

# The multi-tenancy queueing system “QuAntri” for public service mall

Wiwin Suwarningsih, Ana Heryana, Dianadewi Riswantini, Ekasari Nugraheni, Dikdik Krisnandi

Research Center for Information and Data Sciences, National Research and Innovation Agency, Bandung, Indonesia

## Article Info

### Article history:

Received Jul 30, 2021

Revised Jul 1, 2022

Accepted Jul 5, 2022

### Keywords:

COVID-19

Multi-tenancy system

New normal

Public service

Queueing system

## ABSTRACT

In the new-normal era, public services must make various adjustments to keep the community safe during the COVID-19 pandemic. The Public Service Mall is an initiative to put several public services offices in a centralized location. However, it will create a crowd of people who want access to public service. This paper evaluates multi-tenant models with the rapid adaptation of cloud computing technology for all organizations' shapes and sizes, focusing on multi-tenants and multi-services, where each tenant might have multiple services to offer. We also proposed a multi-tenant architecture that can serve queues in several places to prevent the spread of COVID-19 due to the crowd of people in public places. The design of multi-tenants and multi-services applications should consider various aspects such as security, database, data communication, and user interface. We designed and built the "QuAntri" business logic to simplify the process for multi-services in each tenant. The developed system is expected to improve tenants' performance and reduce the crowd in the public service. We compared our agile method for system development with some of the previous multi-tenant architectures. Our experiments showed that our method overall is better than the referenced model-view-controller (MVC), model-view-presenter (MVP), and model-model-view-presenter (M-MVP).

This is an open access article under the [CC BY-SA](#) license.



## Corresponding Author:

Wiwin Suwarningsih

Research Center for Informations and Data Science, National Research and Innovation Agency

St. Sangkuriang 21 Bandung 40135, Indonesia

Email: wiwin.suwarningsih@brin.go.id

## 1. INTRODUCTION

The emergence of the COVID-19 pandemic shocked and significantly impacted the world in almost all aspects of life. Almost nothing can escape the impact of this pandemic. The government issued several policies to prevent the spread of disease, including by maintaining a social and physical distance. Activities that involve large numbers of people should be avoided for the time being, including activities that the government gives to the community. The current restrictions reduce the number of queues entering the room, and services must follow the recommended safe distance of at least 1 meter.

The problems that arise in this service are how to handle the queueing system to prevent the crowd, how to manage queue arrivals tailored to the type of service provided and how to manage the service process so that it is not complicated by implementing the standardization of public services. The emergence of various new service procedures that have not been regulated in public service standards during this pandemic affects the quality of services provided. However, the community is still entitled to good public services and they have a good role in public services carried out by public service providers. Currently, public services in Indonesia include online services such as online passports, e-Samsat (vehicle tax), e-billing, online welfare, online employment, and online tax ID number (TIN) management.

Several works showed that online service implementation could utilize the on-premise infrastructure or cloud computing infrastructure. Cloud computing infrastructure can provide software as a service [1], [2] where various services software for multiple customers are provided using virtualization [3]. The system was intended to share cloud resources and achieve a high level of operational efficiency. The main reason for prioritizing efficiency is to get better use of infrastructure resources [4], [5] with a mutual benefit [6], [7] that the system uses a single operating system, single database, and optimizes the resources. When one user leaves, other users can use the same resource [1], so one application serves many customers, regardless of their requirements. However, operational efficiency that needs to be considered is low operational costs. As research conducted by [8], [9], the implementation of multi-tenancy in the cloud with not only structured query language (NoSQL) data storage allowed more effective resource sharing among tenants, therefore lowering the operational costs [10]. Another way to reduce operational costs is by implementing a single database on a multi-tenant software-as-a-service (SaaS) model to serve multiple e-commerce such as online stores [11] and business support [7].

Many efforts have been made to improve customer service using multi-tenancy. The optimization of application, server, and database resources is to produce the ideal multi-tenant model. The widely applied multi-tenant model is focused on the concept of SaaS [1], [9], [11], [12], the design and implementation of multi-tenant database schemas, such as private tables, extension tables, universal tables, distributed databases and database replication [12]-[14], virtual machines [3], network security [10], [15], multi-tenant monitoring [6], [16], e-commerce [1], [11], [17], and Blockchain-based services [2], [18].

Architectural principles commonly used in cloud computing applications include multi-tenant, multi-user and multi-instance, where these applications allow at one time to serve more than one user with various tasks [19], [20]. Multi-tenant is an architectural principle that can be stated to be relatively new [8]. Unlike multi-users and multi-instances, multi-tenants are able to share one application for several tenants with different locations. Abdul *et al.* [21] they can provide great opportunities to harness the power of cloud infrastructure enabling the tenants to increase the consolidation of data layer operational entities to be dedicated, isolated, and shared. The use of infrastructure can also be done in other ways by applying a multi-tenant management pattern [22], [23] such as the model-view-controller (MVC), model-view-presenter (MVP), and model-view-view-model (MVVM). Each pattern includes some different aspects of the tenant management systematics.

The components used in MVC consist of model, view and controller. The model component functions to determine the type of data and business logic [23]. How to retrieve and manipulate data, communicate with controllers, interact with databases, and occasionally update views. The advantages of MVC according to [24], [25] include the separated business logic from the model, supported asynchronous techniques, and no impacts of modifications made in one component on other components [26]. In the application development process using MVP, developers must take the advantage of the available time to sort the architecture into layers [27], [28]. The layer will share the dependencies that are commonly present in the view. The components used in MVC consist of model, view and presenter. MVVM architecture loosens relationships between components. It also works with the concept of observable [29] from one component to another not directly connected but through the reference contained in the observable. Its components consist of model, view and view model. The model component contains a code for the business logic and repositories. The repository's task is to bridge the requests from a view-model to a local or remote data source. View components have no business logic, but user-linked views extensible markup language (XML) only.

In the midst of this COVID-19 pandemic, we attempted to promote the public service mall (PSM) implementation to create new services that enhance a positive image and responsive government performance. Improving the quality of public services in the regions through the development of community participation is still one of the priorities for the expected and sustainable regional development. The development of mobile applications for queuing systems in public service places such as sub-district offices, health centers, or banks is needed. By using this application, one can register the queue from home and come to the location right before the queue number gets service. This will really help to reduce the crowds in public places. The main contributions of this paper are summarized as follows: i) to enable the multi-tenants, as our target, to serve queues for several places in Bandung; ii) to build the QuAntri business logic that simplifies the process for multi-services in each tenant; and iii) to apply this very suitable system to support the prevention of the spread of COVID-19 due to the accumulation of people in accessing public services.

The remainder of this paper is structured as follows: section 2 describes our proposed method. This is followed by section 3, which presents the research method. Section 4 and section 5 respectively describe and present the result and discussion and conclusion.

## 2. PROPOSED METHOD

Multi-tenancy is a cloud architecture that allows each cloud service tenant to share computing power with other tenants [30], [31]. Tenants can be referred to as users or parties who rent a cloud service [32]. Cloud providers have enormous resources and each tenant will use the same resource. This system uses an Android-based smartphone device that bridges users to interact with the system [16]. A multi-tenancy queue system application will be installed on the smartphone device through the Google Play Store facility. Figure 1 illustrates the multi-tenant architecture.

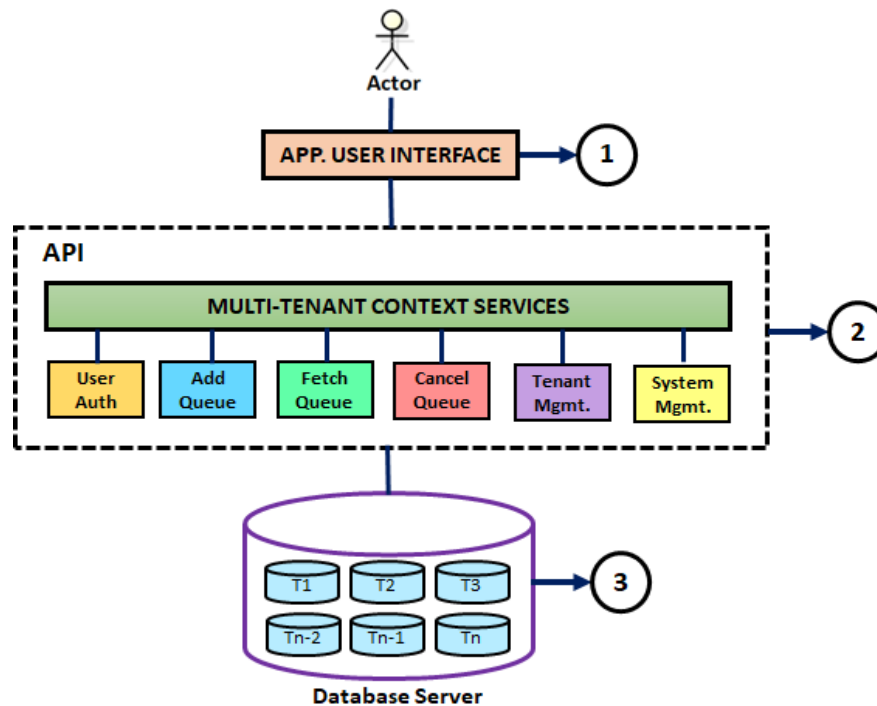


Figure 1. QuAntri multi-tenant SaaS architecture

As seen in Figure 1, the service architecture developed in QuAntri is presented as follows: i) actor through the user interface application (1) will send a to the application programming interface (API) server to request or save data (2); ii) the API is tasked as a bridge for interaction or communication between the user interface application and the database (3); iii) each tenant (T) has its own database, which is managed by the tenant administrator. Requests from users through the application interface will be directed to services that access the specified tenant; iv) on the API server, there is a multi-tenant context services (MTCS) layer, which functions as a gateway or route to access various services on the system; v) every service request from the actor via (1) will be verified and validated by the user authorized service, if it is valid, then it can be continued to other services according to the actor's request; vi) the main service in this API lies in queue handling, which consists of add queue to add queue to the service; fetch queue to retrieve queue data; and cancel queue to cancel the service; and vii) in addition, the service provides a tenant management system.

The QuAntri system is designed for several actors with their respective roles. The actors in QuAntri are: i) QuAntri admin, who acts as a verifier for registered users and tenant master; ii) tenant master can register tenants and tenant officers as well as manages and monitors tenants (services and queues); iii) tenant officer can manage and monitor queues at each tenant service; and iv) users, get tenant services after queuing. The actors and their relationships in the QuAntri system are explained using a use case diagram as shown in Figure 2.

The activities of the actors and their flow during the life cycle of the QuAntri system are master tenants verified by the QuAntri admin can create tenants and tenant officers. The master tenant must set services and queues for each tenant (service name, start, and end hours, service duration and total queues per day), before the tenant is published and can be accessed by users. Users verified by QuAntri admin can request service queues on published tenants. Tenant officers manage service queues by receiving service queue requests from users. The service queuing system for the user will be terminated when the user gets the service.

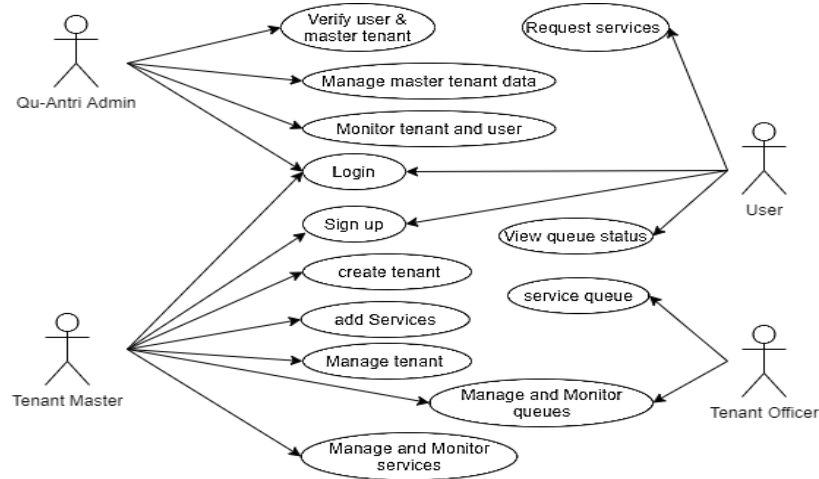


Figure 2. QuAntri use case diagram

### 3. RESEARCH METHOD

QuAntri system is a single software architectural pattern that runs on a server infrastructure in which multiple tenants can access the application. This system requires customization of one instance according to the multi-faceted requirements of multiple tenants where each tenant gets its own application instance. QuAntri provides unlimited services from tenants with an unlimited number of tenants.

The proposed system model was only limited by the server to handle the database. Tenants who register will be verified by the QuAntri admin to prevent fake tenant registrations. Only verified tenants can access and use the features in the QuAntri system. QuAntri as a service is an alternative mode of software distribution in which third parties typically host applications in the cloud. Then, it is made accessible to users via the internet. In fact, QuAntri is a form of SaaS, which is one of the three main wings of cloud computing. One of the main reasons why QuAntri excels in the application development industry is because it offers an efficient architecture and helps to adopt a cost-cutting methodology.

#### 3.1. Business logic layer

The business logic layer is basically to create classes that call methods from the data access layer (DAL) [33]. DAL creates classes and methods that serve to communicate with the database directly. The purpose of making the building logic in QuAntri is to facilitate communication with the database in which it can be done repeatedly. Another purpose of making the business logic is to prevent the presentation level from directly accessing the database to compromise data security. In its implementation, the QuAntri business logic defines classes and methods in the class and performs functions within the method. These functions aim to manipulate data from the DAL method call to be displayed on the presentation layer or vice versa, and the business logic layer receives data from the presentation level and manipulates it to be sent to the database. The features proposed in this business logic layer are to help to cut investment costs for an organization, ease in adding new customers, convenience in using the same application, maximizing resource use, and having multiple tenants in the same time frame. An overview of these features is illustrated in the explanation.

- a. Reduce investment costs, the multi-tenant architecture could reduce the long-term investment costs by using shared available resources. Such as where tenants share the same database and application on scalable server nodes. Thus, acquiring new tenants does not require redeploying the whole service.
- b. Easy to add new customers and new services, new customers can do self-registration even though they need to be verified by the QuAntri administrator. The configurable services allow the customers to specify their services or add new services.
- c. Simple application for all tenants, the application is designed by configuration first in mind. Database and application core configuration were hidden from the customer, while the customers (tenants) can specify their services using tuned configurations.
- d. Efficient resource utilization, by using shared resources, the application can improve resource utilization efficiency and automated resource maintenance.
- e. Multiple tenants in the same period, the multi tenancy architecture does not require each tenant to have an exclusive database and application. This allows single infrastructure to be used by multiple tenants at the same time.

### 3.2. QU-antri security model

QuAntri is a model of cloud services with high-risk types of services and it is vulnerable to attacks from hackers. Many ways can be done to reduce security risks. The QuAntri security system implements a cloud access security algorithm to audit their network for unauthorized cloud services and compromised accounts in which the stages include i) the implementation of the identity and access management; ii) the implementation of data loss prevention, and iii) cloud data encryption. Access management system ensures that end users do not get access to the more resources they need for their jobs. This solution can be done by enabling user access to determine what files and applications a particular user can access. For this, the end-users will only see data they are allowed to see.

User detection software is installed to prevent any sensitive data from being downloaded to private devices and to block any malware or hackers from trying to access and download data. The implementation of data loss prevention can be done. The collaborative monitoring of data sharing is carried out on the QuAntri system so that it is possible for the detection of detailed permissions on files shared with other users, including users outside the organization accessing files via web links. QuAntri also implements data encryption to protect data in storage and data in transit between end users and the cloud or between cloud applications. API testing on the QuAntri service system aims to validate several aspects related to the business logic of the system. API testing is done for functionality, performance, and security vulnerability. The API testing process is carried out from the user interface, and third-party software API testing.

## 4. RESULTS AND DISCUSSION

This section describes the implementation results of the proposed method for QuAntri by comparing the number of detected errors including multi-tenant test scenarios for public service malls. The purpose of this test was to examine 3 multitenant components: applications, infrastructure and networks, all of which are the main components of a multi-tenant program. These are usually thoroughly tested at launch with an update of each component. The focus was on checking the configurable and non-configurable components. Many scenarios were designed in which several tenants were brought in to check basic functionality. Every change occurred was recorded among tenants, server admins and users. In addition, application functionality was checked, depending on the number of tenants and users accessing it. Security tests were also required for QuAntri as multi-tenant applications typically are intended for an isolated use within a specific group. Thus, protocols for permissions, malware and unauthorized access should be checked.

### 4.1. Testing scenarios

The API test scenarios used are:

- a. Functionality test, functional testing is the first piece of API readiness. Functionality testing is done to find out the developed API has met the system requirements. API testing uses a realistic approach, one of which is by applying positive/negative testing.
- b. Security vulnerability test, security vulnerability testing on the API is intended to find out the security mechanisms that have been made on the API. The test methods used are SQL injection attacks, cross-site scripting and token and sessions.
- c. Performance test, testing the performance of the API is intended to find out whether the API responds to all client requests. Furthermore, performance testing also measures the response time of the system to a request. The tests carried out ignore the environment and hardware specifications used by the system.

### 4.2. Result

#### 4.2.1. Results of application improvement

The results of tests carried out on the QuAntri SaaS included data observation and database access [34]. Data observation was done in a way when several users performed the same action simultaneously to send. The data flow from each user was observed and monitored to have the flow as expected. Problems that may arise were when the application was reading to the database at the same time. This was due to the data access session on one of the tenants to the expression database. For this, the handling was done by waiting for a while by imposing a priority scale based on the value of the distance and the smaller access load.

Another test result was when all tenants at the same time retrieve data from the server. Since the resources were shared, it was possible for tenant-1 to receive data from tenant-2. If this occurred, checked were made on the security system to prevent data integrity from being lost. Due to the writing of too many database calls, the system may experience performance degradation. Therefore, it is important to guarantee the data call process to the server so that all the information required is tailored to the needs of each tenant. Eventually, the system is able to complete tasks as intended.

The next result is the validation process for users and tenants who requested access rights. Validation was carried out to ensure the uniqueness of the objects used across all tenants/users. This was

done so that the tenant cannot modify or delete the object. In short, data isolation was clearly defined at all levels.

#### 4.2.2. Result of network data load

The performance of the QuAntri network system has been tested by making the system to be overloaded. This should be done to assist in identifying a bottleneck situation in the use of shared applications for use by several users/tenants. This stage has been successfully carried out and it could be proven that validating system stability with overload did not result in data leakage and the system remained stable.

This endurance test was performed to enable the system to validate resource utilization and response time. Response time was measured to determine system metrics tested and system readiness. This was done for testing and load balancers used in certain environments. This durability testing was also carried out to check the reliability of the software. The reason for thorough testing was because when the number of tenants was increased, then the risk was greater. Therefore, the QuAntri system can be tuned for a better experience and optimization of resources based on these test results.

#### 4.2.3. Result of security testing

Security testing is another component that must be performed as it plays a key role due to the nature of the system. In the QuAntri system, the main emphasis was on the SQL injection process. This was done to see how vulnerable the process was because if the database server/database/table was shared, it would be easier for one tenant/external user to inject queries in the application and exploit other tenants of data.

This test was carried out by doing some validations at the user interface (UI) level so that one of the tenants could try to see the data of other tenants without disturbing the usability of the system. Performing these tests provided sufficient confidence to track emerging feature/functional failures. Data flow and test automation at each node to check for data leaks could be covered and detected early. The summary of the test and realization targets can be seen in Table 1.

Table 1. QuAntri test checklist

Test criteria	Target of achievement (%)	Realized achievement (%)
Functional/feature test: to test the availability of features by users and/or tenants.	95	90
Performance test: to test the response time of the system to a request	96	91
Security vulnerability test: emphasis on SQL injection due to the nature of the system.	100	100

As seen in Table 1, a complete series of tests were carried out prior to delivering the multi-tenant QuAntri system to the relevant agencies. This was done because every failure in production would have a significant impact in terms of reputation and costs because of the involvement of many tenants. On the other hand, the maintenance of the QuAntri system is simple if it is done by defining/following the process according to the test checklist.

### 4.3. Discussion

This multi-tenant server scalability test is to analyze overall performance if one or more loading factors are added. The loading factors themselves generally include the number of users, the amount of data being managed by the application, and the number of transactions. Performance can be seen from the amount of load and response time given by the application. The amount of load is measured by the amount of work that can be done by an application within a predetermined time limit. In addition, the response time is the time interval required between a user in requesting the process and receiving the requested result.

The main concern regarding this multi-tenant is the issue of reliability, that is whether this system is able to provide computing needs as needed and provide stability. Sharing multiple processes together, sending and receiving files together are some of the risks of a cloud computing server. Because it is in one physical server made into several virtual servers. By its own needs, the server becomes a central computer, which makes it must operate more than ordinary computers. As a result, the server has more special hardware specifications. The benefit to be gained from this research is to provide a solution to a virtual public service system that requires many tenants to experiment or support many users who want to get various services at one door. Providing solutions to the government/organizations, each of which requires its own tenants to produce optimal public services.

## 5. CONCLUSION

QuAntri multi-tenant architecture was designed in such a way in which each tenant can share databases, tables, functionality/features, and some non-functional items. The main risks involved in this type of architecture are security vulnerability and the level of system isolation. In this paper, we defined the concept of multi-tenancy from a developer perspective for multiple tenants and multi-users. We have presented a scheme based on the definitions of the queuing system requirements for this pandemic so that it could be used to prevent crowds. In various aspects of being an agile system, we have conducted several tests in order to produce an ideal multi-tenant system.

## ACKNOWLEDGMENT

The author would like to thank all the facilities provided by the National Research and Innovation Agency (BRIN) particularly Research Center for Information and Data Sciences during the process of making this manuscript.




## REFERENCES

- [1] H. Cai, N. Wang, and M. J. Zhou, "A transparent approach of enabling SaaS multi-tenancy in the cloud," *2010 6<sup>th</sup> World Congress on Services*, 2010, pp. 40–47, doi: 10.1109/SERVICES.2010.48.
- [2] I. Weber, Q. Lu, A. B. Tran, A. Deshmukh, M. Gorski, and M. Strazds, "A platform architecture for multi-tenant blockchain-based systems," *2019 IEEE International Conference on Software Architecture (ICSA)*, 2019, pp. 101–110, doi: 10.1109/ICSA.2019.00019.
- [3] Z. Wang, M. M. Hayat, N. Ghani, and K. B. Shaban, "A probabilistic multi-tenant model for virtual machine mapping in cloud systems," *2014 IEEE 3rd International Conference on Cloud Networking (CloudNet)*, Oct. 2014, pp. 339–343, doi: 10.1109/CloudNet.2014.6969018.
- [4] W. Tsai and P. Zhong, "Multi-tenancy and sub-tenancy architecture in software-as-a-service (SaaS)," *2014 IEEE 8<sup>th</sup> International Symposium on Service Oriented System Engineering*, 2014, pp. 128–139, doi: 10.1109/SOSE.2014.20.
- [5] L. Foschini, G. Martuscelli, and R. Montanari, "Simplifying multi-layer and multi-tenant support in openstack: The SACHER use case," *2019 IEEE Symposium on Computers and Communications (ISCC)*, 2019, pp. 1183–1188, doi: 10.1109/ISCC47284.2019.8969678.
- [6] H. Lokawati and Y. Widayani, "Monitoring system of multi-tenant software as a service (SaaS)," *2019 International Conference on Data and Software Engineering (ICoDSE)*, 2019, pp. 1–5, doi: 10.1109/ICoDSE48700.2019.9092741.
- [7] D. Yuan, N. Hong, W. Bingfei, and L. Lei, "Scaling the data in multi-tenant business support system," *2009 Pacific-Asia Conference on Knowledge Engineering and Software Engineering*, 2009, pp. 43–46, doi: 10.1109/KESE.2009.20.
- [8] J. Zeng and B. Plale, "Argus: a multi-tenancy NoSQL store with workload-aware resource reservation," *Parallel Computing*, vol. 58, pp. 76–89, 2016, doi: 10.1016/j.parco.2016.06.003.
- [9] P. -J. Maenhaut *et al.*, "Characterizing the performance of tenant data management in multi-tenant cloud authorization systems," *2014 IEEE Network Operations and Management Symposium (NOMS)*, 2014, pp. 1–8, doi: 10.1109/NOMS.2014.6838232.
- [10] Y. Wang, Q. He, D. Ye and Y. Yang, "Formulating criticality-based cost-effective fault tolerance strategies for multi-tenant service-based systems," in *IEEE Transactions on Software Engineering*, vol. 44, no. 3, pp. 291–307, Mar. 2018, doi: 10.1109/TSE.2017.2681667.
- [11] F. Shaikh and D. Patil, "Multi-tenant e-commerce based on SaaS model to minimize IT cost," *2014 International Conference on Advances in Engineering & Technology Research (ICAETR - 2014)*, 2014, pp. 1–4, doi: 10.1109/ICAETR.2014.7012861.
- [12] G. Karataş, F. Can, G. Doğan, C. Konca, and A. Akbulut, "Multi-tenant architectures in the cloud: A systematic mapping study," *2017 International Artificial Intelligence and Data Processing Symposium (IDAP)*, pp. 6–9, 2017, doi: 10.1109/IDAP.2017.8090268.
- [13] J. Xu, X. Li and X. Zhao, "Design of database architecture in the saas-based multi-tenant educational information system," *2011 6th International Conference on Computer Science & Education (ICCSE)*, 2011, pp. 114–119, doi: 10.1109/ICCSE.2011.6028597.
- [14] F. R. C. Sousa and J. C. Machado, "Towards elastic multi-tenant database replication with quality of service," *2012 IEEE Fifth International Conference on Utility and Cloud Computing*, Nov. 2012, pp. 168–175, doi: 10.1109/UCC.2012.36.
- [15] J. Ye, H. Lu, and K. Maeda, "TOCA: a tenant-oriented control architecture for multi-domain cloud networks," *2016 18<sup>th</sup> Asia-Pacific Network Operations and Management Symposium (APNOMS)*, 2016, pp. 1–4, doi: 10.1109/APNOMS.2016.7737282.
- [16] P. Aghera, S. Chaudhary, and V. Kumar, "An approach to build multi-tenant SaaS application with monitoring and SLA," *2012 International Conference on Communication Systems and Network Technologies*, 2012, pp. 658–661, doi: 10.1109/CSNT.2012.146.
- [17] X. Wang, Q. Li, and L. Kong, "Multiple sparse tables based on pivot tables for multi-tenant data storage in SaaS," *2011 IEEE International Conference on Information and Automation*, 2011, pp. 634–637, doi: 10.1109/ICINFA.2011.5949071.
- [18] B. Huang *et al.*, "BPS: A reliable and efficient pub/sub communication model with blockchain-enhanced paradigm in multi-tenant edge cloud," *Journal of Parallel and Distributed Computing*, vol. 143, pp. 167–178, 2020, doi: 10.1016/j.jpdc.2020.05.005.
- [19] S. Kalra and T. V. Prabhakar, "Patterns for managing tenants in a multi-tenant application," *Proceedings of the 22nd European Conference on Pattern Languages of Programs*, pp. 1–10, 2017, doi: 10.1145/3147704.3147722.
- [20] S. Kalra, "Implementation patterns for multi-tenancy," *Proceedings of the 24th Conference on Pattern Languages of Programs*, vol. 23, no. 18, pp. 1–16, 2017.
- [21] A. O. Abdul, J. Bass, H. Ghavimi, N. MacRae, and P. Adam, "Multi-tenancy design patterns in saas applications: a performance evaluation case study," *International Journal of Digital Society (IJDS)*, vol. 9, no. 1, pp. 1367–1375, 2018, doi: 10.20533/ijds.2040.2570.2018.0168.
- [22] Z. H. Wang, C. J. Guo, B. Gao, W. Sun, Z. Zhang, and W. H. An, "A study and performance evaluation of the multi-tenant data tier design patterns for service oriented computing," *2008 IEEE International Conference on e-Business Engineering*, 2008, pp. 94–101, doi: 10.1109/ICEBE.2008.60.
- [23] P. Morakos and A. Meliones, "Design and implementation of a cloud SaaS framework for multi-tenant applications," *IISA 2014, The 5th International Conference on Information, Intelligence, Systems and Applications*, 2014, pp. 273–278, doi: 10.1109/IISA.2014.6878755.




- [24] A. Ulalah, "Model-view-controller (MVC) architecture," [Online] Available: <http://www.jdl.co.uk/briefings/MVC.pdf> 28 vol. 1, no. Mvc. [Citado em: 10 de março de 2006].
- [25] M. Kalelkar, P. Churi, and D. Kalelkar, "Implementation of model-view-controller architecture pattern for business intelligence architecture," *International Journal of Computer Applications*, vol. 102, no. 12, pp. 16–21, 2014, doi: 10.5120/17867-8786.
- [26] H. Y. Lee and N. J. Wang, "Cloud-based enterprise resource planning with elastic model-view-controller architecture for Internet realization," *Computer Standards & Interfaces*, vol. 64, pp. 11–23, 2019, doi: 10.1016/j.csi.2018.11.005.
- [27] M. R. J. Qureshi and F. Sabir, "A comparison of model view controller and model view presenter," vol. 25, no. 1, pp. 7–9, 2014, doi: 10.48550/arXiv.1408.5786. [Online]. Available: <http://arxiv.org/abs/1408.5786>.
- [28] S. M. S. Anuar, N. F. M. Azmi, N. Maarop, G. N. Samy, S. Yaacob, and D. W. H. Ten, "Preliminary review of model-view-presenter (MVP) and usability design for the development of postgraduate web portal," *Open International Journal of Informatics*, vol. 5, no. 1, pp. 1–11, 2017, [Online]. Available: <http://apps.razak.utm.my/ojs/index.php/oiji/article/download/83/60>. (accessed: Jul. 4, 2022).
- [29] E. Sørensen and M. I. Mihailesc, "Model-view-viewmodel (MVVM) design pattern using windows presentation foundation (WPF) technology," *MegaByte J.*, vol. 9, no. 4, pp. 1–19, 2010, [Online]. Available: [http://megabyte.utm.ro/articole/2010/info/sem1/InfoStraini\\_Pdf/1.pdf](http://megabyte.utm.ro/articole/2010/info/sem1/InfoStraini_Pdf/1.pdf). (accessed: Jul. 4, 2022).
- [30] R. Buyya, C. S. Yeo, S. Venugopal, J. Broberg, and I. Brandic, "Cloud computing and emerging IT platforms: vision, hype, and reality for delivering computing as the 5th utility," *Future Generation computer systems*, vol. 25, no. 6, pp. 599–616, 2009, doi: 10.1016/j.future.2008.12.001.
- [31] X. Li, L. Zhou, Y. Shi, and Y. Guo, "A trusted computing environment model in cloud architecture," *2010 International Conference on Machine Learning and Cybernetics*, 2010, pp. 2843–2848, doi: 10.1109/ICMLC.2010.5580769.
- [32] X. Shang, X. Liu, G. Xiong, C. Cheng, Y. Ma, and T. R. Nyberg, "Social manufacturing cloud service platform for the mass customization in apparel industry," *Proceedings of 2013 IEEE International Conference on Service Operations and Logistics, and Informatics*, 2013, pp. 220–224, doi: 10.1109/SOLI.2013.6611413.
- [33] P. Guleria, "Data access layer: a programming paradigm on cloud," *International Journal of Computers & Technology*, vol. 11, no. 3, pp. 2341–2345, 2013.
- [34] W. Tsai, Y. Huang, and Q. Shao, "Testing the scalability of SaaS applications," *2011 IEEE International Conference on Service-Oriented Computing and Applications (SOCA)*, 2011, pp. 1–4, doi: 10.1109/SOCA.2011.6166245.

## BIOGRAPHIES OF AUTHORS






**Wiwin Suwarningsih**    she was graduated from her bachelor degree at Informatics Program, Adityawarman Institute of Technology Bandung in 1996. She got graduated from her master education in 2000 at the Informatics Study Program, Bandung Institute of Technology and doctoral degree in 2017 the School of Electrical and Informatics Engineering, Bandung Institute of Technology. Since 2006 until now she has been a researcher at Research Center for Information and Data Science, National Research and Innovation Agency, Indonesia. His research interests are artificial intelligence, computational linguistics, particularly in Indonesian natural language processing and Indonesian text mining, information retrieval, and question answering systems. She can be contacted at email: [wiwin.suwarningsih@brin.go.id](mailto:wiwin.suwarningsih@brin.go.id).






**Ana Heryana**    he was the obtain his Master degree from Informatics department, Bandung Institute of Technology. Currently he is working as a researcher at Research Center for Information and Data Science, National Research and Innovation Agency, Indonesia. His research on software engineering, embedded system, and machine learning. He can be contacted at email: [ana.heryana@brin.go.id](mailto:ana.heryana@brin.go.id).






**Dianadewi Riswantini**    received a scholarship from the Overseas Fellowship Program, a collaboration between the Indonesian Government and World Bank, for a Bachelor's and Master's degree in computer science from the Delft University of Technology, the Netherlands, completed in 1994. She is currently a Ph.D. candidate in the School of Business and Management, Bandung Institute Technology, Indonesia, engaged in Big Data analytics for business and management. She is joining the Information Retrieval Research Group at the Research Center for Data and Information Sciences, National Agency of Research and Innovation (Indonesia). Her research interests include data analytics, text mining, natural language processing, and machine learning in the fields of social and medical informatics. She can be contacted at the email: [dianadewi.riswantini@brin.go.id](mailto:dianadewi.riswantini@brin.go.id).





**Ekasari Nugraheni**    obtained a Bachelor's degree in information management from the AKRIND Institute of Technology Yogyakarta in 1996. She received a scholarship from the Indonesian Ministry of Research and Technology for her Master's degree in informatics from the Bandung Institute of Technology, Indonesia, completed in 2016. Currently, she is joining the Information Retrieval Research Group at the Research Center for Data and Information Sciences, National Research and Innovation Agency (Indonesia). Her research interests include data analysis, data mining, deep learning, and natural language processing. She can be contacted at email: [ekasari.nugraheni@brin.go.id](mailto:ekasari.nugraheni@brin.go.id).



**Dikdik Krisnandi**    obtained his Master's degree from the electrical engineering department, Bandung Institute of Technology. Currently, he is working as a researcher at Research Center for Information and Data Science, National Research and Innovation Agency, Indonesia. His research on artificial intelligence, machine learning, and embedded system. He can be contacted at email: [dikdik.krisnandi@brin.go.id](mailto:dikdik.krisnandi@brin.go.id).