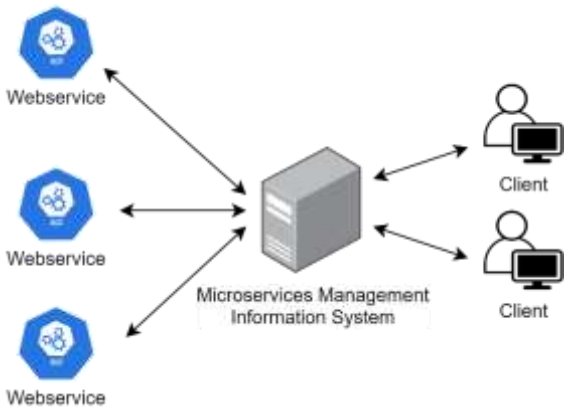

Figure 3. General Design of Microservices Management Information System

2.3 SOA Architecture Structure

As shown in Figure 4, the design structure produced by implementing Service Oriented Architecture (SOA) consists of two major components: the Backend Service and the Frontend GUI. On the Backend Service side, an Authorization Server regulates user authorization and authentication processes. The API Gateway controls route endpoints, verifies input parameters, handles errors, records logs, and modifies response data. The Frontend section is a visually accessible information system. This information system regulates API management and user management.

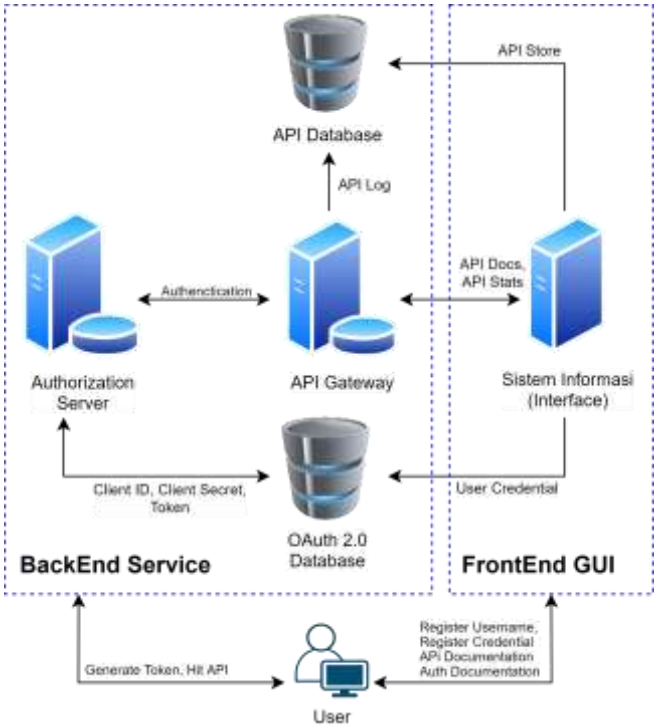

Figure 4. SOA Architecture Structure

2.4 Testing

The testing phases of this study employed three test models: system performance testing, white box testing method, and system feasibility test cases utilizing questionnaire distribution.

For white box testing, this study used the Basis Path method. In the technique of basis path testing, the test was conducted by generating a test case based on the cyclomatic complexity value. The value of cyclomatic complexity was derived from a flowchart based on the microservices management information system flowchart. The test case results were then used to create a result table that calculates the number of valid or appropriate scenario tests versus those that are invalid or not in accordance with the results [8].

The subsequent testing phase involved testing the data input control in accordance with Character checks (U1), Numeric Value Checks (U2), Check Digit (U3), Limit Tests (U4), Reasonableness Tests (U5), Internal Compatibility (U6), Cross Checks with data in other applications (U7), Table Look Ups (U8), Existence of Required Data (U9), Confirmation Screens (U10), dan Field lengths and Overflow checks (U11). Tests were conducted to determine whether the system performed input validation.

The last test used usability testing, which according to ISO 9241-11, is a test process to find out the extent to which the product is easy and can be used and operated by users in terms of achieving a specified goal with conditions of effectiveness, efficiency, and satisfaction in a particular context of use [9]. In this study, the usability testing process employed 2 (two) methods, namely Single Ease Question (SEQ) and System Usability Scale (SUS).

3. Literature Study

The concept in this study comes from a literature review in the form of articles, scientific journals, research reports, and books that are used as references in this study.

3.1 Service Oriented Architecture (SOA)

Microservices are the smallest components of the software or specialized applications for a specific task that collaborate to accomplish high-level objectives [10]. With an API (Application Programming Interface) interface, applications are organized so that they are separated into small independent services, have specific functions (high cohesion), and do not depend on other program components (loose coupling) [11].

Microservices architecture consists of simply dividing applications into small services based on their individual requirements so that they can function uniquely. The application is designed so that each function operates independently, and each function can use the technology stack based on its requirements, which means that different technologies will be utilized in a single large application [12].

3.2 REST API

REST (Representational State Transfer) is a technical or architectural rule that performs the process of communicating data and information through an interface such as HTTP. The REST API works very much like a regular web application. The client can send a request to the server via the HTTP protocol, and after that, the server responds by giving a response back to the client. REST was created and developed by Roy Fielding, the founder of the Apache HTTP Server Project [13].

3.3 State of The Art

Several previous studies have examined the success of data integration and the use of Service-Oriented Architecture (SOA) in data integration development. The first study by Dewi titled "*Analisis Dampak Integrasi Data Terhadap Kecepatan Pelayanan Publik di Kota Surabaya*" In the study, the author innovated public services by implementing e-Government to integrate population data for residents of Surabaya City. In the integration process, 10 data were presented in JSON format and used as entry data in other information systems, resulting in a population data filling process speed of 80.7% [14].

The second study by Megargel & Shankarararman titled "Digital Banking Accelerator: A Service-Oriented Architecture Starter Kit for Banks." In the study, the obstacle in designing a banking system service was the inability to update existing information systems. The study proposed using the Digital Banking Accelerator (DBA) as a "starter kit" with the SOA method. The study found that SOA-based services used for digital bank startups are more innovative and efficient [15].

The third study by Raj & Ravichandra titled "Microservices: A perfect SOA based solution for Enterprise Applications compared to Web Services." In this study, Service-Oriented Architecture has had a significant impact on how software applications are built. The study discussed the main weaknesses of web services and the benefits of microservices through SOA-based services. The study found that microservices are more suitable for developing SOA-based applications than using web services, at 36% [16].

4. Result and Discussion

The results and discussion resulted in the program's design are divided into two main structures: the Backend as the main service running the API and the Frontend as the API documentation display.

4.1 Service Data Flow

The initial phase of microservices management development begins with creating an API Group for all API endpoint conversion results. As shown in Figure 5, the process begins with converting all data source endpoints into endpoints with the first segment URL as a group and the second segment URL as a function of the API.

p-ISSN: 2252-3006
e-ISSN: 2685-2411

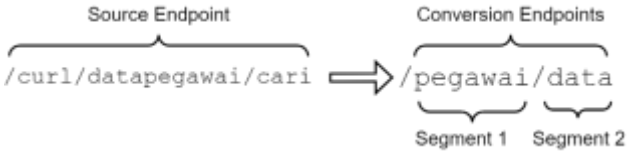

Figure 5. Conversion of End Point on Segment

When a user or client makes a request, traveling data flow is initiated. Every user registered with the Microservices Management Information System is validated to possess a credential key consisting of a Client ID and Client Secret, which is used to authorize API Gateway access. The process depicted in Figure 6 can be described as follows:

1. The user performs the authentication process by sending an encrypted credential key as Authorization Basic to the Authentication Server.
2. The Authentication Server receives a response by sending an Access Token to the user.
3. The user sends request data to the API endpoint that is intended for the API Gateway, accompanied by inserting an access token in the header.
4. API Gateway sends the user token as Authorization Bearer to Authentication Server. The response generated by the Authentication Server is in the form of token validation, which the API Gateway then receives.
5. If the validation result is invalid, then the API Gateway will display a response code "401" with the status "Unauthorized". The API Gateway will forward the user request to the source Webservice if the token validation is valid.
6. The response generated by the source Webservice is then modified by adding the response code, process time, and status.

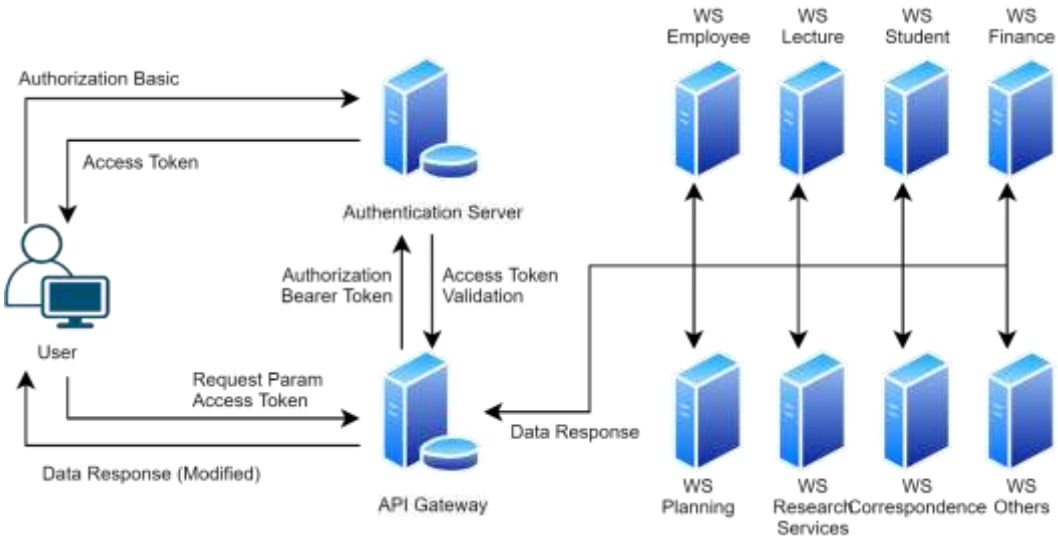

Figure 6. API Data Flow

4.2 Interface Implementation

Before using this API service, the user must understand how to obtain the access token. Figure 7 shows information such as the URL to be accessed, the method to be used, the header parameters to be embedded, and the body variables to be embedded. The "Try it out!" button allows users to test the system by entering the encrypted authorization value and login password.

p-ISSN: 2252-3006
e-ISSN: 2685-2411

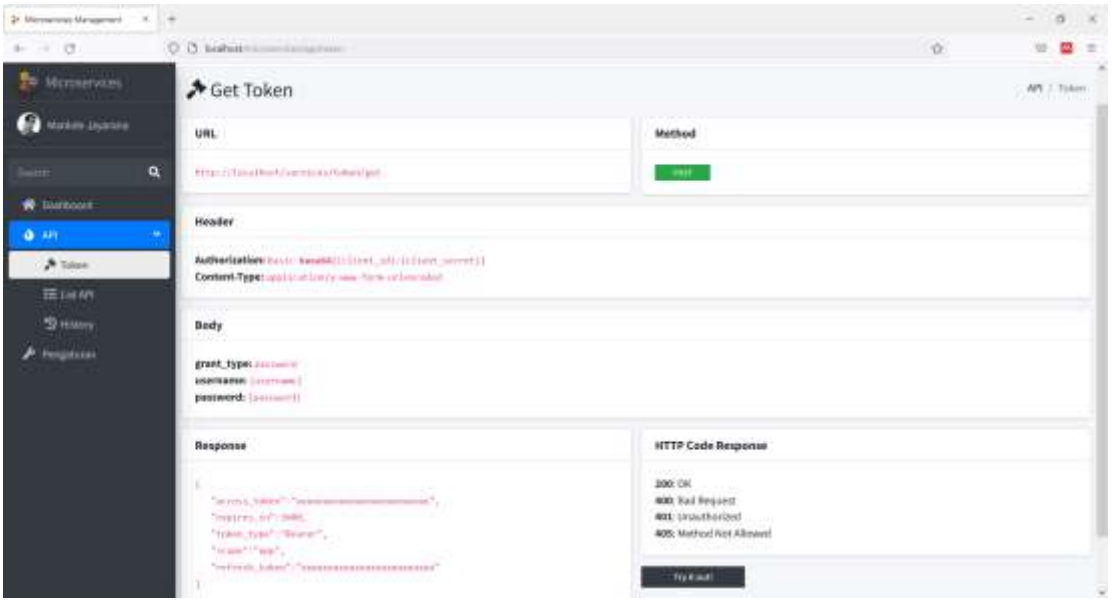

Figure 7. Get Token Interface Implementation

Figure 8 depicts the documentation's interface in the form of a List API. This menu provides access to all API information, including parameters, responses, and their functionality.

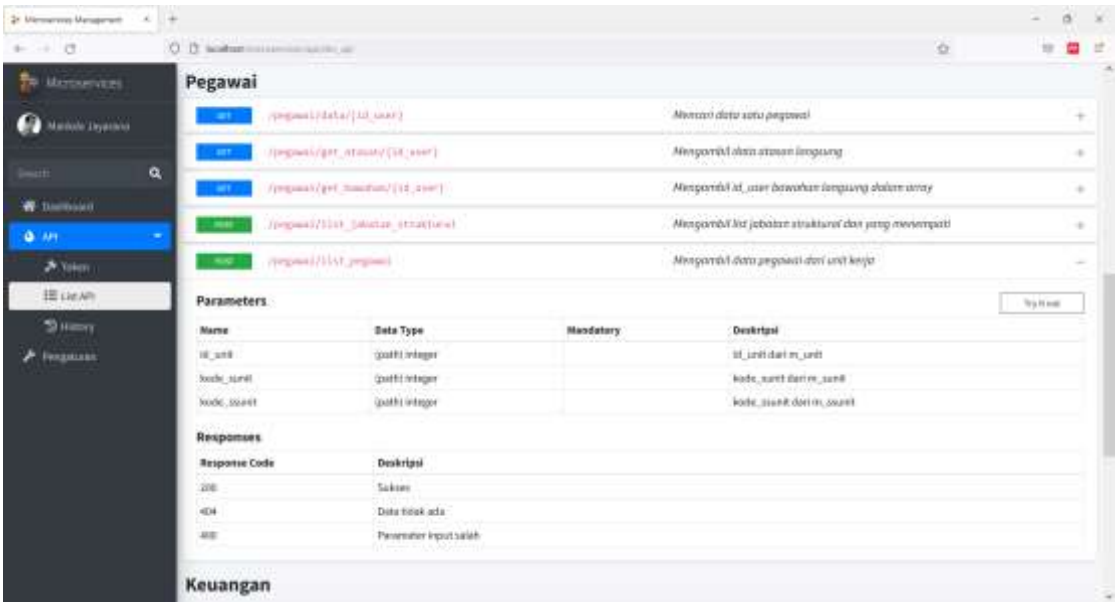

Figure 8. Implementation of API Documentation List Interface

4.3 System Performance Analysis

The microservices management system is analyzed to determine the system's capacity to support process performance. Components are tested for data accuracy, processing time, and data security during the analysis stages to identify efficiency.

1) Data Accuracy

Data accuracy on microservices is measured by analyzing several functionalities such as response code, response headers, API time-outs, JSON validation, and text in JSON responses. Figure 9 shows the output generated in the modified success response.

p-ISSN: 2252-3006
e-ISSN: 2685-2411

Figure 9. Modified Success Response of JSON

2) Processing Time

According to Juliver [17], APIs with an average response time of 0.1 and 1 second are considered high-performance. Twenty attempts are made against a single API endpoint to conduct process time trials. In the Microservices Management System, source and endpoint web services were subjected to testing. The processing time difference between the source web services and microservices management ranged from a minimum of 0.147 seconds to a maximum of 0.569 seconds, with an average processing time difference of 0.199 seconds. SOA processing, where an Authentication Server controls the authorization and user authentication processes and the API Gateway controls route endpoints, input parameters, error handling, recording logs, and modifying response data, slows down the processing time for microservices management data. Figure 10 depicts a comparison chart of API processing time between data source web services and microservices management.

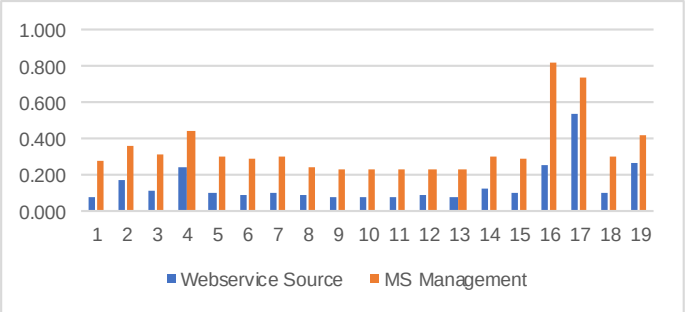

Figure 10. Comparison of Data Processing Time

3) Data Security

The process of all data traffic and transactions in microservices management is carried out using a security protocol with OAuth2 authentication. As shown in Figure 11, each user has a Client ID and Client Secret, where the credential key given is a 32-digit random value between A-Z, a-z, and 0-9 [18].

Figure 11. Authorization Key in User

Before the user makes a hit request for token access to the Authentication Server, the authorization key value is obtained by encrypting the Client ID and Client Secret with the Base64 method and then placing it in the header. The response from the Authentication Server will include an access token with an expiration date of 3600 seconds. Figure 12 represents the authentication server's response when the user successfully authorizes the token.

p-ISSN: 2252-3006
e-ISSN: 2685-2411


```
{
    "access_token": "0745fd5bcab3e864c96141942b89406a78735d72",
    "expires_in": 3600,
    "token_type": "Bearer",
    "scope": "app",
    "refresh_token": "8b61a709c669d985218a68de725bf1c7f031d592"
}
```


Figure 12. Response Authentication Server

4.4 Testing Results

The first test was conducted using the white box method to determine the risk level posed by the microservices management system [19]. The obtained cyclomatic complexity value is then used to determine the level of risk using the categories CC 1-10 for low risk, 11-20 for moderate risk, 21-50 for high risk, and >50 for extremely high risk [20]. As shown in Table 2, the test determined that all risks were minimal.

1) Register V(G) = 17 Edge - 11 Node + 2 = 8
2) Login V(G)=9 Edge-8 Node+2=3
3) Request Token V(G)=17 Edge-13 Node+2=6
4) Hit API V(G)=20 Edge-16 Node+2=6

Table 2. Relationship of Cyclomatic Complexity and Risk Value

No	Process	Cyclomatic Complexity Value	Risk Value
1	Register	8	Low
2	Login	3	Low
3	Request Token	6	Low
4	Hit API	6	Low

Figure 13 shows the input validation test, which aims to prevent the web service from processing invalid parameters and thereby increasing server load by executing invalid parameters.

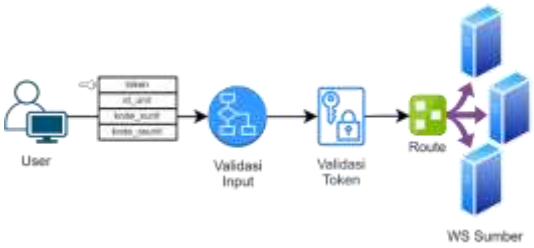

Figure 13. Input Parameter Validation Flow

The experiment was conducted by executing 15 input samples with improper parameter modifications. From the 15 input experiments conducted, it was determined that the web service source of the data regarding the suitability of the expectations contributed only 3 out of 15 or 20% of the total value. Table 3 shows experiments on the microservices system of data sources met expectations 14 times out of 15, or 93.33 percent of the time.

Table 3. Data Input Experiment on Microservices

Endpoint	Trial Type	Microservices Output		Desired Hope		Suitability
		HTTP Code	Response	HTTP Code	Response	
/pegawai/data	Value with a null value	400	User ID cannot be empty	400	Empty entries	TRUE
/pegawai/data	Value with letter grades	400	User ID must be integer	400	Input harus integeInput must be integerr	TRUE
/pegawai/data	Value with a	400	Please input	400	Incorrect input	TRUE

p-ISSN: 2252-3006
e-ISSN: 2685-2411

	minus value		User ID correctly			
/pegawai/data	Value with data return null	404	No data available	404	No data available	TRUE
/pegawai/data	A valid value with a return value	200	Response data is valid	200	Return response	TRUE
/keuangan/pengeluaran	Value with a null value	400	Date cannot be empty	400	Empty entries	TRUE
/keuangan/pengeluaran	Value with wrong date format	400	The date format must be correct	400	Incorrect date format	TRUE
/keuangan/pengeluaran	Value with data return null	404	No data available	404	No data available	TRUE
/keuangan/pengeluaran	Value with letter grades	400	The date format must be correct	400	Incorrect date format	TRUE
/keuangan/pengeluaran	A valid value with a return value	200	Response data is valid	200	Return response	TRUE
/dosen/data	Value with a null value	400	Lecturer ID cannot be empty	400	Empty entries	TRUE
/dosen/data	Value with letter grades	400	Lecturer ID must be integer	400	Input must be integer	TRUE
/dosen/data	Value with a minus value	400	Please input the Lecturer ID correctly	400	Incorrect input	TRUE
/dosen/data	Value with data return null	400	The Requested URL Returned Error	404	No data available	FALSE
/dosen/data	A valid value with a return value	200	Response data is valid	200	Return response	TRUE

The following experiment utilized Single Ease Question (SEQ) and System Usability Scale (SUS) with 15 programmer respondents. Respondents were asked to complete eight tasks, and Figure 14 depicts the overall relative efficiency value.

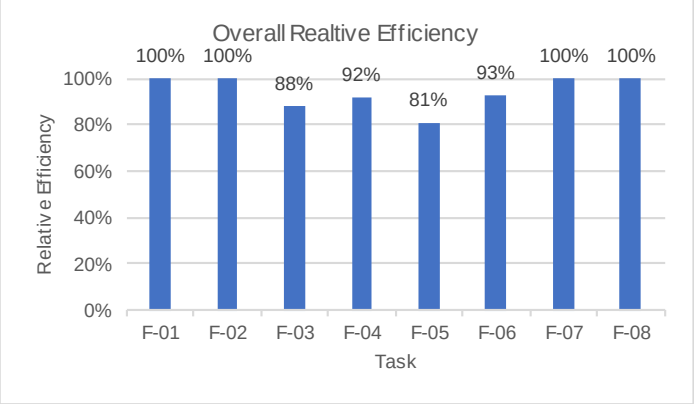

Figure 14. Overall Relative Efficiency

Table 4 shows the respondents were then asked to complete the SUS questionnaire, which yielded a score of 81. Based on the data, the microservices management system's

p-ISSN: 2252-3006
e-ISSN: 2685-2411

usability was rated as acceptable and adequate, with a B score on a scale from excellent to poor [21].

Table 4. Questionnaire Scatter Count Result Score

No	Respondents	Count Result Score										Amount	Mark (Amount x 2.5)
		Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9	Q10		
1	R-1	4	4	4	4	4	3	4	4	3	4	38	95
2	R-2	4	3	4	2	4	2	4	2	2	2	29	73
3	R-3	3	3	4	3	3	3	3	3	3	3	31	78
4	R-4	4	3	4	2	4	3	4	2	4	2	32	80
5	R-5	3	3	3	3	3	3	3	3	3	3	30	75
6	R-6	3	4	3	4	4	4	4	3	4	3	36	90
7	R-7	3	4	3	3	3	3	3	3	2	3	30	75
8	R-8	4	3	4	3	4	2	4	3	2	3	32	80
9	R-9	3	4	3	4	4	2	4	3	3	3	33	83
10	R-10	4	3	4	3	4	3	3	2	2	4	32	80
11	R-11	3	3	4	3	3	3	3	3	3	3	31	78
12	R-12	3	3	3	3	3	2	3	3	3	3	29	73
13	R-13	4	4	4	4	4	4	4	4	4	4	40	100
14	R-14	3	3	4	3	4	3	4	3	3	3	33	83
15	R-15	3	3	4	4	3	3	3	3	3	3	32	80
Average Score (Final Result)													81

5. Conclusion

Based on the outcomes of the integration process that was conducted using the Service Oriented Architecture (SOA) architectural model, it can be concluded that several modifications and design outcomes have met the following SOA criteria:

1. Services. The microservices management system developed has been in the form of an API service so that other systems or applications that will interact can only communicate with the API.
2. Best Practices. The microservices management system is already in the form of an API so that users with various programming languages can access API services.
3. Process. The output produced by the microservices management system uses the JSON output format to be read by all programming languages.
4. Users. The microservices management system that has been developed has fulfilled all user orientations by displaying API documentation that can be viewed and accessed directly by the user.
5. Platforms. The microservices management system is already in the form of an API. The desktop, mobile, and web platforms can directly access API services.

The purpose of this research is to transform the traditional information system integration scheme into an SOA architecture-based system integration by developing a microservices management information system. In addition, this study also performed system performance analysis and testing of the built microservices management system using white box, input and output testing, and usability testing methods. The analysis showed that the microservices management system is highly effective and efficient in improving performance processes. The testing results revealed that the microservices system has a low risk level, good control test validation, and usability with a grade of B, indicating that the system is good and suitable for use [22].

This research can be continued by developing on the program side to become Full Lifecycle API Development, Integration, and Management, eliminating the need for direct program coding. In addition, it is possible to investigate the development of SOA architecture by incorporating the concepts of load balancing and fail-over API involving multiple servers to improve the API service quality in the event of an API server disruption.

References

[1] D. Krisnandari, D. M. Wiharta, and N. P. Sastra, "Penerapan Teknologi Informasi dalam Reformasi Birokrasi pada Bidang Pendidikan," *Majalah Ilmiah Teknologi Elektro*, vol. 18, no. 2, 2019, doi: 10.24843/mite.2019.v18i02.p19.
[2] F. Ponce, G. Marquez, and H. Astudillo, "Migrating from monolithic architecture to microservices: A Rapid Review," in *Proceedings - International Conference of the Chilean Computer Science Society, SCCC*, 2019. doi: 10.1109/SCCC49216.2019.8966423.
[3] J. S. P. Hantana, "Pendekatan Service Oriented Architecture (SOA) Pada Pelaksanaan E-Government di Kementerian Hukum dan HAM RI," *Jurnal Nasional Pendidikan Teknik Informatika (JANAPATI)*, vol. 2, no. 3, 2013, doi: 10.23887/janapati.v2i3.9813.
[4] X. Larrucea, I. Santamaria, R. Colomo-Palacios, and C. Ebert, "Microservices," *IEEE Softw*, vol. 35, no. 3, pp. 96–100, May 2018, doi: 10.1109/MS.2018.2141030.
[5] D. K. Sasidharan and S. Kumar N, *Full Stack Development with JHipster*. 2018.
[6] Sugiyono, "Metodologi Penelitian Bisnis: Pendekatan Kuantitatif, Kualitatif, Kombinasi, dan R&D," *Alfabeta*. 2017.
[7] T. Erl, "Service-Oriented Architecture Analysis and Design for Services and Microservices," *Prentice Hall Press*. 2016.
[8] M. F. Londjo, "Implementasi White Box Testing Dengan Teknik Basis Path Pada Pengujian Form Login," *Jurnal Siliwaangi*, vol. 7, no. 2, 2021.
[9] R. Rahmi, I. M. A. Pradnyana, and M. W. A. Kesiman, "Usability Testing Berbasis ISO 9241-11 Pada Aplikasi Salak Bali (Studi Kasus: Polres Buleleng)," *Kumpulan Artikel Mahasiswa Pendidikan Teknik Informatika (KARMAPATI)*, vol. 8, no. 3, 2019.
[10] D. Gonzalez, *Developing microservices with Node.js : learn to develop efficient and scalable microservices for server-side programming in Node.js using this hands-on guide*. 2016.
[11] S. Newman, *Building Microservices*. 2015.
[12] Y. Jayawardana, R. Fernando, G. Jayawardena, D. Weerasooriya, and I. Perera, "A full stack microservices framework with business modelling," in *18th International Conference on Advances in ICT for Emerging Regions, ICTer 2018 - Proceedings*, 2019. doi: 10.1109/ICTER.8615473.
[13] B. Ç. Agon Memeti, Florinda Imeri, "REST vs. SOAP: Choosing the best web service," *Journal of Natural Sciences and Mathematics of UT,Vol 2,No 3 2017*, vol. 2, no. April, 2015.
[14] R. S. Dewi, "Analisis Dampak Integrasi Data terhadap Kecepatan Pelayanan Publik di Kota Surabaya," *Jurnal Sistem Informasi*, vol. 14, no. 2, 2018, doi: 10.21609/jsi.v14i2.639.
[15] A. Megargel and V. Shankarararman, "Digital Banking Accelerator: A Service-Oriented Architecture Starter Kit for Banks," *IEEE Softw*, vol. 38, no. 3, 2021, doi: 10.1109/MS.2020.3029876.
[16] V. Raj and S. Ravichandra, "Microservices: A perfect SOA based solution for enterprise applications compared to web services," in *2018 3rd IEEE International Conference on Recent Trends in Electronics, Information and Communication Technology, RTEICT 2018 - Proceedings*, 2018. doi: 10.1109/RTEICT42901.2018.9012140.
[17] J. Juliver, "API Response Time, Explained in 1000 Words or Less," *HubSpot*, 2022. https://blog.hubspot.com/website/api-response-time (accessed Jan. 22, 2023).
[18] Ed. D. Hardt, "RFC 6749 - The OAuth 2.0 Authorization Framework," 2012. https://datatracker.ietf.org/doc/html/rfc6749 (accessed Jun. 04, 2022).
[19] ~ Wwjmrd, "A Research Paper on White Box Testing," *International Journal Peer Reviewed Journal Refereed Journal Indexed Journal UGC Approved Journal Impact Factor*, vol. 4, no. 2, 2018.
[20] H. L. P. Y. T. A. Candra, "Cyclomatic Complexity Test Design Flowgraph Registration of Emergency Installation Patients in Wava Husada Hospital Using SEM," *International Journal of Science and Research (IJSR)*, vol. 6, no. 8, 2017.
[21] R. W. Bambang Pudjoatmodjo, "Tes Kegunaan (Usabilty Testing) Pada Aplikasi Kepegawaian Dengan Menggunakan System Usabilty Scale (Studi Kasus : Dinas Pertanian Kabupaten Bandung)," *Seminar Nasional Teknologi Informasi dan Multimedia*. 2016.

p-ISSN: 2252-3006
e-ISSN: 2685-2411

[22] N. A. N. Ahmad and M. Hussaini, “A Usability Testing of a Higher Education Mobile Application Among Postgraduate and Undergraduate Students,” *International Journal of Interactive Mobile Technologies*, vol. 15, no. 9, 2021, doi: 10.3991/ijim.v15i09.19943.