Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 124 (2017) 736–743

Procedia Computer Science

www.elsevier.com/locate/procedia

4th Information Systems International Conference 2017, ISICO 2017, 6-8 November 2017, Bali, Indonesia

Design and Development of Backend Application for Public Complaint Systems Using Microservice Spring Boot

Hatma Suryotrisongko*, Dedy Puji Jayanto, Aris Tjahyanto

Institut Teknologi Sepuluh Nopember, Kampus ITS Sukolilo-Surabaya 60111, Indonesia

Abstract

E-government is an implementation of ICT (information and communication technologies) in the field of governance in improving services to the public by a government or public sector. For example, Smart City, online licensing services, community complaint services, etc. The purpose of this research is to develop public complaint service application based on web application which uses springboot microservice architecture. Microservice architecture was used to divide the application functionality into many parts, or many micro-services based on business process and the services are interconnected, becoming a single application with a complete business process. One of the advantages of this architecture is that more microservice can be added without affecting others. The application was deployed in a cloud environment that can be accessed through a browser.

Keywords: e-Government; Microservices; Complaint Services; Spring Boot

1. Introduction

E-government consists of the use of electronic communication technologies such as internet, for improving citizens access to public services [1]. Implementation of E-Government provides the efficiency and speed of management on the reporting administration system, as well as the transparency of processes that occur in government administration. Through it, an aspect emerged which is called good governance. However, the implementation of E-government in Indonesia faces many issues such as expense issues for developing and

* Corresponding author. Tel.: +62-31-5999-944; fax: +62-31-5964-965.
E-mail address: hatma@is.its.ac.id

operating e-government applications, technical issues such as security issues, privacy, and system update and human resource issues in which there is lack of capability to manage it. Cloud technology becomes one of the alternative answers to tackle those issues. This model allows consumers to use applications that exist in the cloud online through providers that can be accessed in various kinds of devices without worrying about those issues.

The use of microservice technology can give some advantages to the e-government system in the cloud. The concept of modularity in microservice allows the management of services that exist separately in an application. The impact to a development of a particular service is that it will not interfere with other services. Capacity building of a service can be distinguished among other services so that the resource is used appropriately. Also, the development of services can be developed with different programming languages [2].

2. Related works

Sam Newman [3] developing a microservice application, first to conduct is determining the boundaries of the environment or bounded context. Simply, to specify the bounded context of the application that can be seen from the application business process which can then be grouped according to functional groups from the user, e.g., the finance department is in charge of the payment, and the warehouse department is engaged in customer orders. Then each function will be put together into one module. The module then becomes the bounded context to create microservices that match the purpose of the module made. The microservices that have been made will apply the concept of loose coupling between other modules and high cohesion microservice that are interconnected with the module made. Finally, the part which will do data writing to the database and read data from the database can be determined.

Purnama & Yatini [4] developed a thesis management application using Node.js which aims to avoid any similarity of topic or thesis title which is often the case of plagiarism. Node.js was built using microservice architecture purposing the ease of development of the application. When there is a case of new functional addition, re-creating the application is unnecessary, and the function can be added independently. It takes less time for further development.

Janssen & Joha [5] explained that the use of the Software as a Service (SaaS) model in the public sector is still infrequent. Although SaaS in the public/e-government sector promises many advantages, such as cost-saving and effectiveness, the challenges are severe, e.g., quality, security, privacy, and also the need to customize different systems in the region One with other areas.

3. Methodology

3.1. Functional requirement analysis

Table 1. Functional requirements.

ID	Actor	Functional Requirements
FR1	Administrator	Customer registration
FR2	Administrator	Citizen ID management
FR3	Administrator	Category management
FR4	Citizen	ID verification
FR5	Citizen	Sent a complaint
FR6	Citizen	Check complaint
FR7	Government Work Unit	Show complaint recap
FR8	Government Work Unit	Answer a complaint
FR9	Government Work Unit	Delete a complaint
FR10	Vendor	View all customer

Functional requirement analysis was conducted by looking at some similar applications. For example, the public complaint web application in Kediri city [6]. The analysis was attained through looking at design document of the application. Also, the analysis was made from literature studies to find some information related to the functional requirement of e-government public reporting applications.

Functional requirement consists of actor requirement and functional requirement. The actor is a user who will use the application. Few actors who use the application include an administrator, vendor, citizen, and the government work unit. The functional requirement itself is the function of the application. A list of some functional requirements of the application is in Table 1.

3.2. Model the microservice

After a functional requirement was made, the next step is to model the microservice. This step splits the functional requirement which is also known as bounded context into some microservices that match the purpose of the bounded context. In other words, the microservice is a group consisting of one or more microservices, and they are interconnected to perform a business process or a function. The microservices in this application is in Table 2.

Table 2. Microservice requirements.

ID	Microservice Requirements
FR1	Create a new customer data
FR2	Create citizen ID data
FR2	Show list of citizen ID data
FR2	Delete citizen ID data
FR3	Create a category
FR3	Delete a category
FR3	Show list of category
FR4	Get a citizen ID data
FR5	Create a new complaint
FR5	Get a ticket number
FR6	Show complaint status by ticket
FR7	Show all complaint with particular status
FR8	Update answer in a complaint data
FR9	Delete a complaint
FR10	Show all registered customer

3.3. Designing use case

The designing use case was generated from the analysis of functional requirement. Use case shows the interaction between the actor with the application like the Fig. 1.

3.4. Application development

This phase is the process of coding the application based on functional requirement, microservices requirement, and use case. This research focuses on developing the back-end application using java programming language and springboot framework. Springboot framework was favored for it has several advantages.

Spring has supported MVC and provides a RESTful Web Service feature. A database connection has also been provided in the spring package. Spring framework also supports a dependency injection. Dependency injection is the ease of configuration dependency in the application so that in the application development process becomes more convenient. Each spring framework also supports Aspect Object Programming (AOP) [7]. The additional advantage using spring boot is that a tomcat server could easily be included and can be run directly [8].

Developing the front-end application is also needed for communicating to the back-end for making business process. The front-end development is made using typescript programming language and Angular2 framework.

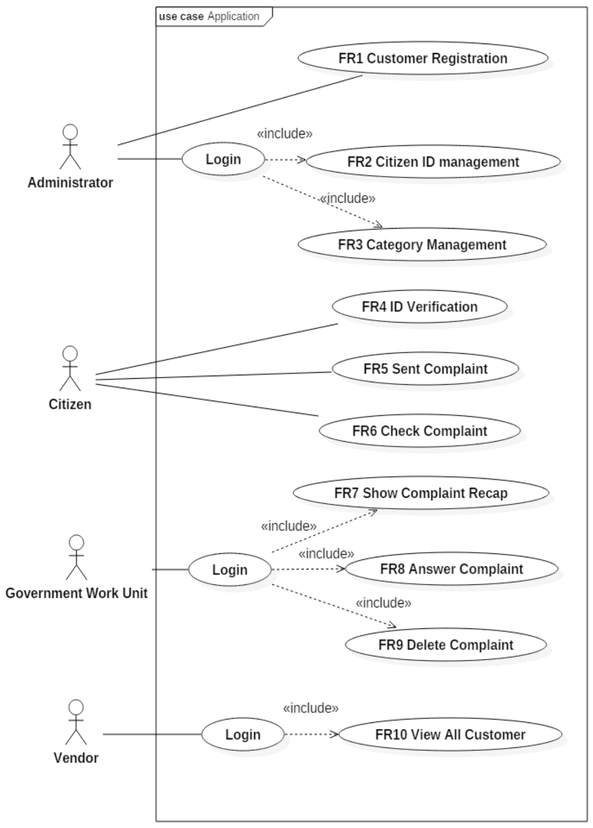

Fig. 1. Application use case.

3.5. Microservice blackbox testing

Blackbox testing is a test conducted to check whether the functional application is running correctly without knowing the processes occur within the app [9]. Blackbox testing was done by creating a test case in the form of test input and output expected from functional applications. Testing can be done on applications that do not use algorithms or low granularity levels [10], so it does not take much time [11]. Although for testing microservice two testings are needed, which are white box and black box testing, this research just performs the black box test because the application does not use any sophisticated algorithm and has low granularity details then the black box test is enough.

4. Application business process

The public complaint application is operated and managed by a specific unit or team. In some cases, in areas such as Bandung city [12], this kind of application is operated and managed in command center so that its work is optimal.

This public complaint application is a cloud-based application. Local governments can rent the application services by paying some money to the vendor. An administrator can be assigned by the government for operating the applications. The administrator can upload citizen ID data.

Citizens create complaint via the application in web. To begin with, citizens must verify their ID based on the uploaded data ID by the administrator. After that, their complaint will be saved on the database and ready to be managed by the government work unit. First, there is a selection stage by the system to filter the content of the complaint. The selection process is managed by the government work unit in the command center. Valid complaint content will be classified by category that has been created by the administrator and then sent to the right government unit outside the command center so that that complaint can be answered with the right answer and right follow-up in the field. If the complainant report is addressed or handled in the field, then the citizen who sent the complaint may change the complaint status to completion to inform the government that the complaint has been adequately addressed.

5. Application architecture

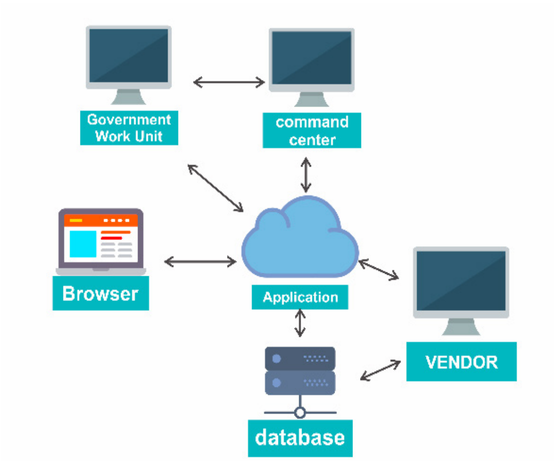

Fig. 2. Application architecture.

The public complaint application that uses cloud technology has a cloud architecture like Fig. 2. Front-end applications will be stored in the cloud as well as back-end applications along with application databases in the same cloud service to save resource usage. The application is accessible to the public on the web through a browser. Complaints submitted through the web application will be sent into the database for further management by the government which in this case is the command center and other government work unit. Vendors access the applications either back-end or front-end to perform maintenance on the application.

6. Microservice architecture

There is a tool that can make it easier for developers to see all the microservice available on the application. The tool is called swagger. Swagger is a standard framework that allows developers to quickly find and understand all the services on the application without accessing the program code, application development documentation, and without requiring inspection of the application service network [13]. Swagger maps the microservice based on the controller class that has been created using springboot framework. Class controller (or can be called a controller) is a class file in springboot which consists of methods to perform an input-output process according to business needs made. In the controller class, the methods can be made into a REST API. Swagger shows all the controllers that have been created in the application. In each swagger controller the REST API can be accessed with the description of the method used (the REST method used is usually GET, POST, DELETE, PUT), then there is the REST API

URL, and also the method name in the controller class. REST API is what will be used by the front-end application as access to a microservice with a predefined communication method that has been made.

6.1. Class controller

Table 3 shows class controllers that exist in the application with the associated microservices.

Table 3. Controller class.

Controller Class	Microservice Requirements
Customer Controller	Create a new customer data
Citizen Controller	Create citizen ID data
Citizen Controller	Show list of citizen ID data
Citizen Controller	Delete citizen ID data
Category Controller	Create a category
Category Controller	Delete a category
Category Controller	Show list of category
Citizen Controller	Get a citizen ID data
Complaint Controller	Create a new complaint
Complaint Controller	Get a ticket number
Complaint Controller	Show complaint status by ticket
Complaint Controller	Show all complaint with particular status
Complaint Controller	Update answer in a complaint data
Complaint Controller	Delete a complaint
Customer Controller	Show all registered customer

6.2. Rest method

Table 4 is a table containing REST method information that exists in the application with the associated microservices.

Table 4. REST method.

REST Method	Microservice Requirements
POST	Create a new customer data
POST	Create citizen ID data
GET	Show list of citizen ID data
DELETE	Delete citizen ID data
POST	Create a category
DELETE	Delete a category
GET	Show list of category
POST	Get a citizen ID data
GET	Create a new complaint
GET	Get a ticket number
GET	Show complaint status by ticket
GET	Show all complaint with particular status
PUT	Update answer in a complaint data
DELETE	Delete a complaint
GET	Show all registered customer

6.3. REST URL

Table 5 contains the URL of the REST. The URL is used to communicate between front end and backend.

Table 5. REST URL.

Rest URL	Microservices Requirements
/customer/new	Create a new customer data
/citizen/upload	Create citizen ID data
/rest/citizen/allby/{idkokab}	Show list of citizen ID data
/rest/citizen/delete/by/{idkokab}	Delete citizen ID data
/rest/admin/category/new	Create a category
/rest/admin/category/delete/{categoryid}	Delete a category
/rest/admin/category/all/{idkokab}	Show list of category
/citizen/checknik	Get a citizen ID data
/complaint/new	Create a new complaint
/complaint/new	Get a ticket number
/complaint/find/ticket/{ticketcode}	Show complaint status by ticket
rest/complaint/kokabandstatus	Show all complaint with particular status
rest/complaint/update/answer/{id}	Update answer in a complaint data
complaint/delete/{id}	Delete a complaint
/rest/user-dev/customer/all	Show all registered customer

6.4. Architecture diagram

The front-end can be divided into four user interfaces (UI), namely admin dashboard (Admin UI) for administrator, Operator UI for government work unit, UI complaint that is used by citizen to create and check compliant, and Vendor UI that is used by the vendor. Each UI will be associated with some microservice as shown in Fig. 3.

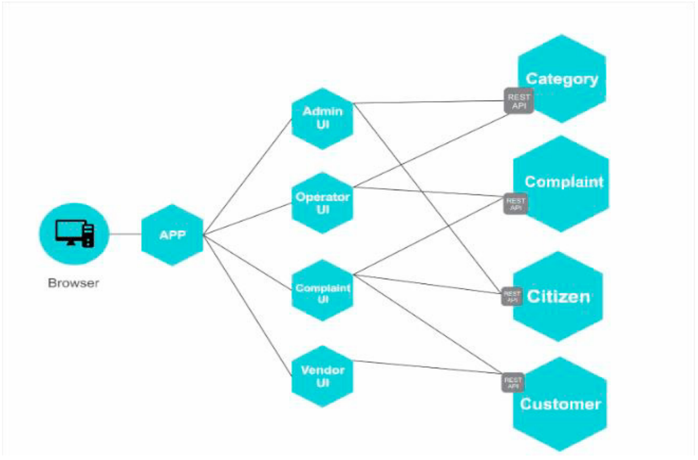

Fig. 3. Microservices achitecture.

6.5. Securing microservice

Fig. 1 depicts that administrator, government work unit, and the vendor have login functionality. The login itself is to be used to secure microservices for those actors. Those actors have to fill their username and password, and then the system will compare them to saved data in the database. If username and password match, the system will send a Jason web token (JWT). JSON Web Token (JWT) is a compact, URL-safe means of representing claims to be transferred between two parties which are the back-end and the front-end [14]. JWT contains some claims such as user scope that refers to the user actor and their dashboard, expired token time. So, when the time comes it will require a login process again and some additional information which is not of sensitive information that was required. The password itself need to be saved in database safely. Hashing function needs to be done so the password in the database is not a just plain text so it can be stolen easily. This research is using a BCrypt algorithm. BCrypt is an irreversible algorithm with low hash collision, using which our password is protected with proper protection [15]. The Java language provides bcrypt library, so this is a convenient way.

7. Conclusion

Based on the processes that have been done in this research, conclusions are as follows: 1) The public complaint is an application as a service developed with microservice architecture using springboot framework. Therefore, this application was deployed in the cloud. In deploying the application, a jar file was made from the back-end. The front-end was also deployed into the same cloud system as well as database applications used. To access the application, it takes a browser that will create communication to applications that are in the cloud. 2) In developing a microservice application, the functional needs should be broken down into several microservices that are interconnected to form a unified application by business processes that have been determined. 3) The advantages of using microservice architecture are that more functional needs can be added and also microservices within it without affecting other microservices. Thus, the developer can add more functionality independently, and that creates a business value which saves time to further development.

Acknowledgement

This research was supported/funded by “Penelitian Laboratorium PUPT ITS 2017” research grant, under contract number: 566/PKS/ITS/2017.

References

[1] D. A. Putra and N. Aminuddin, "Tactical Steps E-Government Development For Regional Government in Pringsewu Regency; Langkah – Langkah Taktis Pengembangan E-Government Untuk Pemerintahan Daerah (Pemda) Kabupaten Pringsewu," Technology Acceptance Model, p. 67, 2014.

[2] S. Sharma, "Mastering Microservices with Java," in Mastering Microservices with Java, Birmingham, PACKT publishing, 2016.

[3] S. Newman, "How to Model Services," in Building Microservices: Designing Fine-Grained Systems, O’Reilly Media, 2014.

[4] H. Purnama and I. Yatini, "Thesis Managemen Application in STMIK AKAKOM Yogyakarta Using Microservice Architecture With Node.Js," in Seminar Riset Teknologi Informasi (SRITI), 2016.

[5] Marijn Janssen and Anton Joha, "CHALLENGES FOR ADOPTING CLOUD-BASED SOFTWARE AS A SERVICE (SAAS) IN THE PUBLIC SECTOR," in European Conference on Information Systems, 2011.

[6] Electronic Data Management Department of Kediri City Government, "Application Development Document of E-Complaint Kediri City".

[7] Mr.Anil Kumar, Dr. Arvind Kumar and Mr. M. Iyyappana, "Applying Separation of Concern for Developing Softwares Using Aspect Programming Concepts," International Conference on Computation, pp. 906-9014, 2016.

[8] I. A. Pradana, "Mengenal Spring Boot," 3 February 2017. [Online]. Available: https://www.codepolitan.com/spring-boot-pengenalan-588da0c4bedd1. [Accessed 16 February 2017].

[9] M. Kumar, S. K. Singh and Dwivedi, "A Comparative Study of Black Box Testing and White Box Testing Techniques," International Journal of Advance Research in Computer Science and Management Studies, 2015.

[10] S. R. Jan, S. T. U. Shah, Z. U. Johar, Y. Shah and F. Khan, "An Innovative Approach to Investigate Various Software Testing Techniques and Strategies," IJSRSET, 2016.

[11] A. Orso and G. Rothermel, "Software testing: a research travelogue (2000–2014)," in Proceedings of the on Future of Software Engineering, New York, 2014.

[12] A. Nurmatari, "Aplikasi Seluruh SKPD akan Terintegrasi di Bandung Command Center," 7 December 2016. [Online]. Available: https://news.detik.com/berita-jawa-barat/d-3365344/aplikasi-seluruh-skpd-akan-terintegrasi-di-bandung-command-center. [Accessed 1 Maret 2017].

[13] Swagger, "Getting Started With Swagger," [Online]. Available: https://swagger.io/getting-started/. [Accessed 10 July 2017].

[14] M. B. Jones, "draft-ietf-oauth-json-web-token-32," [Online]. Available: https://tools.ietf.org/html/draft-ietf-oauth-json-web-token-32. [Accessed 26 Juli 2017].

[15] Yung-Feng Lu, Chin-Fu Kuo and Yi-Yen Fang, "Efficient Storage Encryption for Android Mobile Devices," in Proceedings of the International Conference on Research in Adaptive and Convergent System, Odense, 2016.