Implementation of Micro Services Architecture on Comrades Backend

T Suryana¹, A M Bachtiar², C S Budi³
Departemen Sistem Informasi, Universitas Komputer Indonesia, Indonesia
taryanarx@email.unikom.ac.id

Abstract. This research aims to increase the scalability and maintainability of Comrades backend. The experimental method that was used in this research is Domain-Driven Design. Applying Domain-Driven Design, Comrades backend will have more scalable and high-level scalability architecture, namely, microservices architecture. The result of this research is a new model of backend architecture and a prototype to measure the performances of this new model. To achieve this result, the transformation of architecture from monolithic into several context and implementation of the context to be microservices architecture are the main steps of this research. Moreover, complete software architecture and infrastructure will be developed as the output of this research. To make it possible, the research uses several technologies including Docker, Kubernetes, and Gateway API. By using this architecture, Comrades developer are able to create Comrades backend more reliable because each functional of Comrades will have their architecture. They can use the services to give a better experience for Comrades users in accessing the features in comrades. For the conclusion, a new model of Comrades backend architecture has more scalable level without any error.

1. Introduction

Comrades is a smart application in delivering information and education about HIV/AIDS. This application can help people living with HIV/AIDS to trade information, keep supporting each other, and be connected with societies. Comrades have several features, namely: chatting with “Sahabat ODHA,” an article about HIV/AIDS, news and event in HIV/AIDS domain, positive tweet (supporting element) and sticker for supporting people living with HIV/AIDS. To maintain this application, its developer conduct cooperation with communities which have a concern in HIV/AIDS domain. This co-operation causes the increment of Comrades users. A number of users has two sides, and the negative side is decrement on application scalability and availability [1].

In the first development process, Comrades used monolithic architecture. This kind of architecture implements functional aspects from web service which uses code base and uniform technology in its implementation [2]. This happens because of the limitation of development time and a number of human resources in the team. As business needs in the application, monolithic architecture becomes greater than before and on the other hand, monolithic architecture becomes difficult to maintain [3]. From testing result using Blaze meter on Comrades backend, the application needs 2.5 seconds to give back a response. It takes too long for a backend [4]. Other than that, there are several errors when the number of requests is going higher. The performances of Comrades backend can be seen in Table 1.
Based on the literature review, there is another approach for software architecture. That approach is microservices architecture. Microservices is an approach that divides the application into small parts of services based on their functions [5]. Breaking down process is done to creating web services more scalable, resilience, and high-availability [6]. This research focuses on transform architecture of Comrades backend from monolithic into microservices and measure the result of its transformation. Using the result of this research, Comrades developer can make Comrades backend more reliable and maintainable.

The purpose of this study is to increase the scalability and maintainability of Comrades Backend. The experimental method that was used in this research is Domain-Driven Design. Applying Domain-Driven Design, Comrades backend will have more scalable and high-level scalability architecture, namely, micro-services architecture.

2. Method
The methodology used in this research can be seen in Figure 1.

![Research methodology in this research.](image)

Figure 1. Research methodology in this research.

Here are the explanations of each stage:
1. Architecture Analysis of Comrades
   Purpose of this stage is to an understanding about architecture in Comrades backend before the transformation [7].
2. Functional Analysis
   Dividing or breakdown process for microservices is based on functional. Therefore, the identification of functional in Comrades Backend is important [8].
3. Context Identification
   Grouping of analyzed functional into several contexts using domain-driven design is the main focus in this stage [9].
4. Bounded Context Identification
   After identification of context, the main focus is to measure the boundary of each context.
5. Context Mapping
   This stage focus on describes all context and connection between them.
6. Implementation and Testing of New Architecture
   Several technologies will be used to implement the microservices architecture. After the implementation, measurement of implementation result is the last stage for this research.
3. Results and Discussions
In this section, all stages in the research methodology are explained specifically. The explanation is divided into several parts that consist of input, process, and the output for each stage.

3.1 Architecture Analysis of Comrades
The architecture of Comrades backend can be seen in Figure 2.

![Figure 2. Comrades old architecture (monolithic)](image)

Comrade’s backend was using monolithic architecture. Similar to another ordinary system of the backend, Comrades backend was working in one form of web services that have many connections with clients and another service.

3.2 Functional Analysis
Functional explains about business process in a software [10]. Use case diagram will be used as a functional model. From the analysis process, there are 23 functional in Comrades. The examples of Comrades functional can be seen in Table 2.

| Use Case               | Description                                                                 |
|------------------------|-----------------------------------------------------------------------------|
| Register               | Users can register themselves to use the system.                             |
| Supporting tweet       | Positive tweets to supporting people living with HIV/AIDS. This functional is using machine learning to determine which tweets from Twitter are positive tweets. |
| Location for HIV/AIDS Services | People living with HIV/AIDS can find public services and medicines for them. |

3.3 Context Identification
All found functional are grouped based on similarities between them. The group is called by context. From this research, there are 12 contexts, namely:
1. Event
2. Friends
3. Healbox
4. Medicine location
5. Notification
6. Posting
7. Short Message Service (SMS)
8. Twitter
9. Users
10. Email
11. Advanced Encryption Standard (AES)
12. Supporting Sticker.

3.4 Bounded Context Analysis
The characteristic of micro-services architecture is on autonomous of the services. Bounded context analysis is needed to describe it. Main focus of bounded context analysis is for knowing the boundary for each context. There are 21 bounded contexts and examples of bounded context can be seen in Table 3.

| Bounded Context | Description |
|-----------------|-------------|
| Event           | All contexts whose focus is set events on the web services |
| Notification    | All contexts whose focus is sending notification on mobile apps |

3.5 Context Mapping
Snippet from a context map in this research can be seen in Figure 3.

![Context Mapping Diagram]

In this stage, all bounded context is well-defined. To describe the connection between bounded context, there is a diagram called context map.
3.6 Implementation and Testing of New Architecture

The new architecture of Comrades can be seen in Figure 4.

![Micro services architecture on Comrades](image)

**Figure 4.** Micro services architecture on Comrades

This stage focuses on implementing micro services architecture based on a content map. To implement the architecture, there are several non-functional requirements that divided into:

1. Hardware Node
2. MySQL hardware node
3. MongoDB hardware node
4. Software node
5. MySQL software node
6. MongoDB software node.

Other than that, several refactoring in code and API need to be done.

After the implementation, a new architecture was tested using Blaze meter to measure its performances. The performances of new Comrades backend architecture can be seen in Table 4.

| Functional                  | Average Response Time (ms) | Average Latency (ms) | Average Throughput (second) |
|-----------------------------|---------------------------|----------------------|------------------------------|
| HIV/AIDS Articles           | 654.78                    | 483.99               | 0.01                         |
| HIV/AIDS News               | 544                       | 399.14               | 0.01                         |
| News from Admin             | 371.21                    | 285.03               | 0.01                         |
| Medicine Location Detail    | 87.53                     | 87.51                | 0.01                         |

| Functional                  | Average Response Time (ms) | Average Latency (ms) | Average Throughput (second) |
|-----------------------------|---------------------------|----------------------|------------------------------|
| HIV/AIDS Articles           | 3237                      | 2419                 | 0.051                        |
| HIV/AIDS News               | 2720                      | 1995                 | 0.05                         |
| News from Admin             | 1856                      | 1425                 | 0.05                         |
| Medicine Location Detail    | 1299                      | 1091                 | 0.05                         |
From the results, the average response time for Comrades backend is decreasing significantly. These show that breakdown process from a monolithic architecture into microservices can distribute the income request equally and not burden each service in Comrades backend. Another result that is not less important is the lowering number of errors that produce by microservices. It significantly decreases even is almost zero.

4. Conclusion
From all processes in this research, the performances of Comrades are increased. It can be concluded from QoS on the testing stage. Other than that, errors on responses didn’t found at all. The only problems are not all QoS for backend performances was measured in this research, and not all parts of Comrades system were analyzed (only backend). Further research is needed to cover all the mentioned problems.

References

[1] Rajendran, T., and Balasubramanie, P. (2009). Analysis on the study of qos-aware web services discovery. arXiv preprint arXiv:0912.3965.
[2] Messina, A., Rizzo, R., Storniolo, P., and Urso, A. (2016). A simplified database pattern for the microservice architecture. In The Eighth International Conference on Advances in Databases, Knowledge, and Data Applications (DBKDA).
[3] Katuwala, K. (2016). Microservices: A Flexible Architecture for the Digital Age Version 1.0. American Journal of Computer Science and Engineering. 3 23–8
[4] Vera-Rivera, F. H. (2018, November). A development process of enterprise applications with microservices. In Journal of Physics: Conference Series. 1126(1) p. 012017. IOP Publishing.
[5] Familiar, B. (2015). Microservice architecture. In Microservices, IoT, and Azure (pp. 21-31). Apress, Berkeley, CA.
[6] Mazurov, A., Farley, N., Couturier, B., and Popov, D. (2017). IOP: Microservices for systematic profiling and monitoring of the refactoring process at the LHCb experiment. In J. Phys.: Conf. Ser. 898, p. 072037).
[7] Bachtiar, A. M., and Laksmiwati, H. (2014, May). Development of requirement elicitation model for prediction of student achievement in university. In 2014 2nd International Conference on Information and Communication Technology (ICoICT) (pp. 47-52). IEEE.
[8] Thompson, C., White, J., Dougherty, B., and Schmidt, D. C. (2009, November). Optimizing mobile application performance with model–driven engineering. In IFIP International Workshop on Software Technologies for Embedded and Ubiquitous Systems (pp. 36-46). Springer, Berlin, Heidelberg.
[9] Pettrasch, R. (2017, July). Model-based engineering for microservice architectures using Enterprise Integration Patterns for inter-service communication. In 2017 14th International Joint Conference on Computer Science and Software Engineering (JCSSE) (pp. 1-4). IEEE.
[10] Suhendra, A., and Bachtiar, A. M. (2016). Migration Code Pada Backend Crimezone Dari Php Ke Scala. Jurnal Sistem Informasi, 12(2), 74-81.