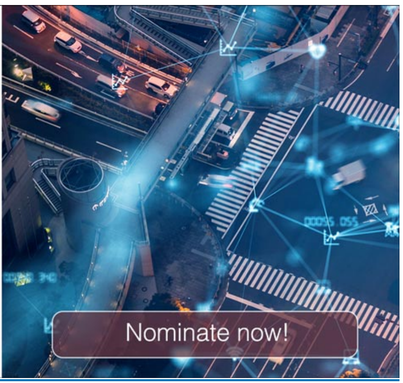
Microservice API Implementation For E-Government Service Interoperability



Microservice API Implementation For E-Government Service Interoperability

N Puspitasari*[1], E Budiman[2], Y N Sulaiman[3], M B Firdaus[4]

[1,2,3,4]Department of Informatics Engineering, Faculty of Computer Science & Information Technology, Mulawarman University, Indonesia

*Coressponding author: novia.ftik.unmul@gmail.com, edy.budiman@fkti.unmul.ac.id

Abstract. To improve e-government services released by Communication and Information Technology Office of Samarinda City, each system needs to be able to interoperate even when developed by different developers. Interoperation can be achieved by using one data source which is API (Application Programming Interface) for general data objects such as announcements. Given this condition, API built by using microservice can support further enhancement even if API is developed by developers who use different programming languages. The result shows that microservice API can be used to interoperate in relaying data between e-government services and can be developed using more than one programming language and base codes. Further development of this API can be done by adding more data objects, using AWS Cognito as authorization management, adding AWS Elasticsearch to load and filter data, and by showing data objects in real-time on the front end.
Keyword: architecture microservice, API, interoperability, smart city.

1. Introduction

Announcement is a notice given to the public about information in the written or verbal form. The purpose of the announcement is to convey information to be known by the public [1]. Announcements can be delivered through a website page. This is in line with smart city action which is a city that has integrated information and communication technology in day-to-day activity, with the purpose of improving effectivity, fixing public services, and improving the welfare of citizens [2, 3]. To improve the public services, in the form of e-government service, each application in e-government service from Communication and Information Technology Office of Samarinda City needs to interoperate with each other. The ability to interoperate is called interoperability, technically defined by IEEE Standard Computer Dictionary as the ability of two or more system to interchange data or information and ability to use said data and information [4]. Interoperability may be achieved by using varied hardware and software both the operating system, database, and programming language used, especially ones used in government instances [5, 6]. Interoperability is achieved using data exchange format standardization. Each related party has to use the defined standard as a joint reference.

Interoperability in relaying data objects, in this case, the announcement as a data object, can be supported by using API (Application Programming Interface). An Application Programming Interface (API) is a particular set of rules and specifications that a software program can follow to access and make use of the services and resources provided by another particular software program that implements that API [7, 8]. It serves as an interface between different software programs and facilitates their interaction, similar to the way the user interface facilitates interaction between humans and computers. Microservice architecture for the API was chosen to facilitate API development because microservice is a software architecture to develop an application formed by layers of small services and could be developed by using multiple programming languages (polyglot) therefore allowing development by multiple developers using different languages. This condition was needed in case of developer changes happening in Communication and Information Technology Office of Samarinda City.

Therefore, a microservice API was built so data could be relayed between e-government services released by Communication and Information Technology Office of Samarinda City and services may interoperate and API still able to be developed even if the developers were different or used different programming language.

2. Literature Review

2.1. Interoperability

Interoperability technically defines the ability of two or more systems to exchange data or information and ability to use said exchanged information. Interoperability may be achieved by using varied hardware and software be operating system, database, and programming language used, especially ones used in government instances. Interoperability is achieved using data exchange format standardization. Each related party has to use defined standard as a joint reference [4].

2.2. Application Programming Interface

An Application Programming Interface (API) is a particular set of rules and specifications that a software program can follow to access and make use of the services and resources provided by another particular software program that implements that API. It serves as an interface between different software programs and facilitates their interaction, similar to the way the user interface facilitates interaction between humans and computers [7, 8].

2.3. Microservice

Microservice, also known as microservice architecture is an architectural style that structures an application as a collection of services that are highly maintainable and testable, loosely coupled, independently deployable, and organized around business capabilities [9-11]. There are a few differences between microservice and monolith architectures, as explained in Table 1 [12].

Table 1. Comparing Monolith and Microservice Architecture

Category	Monolith Architecture	Microservice Architecture
Code	One base code for entire application	Multiple base codes. Each microservice has its own base code.
Understandability	Confusing and hard to maintain	Much better readability and much easier to maintain.
Deployment	Complex deployments with maintenance windows and schedules downtimes.	Complex deployments with maintenance windows and schedules downtimes.
Language	Typically, entirely developed in one programing language.	Each microservice can be developed in a different programing language.
Scaling	Requires you to scale the entire application even though bottlenecks are localized	Enables you to scale bottlenecked services without scaling the entire application.

3. Old and New System Comparison

On the existing system, all systems were built monolith—each system had its base codes and databases [13]. If there were a new announcement that had to be spread in multiple systems, data had to be manually inputted in each system. By implementing microservice API for e-government, announcement data was directed to and from one system by API. Microservice architecture is a Function as a Service. Therefore, a service is divided into each of their functions. The announcement module as a service was separated by each of their functions, resulting in add announcement, edit announcement, delete announcement, list announcement, and detail announcement. These functions were modelled with use case diagram, making announcement service shown in Figure 1.

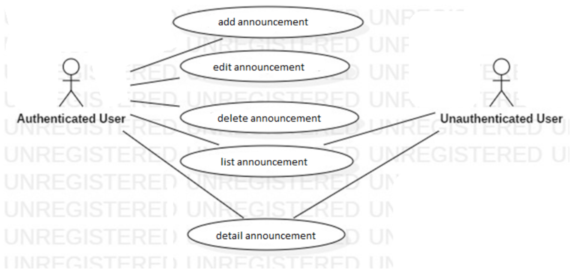


Figure 1. Use Case Diagram

Microservice supports system development with multiple languages and base codes. Microservice API implementation for e-government service interoperability simulated conditions where multiple developers were using different programming languages, simulated by using Node.js [14] and Go. Functions and language used were detailed as follows: a) Add, edit and delete functions were developed using Node.js and may be accessed by an authenticated user with API key; b) List and detail functions were developed using Go and may be accessed by the unauthenticated user without an API key.

Furthermore, throwaway prototypes are used for the implementation of the microservice API. Throwaway prototyping is a prototyping method to demonstrate ability or interface simulations but not to be used as a final version. The prototyping method can decrease project risks [15, 16]. Throwaway prototyping can be done by planning and pre-analyzing requirements. Continued by analyzing, designing, and implementing the program into dummy prototype, and iteration based on requirement changes and ended by implementing the final version. In this study, testing was done by white-box testing, which is a way to test the external functionality of code by testing code, code structure, and its internal design flow, in the form of unit testing. After white box testing was done, each API endpoint generated was tested with Postman. Those endpoints are [POST] /pengumuman, to add announcement, [PUT] /pengumuman/{id} to edit announcement, [DELETE] /pengumuman/{id} to delete announcement, [GET] /pengumuman to list announcements, and [GET] /pengumuman/{id} to get announcement detail. To test the interoperability, endpoints above were tested by integrating them into prototypes. The system interoperability testing scheme is shown in Figure 2.

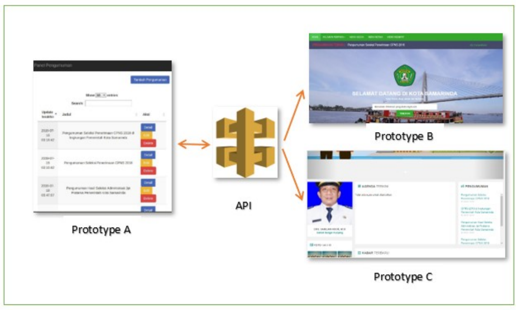


Figure 2. System Interoperability Scheme

The scheme in Figure 2. showed the flow to test system interoperability. The system can be called interoperating if each system used the same data source, which can be accessed through API. Prototype A as an authenticated user was designed as an admin page to add, edit, and delete announcements that would be shown on prototype B and C. Prototype B and C were set as unauthenticated users, designed to simulate the view of existing e-government services released by Communication and Information Technology Office of Samarinda City, used as models to show that each systems are sharing the same resource, which was the API. Prototype B was built to look alike Samarinda City's website (https://samarindakota.go.id/website) and prototype C was built similar to Sungai Kunjang District's website (https://kec-sungai-kunjang.samarindakota.go.id/) to simulate API usage on both websites.

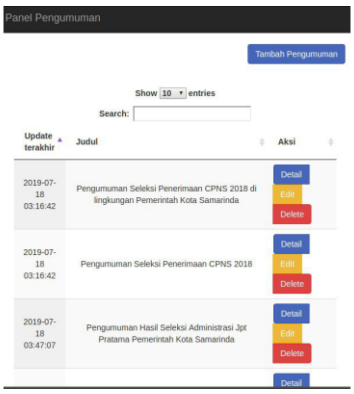
4. Result and Discussion

Creating a throwaway prototype to implement microservice API was started by analyzing requirements with the announcement as data object. Next, a system was designed to simulate a condition where there was more than one developer using different programming languages, which was done by using Node.js dan Go. Implementation was done by creating code base or serverless repository with each language used. After the implementation process, white box testing in the form of unit testing was done to test the internals of the code, with test items and results in Table 2.

Table 2. White-box Test Result

Test Class	Test Case	Result
System is able to add announcement.	Add announcement fails if input header does not have API key	Valid
	Add announcement fails if title is empty or has less than 3 characters.	Valid
	Add announcement succeeds if input header has API key, and title is more than 3 characters	Valid
System is able to edit announcement	Edit announcement fails if input header does not have API key	Valid
	Edit announcement fails if Id is empty	Valid
	Edit announcement fails if Id does not exist in database	Valid
	Edit announcement succeeds if input header has API key, Id is not empty and exists in database	Valid
System is able to delete announcement	Delete announcement fails if input header does not have API key	Valid
	Delete announcement fails if Id is empty	Valid
	Delete announcement fails if Id does not exist in database	Valid
	Delete announcement succeeds if input header has API key, Id is not empty and exists in database	Valid
System is able to give announcement list	Lists announcements from database.	Valid
System is able to give announcement detail	Detail announcement fails if Id is empty	Valid
	Detail announcement fails if Id does not exist in database	Valid
	Detail announcement succeeds if Id is not empty and exists in database	Valid

After coding and unit testing is done, each base code repositories were deployed to AWS separately by using serverless deploy command that would deploy each repository to AWS based on each repository's settings. Deployed functions can be viewed in AWS Lambda. Endpoints for each function can be viewed on AWS API Gateway. Each endpoint was tested using Postman to check whether API relayed data correctly. The test resulted in all endpoints are working properly. These endpoints were implemented on prototype A, B, and C, which were simulations of e-government systems. Results showed Prototype A as an authenticated user were able to add, edit, and remove announcement through API. Prototype A shown in Figure 3.



Update terakhir	Judul	Aksi
2019-07-18 03:16:42	Pengumuman Seleksi Penerimaan CPNS 2018 di lingkungan Pemerintah Kota Samarinda	Detail Edit Delete
2019-07-18 03:16:42	Pengumuman Seleksi Penerimaan CPNS 2018	Detail Edit Delete
2019-07-18 03:47:07	Pengumuman Hasil Seleksi Administrasi Jpt Pratama Pemerintah Kota Samarinda	Detail Edit Delete
		Detail

Figure 3. Prototype A

Data from prototype A as an administrator was able to be relayed through prototype B and C without having to be manually inputted in prototype B and C. Prototype B, built similar to Samarinda City's website, was able to display latest announcement retrieved from API as shown in Figure 4. Furthermore, Prototype B, built similar to Samarinda City's website, was also able to display announcement detail retrieved from API as seen on Figure 5.

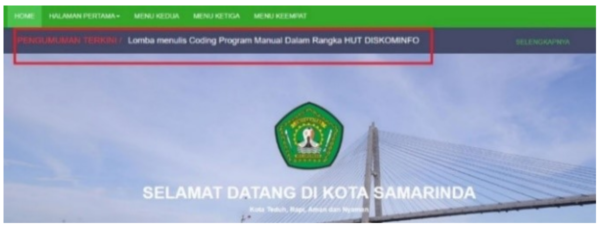
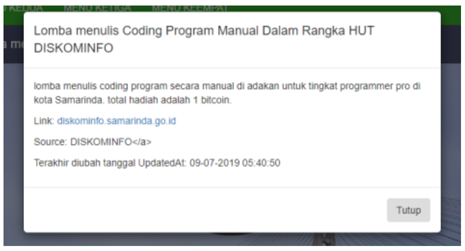

HOME
HALAMAN PERTAMA
MENU KEDUA
MENU KETIGA
MENU KEEMPAT
PENGUMUMAN TERKINI / Lomba menulis Coding Program Manual Dalam Rangka HUT DISKOMINFO
SELENGKAPNYA
SELAMAT DATANG DI KOTA SAMARINDA
Kota Teduh, Rapi, Aman dan Nyaman

Figure 4. Prototype B


Lomba menulis Coding Program Manual Dalam Rangka HUT DISKOMINFO

lomba menulis coding program secara manual di adakan untuk tingkat programmer pro di kota Samarinda. total hadiah adalah 1 bitcoin.

Link: diskominfo.samarinda.go.id

Source: DISKOMINFO</a>

Terakhir diubah tanggal UpdatedAt: 09-07-2019 05:40:50

Tutup

Figure 5. Announcement Detail on Prototype B

Prototype C, built similar to Sungai Kunjang District's website, was able to display list of announcement retrieved from API seen in Figure 6. Furthermore, Figure 7. shows Prototype C built similar to Sungai Kunjang District's website was also able to show announcement detail retrieved from API.

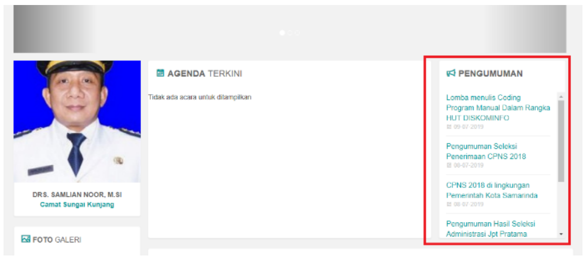
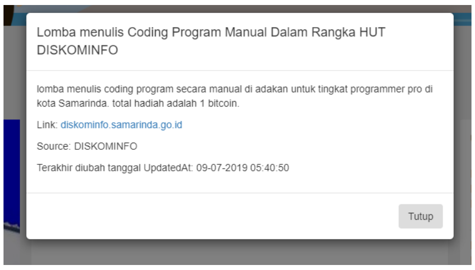

DRS. SAMLIAN NOOR, M.SI
Camat Sungai Kunjang
FOTO GALERI
AGENDA TERKINI
Tidak ada acara untuk ditampilkan
PENGUMUMAN
Lomba menulis Coding Program Manual Dalam Rangka HUT DISKOMINFO
09-07-2019
Pengumuman Seleksi Penerimaan CPNS 2018
08-07-2019
CPNS 2018 di lingkungan Pemerintah Kota Samarinda
08-07-2019
Pengumuman Hasil Seleksi Administrasi Jpt Pratama

Figure 6. Prototype C


Lomba menulis Coding Program Manual Dalam Rangka HUT DISKOMINFO

lomba menulis coding program secara manual di adakan untuk tingkat programmer pro di kota Samarinda. total hadiah adalah 1 bitcoin.

Link: diskominfo.samarinda.go.id

Source: DISKOMINFO

Terakhir diubah tanggal UpdatedAt: 09-07-2019 05:40:50

Tutup

Figure 7. Announcement Detail on Prototype C

Prototype C was also able to show announcement list and detail retrieved from API, consistent with the data source managed in Prototype A. This usage of a single data source is called interoperability. Prototype B and C as unauthenticated users were able to access the announcement list and detail through the API. It can be said that prototype A, B, and C as the e-government system had simulated interoperation for using the same data source. This implementation is a dummy prototype. If there is any iteration done in the future, each base code can be updated and deployed without affecting other base codes.

5. Conclusion
Based on the results, it can be concluded that interoperability in relaying announcement data between e-government system applications can be done by using microservice API. Microservice API can be developed using more than one programming language and more than one base code, therefore suitable for conditions where there is more than one developer using different programming languages. Future research can be done by adding more service for data objects required by multiple systems such as news, events, and personnel information. API authorization can be improved by using AWS Cognito. AWS Elasticsearch can also be added to enhance data searching and filtering. Data from API can also be shown by real-time.

References
[1] A. Baltag, L. S. Moss, and S. Solecki, "The logic of public announcements, common knowledge, and private suspicions," in *Readings in Formal Epistemology*: Springer, 2016, pp. 773-812.

[2] A. Meijer and M. P. R. Bolívar, "Governing the smart city: a review of the literature on smart urban governance," *international review of administrative sciences,* vol. 82, no. 2, pp. 392-408, 2016.
[3] E. Park, A. del Pobil, and S. Kwon, "The role of internet of things (IoT) in smart cities: Technology roadmap-oriented approaches," *Sustainability,* vol. 10, no. 5, p. 1388, 2018.
[4] M. Janssen, E. Estevez, and T. Janowski, "Interoperability in big, open, and linked data-organizational maturity, capabilities, and data portfolios," *IEEE Computer,* vol. 47, no. 10, pp. 44-49, 2014.
[5] H. J. Schnoll, *E-Government: Information, Technology, and Transformation: Information, Technology, and Transformation*. Routledge, 2015.
[6] T. Lodato, E. French, and J. Clark, "Open government data in the smart city: Interoperability, urban knowledge, and linking legacy systems," *Journal of Urban Affairs,* pp. 1-15, 2018.
[7] S. P. Ong *et al.*, "The Materials Application Programming Interface (API): A simple, flexible and efficient API for materials data based on REpresentational State Transfer (REST) principles," *Computational Materials Science,* vol. 97, pp. 209-215, 2015.
[8] M. A. Boillot, "Application programming interface (API) for sensory events," ed: Google Patents, 2012.
[9] I. Nadareishvili, R. Mitra, M. McLarty, and M. Amundsen, *Microservice architecture: aligning principles, practices, and culture*. " O'Reilly Media, Inc.", 2016.
[10] A. Sill, "The design and architecture of microservices," *IEEE Cloud Computing,* vol. 3, no. 5, pp. 76-80, 2016.
[11] Y. Yu, H. Silveira, and M. Sundaram, "A microservice based reference architecture model in the context of enterprise architecture," in *2016 IEEE Advanced Information Management, Communicates, Electronic and Automation Control Conference (IMCEC)*, 2016: IEEE, pp. 1856-1860.
[12] W. H. C. Almeida, L. de Aguiar Monteiro, R. R. Hazin, A. C. de Lima, and F. S. Ferraz, "Survey on Microservice Architecture-Security, Privacy and Standardization on Cloud Computing Environment," *ICSEA 2017,* p. 210, 2017.
[13] A. Bucchiarone, N. Dragoni, S. Dustdar, S. T. Larsen, and M. Mazzara, "From monolithic to microservices: An experience report from the banking domain," *Ieee Software,* vol. 35, no. 3, pp. 50-55, 2018.
[14] M. Villamizar *et al.*, "Infrastructure cost comparison of running web applications in the cloud using AWS lambda and monolithic and microservice architectures," in *2016 16th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid)*, 2016: IEEE, pp. 179-182.
[15] A. M. Davis, "Operational prototyping: A new development approach," *IEEE software,* vol. 9, no. 5, pp. 70-78, 1992.
[16] H.-M. Chen, R. Kazman, and S. Haziyev, "Strategic prototyping for developing big data systems," *IEEE Software,* vol. 33, no. 2, pp. 36-43, 2016.