Analysis and Design of Integration Model for API Management and CI/CD at Directorate General of Taxation

Tri Pramudaya[1]*, Fenni Agustina[2]
Master of Information Systems Management Program [1]
Master of Technology and Engineering Program [2]
Gunadarma University
Jawa Barat, Indonesia
tripramudaya@gmail.com [1], fenni@staff.gunadarma.ac.id [2]

Abstract— **The Directorate General of Taxes (DGT) currently utilizes Application Programming Interface (API) to enhance efficiency in tax data exchange with external parties. DGT is facing challenges due to the rising number of published APIs and the increasing connections from external parties to the DGT system, which necessitates a speedy API issuance process. The objective of this research is to assist the Directorate General of Taxation (DGT) in developing an integrated API management system with Continuous Integration/Continuous Deployment (CI/CD). The system design process is conducted using the Standards and Architectures for E-Government Application (SAGA) framework, encompassing Enterprise Viewpoint, Technology Viewpoint, Computational Viewpoint, Information Viewpoint, and Engineering Viewpoint. A qualitative method is employed, including interviews to gain insights into the existing issues. Additionally, information regarding systems and technologies is documented for gap analysis. The results of this analysis are then utilized to design the architecture of the API management system, applications, and technologies. This research yields a model of the API management system integrated with CI/CD at DGT. The model is developed using 3Scale and Jenkins software. Following validation, the API management system at DGT operates effectively with three DGT API systems and three API users.**

Keywords— API, API Management, CI/CD, Model, SAGA

I. Introduction

The use of Application Programming Interface (API) plays a key role in building connectivity and interconnection among entities in the broader information technology ecosystem. An API is a software-to-software interface that allows applications to communicate over a network without user intervention, serving as a contract between them[1]. An application programming interface (API) serves as a crucial bridge between different software systems, enabling developers to interact with the features and data of a program in a standardized manner. Essentially, an API acts as a set of functions and protocols, defining how software components should interact with each other[2]. By providing a clear set of rules and functionalities, APIs facilitate seamless integration and communication between various applications, allowing developers to harness the full potential of software resources and build innovative solutions[3]. Through API usage, developers can access specific functionalities of a software system without needing to understand its internal workings, streamlining the development process and promoting collaboration across different platforms and services.

APIs require API Management to ensure their security, scalability, and efficient utilization, as well as to provide governance and control over their lifecycle. The API management is a set of technologies, processes, and tools for governing APIs in a scalable and secure service infrastructure. API management solution consists of the following main components : API Portal, API Gateway, API Service Manager, API Monitor and API Billing[4]. Discovering, creating, deploying, utilizing, or running large-scale APIs is the domain of API management, which includes a number of procedures and resources [5] API management encompasses more than simply technical management; it also involves creating an ecosystem that allows different stakeholders to work together, develop, and profit from APIs. There are many API lifecycle management platforms available, each with its own set of features, with 3scale being one of them. Red Hat 3scale API lifecycle Management is a comprehensive solution for managing APIs. It provides a centralized platform for managing API keys, rate limiting, and access control. 3scale also provides a self-service portal for developers to register for keys and manage their own access[6].

API Management (APIM) and Continuous Integration/Continuous Deployment (CI/CD) are closely related in modern software development practices. APIM governs the lifecycle of APIs, ensuring their security, scalability, and efficient utilization. On the other hand, CI/CD automates the process of integrating code changes, testing them, and deploying them to production environments rapidly and consistently. Continuous Integration/Continuous Deployment (CI/CD) consists of a collection of methods and tools that enable teams to automate software product development, testing, and deployment in an efficient and reliable manner. [7]. Jenkins is one of the open-source applications used to automate tasks within the continuous integration and delivery (CI/CD) process of an application[8].

In 2019, [4] conducted a study on 5 API Management

p-ISSN 2301-7988, e-ISSN 2581-0588
DOI : 10.32736/sisfokom.v13i2.2086, Copyright ©2024
Submitted : February 9, 2024, Revised : March 5, 2024, Accepted : March 19, 2024, Published : June 15, 2024

software, namely Axway API Gateway, APIGEE, IBM API Connect, Mashrey API Management, and CA API Gateway. In this research, an in-depth examination was carried out on solutions provided by the top five vendors, with a particular emphasis on industry applications, commonly utilized features, and notable success stories associated with these vendors. Based on a literature review, this research identified the top three industry implementations of API management as "Technology," "Media & Entertainment," and "Service Provider," with the leading API Management Solutions being "CA" and "APIGEE." Survey results supported these findings, highlighting the prevalence of API management in industries such as "Technology," "Government," and "Service Provider," with "APIGEE" and "CA" emerging as the top two solutions.

The research conducted by [9] aimed to extensively examine the vital role of API management in the field of product management. By adopting optimal approaches in API design, documentation, security, and monitoring, product managers could effectively address challenges, streamline product development, and play a significant role in the overall success of their digital products. In conclusion, it was essential for product managers to have the capability to strategically decide on API design, partnerships, and integrations in order to stimulate innovation and guarantee the success of their products

In 2020, [10] described a Systematic Literature Review (SLR) that had the goal of collecting API Management practices and capabilities related to API Management. This research described a Systematic Literature Review (SLR) aimed at gathering API Management practices and capabilities associated with API Management, alongside proposing a comprehensive definition of the topic. Within this research, a practice is defined as any action intended to enhance, promote, and oversee the utilization of APIs. Capabilities refer to the proficiency in achieving specific goals related to API Management. Throughout this research, 24 unique definitions for the topic of API Management were compiled, along with 114 API Management practices and 39 capabilities of API Management.

The issue discussed in reference [11] was the automation of the myITS application deployment process on the server using Jenkins pipeline.The myITS Single Sign On application is used by ITS to interact with other applications such as Classroom, Academic, and Scholarships on myITS. This research resulted in good functional testing, with all CI/CD pipeline steps running normally. For performance testing, it was found that the average time for the pipeline to run was 355.4 seconds or approximately 5.93 minutes. Therefore, further research can be developed by optimizing the build image process to shorten the build duration in the pipeline.

The research conducted by [12] discussed the challenges faced when implementing CI/CD with Jenkins, the benefits of Jenkins, methods to improve the process like micro-pipelines, and the plugins available in Jenkins. Employing CI/CD in software development is very important as it saves time in finding bugs from a large source code and fixing them.

The Directorate General of Taxation (DGT) is currently developing interoperability with both internal and external entities.Interoperability refers to the ability of systems to exchange and utilize information seamlessly[13]. By establishing interoperability with other systems, DGT aims to improve efficiency and foster better collaboration within the ecosystem of information technology and tax services. One of the key technologies enabling this interoperability is Application Programming Interface [14]. APIs allow DGT's system to communicate and interact with external systems, such as those belonging to government agencies, financial institutions, or business entities. This facilitates the efficient and secure exchange of data between DGT and these various entities.

Based on Regulation Number PER-10.PJ/2020 of the Directorate General of Taxation, which pertains to amendments to Regulation Number PER-11/PJ/2019 of the Director General of Taxation regarding Tax Application Service Providers, the Directorate General of Taxes offers 9 tax systems in API format to facilitate communication with diverse stakeholders in the tax sector. Additionally, there are 86 banking/post/perception institutions [15], 13 Tax Application Service Provider entities [16], and local government entities that currently have Memoranda of Understanding (MOUs) with 367 Local Governments [17] connected to DGT's API.

DGT faces challenges in maintaining its published APIs due to the traditional approach to API maintenance without support from API management software. In this context, DGT's numerous APIs and connections to various stakeholders can create a complex network of interactions, making management difficult. This complexity can result in what is termed a "hyperconnectivity mess,[18]" where the sheer number of connections and interactions creates confusion, inefficiency, and potential security risks. However, DGT also encounters difficulties in the API deployment process, necessitating efficiency in speed. In API development, the delivery and deployment processes are carried out manually, with developers compiling the application on their computers, and then releasing the compiled API results into the server. However, this is considered very troublesome, especially if the changes released are minor and code changes occur more frequently.

Based on this background, there is a significant need for implementing API management integrated with continuous integration and continuous deployment (CI/CD) at DGT. The research aims to address this need by analyzing the current situation and developing a model for an integrated API management system with CI/CD specifically tailored to the DGT environment. With the API management system model in DGT, a system prototype will be produced as an intermediary between developers and users to interact in the process of information system development activities[19]. This research can provide insights into the unique needs and challenges of combining API Management and CI/CD in the tax environment.

II. Research Method

In this study, qualitative methods were employed, focusing on a specific case study at DGT. This method involved direct data collection through interviews and observations to understand the complexity of API implementation at DGT, with a focus on operational challenges and stakeholder perceptions[20]. Qualitative methods also involve gathering feedback from users or stakeholders. In this study, the design of an information system was carried out, resulting in a model of API management. This model serves to ensure that the designed system can assist in the development of appropriate software and facilitate management[21]. The scope of activities for creating that model is illustrated in Fig. 1:

1. *Problem Identification: This step involves identifying and defining the specific problems or challenges in deploying APIs and API management that have been published by DGT.*
2. *System Analysis: In this step, a comprehensive analysis is conducted to understand the requirements, constraints, and objectives of the system being modeled.*
3. *System Design: This step focuses on designing the structure, components, and functionalities of the system based on the analysis conducted in the previous step.*
4. *System Validation: In this final step, the model is validated to ensure that it meets the specified requirements and effectively addresses the identified problems or challenges.*

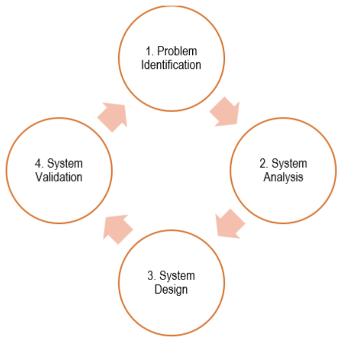


Fig. 1. Research Process Flow

III. Result And Discussion

In general, the Results and Discussion chapter aims to explain and interpretation the research findings.

A. *System Analysis*

System analysis is conducted using the following steps:

1. *Identification Of System Requirements*

Identification of business problems results in three main things: the need for a system model, the need for an API management system, and the need for accelerated deployment. The model is needed as a tool for describing API management that will be carried out at DGT. An API management system is needed because currently there is no such system in the DGT. To increase development efficiency and spread API changes quickly, CI/CD is needed.

2. *API Inventory Identification*

The model to be built is planned to have 10 (ten) APIs installed, consisting of 1 (one) API for the Taxpayer Status Confirmation system, 3 (three) APIs for the e-Filing system, and 6 (six) APIs for the withholding system.

3. *CI/CD Workflow Identification*

The CI/CD implementation will be applied to the process of installing the API on the server in five stages, namely pulling the source code from the repository, compiling the application, creating the application image, pushing the application image to the repository, and installing the API [22].

4. *Selection of Tools and Technologies*

While API management needs may vary across organizations, the fundamental functions of API management include security, monitoring, and version control. Defining the functional areas of API management software according to [4], [23] and [24], a priority scale is made based on the Moscow technique [25], as shown in (TABLE I. API Management Platform Requirements).

TABLE I. API Management Platform Requirements

No	Area	Priority
1	Architecture	Must Have
2	Developer Portal	Must Have
3	API Gateway	Must Have
4	API Security	Must Have
5	API Analytics	Must Have
6	API SDLC	Must Have
7	API Innovation	Could Have
8	Training & Support	Should Have
9	Industry Experience	Could Have

The system will be designed to meet needs in the "Must Have" category only. Based on the above needs, the API management platform chosen was 3Scale, the CI/CD software was Jenkins [26], and other tools were Nexus, Openshift Container Platform, and GitLab.

5. *Integration Design*

The integration workflow between API management and CI/CD is designed to start with the developer building an API and, after completing programming, sending the source code to the repository. CI/CD can be run by starting by pulling the source code from the repository and then building the API in application form. The next stage is to build an image so that it can be run independently. The image is first placed in the repository as a backup and then deployed in the environment (development, simulation, or production). The API address, in the form of a uniform resource locator (URL), will be connected to API Management. The API will be published via the API Management tool so that it will appear on the developer portal website, which is accessed by API user developers. The integration point between the CI/CD workflow and API Management is accessing API addresses in the backend to be published by API Management (Fig. 2 Integration Design).

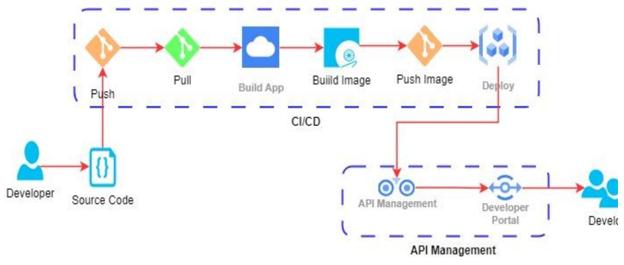


Fig. 2. Integration Design

B. System Design

A specific design model being proposed for managing APIs within the Directorate General of Taxation (DGT) environment, which incorporates continuous integration and continuous deployment (CI/CD) practices. This model is structured and presented using the SAGA framework, which likely provides a systematic approach or structure for organizing and implementing API management processes within the DGT context.

1. *Enterprise Viewpoint :*The enterprise viewpoint specifies the aims, scope, processes, and policies of an application[27]. A new policy needs to be established because previously there was no working procedure for managing API users. The DGT API Developer Portal is intended for public access, but only third parties who meet specific requirements are granted API access. The work procedure for granting DGT API user access is determined as follows:

1) *The third party registers new users through the DGT portal developer*

2) *Employees at the Directorate of Information and Communication Technology carry out verification. If approved, the employee will carry out approval on the API Management admin page and send an email to the third party.*

3) *Users subscribe to an API provided by DGT via the developer portal.*

4) *The third party registers the Third Party's application on the developer portal page.*

5) *Employees at the Directorate of Information and Communication Technology will verify the application submission and approve it if it meets the requirements.*

6) *Third parties can view the API key via the DGT portal developer page for approved applications.*

2. *Information Viewpoint :* The information viewpoint describes the structure of the API that has to be developed in the context of API development. API specifications provide clear technical guidance for developers who will use or integrate APIs. Therefore, it is determined that the API specifications to be built include the following (TABLE II. API Specifications) :

TABLE II API SPECIFICATIONS

No	Element	Spesifikasi
1	Architecture	Representational State Transfer (REST)
2	Data Formats	JavaScript Object Notation (JSON).
3	HTTP Method	POST, GET, DELETE and PUT
4	Response Code	1) 200 (Success) for successful request 2) 400 (Bad Request) for data used when the request is not appropriate 3) 401 (Unauthorized) for a code indicating that the request requires authentication (such as username and password) 4) 404 (Not Found) for the requested URL not found 5) 500 (Internal Server Error) to inform you that there is a technical error on the API or server-side 6) Specific code according to API requirements
5	Response Message	1) Status Code which contains the return code from the API or response code 2) Message Status which contains response messages or error messages 3) Data containing returned data in JSON format
6	Framework	springboot, however, if there are other considerations, an appropriate framework can be used.
7	Programming Language	Java, however, if there are other considerations, another programming language can be used.
8	Documentation	Swagger 2.0 atau Open API 3.0
9	Database Connection	HikariCP, however, if there are other considerations, another library can be used.
10	Logging	sl4j or log4J.
11	Uniform Resources Indentifier (URI) Path	URI format: /api.[service_grup].[environment].pajak.go.id/version/nama_produk 1) "service_grup" to represent the system group name of the API 2) "environment" to represent the installed location, namely dev for development, sim for simulation, and prod for production (live) 3) version" is written with the letter v followed by the version number without a dot 4) "product_name" to represent the API name example: api.kswp.apps.dev.pajak.go.id /v2/wp
12	Authorization and authentication	Minimum basic auth authorization with Oauth 2.0 is recommended, while authentication can use a user key, application key, or token
13	File Transactions	If an API has a file as a payload, then: 1) If the file size is large, it must be chunked 2) The format of the contents must be in XML form 3) Must be signed with a certificate owned by the Taxpayer 4) Must be signed with a certificate owned by the sending provider

3. *Computational Viewpoint*

The computational viewpoint represents the breakdown of an application into functional modules that include interaction interfaces[27]. 3Scale API Management software has provided a web-based graphical user interface that can be used to manage APIs. However, the developer portal as an interaction tool with developers outside the DGT still has to be adjusted according to the DGT's needs. The interaction relationship between the DGT API management system and system user actors is shown in Figure 3. Developer Portal Use Case. Two actors play a role, namely the DGT Third Party as the portal developer system user and the DGT Administrator as the API Management Manager.

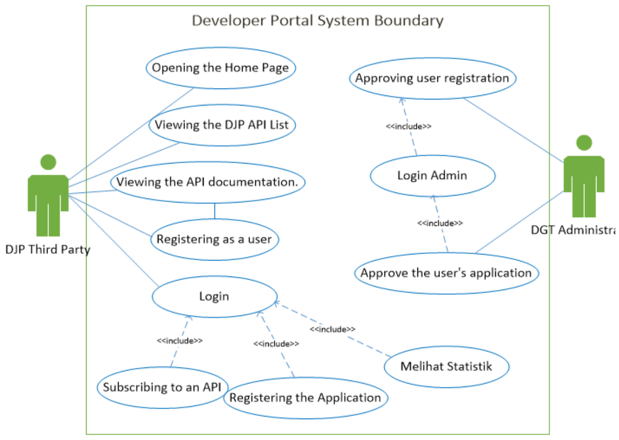


Fig.3. Use Case Developer Portal

The DGT Administrator's job is to approve user developer portal applications and approve applications from users, while the DGT Third Party can register users, view API documentation, see a list of available APIs, subscribe to an API, register applications, and view statistics from API usage.

The DGT developer portal that will be built is web-based. The use of this wireframe makes it possible to define the design, design the layout, and determine the site path [18] from the DGT web developer portal. The DGT portal developer wireframe shows the relationship path of the 9 (nine) pages that will be built, as in Fig. 4. Wireframe Developer Portal.

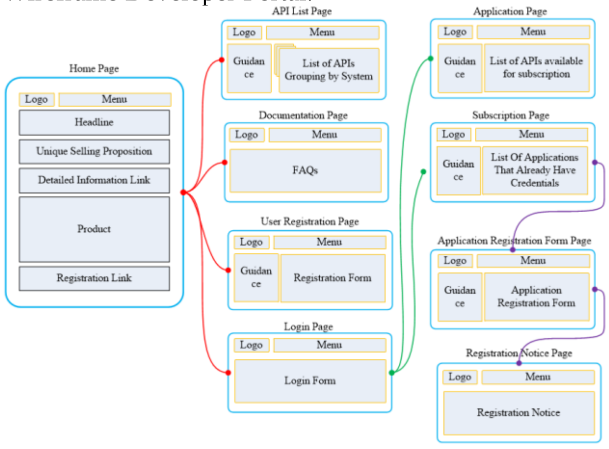

Home Page
Logo
Menu
Headline
Unique Selling Proposition
Detailed Information Link
Product
Registration Link
API List Page
Guidance
List of APIs Grouping by System
Documentation Page
FAQs
User Registration Page
Registration Form
Login Page
Login Form
Application Page
List of APIs available for subscription
Subscription Page
List Of Applications That Already Have Credentials
Application Registration Form Page
Application Registration Form
Registration Notice Page
Registration Notice

Fig.4. Wireframe Developer Portal

4. *Engineering Viewpoint*

1) *Arsitektur API Management*

Engineering Viewpoint describes the distribution of physical resource elements and their relationships[27]. The DGT API architecture has four parts (Fig.5. API Management Architecture).

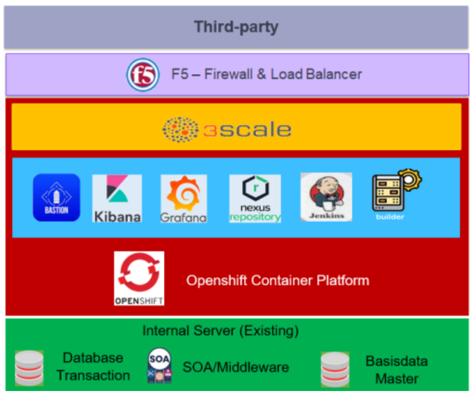


Fig.5. Arsitektur API Management

a. *Internal Server*

Internal Server is the database that will be accessed by the API I.

b. *Openshift Container Platform (OCP)*

OCP is used by many organizations to manage and orchestrate containers. OCP is a large unified system, which includes Bastion, Kibana, Grafana, Nexus, Jenkins, and 3Scale.

c. *Firewall and Load Balancer*

Firewall and Load Balancer use F5 products as software for system and load balancer security.

d. *Third-party*

Third parties are developers outside the DGT who will use the API.

2) *Hardware Design*

The hardware is arranged to meet the minimum production level requirements listed in TABLE III Server Requirements.

TABLE III SERVER REQUIREMENTS

No	Server	vCPU	RAM (GB)	DISK (GB)	OS	Jumlah (unit)
1	Bastion	4	16	200	RHEL7	1
2	Master	8	32	125	CoreOS	4
3	Router	4	16	125	CoreOS	4
4	Storage	16	64	600	CoreOS	4
5	Infra	8	32	300	CoreOS	8
6	Worker	8	32	250	CoreOS	1
7	Nexus	4	16	200	RHEL7	1
8	Jenkins	4	16	125	RHEL7	1
9	GitLab	4	16	300	RHEL7	1
10	Builder	4	16	125	RHEL7	1

3) Network Topology

The physical or logical arrangement of hardware and connections between servers in API Management is outlined in the network topology to be built (Fig.6. Network Topology).

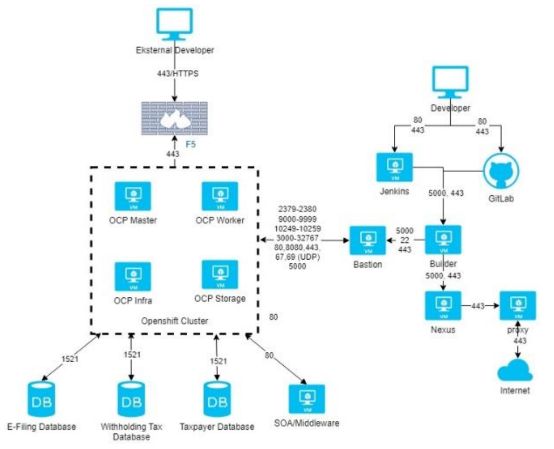


Fig.6. Network Topology

5. *Technology Viewpoint*

The technologies employed to implement the system are described from a technological point of view[27]. The following technologies were employed in the creation of the DGT API management model:

a. *Container*

An API will live in a container, so software is needed to manage the container. The software used is the Openshift Container Platform from RedHat.

b. *API Management*

The software used for API Management is 3Scale from RedHat. Openshift Container Platform and 3Scale are both products from RedHat, so these two software are compatible and easy to integrate.

c. *CI/CD*

The software used to manage CI/CD is Jenkins.

d. *Repository*

The repository tools used are GitLab and Sonatype Nexus Repository (Nexus). GitLab is used to store files from application development as well as a Version Control System (VCS) whose job it is to track changes to files. Nexus is used as image storage for applications and also as a center for libraries needed in developing application images.

e. *Testing*

Testing for the API used by Postman.

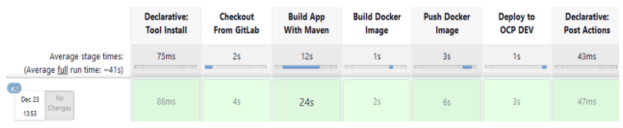
C. *Model Validation*

1. *Validation for API Management Platform Requirements*

Validation of API Management platform requirements is carried out for areas in the “Must Have” category. The specifications of each area are checked for their function in the DGT API management system model that is being built. The results of the validation state that all the required features have been fulfilled in the model.

2. *Validation for CI/CD*

The Jenkins procedure is used for CI/CD validation. If the API has been successfully deployed into the OCP container, then the validation is considered successful. Jenkins displays the CD/CD process, beginning with getting a copy of source code from GitLab, building the application with Maven, creating an image, storing the image in the repository, and finally deploying the API (Fig.7. CI/CD Stage). The CI/CD process in the development environment has successfully completed the ten (10) APIs that were turned into models.



	Declarative: Tool Install	Checkout From GitLab	Build App With Maven	Build Docker Image	Push Docker Image	Deploy to OCP DEV	Declarative: Post Actions
Average stage times: (Average full run time: ~41s)	75ms	2s	12s	1s	3s	1s	43ms
#2 Dec 23 13:53 No Changes	88ms	4s	24s	2s	6s	3s	47ms

Fig. 7. CI/CD Stage

3. *Validation for API Management*

Validation of API management is seen from two sides, namely the DGT administrator side and the API user side (DGT Third Party). From the DGT administrator side, validation was carried out on the system model that was built, where the model had successfully published 10 (ten) APIs. Model validation from the API user side is carried out on the developer portal, which is accessed by the DGT Third Party. The development of the DGT API portal developer resulted in nine web pages, all of which functioned well. The testing process uses three dummy users as DGT external parties who access the DGT API through the developer portal.

IV. CONCLUSION

After examining the Directorate General of Taxes' API management system, a number of conclusions can be made based on research on the integration of application programming interface management with the continuous integration/continuous deployment process. This study develops an API management system model that is integrated with an automatic application installation procedure that employs the continuous integration/continuous deployment methodology. The DGT developer portal and API Management interface are compatible with the DGT API management system concept. While DGT may manage its APIs through the API Management interface, which utilizes 3Scale software, the DGT portal developer has served as an interface between DGT and parties outside DGT.

However, it is important to note that this research focuses solely on the proposed design model for managing APIs within the Directorate General of Taxation (DGT) environment. This research still has many shortcomings, but it is hoped that this can be the first step toward a better DGT. API integration certainly involves many parties. This research has not yet analyzed the level of user acceptance of the API management system model organized by the DGT. Performance testing has also not been carried out in this research. Standardization is needed in API management, so it can replicate the National Open API Payment Standard (SNAP) established by Bank Indonesia in the payment system business.

REFERENCES

[1] Weir , L., Enterprise API Management. *Birmingham: Packt Publishing*

Ltd. 2019

[2] O. O. Efuntade and A. O. Efuntade, “Application Programming Interface (API) And Management of Web-Based Accounting Information System (AIS): Security of Transaction Processing System, General Ledger and Financial Reporting System,” *Journal of Accounting and Financial Management*, vol. 9, no. 6, 2023, doi: 10.56201/jafm.v9.no6.2023.pg1.18.

[3] N. K. Akmal and M. N. Dasaprawira, “Rancang bangun Application Programming Interface (API) menggunakan gaya arsitektur Graphql untuk pembuatan sistem informasi pendataan anggota Unit Kegiatan Mahasiswa (UKM) studi kasus UKM Starlabs,” *Jurnal SITECH : Sistem Informasi dan Teknologi*, vol. 5, no. 1, 2022, doi: 10.24176/sitech.v5i1.7937.

[4] S. Moiz Ali and T. Rahim Soomro, “Comparative Study of API Management Solutions,” *Proceedings of The 6th International Conference on Innovation in Science and Technology*, 2019. doi: 10.33422/6th-istconf.2019.07.411.

[5] M. Mehdi, E. Wilde, R. Mitra, and M. Amundsen, Continuous API Management: Making the Right Decisions in an Evolving Landscape, First Edition. Sebastopol, CA : O’Reilly Media, Inc., 2019.

[6] V. Srivastava, “10 API Lifecycle Management Platforms”, January 19, 2023, https://nordicapis.com/9-api-lifecycle-management-platforms/, (accessed March 3, 2024).

[7] N. Singh, “CI/CD Pipeline for Web Applications,” *International Journal for Research in Applied Science and Engineering Technology*, vol. 11, no. 5, 2023, doi: 10.22214/ijraset.2023.52867.

[8] Malathi. S | Ganeshan. M, “Building and Deploying a Static Application using Jenkins and Docker in AWS,” *International Journal of Trend in Scientific Research and Development*, vol. 4, no. 4, 2020, doi: 10.1007/978-3.

[9] L. S. Daggubati, “API Management for Product Managers”,

International Journal of Management (IJM), 14(7), 2023, pp. 137-141.

[10] Mathijssen, Max, Michiel Overeem and Slinger Jansen. “Identification of Practices and Capabilities in API Management: A Systematic Literature Review.” ArXiv abs/2006.10481 , 2020.

[11] R. A. Parama, H. Studiawan, and R. J. Akbar, “Implementasi Continuous Integration dan Continuous Delivery Pada Aplikasi myITS Single Sign On,” *Jurnal Teknik ITS*, vol. 11, no. 3, 2022, doi: 10.12962/j23373539.v11i3.99436.

[12] S.K, Arpita, Amrathesh Amrathesh and Dr. Govinda Raju M. “A review on Continuous Integration, Delivery and Deployment using Jenkins.” *Journal of University of Shanghai for Science and Technology*, 2021, https://doi.org/10.51201/jusst/21/05376.

[13] D. Marsh-Hunn, S. Trilles, A. González-Pérez, J. Torres-Sospedra, and F. Ramos, “A Comparative Study in the Standardization of IoT Devices Using Geospatial Web Standards,” *IEEE Sensors Journal*, vol. 21, no. 4, 2021, doi: 10.1109/JSEN.2020.3031315.

[14] N. Puspitasari, E. Budiman, Y. N. Sulaiman, and M. B. Firdaus, “Microservice API Implementation for E-Government Service Interoperability,” *Journal of Physics: Conference Series*, 2021. doi: 10.1088/1742-6596/1807/1/012005.

[15] D.Sudarmawan, “MPN G3 Solusi Kemudahan Penyetoran Penerimaan Negara Di saat Pandemi Covid-19”, January 12, 2021, https://DGTb.kemenkeu.go.id/kppn/solok/id/data-publikasi/artikel/2979-mpn-g3-solusi-kemudahan-penyetoran-penerimaan-negara-di-saat-pandemi-covid-19.html, (accessed February 20, 2024).

[16] Direktorat Jenderal Pajak, “Daftar PJAP”, 2022, https://ww.pajak.go.id/id/index-pjap, 2022, (accessed February 20, 2024).

[17] R. Yunita, “Wujud Kongkrit Sinergi Dalam rangka Optimalisasi Pajak Pusat Dan Pajak Daerah”, August 23, 2023, . https://DGTk.kemenkeu.go.id/wp-content/uploads/2023/08/SP-PKS-DGT-DGTK-Pemda_release.pdf , accessed February 20, 2024).

[18] W. Luis, “Avoiding a hyperconnectivity mess,” https://subscription.packtpub.com/book/web-development/9781787284432/1/ch01lvl1sec04/avoiding-a-hyperconnectivity-mess, 2019, (accessed February 20, 2024).

[19] A. Susanto and Meiryani, “System Development Method with The Prototype Method,” *International Journal of Scientific and Technology Research*, vol. 8, no. 7, 2019.

[20] Rusandi and Muhammad Rusli, “Merancang Penelitian Kualitatif Dasar/Deskriptif dan Studi Kasus,” *Al-Ubudiyah: Jurnal Pendidikan dan Studi Islam*, vol. 2, no. 1, 2021, doi: 10.55623/au.v2i1.18.

[21] S. Susanto, J. P. Manurung, and F. Wnditya Setyawan, “Information System Design COVID-19 with Prototype Model,” *Buana Information Technology and Computer Sciences (BIT and CS)*, vol. 1, no. 2, 2020, doi: 10.36805/bit-cs.v1i2.1074.

[22] M. K. Abhishek, D. R. Rao, and K. Subrahmanyam, “Framework to Deploy Containers using Kubernetes and CI/CD Pipeline,” *International Journal of Advanced Computer Science and Applications*, vol. 13, no. 4, 2022, doi: 10.14569/IJACSA.2022.0130460.

[23] TIBCO, “The Ultimate Implementation Guide to API Management: Strategies, Insights, and Best Practices for API Product Leaders,” United State, 2021.

[24] B. De, “API Management: An Architect´s Guide to Developing and Managing APIs for Your Organization”, Apress, vol. First Edition. 2017.

[25] A. Hudaib, R. Masadeh, M. H. Qasem, and A. Alzaqebah, “Requirements Prioritization Techniques Comparison,” *Modern Applied Science*, vol. 12, no. 2, 2018, doi: 10.5539/mas.v12n2p62.

[26] Pathania, Nikhil. (2017) 2017. “Learning Continuous Integration with Jenkins”, Second Edition. 2nd ed. Packt Publishing. https://www.perlego.com/book/578794/learning-continuous-integration-with-jenkins-second-edition-pdf.

[27] A. Purwanto, “Perancangan Arsitektur Sistem Informasi Pariwisata Menggunakan Framework Saga (Studi Kasus: Dinas Pariwisata Kabupaten Belitung),” *Infotronik : Jurnal Teknologi Informasi dan Elektronika*, vol. 8, no. 1, 2023, doi: 10.32897/infotronik.2023.8.1.2735.

p-ISSN 2301-7988, e-ISSN 2581-0588
DOI : 10.32736/sisfokom.v13i2.2086, Copyright ©2024
Submitted : February 9, 2024, Revised : March 5, 2024, Accepted : March 19, 2024, Published : June 15, 2024