Terakreditasi SINTA Peringkat 2
Surat Keputusan Dirjen Penguatan Riset dan Pengembangan Ristek Dikti No. 10/E/KPT/2019
masa berlaku mulai Vol. 1 No. 1 tahun 2017 s.d Vol. 5 No. 3 tahun 2021

Terbit online pada laman web jurnal: **http://jurnal.iaii.or.id**

JURNAL RESTI
(Rekayasa Sistem dan Teknologi Informasi)
Vol. 4 No. 4 (2020) 745 - 750 ISSN Media Elektronik: 2580-0760

Implementasi Golang dan *New Simple Queue* pada Sistem Sandbox Pihak Ketiga Berbasis REST API

Albertus Ari Kristanto[1], Yulius Harjoseputro[2], Joseph Eric Samodra[3]
[123]Program Studi Informatika, Fakultas Teknologi Industri, Universitas Atma Jaya Yogyakarta
yulius.harjoseputro@uajy.ac.id*

Abstract

A good application development requires a testing phase to ensure there are no errors before it's released to public. But testing phase becomes difficult if the application development involves features from third parties. The idea to resolve the problem for Dhanapala application under the auspices of PT. Semangat Gotong Royong is to make the Sandbox system which is a system designed to resemble the characteristics of a third party. The Sandbox system will be developed into a REST API and written using the Golang programming language. In conducting communications with other systems New Simple Queue (NSQ) is also used that can support concurrency and prevent data transmission failures. As a result, the Sandbox system can receive requests and will process responses that are similar to functions from third parties. All forms of feature calls to third parties can be transferred to the Sandbox system so that all the data needs on some functions involving third parties can be fulfilled and the Dhanapala application can be run without its dependence on third parties

Keywords: Sandbox, Golang, NSQ, REST API, third party

Abstrak

Pengembangan aplikasi yang baik memerlukan tahap pengujian untuk memastikan tidak ada galat sebelum digunakan oleh user. Tetapi pengujian menjadi sulit dilakukan jika dalam pengembangan aplikasi tersebut melibatkan fitur dari pihak ketiga. Pada studi kasus aplikasi Dhanapala dibawah naungan PT. Semangat Gotong Royong ini, dibuatlah sistem Sandbox yang merupakan sebuah sistem yang dirancang sedemikian rupa untuk menyerupai karakteristik dari pihak ketiga. Sistem *Sandbox* akan dikembangkan menjadi sebuah REST API dan ditulis menggunakan bahasa pemrograman Golang. Dalam melakukan komunikasi dengan sistem lain pun digunakan *New Simple Queue* (NSQ) yang dapat mendukung konkurensi dan mencegah kegagalan pengiriman data. Hasilnya, sistem *Sandbox* dapat menerima request dan akan memproses response yang mirip dengan fungsi dari pihak ketiga. Segala bentuk pemanggilan fitur pada pihak ketiga dapat dialihkan ke sistem *Sandbox*, sehingga segala kebutuhan data pada beberapa fungsi yang melibatkan pihak ketiga dapat terpenuhi dan aplikasi Dhanapala dapat dijalankan tanpa ketergantungannya terhadap pihak ketiga.

Kata kunci: *Sandbox*, Golang, NSQ, REST API, Pihak Ketiga

1. Pendahuluan

Ide dan inovasi yang dimiliki para pengembang aplikasi pada saat ini sangatlah kompleks dan beragam [1]. Dalam membuat sebuah aplikasi, diperlukan fitur-fitur yang saling terintegrasi untuk mendukung jalannya sebuah aplikasi agar lebih nyaman digunakan oleh pengguna. Pengembangan fitur pendukung dalam suatu aplikasi kadang hanya membuat waktu seorang pengembang. Pada saat ini, banyak sekali pihak ketiga yang menawarkan fitur-fitur pendukung jalannya aplikasi, seperti pembayaran secara *cashless*, pengiriman *One Time Password* ke *email* dan nomor ponsel, hingga fitur yang menawarkan informasi yang bersifat pribadi. Tetapi melakukan kerja sama dengan pihak ketiga dapat menciptakan ketergantungan selama masa pengembangan maupun pengujian yang menyebabkan beberapa kendala bagi pengembang. Padahal tahap pengujian merupakan salah satu tahapan yang berhubungan dengan kualitas aplikasi yang akan dibangun [2].

Namun, melihat dari permasalahan yang ditemukan pada saat pengembangan aplikasi *finance* Dhanapala di PT. Semangat Gotong Royong (SGR) ini, aplikasi menggunakan beberapa fitur yang didapatkan dari pihak ketiga, seperti tanda tangan digital, pembayaran menggunakan *virtual account*, dan juga verifikasi data

Diterima Redaksi : 26-07-2020 | Selesai Revisi : 07-08-2020 | Diterbitkan Online : 20-08-2020

diri dari pengguna yang terdaftar. Saat fitur yang dikembangkan telah selesai, pengujian dirasa begitu memakan waktu, karena ada beberapa fitur dari pihak ketiga yang hanya dapat dijalankan pada *environment Staging* (*environment* yang dibentuk untuk pengujian data nyata sebelum masuk ke *environment Production*), dimana harus dilakukan proses *whitelisting* terlebih dahulu agar dapat melakukan request pada pihak ketiga [3]. Sehingga hal ini menyebabkan pengembang kesulitan dalam melakukan pengujian alur jika kedepannya terdapat perubahan.

Oleh karena itu, dengan melihat permasalahan tersebut, maka dikembangkanlah sistem yang disebut dengan *Sandbox* yang merupakan sebuah sistem yang dirancang sedemikian rupa untuk menyerupai karakteristik pada *environment Production*.

Keterbaruan dari penelitian ini adalah pengembangan sebuah sistem *Sandbox* yang mengimplementasi *REST API* sebagai *web service* dan sistem antrian dengan menggunakan *NSQ*. Dalam hal ini, *Sandbox* juga sering digunakan untuk mengeksekusi kode yang belum teruji [4]. Sistem *Sandbox* dibuat menjadi sebuah *web service* berbasis REST API agar dapat melakukan komunikasi antar aplikasi [5][6]. Tentunya hal ini digunakan untuk membantu tahap pengembangan maupun pengujian pengujian, sehingga segala bentuk pemanggilan terhadap pihak ketiga dapat dialihkan ke sistem *Sandbox*. Hal ini dapat mengurangi ketergantungan dan resiko saat memanggil fitur yang ada di pihak ketiga tanpa mengganggu jalannya proses pengembangan dan pengujian aplikasi itu sendiri.

2. Metode Penelitian

Sebelum membahas mengenai metode penelitian yang digunakan dalam penelitian ini, akan disajikan beberapa literatur review dan teori yang mendekati dengan penelitian ini.

Pada penelitian yang dilakukan oleh Putra pada tahun 2019, dikatakan bahwa membuat sebuah lingkungan pengujian terkontrol menggunakan *Sandbox* dapat mendeteksi adanya aktivitas *malware* pada *device* Android. *Sandbox* merupakan sebuah mekanisme keamanan untuk memisahkan program yang sedang berjalan dan biasanya digunakan untuk mengeksekusi program yang sedang dalam tahap pengembangan dan belum teruji. Dikatakan juga bahwa konsep utama dari pembangunan *Sandbox* adalah untuk melakukan pengujian di suatu lingkungan yang dapat dikontrol dengan mudah oleh pengembang sehingga perilaku yang dilakukan oleh program yang diuji dapat dipelajari dan dianalisis secara lebih dalam lagi [4].

Adapaun penelitian lain yang membahas mengenai *Sandbox*, yakni oleh Indradevi, dkk. pada tahun 2018, dikatakan bahwa pembangunan *Sandbox* digunakan untuk memisahkan setiap proses yang akan diuji ataupun proses yang tidak terpercaya untuk dialihkan ke suatu *environment Sandbox* yang terpisah dengan *environment* utamanya, sehingga segala proses yang belum teruji tersebut tidak akan mempengaruhi sistem utama namun pengujian masih dapat dilakukan dengan baik [7].

Setelah mengkaji penelitian mengenai *Sandbox*, dikaji literatur untuk pembangunan *web service* pada sistem *Sandbox*. Pada penelitian yang dilakukan oleh Briones, dkk. pada tahun 2016, dikatakan bahwa RESTful merupakan salah satu implementasi pada arsitektur *Representational State Transfer* (REST) dalam pengembangan *web service*. Arsitektur *RESTful* sendiri berfokus untuk menyediakan metodologi yang sederhana dalam melakukan pemrosesan data melalui *web*. Komunikasi yang dilakukan juga menggunakan protokol *stateless*, dimana segala bentuk data yang akan diproses didapatkan dari permintaan klien sehingga tidak membebani *server*. Adapun juga dalam pengembangan *web service* tersebut digunakan bahasa pemrograman Golang yang mendukung konkurensi, *unit testing* yang terintegrasi dan mudah digunakan, serta merupakan bahasa yang sangat mudah dimengerti manusia namun masih memiliki kecepatan kompilasi yang tinggi [8].

Penelitian juga mengambil literatur mengenai NSQ sebagai alat komunikasi antar layanan pada sistem *Sandbox*. Pada penelitian yang dilakukan oleh Großmann, dkk. pada tahun 2019, dikatakan bahwa penggunaan NSQ sebagai sistem antrian menawarkan pendistribusian pesan secara *realtime* yang tentunya dapat mempermudah komunikasi antar layanan dengan skalabilitas horizontal yang tinggi. NSQ mendukung topologi terdistribusi tanpa adanya titik kegagalan, melakukan komunikasi TCP sehingga lebih *reliable* [9], penggunaan dalam berbagai bahasa pemrograman, dan melakukan komunikasi dengan pola *publish-subscribe* [10].

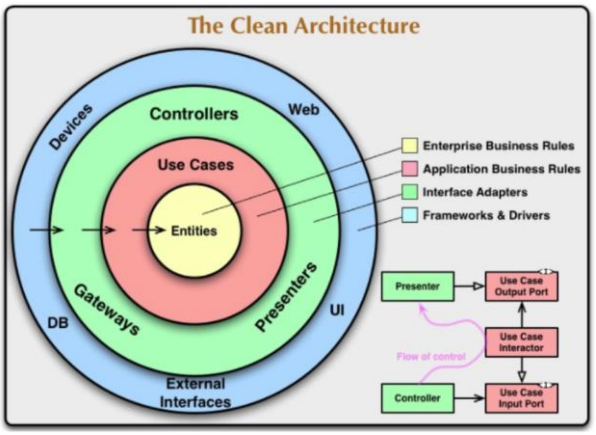
2.1. Golang

Golang (Go *Language*) merupakan sebuah bahasa pemrograman yang mengkombinasikan keamanan dan performa [11] untuk pengembangan sistem yang bersifat *open source* dan dikembangkan di Google oleh Rob Pike, Robert Griesemer, dan Ken Thompson [12] beserta kontributor lainnya dalam komunitas pengembang *open source* [8]. Hingga saat ini, Golang mulai banyak digunakan pada perusahaan-perusahaan besar maupun startup yang bergerak di bidang teknologi. Hal ini dikarenakan pemrograman dengan Golang ini memiliki beberapa kelebihan, antara lain [11][12][13][14]: (1) Mendukung konkurensi dalam sistem pemrograman dengan sangat baik dengan pengaplikasiannya sendiri yang cukup mudah. (2) Merupakan bahasa pemrograman yang bersifat *open source*. (3) Memiliki sistem *garbage collection* yang baik dengan memanfaatkan bantuan *built-in garbage collector process* (Goroutines). (4) Memiliki sintaks yang bersifat bersih, tidak mengotori sistem terlalu berlebihan. (5)

Bahasa pemrograman yang *reliable* dan cepat dalam skala besar.

2.2. *Clean Architecture*

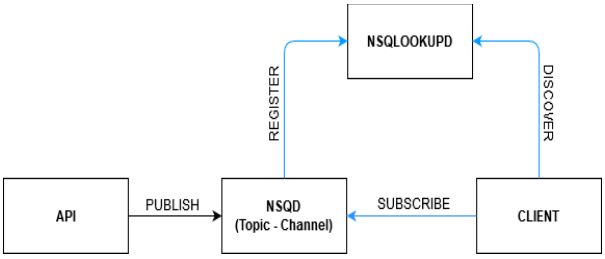
Clean Architecture yang ditunjukkan pada gambar 1 merupakan salah satu arsitektur pemrograman ini yang akan membagi sebuah sistem menjadi beberapa komponen, sehingga lebih tertata dan terlihat jelas tugas dari masing-masing komponennya [15]. Penggunaan *Clean Architecture* sendiri dikarenakan setiap komponen bersifat *independent*, dimana perubahan yang terjadi pada komponen satu tidak akan memberikan dampak pada komponen lainnya. Kunci penting dari arsitektur ini, pengembang sistem harus mengingat betapa pentingnya aturan dependency agar tiap komponen benar mengerjakan tugas mereka masing-masing [16].



Gambar 1. Clean Architecture [16]

2.3. *New Simple Queue* (NSQ)

NSQ atau yang disebut juga dengan *New Simple Queue* merupakan sebuah sistem yang berjalan di belakang layar sebagai sebuah *realtime distributed messaging platform* yang dibuat oleh BitLy dan ditulis dengan bahasa pemrograman Golang. Dalam komunikasinya sendiri, NSQ menggunakan pola *publish-subscribe* untuk mengirim dan menerima pesan [10]. Dimana *publisher* merupakan sistem yang meng-*generate* konten yang akan dikirimkan ke sistem NSQ, dan *subscriber* merupakan sistem yang menginginkan konten tersebut [17]. Pada gambar yang ditunjukkan pada gambar 2, terdapat beberapa komponen dalam NSQ yang mengatur jalannya antrian, antara lain: (1) NSQD – sebuah *daemon* yang bertanggung jawab untuk menerima, mengantrikan, dan mengirimkan pesan ke aplikasi atau servis lainnya [18] yang diatur dalam komponen *Topic(s)* dan *Channel(s)* yang merupakan sebuah *node* tempat menyimpan pesan yang di-*publish*. (2) NSQLOOKUPD yang mengatur topologi jaringan pada sistem NSQ.



Gambar 2. Arsitektur NSQ

Adapun beberapa hal yang membuat NSQ lebih baik daripada sistem antrian lainnya, antara lain [5]: (1) Menawarkan skalabilitas horizontal tinggi. (2) Memberikan kemudahan penggunaan untuk mewujudkan komunikasi antar aplikasi maupun antar servis dengan fitur NSQAdmin yang merupakan antarmuka *realtime* pengelolaan NSQ. (3) Pesan yang di distribusikan bersifat *realtime*. (4) Mendukung penggunaan dalam beberapa bahasa pemrograman. (5) Menjamin pesan akan tersampaikan setidaknya satu kali kepada Klien. (6) Dapat melakukan komunikasi melewati *port* TCP.

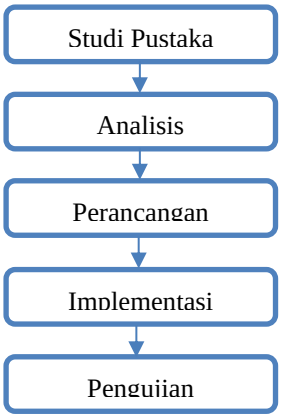
2.4. *Sandbox*

Sandbox adalah sebuah mekanisme keamanan untuk memisahkan program utama dengan program yang akan digunakan untuk mengeksekusi kode yang belum teruji [3]. Secara prinsip, *Sandbox* dapat dikatakan sebuah *service virtualization*, dimana *Sandbox* akan mengimitasi fungsi asli dari suatu produk untuk digunakan pengembang dalam melakukan pengembangan. Beberapa keuntungan penggunaan *Sandbox*, antara lain: (1) Mengurangi ketergantungan terhadap pihak ketiga selama tahap pengembangan. (2) Memungkinkan untuk melakukan pengujian dan pengembangan dalam waktu yang bersamaan untuk mempercepat siklus pengembangan perangkat lunak [19]. (4) Dapat mensimulasikan segala bentuk skenario yang dapat terjadi pada tahap produksi.

2.5. Metode yang diusulkan

Dalam penyusunan penelitian ini dilewati berbagai langkah metodologi penelitian yang ditunjukkan pada gambar 3, seperti: (1) Studi Pustaka - Pada tahapan ini dilakukan pencarian literatur yang berkaitan dengan sistem yang akan dibuat. (2) Analisis - Pada tahapan ini dilakukan analisa mengenai sistem yang akan dibuat, seperti fungsi-fungsi yang terdapat dalam sistem *Sandbox* ataupun fitur yang mungkin akan diberikan. (3) Perancangan - Pada tahapan ini akan dilakukan perancangan sistem *Sandbox* yang sudah dibentuk secara kasar dalam tahap analisis, seperti membuat arsitektur sistem, perancangan antarmuka, pembuatan *use case*, dll. (4) Implementasi - Pada tahapan ini akan dilakukan pengimplementasian atas rancangan yang telah dibuat pada tahap sebelumnya. Hasil dari tahapan ini adalah sebuah sistem *Sandbox* yang sesuai dengan tujuan penelitian. (5) Pengujian - Pada tahapan ini akan

mengevaluasi sistem apakah telah sesuai dengan tujuan penelitian atau belum, ataukah ada kesalahan sistem saat sistem tersebut diintegrasikan dengan aplikasi. Pengujian dilakukan berdasarkan parameter yang telah ditentukan.



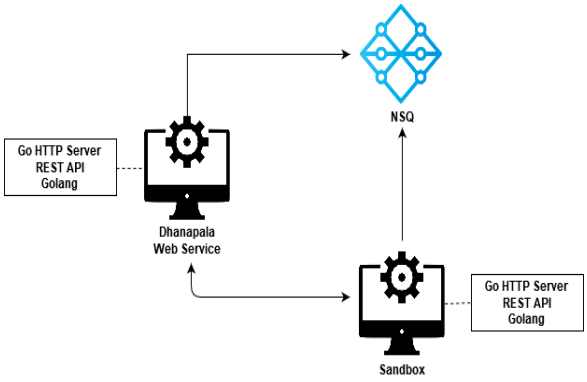
Gambar 3. Metode Penelitian

3. Hasil dan Pembahasan

Pada *section* ini akan dibahas mengenai analisis sistem dan perancangan, serta hasil implementasi dari penelitian ini.

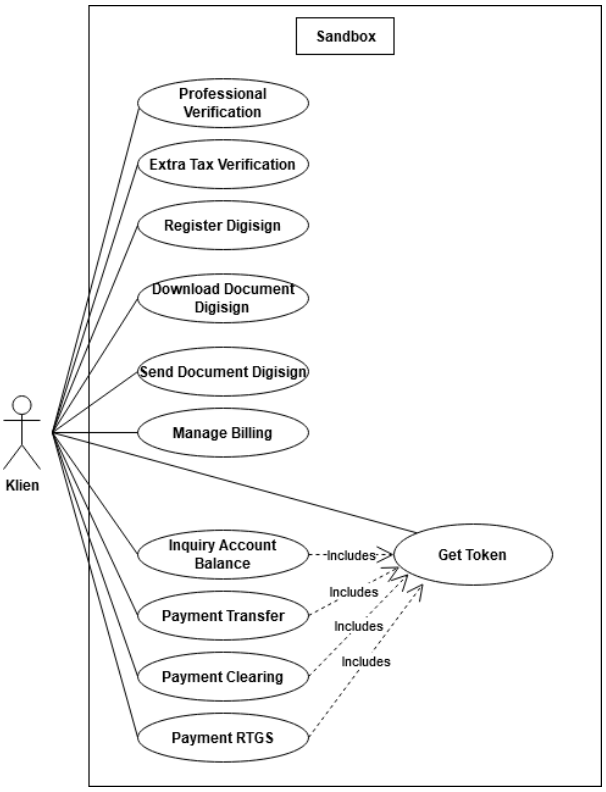
3.1. Analisis dan Perancangan Sistem

Secara garis besar, sistem *Sandbox* yang akan dikembangkan merupakan sebuah REST API yang berjalan pada Go HTTP Server dan ditulis dengan bahasa pemrograman Golang, seperti yang ditunjukkan pada gambar 4. Aplikasi Dhanapala, dapat melakukan komunikasi secara langsung dengan sistem *Sandbox* dengan mengirimkan *request* pada *endpoint Sandbox* yang telah disediakan dan akan mendapatkan *response* dari *Sandbox* secara langsung untuk diproses lebih lanjut. Selain itu Dhanapala juga menggunakan NSQ dalam mengirimkan pesan ke sistem *Sandbox* untuk menggunakan beberapa fitur yang tidak dapat ditoleransi jika terjadi *error*.



Gambar 4. Analisis Sistem

Setelah mengetahui kebutuhan dari sistem ini sendiri, maka dibuatlah sebuah diagram *use case*, seperti yang ditunjukkan pada gambar 5, dimana menunjukkan semua fungsi yang ada pada sistem *Sandbox* ini. Fungsi yang ada pada sistem *Sandbox* hanya mencakup pihak ketiga yang telah melakukan kerja sama dengan aplikasi Dhanapala.

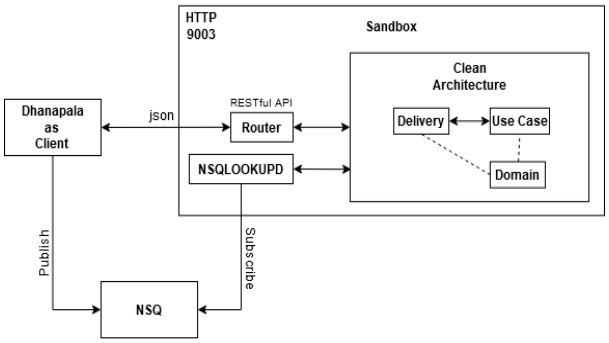


Gambar 5. Diagram *Use Case*

Adapun dilakukannya perancangan sistem untuk menunjukkan lebih dalam lagi bagaimana komunikasi setiap komponen di dalam sistem *Sandbox* ini terjadi. Dalam perancangan sistem *Sandbox*, *Sandbox* akan dibangun sebagai sebuah API berbasis web service yang dirancang dengan arsitektur REST dan dijalankan pada protokol HTTP port 9003. Sehingga komunikasi dari pihak luar dapat dilakukan melalui API ataupun NSQ yang telah disediakan. Pengiriman data juga dilakukan dengan format JSON yang berukuran kecil dan telah didukung oleh banyak bahasa pemrograman [20].

Sistem *Sandbox* dibangun dengan beberapa lapisan, *yang ditunjukkan pada gambar 6*, yang memiliki tugasnya masing-masing. Lapisan Router yang digunakan sebagai jalur akses terhadap fungsionalitas yang terdapat pada sistem *Sandbox* lewat API yang telah disediakan. Lapisan NSQLOOKUPD juga digunakan sebagai jalur akses fungsionalitas namun digunakan untuk melihat secara berkala pesan yang masuk pada NSQ. Setelah melewati Router ataupun NSQLOOKUPD, request akan diteruskan ke fungsionalitas sistem yang dirancang dengan arsitektur clean, dimana terdapat lapisan Delivery, Use Case, dan Domain. Lapisan Delivery merupakan lapisan terluar yang akan menerima request untuk diteruskan ke Use Case dan mengembalikan response yang telah diproses.

Lapisan Use Case merupakan lapisan y ang berisi business logic, dimana segala bentuk pemrosesan data terjadi disini. Lapisan Domain merupakan lapisan yang menyimpan struktur suatu object yang akan digunakan pada lapisan lainnya.



Gambar 6. Arsitektur Sistem

3.2. Implementasi Sistem

Dari perancangan yang telah dibuat, maka dibuatlah sebuah sistem berbasis *web service* dengan satu antarmuka yang dapat menjalankan semua fungsi pada sistem *Sandbox* ini. Adapun untuk menjalankan *web service* yang dijalankan pada server menggunakan satu perintah yang dijalankan pada terminal:

Perintah Menjalankan *Web Service*

```
go run *.go
```

Lalu pada gambar 7 menunjukkan implementasi antarmuka pada sistem *Sandbox*. Antarmuka ini digunakan sebagai pengganti aplikasi Dhanapala untuk melakukan pengujian pada sistem *Sandbox*. Pada bagian kiri terdapat daftar fungsi yang dapat dipilih untuk mengubah pilihan fungsi yang akan dijalankan oleh sistem. Jika ingin menggunakan URL pihak ketiga yang asli, pengguna dapat mematikan fitur *Sandbox* pada *selector* yang disediakan pada bagian atas. Pengguna akan menulis *parameter request* secara pribadi pada *text field* yang telah disediakan ataupun dapat menggu nakan tombol "*Auto Fill*" untuk mengisi *parameter request* secara otomatis. Pengguna juga dapat menggunakan tombol "*Clear Request*" untuk menghapus *parameter* yang telah diisi, ataupun menggunakan tombol "*Clear Response*" untuk menghapus bersih *log response* yang telah didapat. Pengguna juga dapat menggunakan tombol "JSON" untuk melihat *request* dalam format JSON. Jika sudah selesai melengkapi *request*, pengguna dapat menggunakan tombol "*Execute*" untuk melakukan *request* HTTP pada URL yang tertera di bagian kanan atas. Pengguna dapat melihat *log response* sebagai hasil pemrosesan dari URL tersebut pada bagian kanan halaman.

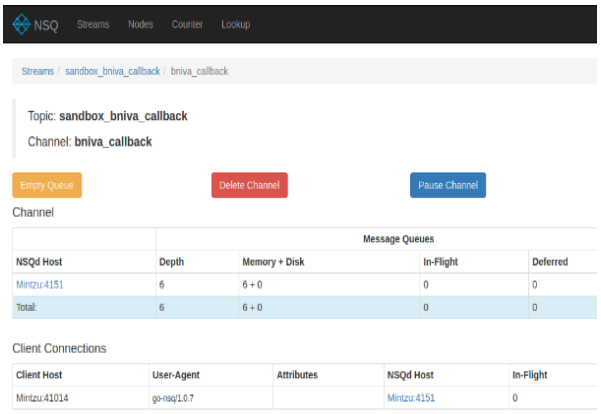
Gambar 7. Implementasi Antarmuka Sistem *Sandbox*

Komunikasi pada sistem *Sandbox* juga dilakukan menggunakan NSQ. Untuk menjalankan NSQ pada *server* diperlukan beberapa perintah pada terminal:

Perintah Menjalankan NSQ

```
nsqlookupd & nsqd --lookupd-tcp-address=127.0.0.1:4160 & nsqadmin --lookupd-http-address=127.0.0.1:4161
```

Setelah menjalankan perintah tersebut, maka pengembang dapat mengakses fitur-fitur yang dimiliki oleh NSQ. Adapun gambar 7 menunjukkan antarmuka NSQAdmin yang digunakan untuk mengelola antrian pada sistem *Sandbox*, dimana pengguna dapat melihat pesan yang terdapat pada antrian, menghapus antrian, ataupun menghapus NSQD yang telah dibentuk.



Gambar 8. NSQAdmin

Sistem *Sandbox* berhasil dibangun menjadi sebuah *web service* berbasis REST API. Penggunaan arsitektur REST membuat sistem *Sandbox* dapat dengan mudah diintegrasikan dengan sistem lainnya. Hal ini dikarenakan REST dapat diakses lewat berbagai protokol, dimana dalam kasus ini menggunakan protokol HTTP. Sistem yang ingin menggunakan servis yang terdapat pada sistem *Sandbox* dapat memanggil URL yang telah disediakan. Selain itu sistem *Sandbox*

juga menggunakan NSQ untuk pengiriman data antar servisnya. Dengan menggunakan NSQ, tentunya data yang dikirimkan lebih aman karena melewati protokol TCP, dan juga akan melakukan *requeue* jika data tidak berhasil terkirimkan.

Sistem *Sandbox* yang meniru karakteristik pihak ketiga memungkinkan sistem yang terintegrasi dengan *Sandbox* sendiri dapat menggunakan beberapa fitur yang mirip dengan pihak ketiga dengan melakukan pengiriman data ke URL yang telah disediakan. Hal ini tentunya dapat dilakukan pada *environment* manapun tanpa harus memikirkan *token* ataupun hak akses yang dibutuhkan seperti pada saat memanggil fitur pada pihak ketiga yang sebenarnya. Kebutuhan data terhadap pihak ketiga pun dapat terpenuhi karena hasil *response* yang diberikan oleh sistem *Sandbox* memiliki struktur yang sama dengan pihak ketiga.

4. Kesimpulan

Dari penelitian yang dilakukan, (1) Sistem *Sandbox* dapat membantu proses pengujian maupun pengembangan yang berhubungan dengan pihak ketiga karena memiliki karakteristik struktur data yang sama dengan pihak ketiga dari segi *request* maupun *response*. (2) Sistem *Sandbox* dapat mengurangi ketergantungan terhadap pihak ketiga selama tahap pengembangan maupun pengujian karena dapat digunakan pada *environment* manapun dan dapat diakses oleh sistem lain dengan mudah tanpa perlu dilakukannya *whitelisting* sistem terlebih dahulu.

Adapun beberapa saran yang dapat diberikan, ada baiknya jika sistem *Sandbox* dapat dikembangkan setiap fungsionalitasnya agar dapat diatur untuk mengembalikan *response* yang *error*, sehingga pengguna sistem *Sandbox* juga dapat mensimulasikan kasus yang berpotensi *error* juga. Komunikasi sistem *Sandbox* dengan Dhanapala juga lebih baik dilakukan pada protokol HTTPS dan menggunakan akses *token*, sehingga data yang dikirim terenkripsi dan lebih menyerupai komunikasi dengan pihak ketiga.

Daftar Rujukan

[1] A. Umar, R. Pakaya, and I. Karim, "Estimasi Perhitungan Bandwidth Untuk Aksesibilitas Aplikasi Berbasis Web," *J. Teknol. Inf. Indones.*, vol. 2, no. 2, pp. 6–9, 2017.

[2] E. Rosi Subhiyakto and D. Wahyu Utomo, "Strategi, Teknik, Faktor Pendukung Dan Penghambat Pengujian Untuk Pengembang Perangkat Lunak Pemula," *Semin. Nas. Teknol. Inf. dan Komun.*, vol. 2016, no. Sentika, pp. 2089–9815, 2016.

[3] C. Sashikanth, W. Yilei, P. Adiddam, X. Zhihong, and C. Varouj, "Network Security Based On Proximity With IP Whitelisting," 2017.

[4] R. L. Putra, "Analisis Aktivitas Malware Pada Ram Android Dan Sandbox Environment," 2019.

[5] S. Al Ghozaly and E. I. Sela, "Implementasi Rest Api Pada Pusat Informasi Mahasiswa Universitas Teknologi Yogyakarta," 2019.

[6] Y. Harjoseputro, Y. D. Handarkho, and H. T. R. Adie, "The Javanese Letters Classifier With Mobile Client-Server Architecture And Convolution Neural Network Method," *Int. J. Interact. Mob. Technol.*, vol. 13, no. 12, pp. 67–80, 2019.

[7] K. A. R. Indradevi, P. Sukarno, and E. M. Jadied, "Analisis Performansi Aplikasi Sandbox pada Sistem Operasi Windows," in *eProceedings of Engineering*, 2018, vol. 5, no. 3, pp. 7536–7543.

[8] D. Igou and A. Throckmorton, "RESTful API Framework Golang Proof of Concept," 2016.

[9] A. Qonita, "Layanan Dari TCP dan UDP Protocol," 2017.

[10] M. Großmann, S. Illig, and C. L. Matějka, "SensIoT: An extensible and General Internet of Things Monitoring Framework," *Wirel. Commun. Mob. Comput.*, vol. 2019, pp. 1–15, 2019.

[11] I. Norwandi, W. Suadi, and B. A. Pratomo, "Implementasi Database Abstraction Layer untuk MySQL Menggunakan Google Go," 2011.

[12] M. D. Lusita, H. Hurnianingsih, and E. Rihyanti, "Aplikasi Bot Akademik BAAK STMIK Jakarta STI&K Platform Line Messenger Menggunakan Go Languages," *J. Teknol. Sist. Inf. dan Apl.*, vol. 3, no. 1, p. 1, 2020.

[13] W. Ananda, M. Arif, and F. Ridha, "Pengembangan Cloud Computing Platform As A Service Untuk Bahasa Pengembangan Cloud Computing Platform As A Service Untuk Bahasa Pemrograman Go," *J. Aksara Komput. Terap.*, vol. 5, no. 2, 2016.

[14] A. Dunan and E. Prihantoro, "Interaksi Universitas-Pemerintah-Industri Dalam Inovasi Inkubator Bisnis: Studi Kasus Pada Universitas Gajah Mada Yogyakarta," *J. Masy. Telemat. dan Inf.*, vol. 7, no. 2, pp. 135–144, 2016.

[15] Z. Lin, "Towards a Clean Architecture For TechLauncher Projects," 2019.

[16] Tung Bui Du, "Reactive Programming and Clean Architecture in Android Development," 2017.

[17] V. Ramasubramanian, R. Peterson, and E. G. Sirer, "Corona: A high performance publish-subscribe system for the world wide web," *Proc. Networked Syst. Des. Implement. NSDI*, pp. 15–28, 2006.

[18] S. Raje, "Performance Comparison of Message Queue Methods," 2019.

[19] D. Subbiah, B. Arulmozhi, and H. Maruthamuthu, "Constraint Free Testing using Service Virtualization," *Int. J. Comput. Appl.*, vol. 105, no. 17, pp. 14–17, 2014.

[20] D. S. Wiyono and A. Wijayanto, "Implementasi Rest Web Service Dengan Menggunakan Json Pada Aplikasi Mobile Enterprise Resource Planning," *PERFORMA Media Ilm. Tek. Ind.*, vol. 11, no. 2, pp. 143–152, 2012.