PEMBANGUNAN SISTEM SANDBOX PADA PENGUJIAN TERHADAP PIHAK KETIGA BERBASIS REST API MENGGUNAKAN GOLANG DAN NSQ

Tugas Akhir

Diajukan untuk Memenuhi Salah Satu Persyaratan Mencapai Derajat Sarjana Informatika

Dibuat Oleh:

ALBERTUS ARI KRISTANTO

16 07 08916

PROGRAM STUDI INFORMATIKA

FAKULTAS TEKNOLOGI INDUSTRI

UNIVERSITAS ATMA JAYA YOGYAKARTA

2020

HALAMAN PENGESAHAN

Tugas Akhir Berjudul

PEMBANGUNAN SISTEM SANDBOX PADA PENGUJIAN TERHADAP PIHAK KETIGA BERBASIS REST API MENGGUNAKAN GOLANG DAN NSQ

yang disusun oleh

ALBERTUS ARI KRISTANTO

160708916

dinyatakan telah memenuhi syarat pada tanggal 06 Juli 2020

		Keterangan
Dosen Pembimbing 1	: Yulius Harjoseputro, ST., MT.	Telah menyetujui
Dosen Pembimbing 2	: Joseph Eric Samodra, S.Kom, MIT.	Telah menyetujui
Tim Penguji		
Penguji 1	: Yulius Harjoseputro, ST., MT.	Telah menyetujui
Penguji 2	: Thomas Adi Purnomo Sidhi, ST., MT.	Telah menyetujui
Penguji 3	: Eduard Rusdianto, ST., MT.	Telah menyetujui

Yogyakarta, 06 Juli 2020

Universitas Atma Jaya Yogyakarta

Fakultas Teknologi Industri

Dekan

ttd

Dr. A. Teguh Siswantoro, M.Sc

PERNYATAAN ORISINALITAS & PUBLIKASI ILMIAH

Saya yang bertanda tangan di bawah ini:

Nama Lengkap	: Albertus Ari Kristanto
NPM	: 16 07 08916
Program Studi	: Informatika
Fakultas	: Teknologi Industri
Judul Penelitian	: Pembangunan Sistem Sandbox pada Pengujian Terhadap Pihak Ketiga Berbasis REST API Menggunakan Golang dan NSQ

Menyatakan dengan ini:

1. Tugas Akhir ini adalah benar tidak merupakan salinan sebagian atau keseluruhan dari karya penelitian lain.
2. Memberikan kepada Universitas Atma Jaya Yogyakarta atas penelitian ini, berupa Hak untuk menyimpan, mengelola, mendistribusikan, dan menampilkan hasil penelitian selama tetap mencantumkan nama penulis.
3. Bersedia menanggung secara pribadi segala bentuk tuntutan hukum atas pelanggaran Hak Cipta dalam pembuatan Tugas Akhir ini.

Demikianlah pernyataan ini dibuat dan dapat dipergunakan sebagaimana mestinya.

Yogyakarta, 24 Juni 2020

Yang menyatakan,

Albertus Ari Kristanto

16 07 08916

PERNYATAAN PERSETUJUAN DARI INSTANSI ASAL PENELITIAN

Saya yang bertanda tangan di bawah ini:

Nama Lengkap Pembimbing : Lucky Ong

Jabatan : Engineering Manager

Departemen : Financial Technology

Menyatakan dengan ini:

Nama Lengkap : Albertus Ari Kristanto

NPM : 16 07 08916

Program Studi : Informatika

Fakultas : Teknologi Industri

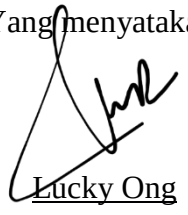
Judul Penelitian : Pembangunan Sistem Sandbox pada Pengujian Terhadap Pihak Ketiga Berbasis REST API Menggunakan Golang dan NSQ

1. Penelitian telah selesai dilaksanakan pada perusahaan.
2. Perusahaan telah melakukan sidang internal berupa kelayakan penelitian ini dan akan mencantumkan lembar penilaian secara tertutup kepada pihak universitas sebagai bagian dari nilai akhir mahasiswa.
3. Memberikan kepada Instansi Penelitian dan Universitas Atma Jaya Yogyakarta atas penelitian ini, berupa hak untuk menyimpan, mengelola, mendistribusikan, dan menampilkan hasil penelitian selama tetap mencantumkan nama penulis.

Demikianlah pernyataan ini dibuat dan dapat dipergunakan sebagaimana mestinya.

Jakarta, 24 Juni 2020

Yang menyatakan,



Lucky Ong

Engineering Manager

HALAMAN PERSEMBAHAN

“

Tugas Akhir ini tentunya kupersembahkan untuk

Semua pihak yang telah membantu proses penyusunan

Terutama Keluarga

Teman

Dan

Tentunya Pacar

Oh iya..

Ada pesan dariku..

Teruntuk Aku di masa yang akan datang

Begitu banyak hal yang telah Kamu lalui selama ini

Jika satu saat nanti Kamu berpikir untuk menyerah

Lihatlah ini dan lihatlah betapa besar perjuanganmu

Dan janganlah lupa akan orang-orang yang telah berjuang bersamamu

God bless you

”

KATA PENGANTAR

Puji dan syukur penulis haturkan kepada Tuhan Yang Maha Esa karena berkat rahmat dan karunia-Nya penulis dapat menyelesaikan pembuatan tugas akhir "Pembangunan Sistem Sandbox pada Pengujian Terhadap Pihak Ketiga Berbasis REST API Menggunakan Golang dan NSQ" ini dengan baik.

Penulisan tugas akhir ini bertujuan untuk memenuhi salah satu syarat untuk mencapai derajat sarjana Informatika dari Program Studi Informatika, Fakultas Teknologi Industri di Universitas Atma Jaya Yogyakarta.

Penulis menyadari bahwa dalam pembuatan tugas akhir ini penulis telah mendapatkan bantuan, bimbingan, dan dorongan dari banyak pihak. Untuk itu, pada kesempatan ini penulis ingin mengucapkan terima kasih kepada:

1. Tuhan Yesus Kristus yang selalu membimbing dalam iman-Nya, memberikan berkat-Nya, dan menyertai penulis selalu.
2. Bapak Dr. A. Teguh Siswantoro, M.Sc., selaku Dekan Fakultas Teknologi Industri, Universitas Atma Jaya Yogyakarta.
3. Bapak Yulius Harjoseputro, S.T., M.T., selaku dosen pembimbing I yang telah membimbing dan memberikan masukan serta motivasi kepada penulis untuk menyelesaikan tugas akhir ini.
4. Bapak Joseph Eric Samodra, S.Kom., MIT., selaku dosen pembimbing II yang telah membimbing dan memberikan masukan serta motivasi kepada penulis untuk menyelesaikan tugas akhir ini.

Demikian laporan tugas akhir ini dibuat, dan penulis mengucapkan terima kasih kepada semua pihak. Semoga laporan ini dapat bermanfaat bagi pembaca.

Yogyakarta, 24 Juni 2020

Albertus Ari Kristanto

16 07 08916

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

INTISARI

PEMBANGUNAN SISTEM SANDBOX PADA PENGUJIAN TERHADAP PIHAK KETIGA BERBASIS REST API MENGGUNAKAN GOLANG DAN NSQ

Albertus Ari Kristanto

16 07 08916

Perkembangan teknologi di era digital membuat begitu banyak orang melakukan pengembangan aplikasi yang bertujuan untuk membantu keberlangsungan hidup manusia. 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, seperti aplikasi yang dikembangkan di PT. Semangat Gotong Royong (SGR) ini, Dhanapala. Terdapat beberapa fungsi yang memerlukan data dari pihak ketiga namun tidak bisa dilakukan pada *environment* lokal. Hal ini membuat pengembang mengalami kesulitan dalam melakukan pengembangan maupun pengujian.

Permasalahan tersebut menjadi dasar pengembangan sistem Sandbox yang akan dikembangkan. Sistem Sandbox merupakan sebuah sistem yang dirancang sedemikian rupa untuk menyerupai karakteristik pada lingkungan produksi dari pihak ketiga. Sistem Sandbox akan dikembangkan menjadi sebuah *web service* API yang dirancang dengan arsitektur REST dan ditulis menggunakan bahasa pemrograman Golang. Dalam melakukan komunikasi dengan perangkat lain pun digunakan NSQ yang dapat mendukung konkurensi dan mencegah kegagalan pengiriman data.

Hasilnya, sistem Sandbox yang merupakan *service virtualization* dapat menerima *request* dari aplikasi Dhanapala dan akan memproses *response* yang mirip dengan fungsi dari pihak ketiga. Pengembangan dan pengujian pada aplikasi Dhanapala dapat dilakukan lebih mudah setelah sistem Sandbox terintegrasi dengan aplikasi Dhanapala, 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

Dosen Pembimbing I : Yulius Harjoseputro, S.T., M.T.

Dosen Pembimbing II : Joseph Eric Samodra, S.Kom., MIT.

Jadwal Sidang Tugas Akhir : 6 Juni 2020

BAB I. PENDAHULUAN

1.1. Latar Belakang

Kemajuan teknologi di era digital ini membuat semua orang mendambakan aplikasi yang dapat membantu segala pekerjaan mereka tanpa kekurangan suatu apapun. Hal inilah yang menjadi tantangan bagi para pengembang aplikasi untuk menciptakan aplikasi yang dapat berjalan secara sempurna tanpa adanya galat yang menganggu pengguna saat menggunakan aplikasi tersebut. Dalam melakukan pengembangan aplikasi mencakup beberapa tahapan seperti pengumpulan kebutuhan, analisis, perancangan, implementasi, pengujian, dan pemeliharaan. Tahap pengujian merupakan salah satu tahapan yang berhubungan dengan kualitas aplikasi yang akan dibangun [1]. Maka dari itu, pengembangan aplikasi harus melewati tahap pengujian terlebih dahulu hingga tidak menemukan galat sebelum aplikasi tersebut diluncurkan ke publik. Hal ini sangat penting untuk meningkatan *user experience* atau pengalaman yang akan didapatkan pengguna saat menggunakan aplikasi tersebut.

Mengenai perkembangan aplikasi sendiri, ide dan inovasi yang dimiliki para pengembang aplikasi pada saat ini sangatlah kompleks dan beragam [2]. Poin penting yang selalu dilakukan oleh para pengembang adalah memfokuskan pada pengembangan fitur spesial yang akan ada di aplikasinya. Manajemen waktu merupakan suatu aspek penting dalam pengembangan aplikasi, melakukan pengembangan fitur sampingan dapat memakan waktu yang berlebih. Maka dari itu, banyak dari para pengembang memutuskan untuk melakukan beberapa kerja sama untuk menghadirkan fitur sampingan tersebut. Pada saat ini, banyak sekali *third-party* atau pihak ketiga yang menawarkan fitur-fitur sampingan yang mendukung jalannya aplikasi, seperti pembayaran secara *cashless*, pengiriman *One Time Password* ke *email* dan nomor ponsel, hingga fitur yang menawarkan informasi yang bersifat privasi.

Tentunya melakukan kerja sama dengan pihak ketiga dapat menciptakan ketergantungan selama masa pengembangan. Dalam mencapai titik kepuasan yang

maksimal bagi pengguna, pengujian integrasi pemanggilan fitur pihak ketiga pada aplikasi yang dikembangkan dapat dibilang mengeluarkan usaha lebih dalam melakukan setiap kasus ujinya. Hanya untuk melakukan pengujian sendiri saja dibutuhkan kasus uji yang cukup banyak, dari kasus positif hingga kasus negatif pun perlu diuji untuk mengetahui apakah integrasi terhadap pihak ketiga telah sempurna atau belum.

Melihat dari permasalahan yang ditemukan pada saat pengembangan aplikasi kredit 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 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*). Sehingga hal ini menyebabkan pengembang kesulitan dalam melakukan pengujian alur jika kedepannya terdapat perubahan.

Maka dari itu dikembangkanlah sistem yang disebut dengan Sandbox yang merupakan sebuah sistem yang dirancang sedemikian rupa untuk menyerupai karakteristik pada lingkungan produksi dimana lingkungan tersebut merupakan lingkungan saat aplikasi rilis ke publik. Sandbox juga sering digunakan untuk mengeksekusi kode yang belum teruji [3]. Tentunya hal ini digunakan untuk membantu tahap 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 aplikasi itu sendiri.

Sistem ini pun akan dibentuk dengan berbasis sebuah *web service* API yang dirancang dengan arsitektur REST dan ditulis menggunakan bahasa pemrograman Golang yang dikenal sebagai bahasa pemrograman efisien, ringan, dan memiliki kemampuan konkurensi yang baik [4]. Salah satu penerapan konkurensinya pun menggunakan sebuah sistem antrian bernama NSQ yang mendukung penggunaannya dengan Golang. Mengingat pembangunan sistem Sandbox akan

dibuat terpisah dengan aplikasi Dhanapala, maka sistem antrian NSQ ini akan dimanfaatkan sebagai alat komunikasi antar sistem yang menawarkan skalabilitas yang tinggi [5].

1.2. Rumusan Masalah

Berdasarkan latar belakang permasalahan, maka dapat diidentifikasikan permasalahan yang akan dibahas adalah sebagai berikut:

1. Bagaimana sistem Sandbox dapat menyerupai karakteristik fungsionalitas pihak ketiga?
2. Bagaimana sistem Sandbox dapat mengurangi ketergantungan aplikasi Dhanapala terhadap pihak ketiga dalam tahap pengembangan maupun pengujian?

1.3. Batasan Masalah

Berdasarkan permasalahan diatas, maka pengembangan sistem ini akan dibatasi pada beberapa hal, antara lain:

1. Sistem Sandbox yang dibuat hanyalah digunakan untuk menyerupai pihak ketiga yang telah melakukan kerjasama dengan aplikasi yang dikembangkan. Sehingga sistem Sandbox yang dibuat didasari pada dokumentasi yang diberikan pihak ketiga. Maka dari itu, jika pihak ketiga melakukan perubahan terhadap fitur yang diberikan, sistem yang dikembangkan juga harus mengikuti perubahan tersebut.
2. Sistem Sandbox yang dibuat merupakan sebuah *Representational State Transfer Application Programming Interface* (REST API) yang ditulis menggunakan bahasa Golang dengan arsitektur *clean* dan menggunakan sistem antrian bernama *New Simple Queue* (NSQ).
3. Pengujian sistem Sandbox akan dilakukan dengan antarmuka yang berbeda dengan kasus nyata, hal ini dikarenakan aplikasi Dhanapala tidak dapat digunakan untuk keperluan tugas akhir yang berada diluar area perusahaan.

4. Sistem Sandbox merupakan sistem yang dibuat berdasarkan studi kasus yang ditemukan pada PT. Semangat Gotong Royong (SGR).

1.4. Tujuan Penelitian

Dalam penyusunan penelitian ini terdapat beberapa tujuan penting yang diharapkan dapat tercapai, seperti:

1. Sistem Sandbox yang dikembangkan dapat menyerupai karakteristik fungsionalitas pihak ketiga.
2. Sistem Sandbox yang dikembangkan mampu mengurangi ketergantungan aplikasi Dhanapala terhadap pihak ketiga selama tahap pengembangan maupun pengujian.

1.5. Metode Penelitian

Dalam penyusunan penelitian ini dilewati berbagai langkah metodologi penelitian, seperti:

1. Studi Pustaka

   Pada tahapan ini dilakukan pencarian literatur yang berkaitan dengan sistem yang akan dibuat. Pada penelitian ini, literatur yang dicari berpusat pada pengembangan sistem Sandbox yang telah ada sebelumnya. Selain itu juga diperlukan literatur yang dibutuhkan penulis untuk membuat sistem ini sendiri seperti arsitektur REST, bahasa pemrograman Golang, dan sistem antrian NSQ.
2. Analisis

   Pada tahapan ini dilakukan analisa mengenai sistem yang akan dibuat, seperti fungsi-fungsi yang terdapat dalam sistem Sandbox, fitur yang mungkin akan diberikan, bagaimana cara sistem Sandbox ini menjadi lebih mudah digunakan untuk membantu pengujian aplikasi, serta menganalisis data-data apa saja yang akan digunakan di dalam sistem ini pada dokumentasi yang diberikan oleh pihak ketiga.

3. Perancangan

   Pada tahapan ini akan dilakukan perancangan sistem Sandbox yang sudah dibentuk secara kasar dalam tahap analisis. Perancangan meliputi pembuatan fungsi-fungsi yang terdapat dalam sistem, serta melakukan integrasi dari sistem Sandbox ke aplikasi yang menggunakan fitur pihak ketiga.

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 dilakukan pengujian sistem yang telah dibuat pada tahap implementasi. Pengujian dilakukan untuk 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. Hasil dari tahapan ini dijadikan evaluasi untuk penelitian selanjutnya.

1.6. Sistematika Penelitian

Dalam penyusunan penelitian ini dilewati berbagai langkah metodologi penelitian, seperti:

PENDAHULUAN

Pada bab ini akan dibahas mengenai latar belakang, rumusan beserta batasan masalah yang akan diteliti, tujuan penelitian dan metode yang dilakukan dalam penelitian ini.

TINJAUAN PUSTAKA

Pada bab ini akan dibahas mengenai literatur-literatur penelitian terdahulu yang digunakan untuk membantu penulis dalam melakukan pengembangan sistem. Literatur yang dibahas seperti penelitian mengenai Sandbox, arsitektur *web service*, dan bahasa pemrograman akan digunakan. Dibagian

ini juga terdapat perbandingan yang membedakan penelitian yang ditulis dengan penelitian yang sudah ada sebelumnya.

LANDASAN TEORI

Pada bab ini akan dibahas mengenai teori penunjang penelitian seperti konsep Sandbox, REST API sebagai arsitektur *web service*, bahasa pemrograman Golang yang digunakan dalam pengembangan sistem, sistem antrian NSQ, dan juga membahas konsep pengujian perangkat lunak selama tahap pengembangan.

ANALISIS DAN PERANCANGAN SISTEM

Pada bab ini akan dibahas mengenai analisis kebutuhan dan perancangan sistem yang akan dibuat oleh penulis.

IMPLEMENTASI DAN PENGUJIAN SISTEM

Pada bab ini akan dibahas mengenai pengimplementasian dan pengujian sistem yang dibuat dalam kasus nyata yang dilakukan oleh penulis.

PENUTUP

Pada bab ini akan dituliskan mengenai kesimpulan dan saran dari pengembangan sistem yang telah dilakukan oleh penulis.

BAB II. TINJAUAN PUSTAKA

Di dalam melakukan sebuah penelitian, tentunya diperlukan teori-teori sebagai aspek pendukung yang didasari oleh literatur-literatur penelitian yang sudah ada sebelumnya. Hal ini diperlukan agar dapat memaksimalkan penelitian dengan meneliti lebih dalam hal-hal yang telah diteliti sebelum-sebelumnya dan menjauhkan penelitian dari kesalahan yang dulunya pernah terjadi sehingga tidak terjadi kembali. Adapun beberapa literatur yang akan ditinjau akan membahas mengenai Sandbox, arsitektur dalam pengembangan sebuah *web service*, dan bahasa pemrograman yang digunakan dalam pengembangan.

Literatur yang pertama adalah sebuah penelitian yang ditulis oleh Rizky Laksamana Putra pada tahun 2019 dengan judul penelitiannya adalah "Analisis Aktivitas Malware Pada RAM Android dan Sandbox Environment". Di dalam literatur tersebut, peneliti membuat sebuah lingkungan pengujian terkontrol menggunakan Sandbox untuk 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 [3].

Penerapan yang tidak jauh berbeda pun dilakukan oleh Indradevi, dkk. dalam literatur penelitian yang dilakukan pada tahun 2018 dengan judul "Analisis Performansi Aplikasi Sandbox pada Sistem Operasi Windows". Dalam literatur tersebut dijelaskan bahwa pembangungan 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 [6]. Kedua literatur diatas memiliki banyak kesamaan dalam hal penerapan Sandbox beserta penjelasan

penelitiannya, namun Sandbox yang dibuat dalam penelitian tersebut sedikit berbeda dengan apa yang akan dibuat oleh penulis. Kedua penelitian tersebut menggunakan aplikasi Sandbox yang terdapat pada sistem operasi Linux dan Windows untuk meneliti perilaku proses yang tidak terpercaya, sedangkan penulis akan membuat sistem Sandbox sendiri yang berupa *web service* untuk mempelajari perilaku fungsi yang dijalankan sebelum dijalankan pada *environment* utama.

Dalam membangun sistem Sandbox yang berbasis *web service*, diperlukan juga untuk mengetahui perkembangan arsitektur yang digunakan dalam membangun *web service* itu sendiri. Dalam literatur yang dikemukakan oleh Gerard Briones, David Igou, dan Aaron Throckmorton pada tahun 2016 dengan judul "*RESTful API Framework: Golang Proof of Concept*", 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 peneliti menggunakan bahasa pemrograman Golang yang dikembangkan oleh Google dan pengembang lainnya karena bersifat *open source*. Penggunaan Golang didasari dengan begitu banyaknya kelebihan yang dimiliki daripada bahasa pemrograman lainnya, seperti konkurensi yang menjadi fitur *built-in*, *unit testing* yang terintegrasi dan mudah digunakan, serta merupakan bahasa yang sangat mudah dimengerti manusia namun masih memiliki kecepatan kompilasi yang tinggi [4]. Literatur ini sangat baik untuk menjadi referensi pengembangan sistem yang akan dibuat oleh penulis karena menggunakan arsitektur *web service* dan bahasa pemrograman yang sama.

Sistem Sandbox yang akan dibangun juga dapat melakukan komunikasi dengan layanan lainnya, maka dari itu dibutuhkan sebuah komunikasi yang dapat menghantarkan pesan antar layanan dengan memastikan pesan tersebut diantrikan sebelum layanan yang menerima memproses pesan tersebut. Sistem antrian sebagai komunikasi antar layanan ini juga dibahas pada literatur yang ditulis oleh Marcel

Großmann, dkk. pada tahun 2019 dengan judul "SensIoT: *An extensible and General Internet of Things Monitoring Framework*", dimana penelitian tersebut mengimplementasikan NSQ sebagai sistem antriannya. 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 [5], jika ada pesan yang gagal terkirim maka pesan tersebut akan dimasukkan kembali antrian, sehingga hal ini dapat mengurangi kegagalan pengiriman data. Selain itu NSQ juga mendukung skalabilitas horizontal yang tinggi tanpa adanya perantara untuk menambahkan titik baru, komunikasi lewat TCP yang aman, dan juga mendukung penggunaan dalam beberapa bahasa pemrograman. NSQ berkomunikasi dengan mengimplementasikan pola *publish-subscribe* [5]. Layanan yang ingin mengirimkan pesan dapat melakukan *publish* pesan tersebut ke sistem NSQ, dan bagi layanan lainnya yang ingin mendapatkan pesan tersebut dapat melakukan *subscribe* ke topik NSQ terkait. Literatur ini dapat menjadi referensi yang baik karena menjelaskan dengan sangat dalam mengenai NSQ, seperti komponen yang terdapat pada NSQ hingga arsitektur NSQ dalam melakukan komunikasi. Namun literatur ini tidak membahas mengenai Sandbox dan bahasa pemrograman yang digunakan, Golang.

Tabel 2.1. Tabel Perbandingan Penelitian

NO	PENELITI	JUDUL	SANDBOX	ARSITEKTUR API	BAHASA PEMROGRAMAN	SISTEM ANTRIAN	HASIL PENELITIAN
1	[3]	Analisis Aktivitas Malware Pada RAM Android dan Sandbox Environment	Ya	-	-	-	Sandbox berhasil dibangun dan terintegrasi dengan baik sebagai wadah eksekusi dan analisis *sample.*
2	[6]	Analisis Performansi Aplikasi Sandbox pada Sistem Operasi Windows	Ya	-	-	-	Memaparkan hasil perbandingan setiap aplikasi Sandbox yang digunakan pada sistem operasi Windows.
3	[4]	*RESTful API Framework: Golang Proof of Concept*	-	REST	Golang	-	Mengembangkan sistem RESTful API menggunakan Go yang berfungsi untuk menyederhanakan akses *website* ke Capital One.

4	[5]	*SensIoT: An Extensible and General Internet of Things Monitoring Framework*	-	-	-	NSQ	SensIoT menjalankan beberapa NSQ dan *consumer* untuk menghasilkan skalabilitas tinggi bahkan jika salah satu perangkat tidak bisa beroperasi.
5	Kristanto (2020) *	PENGEMBANGAN SISTEM SANDBOX PADA PENGUJIAN TERHADAP PIHAK KETIGA BERBASIS REST API MENGGUNAKAN GOLANG DAN NSQ (STUDI KASUS PT. SEMANGAT GOTONG ROYONG)	Ya	REST	Golang	NSQ	Berhasil membangun sistem Sandbox yang memiliki karakteristik yang mirip dengan pihak ketiga yang digunakan oleh aplikasi Dhanapala yang dikembangkan di PT. SGR ini.

* penelitian yang dilakukan

BAB III. LANDASAN TEORI

Adapun akan dikemukakan beberapa teori yang berkaitan dengan permasalahan dan ruang lingkup pembahasan sebagai dasar dalam penelitian ini.

3.1. Golang

Pemrograman dengan bahasa Go atau yang sering disebut dengan Golang (*Go Language*) merupakan sebuah bahasa pemrograman untuk pengembangan sistem yang bersifat *open source* dan dikembangkan oleh Google beserta kontributor lainnya dalam komunitas pengembang *open source* [4]. Secara sintaks, Golang dapat dibilang sangat menyerupai karakteristik bahasa pemrograman C dan C++ yang efisien dan ringan. Golang sendiri juga tidak menggunakan konsep *Object Oriented Programming* (OOP) sepenuhnya, sehingga pengguna Golang dapat dengan mudah membuat programnya tanpa memikirkan segmentasi-segmentasi seperti pada konsep OOP [7].

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 [7]:

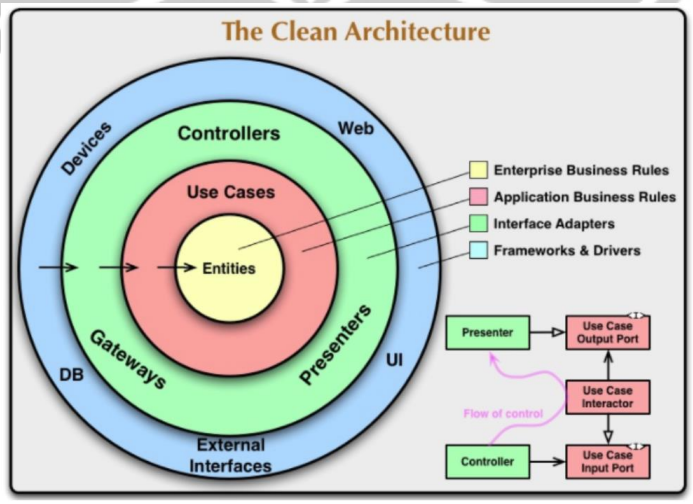
a. Mendukung konkurensi dalam sistem pemrograman dengan sangat baik dengan pengaplikasiannya sendiri yang cukup mudah. Sehingga Golang mampu melakukan beberapa proses yang berbeda-beda di waktu yang bersamaan (*parallel processing*).
b. Merupakan bahasa pemrograman yang bersifat *open source,* sehingga banyak sekali pengembang lain yang ikut membantu membuat modul-modul tambahan yang dapat dimanfaatkan publik.
c. Memiliki sistem *garbage collection* yang baik dengan memanfaatkan bantuan *built-in garbage collector process* (Goroutines). Golang sangat membantu dalam manajemen Goroutines dan manajemen memori. Hal ini

juga mendukung konkurensi, karena konkurensi dapat mudah dilakukan dengan adanya sistem *garbage collection* ini.

d. Memiliki sintaks yang bersifat bersih, tidak mengotori sistem terlalu berlebihan. Golang tidak menggunakan konsep hirarkis seperti OOP yang harus membentuk hirarki terlebih dahulu hingga program dapat dijalankan. Sintaks yang digunakanpun sangat singkat dan sangat mudah dipahami karena Golang mempunyai prinsip *Reduce Typing*. Maka dari itu Golang dapat dibilang menjadi bahasa pemrograman yang ringan dan efisien untuk dijalankan pada sistem manapun dengan waktu kompilasi yang minimum.

3.2. Clean Architecture

Begitu banyak arsitektur pemrograman yang dikemukakan untuk membuat sebuah sistem yang baik. Setiap arsitektur pemrograman memiliki tujuan yang sama, yaitu membuat sebuah sistem menjadi terbagi ke dalam beberapa bagian yang berbeda, sehingga setiap bagian dapat melakukan tugasnya masing-masing dan menjadi lebih tertata. *Clean Architecture* merupakan salah satu arsitektur pemrograman yang memiliki sifat *independent* untuk setiap komponen di dalamnya.

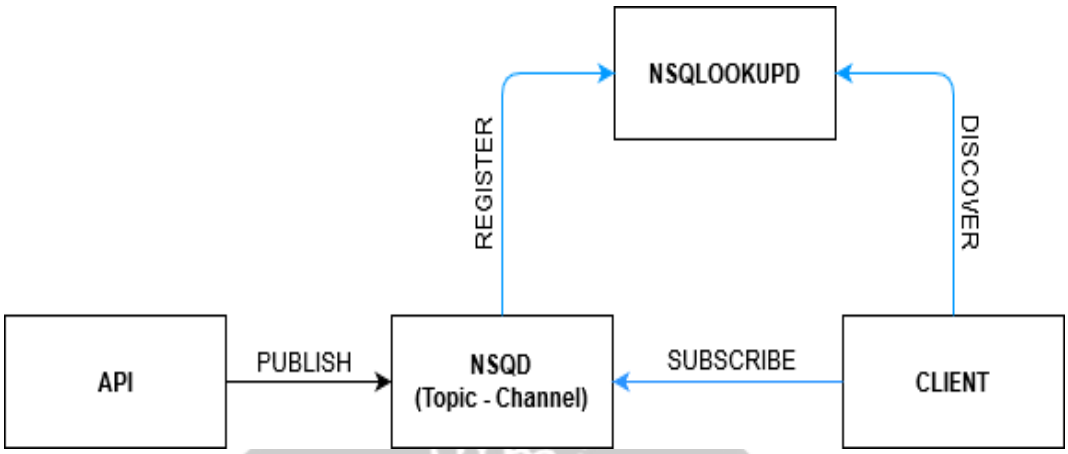


Gambar 3.1. *Clean Architecture* [8]

Gambar 3.1 merupakan penggambaran dari *Clean Architecture*, dimana terdapat beberapa komponen yang membentuk arsitektur ini. Komponen *Entities* merupakan bagian yang menyimpan model seperti objek *struct* dan *method*. Komponen *Use Case* menangani *business logic*, perubahan pada komponen ini seharusnya tidak mempengaruhi komponen *Entities*. Selain itu, komponen *use case* juga tidak boleh terpengaruh oleh perubahan eksternal. Lalu terdapat komponen *Interface Adapters* yang menghubungkan *user* dengan sistem. Tugas lain dari komponen ini adalah menjadi dinding penghubung antara *user* dan sistem, dimana akan menerima segala input dan validasi input sesuai standar yang digunakan dan menjadi komponen yang mengeluarkan hasil pemrosesan pada *user*. Terakhir, komponen *Frameworks and Drivers* terluar terdiri dari *framework* dan alat-alat yang digunakan, seperti *database*. Kunci penting dari arsitektur ini, pengembang sistem harus mengingat betapa pentingnya aturan dependency agar tiap komponen benar mengerjakan tugas mereka masing-masing [8].

3.3. NSQ

NSQ atau yang disebut juga dengan *New Simple Queue* merupakan sebuah sistem yang berjalan di belakang layar sebagai sebuah *real-time 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 [5].



Gambar 3.2. Arsitektur NSQ

Seperti pada gambar 3.2, NSQ memiliki sebuah *daemon yang* bertanggung jawab untuk menerima, mengantrikan, dan mengirimkan pesan ke aplikasi atau servis lainnya yang diatur dalam komponen yang bernama NSQD [9], dimana NSQD terbagi lagi menjadi *topic(s)* yang terdapat *channel(s)* di dalamnya. *Topic* merupakan bagian yang menampung pesan yang di-*publish* lewat API, dan akan menyalurkan pesan-pesan tersebut ke setiap *channel* yang terdapat pada *topic* terkait. Setiap NSQD yang terbentuk akan didaftarkan ke NSQLOOKUPD yang menjadi sebuah komponen yang mengatur topologi jaringan pada sistem antrian ini. Klien dapat melakukan *query* ke NSQLOOKUPD untuk melihat *topic* apa saja yang terdaftar dan dapat melakukan *subscribe* pada *topic* tersebut, sehingga ketika *topic* terkait terdapat antrian pesan, maka pesan tersebut akan dikirim ke klien secara *realtime*.

Adapun beberapa hal yang membuat NSQ lebih baik daripada sistem antrian lainnya, antara lain [5]:

a. Menawarkan skalabilitas horizontal tinggi dengan tanpa adanya perantara sehingga memungkinkan untuk melakukan penambahan NSQD dan Klien secara mandiri.
b. Memberikan kemudahan penggunaan untuk mewujudkan komunikasi antar aplikasi maupun antar servis dengan fitur NSQAdmin yang merupakan antarmuka *real-time* untuk melihat statistik dari sebuah *topic(s)* dan juga dapat mengelola *topic(s)* ataupun menghapus antrian [9].

c. Pesan yang di distribusikan bersifat *realtime*.
d. Mendukung penggunaan dalam beberapa bahasa pemrograman.
e. Menjamin pesan akan tersampaikan setidaknya satu kali kepada Klien. Dan jika terjadi kegagalan dalam pengiriman pesan, sistem akan mengembalikan pesan tersebut ke antrian *topic* terkait.
f. Dapat melakukan komunikasi melewati *port* TCP, sehingga lebih *secure* dan terdapat komunikasi antar perangkat, jika terjadi kesalahan pada sistem, perangkat yang terhubung dapat diberi pemberitahuan.

3.4. REST API

Application Programming Interface (API) merupakan suatu kumpulan yang terdiri dari *interface*, fungsi, kelas, struktur dan sebagainya untuk membangun sebuah perangkat lunak yang digunakan untuk menghubungkan suatu aplikasi dengan aplikasi lainnya [10]. API biasanya akan menerima *request* dan akan mengembalikan *response* saat dipanggil. Sebagai contohnya yaitu Google API, dimana salah satu API yang ditawarkan adalah penggunaan sistem peta yang dimiliki oleh Google. Maka saat kita ingin membuat sistem peta untuk mencari jarak dari dua buah lokasi, dengan mudahnya kita memberikan *request* berupa dua buah nama lokasi yang kita inginkan ke *end point* Google API, maka kita akan mendapatkan *response* berupa jarak antara dua buah lokasi tersebut.

Representational State Transfer **(**REST) sendiri merupakan salah satu arsitektur yang sering digunakan dalam pengembangan API. Arsitektur REST berfokus untuk menyediakan metodologi yang sederhana untuk mendapatkan data lewat jaringan [4]. REST akan mendefinisikan aturan-aturan yang digunakan dalam membuat API. Jika ada *request* dengan aturan yang tidak sesuai dengan yang telah dibuat oleh REST *Server*, maka API akan mengembalikan *response* berupa galat. Komunikasi REST juga dilakukan melalui protokol *stateless*, dimana setiap data yang diperlukan untuk melayani permintaan tersebut dikirim oleh klien itu sendiri [4].

Adapun pada arsitektur ini digunakan beberapa metode untuk mengakses API yang sering digunakan, yaitu [11]:

a. GET

   Metode ini biasanya digunakan untuk mendapatkan data dari REST *Server*. Tetapi metode ini juga memungkinkan untuk memberikan tambahan *request parameter* pada REST *Server*. Hal ini sangatlah tidak *secure*, karena dengan metode GET, parameter akan dilempar melalui URL yang tertera. Sehingga parameter tersebut dapat terlihat dengan jelas.

b. POST

   Metode ini biasanya digunakan untuk mengirimkan data yang diinginkan sebagai *request parameter* terhadap REST *Server*, sehingga *response* yang diberikan akan sesuai pada data yang diminta user. Metode ini juga memungkinkan untuk melakukan *Update* ataupun *Delete* terhadap data yang terdapat di REST *Server*. Semua itu tergantung pengolahan yang terjadi pada API yang dipanggil.

c. PUT

   Metode ini biasanya digunakan untuk memperbaharui data yang terdapat di REST *Server*.

d. DELETE

   Metode ini biasanya digunakan untuk menghapus data yang terdapat di REST *Server*.

3.5. JSON

Pemrosesan data yang dilakukan oleh sistem membutuhkan data yang masuk dari luar sistem itu sendiri. *JavaScript Object Notation* (JSON) merupakan sebuah format standar berbasis teks untuk melakukan pertukaran data. Format JSON dikenal memiliki ukuran data yang kecil, mudah untuk dibaca, dipahami, dan ditulis oleh manusia serta mudah diuraikan maupun dibuat oleh mesin [12]. Format JSON ini sendiri dibuat dengan dasar pemrograman JavaScript, standar ECMA-262 edisi ketiga. Melakukan

pertukaran data dengan format JSON pun dikatakan ideal karena sudah banyak bahasa pemrograman yang mendukung pembacaan data dengan format JSON.

3.6. Sandbox

Dalam pengembangan perangkat lunak, segala jenis fungsi dengan mudahnya dibuat dalam sebuah API. Ketergantungan ini mempunyai arti bahwa penguji aplikasi harus menguji segala bentuk *request* maupun *response* saat memanggil API tersebut. Hal ini akan susah dilakukan jika API yang digunakan sedang dalam tahap pengembangan apalagi dimiliki oleh pihak ketiga. Pada dasarnya, 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. Dengan adanya Sandbox, pengembangan dalam sebuah aplikasi memiliki beberapa keuntungan yang sangat signifikan, antara lain:

a. Mengurangi ketergantungan terhadap pihak ketiga selama tahap pengembangan. Ketergantungan dapat memakan waktu saat pengujian jika harus menguji begitu banyak kasus uji untuk mendapatkan hasil yang diinginkan sebelum dirilis ke publik.
b. Memungkinkan untuk melakukan pengujian dan pengembangan dalam waktu yang bersamaan untuk mempercepat siklus pengembangan perangkat lunak [13].
c. Dapat mensimulasikan segala bentuk skenario yang dapat terjadi pada tahap produksi, seperti menghitung latensi dari waktu pemberian *response* sebuah API, simulasi *error handling*, atapun dapat membuat aturan untuk menguji *positive case* maupun *negative case* sesuai kasus uji yang dimiliki.

3.7. Software Testing

Dalam melakukan pengembangan perangkat lunak, setiap pengembang tidak luput dari kesalahan, dimana kesalahan tersebut berujung pada defect

maupun bug dalam sistem. Beberapa defect ataupun bug tersebut dapat menyebabkan suatu hal yang fatal, atau mungkin juga tidak. Maka dari itu seorang pengembang harus selalu menguji perangkat lunak yang dikembangkan agar dapat memberi kepuasan kepada penggunanya [14]. Adapun *software testing* ataupun pengujian perangkat lunak dapat diartikan sebagai proses yang bersifat destruktif untuk mencari kesalahan dalam sebuah program. Hal ini dilakukan untuk meyakinkan bahwa aplikasi berjalan sesuai yang diinginkan dan tidak melakukan hal lain yang tidak diinginkan. Pengujian yang sukses adalah pengujian yang dapat menemukan kesalahan yang tak terduga sebelumnya, tentunya didapatkan dengan meng-eksplorasi kesalahan aplikasi secara terus menerus.

Adapun dikemukakan bahwa ada dua cara yang paling sering digunakan dalam pengujian perangkat lunak, yaitu [15]:

a. Black Box

Pengujian *Black Box* adalah pengujian perangkat lunak yang berbasis data. Pengujian ini memperlihatkan bahwa fungsi-fungsi pada perangkat lunak dapat berjalan dengan baik, dimana masukan dan keluaran yang dihasilkan oleh perangkat lunak tersebut adalah data yang diharapkan oleh pembuat perangkat lunak. Maka dari itu pengujian ini tidak menguji bagian kode programnya, melainkan melihat dari sisi masukan dan keluarannya saja.

b. White Box

Pengujian *White Box* adalah pengujian perangkat lunak yang lebih terfokus pada prosedur-prosedur yang ada. Pengujian ini juga memungkinkan untuk melakukan uji struktur internal data apakah sudah benar atau belum. Setelah itu pengujian ini juga menguji algoritma dari kode program apakah sudah menghasilkan output yang sesuai. Jika output masih salah, maka algoritma akan diperiksa kembali hingga menghasilkan output yang diharapkan.

BAB VI. PENUTUP

6.1. Kesimpulan

Berdasarkan hasil perancangan, pembahasan, dan pengujian yang dilakukan pada sistem Sandbox, maka dapat diambil beberapa kesimpulan, antara lain:

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.

6.2. Saran

Sistem Sandbox sebenarnya dibangun untuk mempermudah pengembangan dan pengujian pada aplikasi Dhanapala, sehingga validasi data yang dilakukan hanyalah validasi sederhana. Ada baiknya jika sistem Sandbox juga 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 PUSTAKA

[1] 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.

[2] 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, doi: 10.30w869/jtii.v2i2.278.

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

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

[5] M. Großmann, S. Illig, and C. L. Matějka, "SensIoT: An extensible and General Internet of Things Monitoring Framework," *Wirel. Commun. Mob. Comput.*, vol. 2019, 2019, doi: 10.1155/2019/4260359.

[6] K. Ayu, R. Indradevi, P. Sukarno, and E. M. Jadied, "Analisis Performansi Aplikasi Sandbox pada Sistem Operasi Windows," vol. 5, no. 3, pp. 7536–7543, 2018.

[7] 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, doi: 10.32493/jtsi.v3i1.4130.

[8] Tung Bui Du, "Reactive Programming and Clean Architecture in Android Development," no. April, p. 47, 2017.

[9] S. Raje, "Performance Comparison of Message Queue Methods," no. August, 2019.

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

[11] D. Riyadi, "RANCANG BANGUN REST WEB SERVICE UNTUK PERBANDINGAN HARGA PENGIRIMAN DENGAN METODE WEB

SCRAPPING DAN PEMANFAATAN API," *J. Teknol.*, vol. 56, no. 1, pp. 69–73, 2011, doi: 10.11113/jt.v56.60.

[12] 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, doi: 10.20961/PERFORMA.11.2.13942.

[13] D. Subbiah, "Constraint Free Testing using Service Virtualization," vol. 105, no. 17, pp. 14–17, 2014.

[14] I. M. S. Ardana, "Pengujian Software Menggunakan Metode Boundary Value Analysis dan Decision Table Testing," *J. Teknol. Inf. ESIT*, vol. 14, no. 11, pp. 40–47, 2019.

[15] S. Alfisahrin, "Pendekatan White Box Testing Untuk Menentukan Kualitas Perangkat Lunak Dengan Menggunakan Bahasa Pemrograman C++," *Paradigma*, vol. XIV, no. 1, pp. 69–78, 2012.