RESTFul Web Service Untuk Sistem Pencatatan Transaksi Studi Kasus PT. XYZ

Penidas Fiodinggo Tanaem[#1], Danny Manongga[*2], Ade Iriani[#3]
Magister Sistem Informasi, Fakultas Teknologi Informasi
Universitas Kristen Satya Wacana Salatiga
JL. Diponegoro 52-60, Salatiga 50711, Indonesia
[1]fiodinggo@gmail.com
[2]dmanongga@gmail.com
[3]adeiriani@gmail.com

Abstract – **PT. XYZ is a PT that engaged in marketing of jewelry. There ara several units at PT. XYZ that have a need for information and using of different systems. These condition influence the process of integration and distributing data at PT. XYZ. The impact of these conditions that descrease the performance of the company. The objectives to be achieved from this research is discusses the RESTful Web Service architecture that uses security by implementing JSON Web Token and use was in the system of recording transactions in PT XYZ. This research resulted in a RESTful Web Service architecture applied in PT. XYZ.**

Keywords – ***Web Service, RESTFul, JSON Web Token .***

I. PENDAHULUAN

Pemanfaatan teknologi saat ini, memberikan pengaruh besar baik dalam lingkup individu maupun organisasi. Teknologi selalu dikembangkan dengan tujuan untuk memenuhi setiap kebutuhan bagi para penggunanya. Dalam lingkungan bisnis, proses bisnis saat ini lebih cenderung bersifat dinamis, didistribusikan dan kolaboratif, sehingga perlu untuk beradaptasi dengan proses bisnis yang lain dengan lebih sering, dan integrasi proses melalui batas-batas organisasi [1].

PT. XYZ adalah sebuah PT yang bergerak dalam bidang pemasaran perhiasan yang beroperasi di daerah Semarang semenjak tahun 2012. Terdapat beberapa unit dalam lingkungan perusahan. Masing-masing unit memiliki sistem yang berbeda *platform* dan bahasa pemrograman. Kondisi seperti ini memberikan pengaruh besar dalam penurunan kinerja dari masing-masing unit. Dalam kesehariannya, terdapat transaksi-transaksi yang mempengaruhi jumlah stok barang, pencatatan dan laporan transaksi yang dilakukan. Proses tersbut berdampak pada penyesuaian data antara data pergudangan dan data pencatatan transaksi, dikarenakan terdapat beberapa database yang digunakan secara terpisah dan aplikasi-aplikasi yang berbeda-beda. Permasalahan lain yang terdapat pada PT. XYZ yaitu membangun integrasi antar aplikasi yang terdapat di lingkungan PT. XYZ, dan akses yang akan diberikan kepada para petinggi perusahaan yang berlokasi di luar Indonesia. Hal ini memberikan tantangan tersendiri bagi pihak PT. XYZ sendiri. Penerapan *Web Service (WS)* merupakan alternative terbaik yang digunakan dalam mengatasi kondisi integrasi sistem dan distribusi data yang dialami oleh pihak perusahaan.

WS merupakan suatu bentuk perkembangan dari teknologi saat ini yang dapat digunakan untuk integrasi sistem, terutama dalam lingkungan bisnis. Vasco dan Dustdar memberikan pandangan bahwa manfaat dari membangun sebuah *WS* untuk kebutuhan bisnis yang berkembang dari waktu ke waktu yaitu peningkatan jumlah integrasi dan fleksibilitas untuk pengembangan demi kepentingan dalam mengintegrasikan service ke dalam pendekatan manajemen dan proses bisnis yang ada [2]. Bentuk nyata dari perkembangan *WS* saat ini yaitu dengan hadirnya *WS* berbasis *RESTful*. Meng berpendapat bahwa [3] *RESTFul WS* sangat baik dalam mengoptimalkan kinerja. Ketika *server RESTFul WS* diakses dalam skala besar dan secara bersamaan dalam sistem terdistribusi di internet, karena memiliki kinerja tinggi dan stabil. Namun, keamanan *RESTFul* merupakan salah satu poin penting yang harus di perhatikan. Mengamankan *RESTFul WS* mencakup mengamankan data, serta seluruh komunikasi untuk melindungi kerahasiaan dan integritas data [4]. Salah satu langkah yang dapat di gunakan dalam mengatasi permasalahan tersebut yaitu menggunakan *JSON Web Token* (*JWT*). *JWT* adalah standar (RFC 7519) yang mendefinisikan cara yang simpel dan independen dari transmisi informasi yang aman antar setiap pihak dengan menggunakan format data objek *JSON*. Informasi yang ditransmisikan dengan aman dan dapat diverifikasi karena menggunakan *digital signature*. *Digital signature JWT* dapat menggunakan kunci rahasia (dengan algoritma *HMAC*) atau sepasang kunci publik/privat menggunakan *RSA* [5].

Artikel ini akan dibahas mengenai arsitektur *RESTFul WS* yang memanfaatkan *security* dengan menerapkan *JWT* dan pemanfaatanya dalam pemanfaatannya pada sistem pencatatan transaksi PT XYZ. Faktor penyebab penggabungan ini dikarenakan *RESTFul* memanfaatkan

format data *JSON*, dengan demikian akan sangat memungkinkan dalam proses menggabubgkan *RESTFul* dan *JWT* [6] dengan tujuan yaitu untuk mengamankan distribusi data dengan menggunakan *RESTFul WS*. Penelitian ini menghasilkan sebuah arsitektur *RESTFul WS* yang diterapkan di lingkungan PT. XYZ untuk menunjang proses distribusi data pada sistem yang berbeda-beda.

II. KAJIAN PUSTAKA

Su dan Chiang menekankan bahwa sistem yang mengikuti prinsip-prinsip *REST* sering disebut "*RESTFul*" [7]. Konsep *REST WS* diperkenalkan pada tahun 2000 oleh Roy Fielding. Fielding mengungkapkan bahwa gaya *Transfer Representasi State* (*REST*) merupakan abstraksi dari elemen arsitektur dalam sistem hypermedia terdistribusi. Dalam membangun komunikasi, *REST* menggunakan protokol *HTTP*, Sedangkan Setiap dokumen dan setiap proses dimodelkan sebagai sumberdaya *Web* dengan *URI* yang unik [8]. Dengan kata lain, *RESTFul WS* adalah *WS* sederhana yang dapat dianggap sebagai kumpulan sumber daya yang diimplementasikan menggunakan *HTTP* dan prinsip-prinsip *REST* [7].

A. *Penelitian Terdahulu*

Terdapat penelitian terdahulu dibidang ini. Penelitian Hussain dan Dilber dengan judul "*Restful web services security by using ASP.NET web API MVC based*" dengan tujuan yaitu mengeksplorasi API *Web* ASP.NET yang akan menggunakan framework MVC untuk mengimplementasikan *RESTFul WS*. Penggunaan teknik keamanan yang berbeda seperti *claim based*, *OAuth2* dan *Web token* sederhana dan *delegation based* karena memiliki kendala keamanan yang serius seperti itu *cashable*, *stateless*, *client-server* dll. Hasil dari penelitian ini yaitu mengamankan representasi dari sumber daya informasi [1]. Berbeda dengan penelitian yang dilakukan di PT.XYZ yaitu penelitian ini menggabungkan *RESTFul WS* dengan konsep *JWT* dalam mengamankan pertukaran data atau informasi yang terjadi.

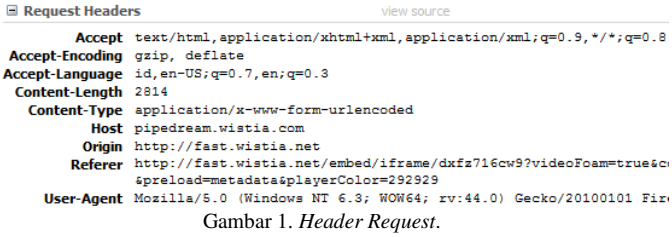
Kariluoma menyajikan penelitian dengan judul "*A RESTFul Architecture For Multiuser Virtual Environments And Simulations*" berfokus pada proses penggabungan pola desain *MOO* (*MUD Object Oriented*) dengan arsitektur terdistribusi dan modifikasi *Java MOO* yang memungkinkan *MOO* yang didistribusikan. Dengan tujuan yaitu mengurangi beban komputasi per-*server* dibandingkan dengan pendekatan *single-server* tradisional. Hasil dari penelitian tersebut yaitu membangun sebuah arsitektur menggunakan *interface RESTFul* dan mendukung pola design *MOO*, menggunakan sumber daya yang minim dari *server-side*, dan dapat diimplementasikan sebagai jaringan *server* terdistribusi [9]. Fokus penelitian saat ini yaitu mendistribusikan data dengan memanfaatkan *interface RESTFul* yaitu format data *JSON*, karena format data *JSON* sangat sederhana dan memiliki ukuran yang lebik kecil dibandingkan dengan *XML*.

B. *RESTFul WS*

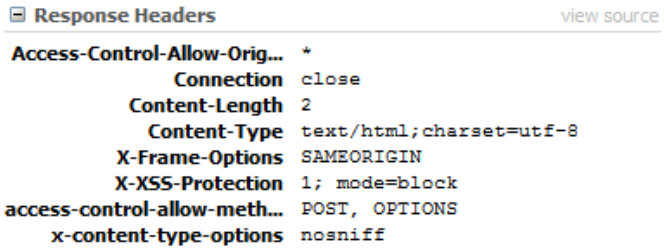
Gagasan utama dari *REST* adalah konsep *resource* sebagai komponen dari aplikasi yang perlu digunakan atau dialamatkan [10]. *REST WS* membangun integrasi dengan cara yang lebih ringan dan sederhana, dan berfokus pada sumberdaya [11]. *REST* dapat dijelaskan dalam lima batasan, diantaranya:

- *Resource Identification*: *Web* bergantung pada *Uniform Resource Identifier* (*URI*) untuk mengidentifikasi sumber daya, sehingga *link* ke sumber daya dapat dibentuk menggunakan skema identifikasi yang mudah untuk dikenali [10].
- *Connectedness*: artinya klien dari *RESTFul* Service seharusnya mengikuti *link* untuk menemukan sumber daya agar dapat berinteraksi dengan *Service* [10].
- *Uniform Interface*: artinya sumber daya harus tersedia melalui antarmuka yang seragam dengan semantik yang mendefinisikan interaksi, seperti *Hypertext Transfer Protocol* (*HTTP*) [10]. *HTTP* mencakup metode *POST*, *GET*, *PUT* dan *DELETE* [12].
- *Self-Describing Messages*: artinya mengekspos *resource* yang ada, *RESTFul* menggunakan lebih dari satu format data (*XML*, *JSON*, *RDF*, dll) dibandingkan dengan *SOAP* (*XML*), namun hal ini tergantung *developer*.
- *Stateless Interactions*: mengharuskan setiap *request* dari klien lengkap, dalam arti bahwa semua informasi untuk melayani *request* ke *server* harus berisi setiap informasi yang dibutuhkan agar *request* dapat dipahami [10], dan tidak ada ketergantungan dengan *state* atau penanda dari *client*.

Terdapat dua bagian pesan yang digunakan untuk membangun komunikasi dengan *server* yaitu pesan *Header* dan pesan *Body*. *HTTP Header*, yang umum meliputi *header request* diilustrasikan pada gambar 1, *header response* pada gambar 2, dan terdapat bidang entitas-*header* [13]. Setiap *request* sumberdaya dari masing-masing *client* dapat dikendalikan dengan memanfaatkan *HTTP Header* [14]. Setiap kolom *Header* terdiri dari nama diikuti dengan titik dua (":") atau *white space* dan konten *field*. Nama *field* bersifat *case-sensitive*. *Header* berisikan semua informasi yang diperlukan untuk mengumpulkan metode *request* [15] dan respon.

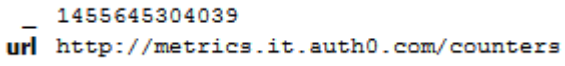


Gambar 1. *Header Request*.

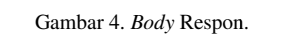


Gambar 2. *Headers Response*.

Sedangkan *HTTP body* mencakup pesan *HTTP* yang digunakan untuk memuat entitas *body* melalui protkol *HTTP* yang berhubungan dengan *request* tuangkan pada gambar 3 dan respon pada gambar 4 [13]. Saat *client* melakukan *request*, *HTTP body* bisanya berisikan informasi setiap parameter yang untuk *request*. Sedangkan respon berisikan informasi yang didapatkan dari hasil *request*.



Gambar 3. *Body Request.*



Gambar 4. *Body* Respon.

REST menentukan sekumpulan prinsip arsitektur yang mana dapat digunakan untuk merancang *WS* yang berfokus pada sumber daya sistem, termasuk bagaimana sumber daya yang dialamatkan dan ditransfer melalui *HTTP* oleh berbagai klien yang ditulis dalam bahasa pemrograman yang berbeda [16]. Dengan demikian *REST* dapat mengoperasikan operasi *CRUD* (c*reate*, *read*, *update* dan *delete*) yang [17] dapat dilakuan dengan memanfaatkan metode *HTTP* antara lain *POST*, *GET*, *PUT* dan *DELETE*. Tabel 1 menyajikan pemetaan operasi *CRUD* ke dalam permohonan *HTTP*.

Format *application/x-www-form-URLencoded* yang digunakan oleh masing-masing metode *HTTP* diantaranya *GET* dan *DELETE* berbeda dengan *POST* dan *PUT* adalah berbeda, hal ini dikarenakan cara parsing data yang berbeda. Parsing data pada metode *GET* dan *DELETE* dimuai melalui *URL*, sedangkan *POST* dan *PUT* melakukan parsing data melalui *payload HTTP* dengan memanfaatkan *media type* 'application/x-www-form-*URL*encoded'.

Kode status *HTTP* respon *server* terhadap aksi yang dilakukan oleh *client*.

- Kode status 201: *request* telah terpenuhi dan menghasilkan sumber daya yang baru *Create*.
- Kode status 200: respon standar untuk *request HTTP* dari *client* yang dinyatakan sukses oleh *server*. Respon sebenarnya akan tergantung pada metode *request* yang digunakan.

TABEL 1. PEMETAAN METODE *HTTP* PADA *REST*

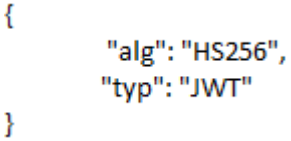
Operasi *CRUD*	*HTTP Method*	Format *Application/x-www-form-URLencoded*	*HTTP* Status
Create	*POST*	*HTTP Form Encoded*	Status 201 *CREATED*
Read	*GET*	*None*	Status 200 Ok
Update	*PUT*	*HTTP Form Encoded*	Status 200 Ok
Delete	*DELETE*	*None*	Status 200 Ok

C. *JSON Web Token (JWT)*

JSON Web Token (*JWT*) adalah keamanan berbasis *JSON token encoding* yang memungkinkan identitas dan keamanan informasi untuk dibagikan di seluruh domain keamanan [18]. Hal ini memungkinkan *client* untuk mendapatkan *token* dengan memberikan *username* dan *password* mereka [19]. Sebuah *token* umumnya diterbitkan oleh penyedia layanan dan dikonsumsi oleh pihak yang mengandalkan konten untuk mengidentifikasi subjek *token* dengan tujuan yang berhubungan dengan keamanan [18]. *JWT* adalah Format representasi yang sederhana yang ditujukan untuk ruang yang dapat digunakan seperti *HTTP Header Authorization* dan parameter *URI Request* [20]. *JWT* adalah suatu cara merepresentasikan klaim yang akan ditransfer antara dua pihak [6],[21]. Kelebihan yang ditawarkan oleh *JWT* adalah *stateless*. *Stateless JWT* memungkinkan pengguna untuk berbagi, menggunakan kembali dan kontrol akses ke sumberdaya [22].

Gambar 5. *JWT*

JSON Web Token terdiri dari tiga bagian yang dipisahkan oleh titik "." (gambar 5), yaitu *Header*, *Payload* dan *Signature*. *Header* biasanya terdiri dari dua bagian: jenis *token* , yaitu *JWT*, dan algoritma *hashing* seperti *HMAC SHA256* (gambar 5) [5].



Gambar 6. *JWT Header*.

Bagian kedua dari *JWT* adalah *payload*, yang berisi klaim. Klaim adalah pernyataan tentang suatu entitas (biasanya, pengguna) dan metadata tambahan [5] seperti yang dituangkan pada gambar 7.

```
{
        "iss": "scotch.io",
        "exp": 1300819380,
        "name": "Chris Sevilleja",
        "admin": true
}
```

Gambar 7. *JWT Payload*.

Bagian ketiga dan terakhir dari kami *JSON Web Token* akan menjadi *signature*. *Signature* ini terdiri dari hash dari komponen-komponen yaitu *Header*, *payload* dan kunci rahasia [5]. Bedasarkan setiap komponen komponen yang terdapat pada *JWT*, akan di *signin* dengan kunci rahasia seperti yang dituangkan pada gambar 8.

```
HMACSHA256(
        base64UrlEncode(header) + "." +
        base64UrlEncode(payload),
        "secret"
)
```

Gambar 8. *JWT* Signature.

Alur kerja enkripsi dari *JWT* yaitu dengan menggunakan *Base64URL encoding* digunakan encripsi *header*, *payload* dan *signature token* yang menghasilkan hasil enkripsi 64bit. Kemudian sebuah signatur yang unik akan dihasilkan dari proses *Base64URL* menggunakan algoritma kriptografi *hash HMAC SHA-256* dan sebuah *private key* yang terdapat pada bagian *signature* untuk integritas data dan otorisasi. *Secure Hash Algorithm* 256 (*SHA-256*) menghasilkan *message digest* dengan panjang 256 bit. *SHA-256* merupakan salah satu fungsi hash satu arah, karena tidak mungkin menemukan pesan dari *message digest* yang dihasilkan karena *SHA-256* menggunakan enam fungsi logika, di mana setiap fungsi beroperasi pada 32-bit [23].

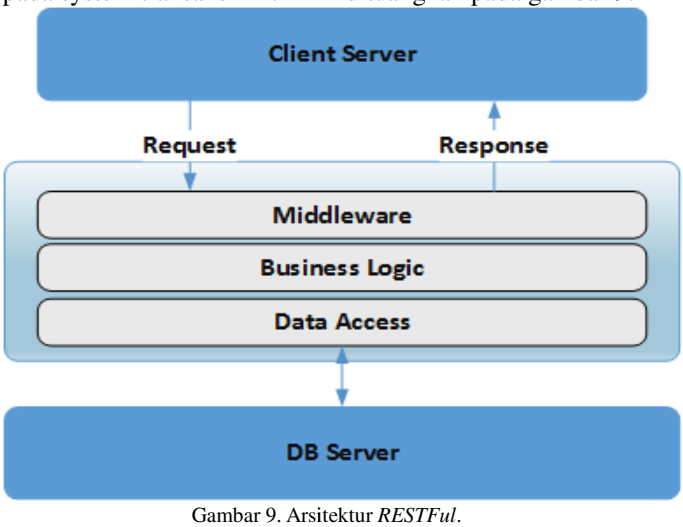
Klaim dalam *JWT* di encode sebagai objek *JSON* yang dapat ditandatangani dan/atau dienkripsi secara *digital* menggunakan *JSON Web Signature* (*JWS*) dan sebagai pilihan dienkripsi menggunakan *JSON Web Encryption* (JWE) [6]. Sebuah *token* adalah *string* yang mengkodekan struktur data *JSON*, dengan demikian sebuah *string* dapat merepresentasikan suatu objek *JSON* yang berisi klaim yang disampaikan oleh *JSON Web Token* [24]. Proses enkripsi dan mendekripsi *token* dilakukan menurut *JSON Web* Algorithm (*JWA*) [19]. Standar spesifikasi *JWT* ditentukan oleh *Internet Engineering Task Force* (IETF), tingkat kebutuhan implementasi ditentukan untuk setiap *JWA* [19].

Penerapan *JWT* dapat digunakan pada 2 kondisi yaitu kondisi saat autentikasi dan pertukaran informasi. Autentikasi merupakan kondisi yang umum ditemukan untuk menggunakan *JWT*, setelah pengguna *login*, setiap permintaan yang dilakukan oleh *client* harus menyertakan *JWT*, yang dapat memungkinkan pengguna untuk akses *router*, *service*, dan sumber daya yang diizinkan dengan *token* yang dibuat [5]. *Single Sign On* merupakan fitur yang banyak menggunakan *JWT* saat ini, karena *overhead* yang kecil dan kemampuannya untuk dengan mudah digunakan pada sistem dan domain yang berbeda [5]. Sedangkan yang dimaksudkan dengan pertukaran informasi menggunakan *JWT* adalah cara yang baik untuk transmisi informasi dengan aman antara setiap pihak, karena dapat ditandatangani, misalnya menggunakan *public/private key*. Selain itu, *signature* dibentuk dengan menggunakan *Header* dan *payload* [5]. Dengan demikian, *JWT* mampu memberikan kemudahan bagi *client* untuk mengakses sumberdaya tanpa harus menginput *username* dan *password* berulang kali [19]. Pendekatan berbasis *token* memungkinkan *client* untuk membuat panggilan *AJAX* ke *server* di domain karena panggilan dapat menggunakan *Header HTTP* untuk mengirimkan informasi pengguna [5].

III. METODOLOGI

Penelitian ini dilakukan di PT. XYZ karena terdapat beberaapa unit dilingkungan perusahaan dan masing-masing unit memiliki sistem yang berbeda-beda berdasarkan kebutuhan tiap-tiap unit. Data atau informasi yang terkumpul dari hasil wawancara dan dijadikan acuan dalam mendesain *WS* yang dibangun. Wawancara dilakukan dengan tujuan untuk melihat lebih jauh berbagai komponen yang dipakai pada sistem yang digunakan meliputi data atau informasi, *hardware*, *software*, jaringan dan sumber daya manusia yang tersedia di lingkungan perusahaan. Berdasarkan hasil wawancara yang dilakukan, dilakukan desain sistem yang menghasilkan desain fungsional. Desain system berupa desain konseptual dari sistem dengan tujuan yaitu menghasilkan spesifikasi yang sesuia dengan kebutuhan pengguna.

Konsep arsitektur yang digunakan pada *RESTFul WS* pada system transaksi PT. XYZ dituangkan pada gambar 9.



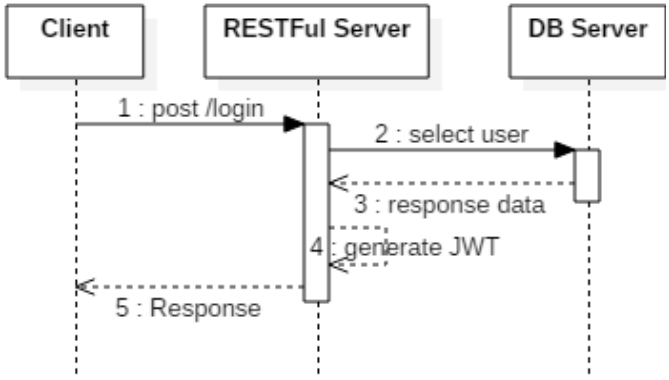
Gambar 9. Arsitektur *RESTFul*.

Arsitektur tersebut terdiri atas lapisan *Middleware*, *Business Logic* dan *data Access*. Lapisan *Middleware* berfungsi untuk memproses setiap komunikasi (*POST*, *GET*, *PUT* dan *DELETE*). Pada penerapannya, *Middleware* akan menerjemahkan setiap *request HTTP Body* dan validasi *token* dari *client*. Sedangkan *Business layer* berfungsi mengimplementasikan fungsionalitas inti dari sistem, dan

merangkum logika bisnis yang relevan. Lapisan *Data access* berperan untuk mengekspos data berdasarkan batasan-batasan yang dimiliki oleh sistem. Data atau informasi yang diekspos menggunakan format data *JSON*. *JSON* adalah bahasa independen dan berdasarkan koleksi pasangan key/value dan mempunyai list value. Struktur ini memungkinkan untuk digunakan dalam setiap bahasa pemrograman modern yang membuatnya menjadi pilihan yang baik untuk berkomunikasi di dunia *Web* [25].

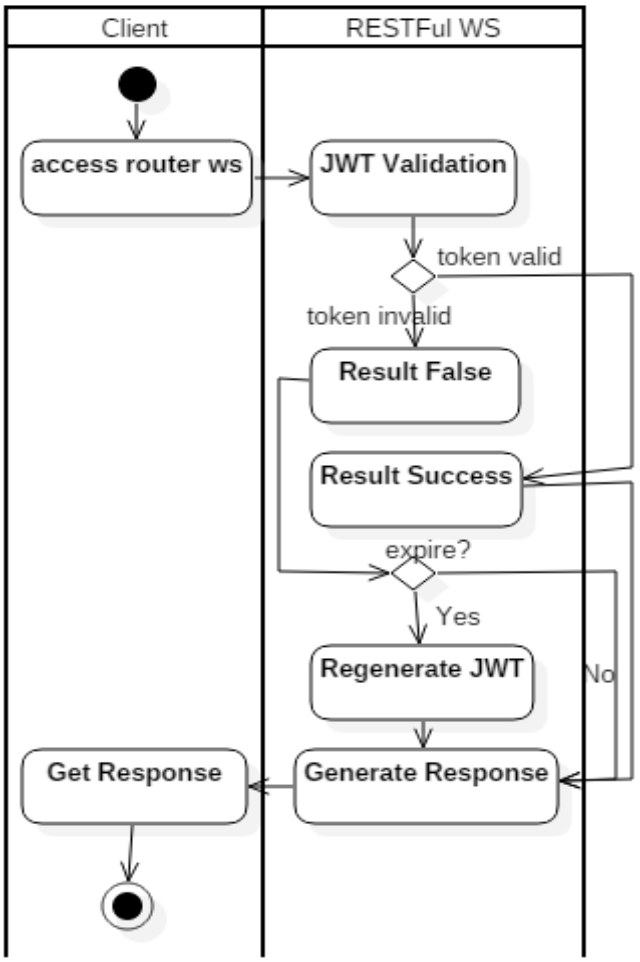
Penelitian ini menggunakan beberapa metode *HTTP* yaitu *POST*, *GET* dan *DELETE*. *POST* digunakan untuk operasi *Create* dan *Update*. Metode *GET* digunakan untuk operatsi *Read*. Sedangkan metode *DELETE* untuk operasi *Delete*. Namun, Price berpendapat bahwa [26] metode *HTTP* yang paling banyak digunakan yaitu metode *request HTTP GET*, kerena *HTTP GET* akan dipanggil setiap kali *browser Web* meminta *URI* dari *Web server*. Dampak yang dirasakan dari situasi tersebut yaitu penurunan kinerja dari *RESTFul*. Dalam mengatasi metode *GET*, *cache* merupakan solusi yang bisa digunakan untuk mengatasi masalah ini. *Caching* bisa diartikan sebagai memiliki informasi, data dan obyek sementara yang disimpan untuk penggunaan yang sering atau berhubungan erat dengan *interval* [27]. Dalam penerapannya, *cache* digunakan pada lapisan *middleware* untuk mengatasi response ke *client*.

Alur kerja dari *RESTFul WS* yang dibangun, dapat dijelaskan pada gambar 10 sebagai berikut:



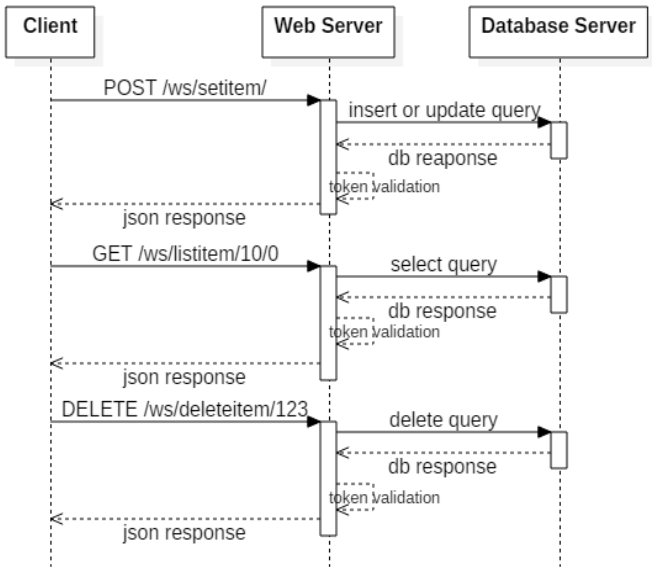
Gambar 10. *GET Token* .

Sebelum *client* melakukan akses kepada setiap sumberdaya yang dimiliki, *client* harus melakukan *login* terlebih dahulu. Respon dari proses *login* yaitu *token* yang akan digunakan pada *HTTP Header* untuk mengakses sumberdaya.



Gambar 11. *JWT* Validation.

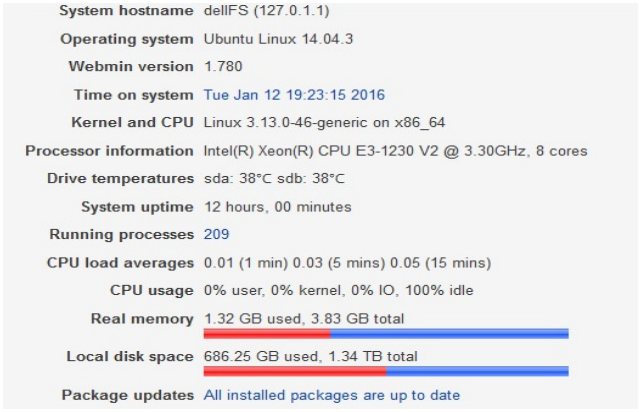
Terdapat grup *router* yang digunakan untuk mengakses sumberdaya yaitu *group* ruter "*/WS*". Hal ini dilakukan untuk menempatkan proses validasi *JWT* untuk setiap *client* yang ingin mengakses data. Sebelum setiap response di tangani, *RESTFul* akan melakukan validasi *token* yang di muat melalui *HTTP Header* oleh *client* (gambar 11).



Gambar 12. Akses *RESTFul WS*.

RESTFul memberikan abstraksi untuk mempublikasikan informasi dan memberikan akses *remote* ke lapisan *Data Access*. Isi *Data Access* yang dianggap sebagai sumber daya yang direferensikan oleh *URI* sederhana dan diwakili oleh *JSON*. Misalnya, data *list item* dengan limit: "10" dan offset: "0" dapat diakses melalui metode *GET HTTP* sederhana dengan *URI*: '/*WS*/listitem/10/0'. Demikian pula, data *item* juga dapat di*application/x-www-form-URLencoded*, diubah dan di hapus menggunakan metode *HTTP POST* ataupun *DELETE* (Gambar 12). Jalur komunikasi yang diterapkan untuk mengakses *RESTFul WS* yang digunakan PT.XYZ pada saat ini yaitu *VPN*. *VPN* (*Virtual Private Network*) menyediakan akses yang aman antara perusahaan dan *Cloud* [28]. Dengan demikian perusahan sepenuhnya memiliki *control* terhadap sumberdaya mereka sendiri.

Dalam penerapannya, *database* yang digunakan untuk mengoleksi sumberdaya yaitu *mysql server*. Aplikasi *server* menggunakan *server* apache 2. Komputer *Server RESTFul* menggunakan *Ubuntu server LTS 14.04*. dengan spesifikasi seperti yang di tuangkan pada gambar 13.



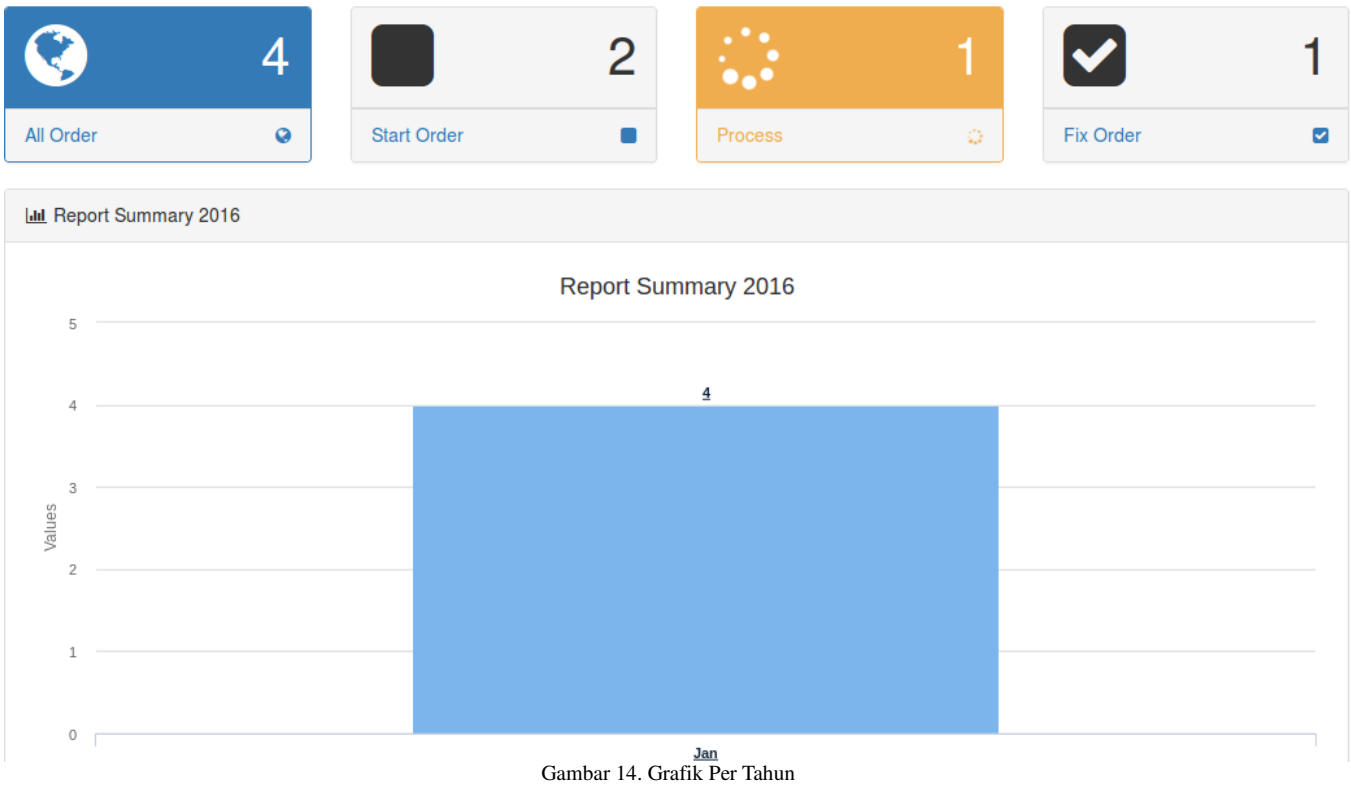
Gambar 13. Spesifikasi *Server*.

IV. HASIL DAN PEMBAHASAN

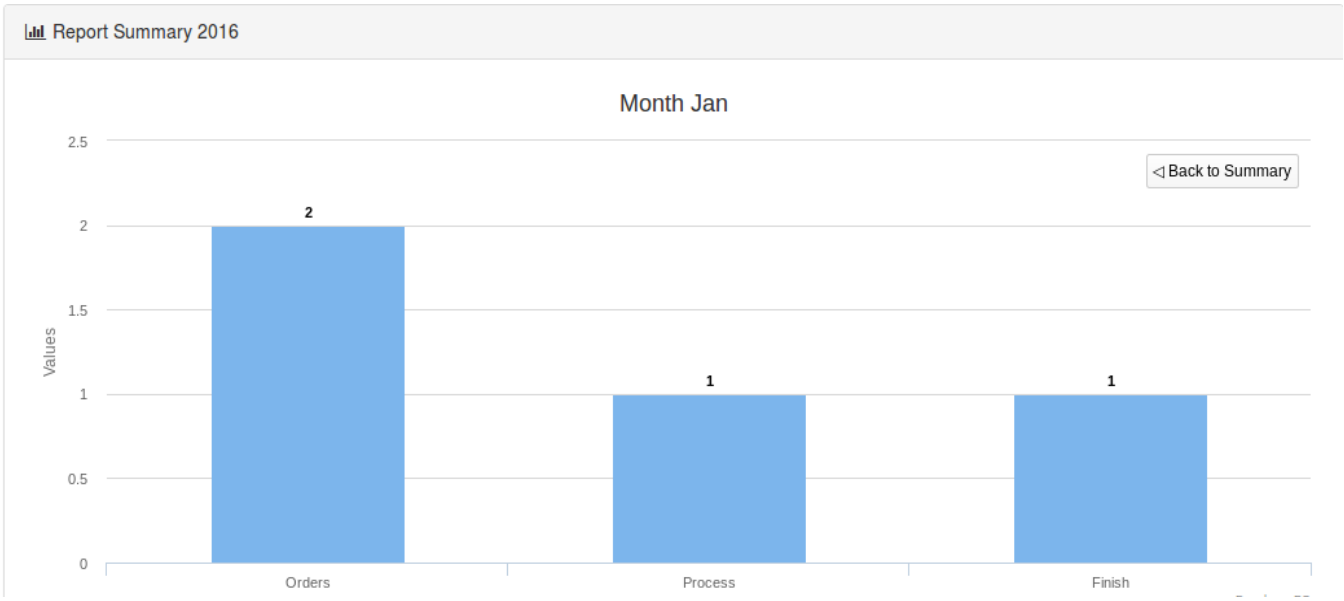
A. Hasil

Sistem pencatatan transaksi yang dimiliki oleh PT.XYZ dibangun dengan tujuan utama yaitu mencatat setiap transaksi yang terjadi. Namun dalam perkembangannya sistem ini dikembangkan agar dapat di integrasikan dengan sistem lain yang dimiliki oleh PT. XYZ dengan tujuan yaitu menunjang pelaksanaan proses bisnis di lingkungan perusahaan secara menyeluruh, yang [29] bukan terbatas pada fungsi bisnis tertentu saja namun dapat bermanfaat bagi sejumlah area fungsional (*across functional areas*).

Berdasarkan implementasi saat ini, *RESTFul WS* dijalankan pada jaringan local. *RESTFul WS* digunakan oleh PT.XYZ untuk mengitegrasikan sistem yang ada dilingkungan PT.XYZ baik sistem yang dijalankan pada jaringan lokal, maupun yang dijalankan pada jaringan internet. Jalur komunikasi yang digunakan saat ini untuk mengintegrasikan *RESTFul WS* dengan sistem yang berada di lingkungan PT. XYZ yaitu dengan menggunakan jaringan *VPN*. Cakupan dari sistem ini yaitu pencatatan setiap transaksi yang terjadi, rekapan tiap produk, pengelolaan data *account user*, rekapan data stamp, pencatatan *Provide Order* / transaksi, *print Provide Order* / transaksi, rekap status *Provide Order* / transaksi, *setting Access control List* dan laporan setiap transaksi yang terjadi / *summary*. Hasil dari pencatatan tiap transaksi akan dituangkan dalam bentuk grafik tahunan maupun bulanan. Grafik tahunan berisikan informasi mengenai rangkuman dari setiap transaksi yang terjadi dalam setahun (gambar 14). Sedangkan grafik bulanan berisiskan informasi mengenai rangkuman hasil transaksi yang terjasi dalam satu bulan (gambar 15).



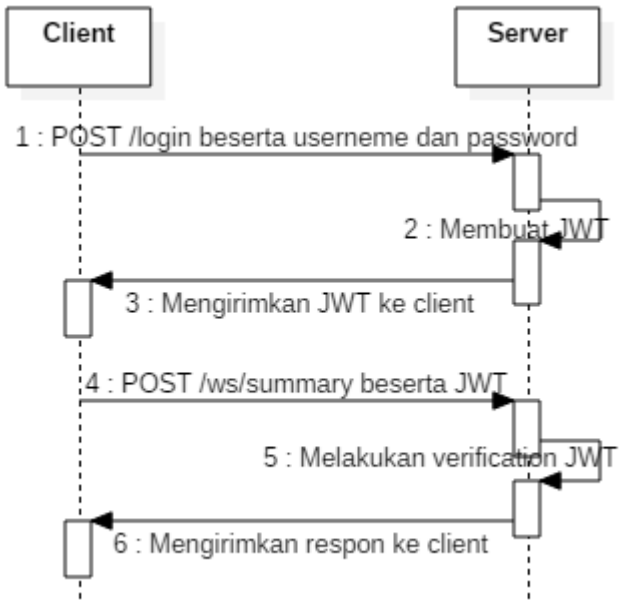
Gambar 14. Grafik Per Tahun



Gambar 15. Grafik Per Bulan.

Pemanfaatan *RESTFul* pada sistem transaksi PT.XYZ difokuskan pada integrasi antara setiap sistem yang berbeda dan sumberdaya dialamatkan. *RESTFul WS* sepenuhnya harus mampu mengatasi setiap integrasi yang ada. Dengan demikian, ketika pengguna ingin memanggil sumberdaya yang ada, sumberdaya harus di distribusikan secara individual atau menambahkan aplikasi untuk mendistribusikan sumberdaya tersebut berdasarkan setiap *request* yang dilakukan *client*. *Request* sumberdaya yang dilakukan oleh *client* memanfaatkan *URL*. Contohnya, untuk mengambil data *summary* berdasarkan tahun, yang dialamatkan dengan *URL* berikut *POST* '*/WS/GET_summary* {tahun}'. Dengan demikian, Setiap *URL* dapat mengandung satu atau lebih sumberdaya [30].

Umumnya pada sistem yang dibangun, setiap *request* terhadap sumberdaya yang dilakukan oleh *client* harus menyertakan data *token client*. Data tersebut kemudian akan di sertakan pada *HTTP header* dan di kirim ke *WS* dengan menggunakan tipe konten *application/x-www-form-URLencoded* untuk di validasi pada lapisan *middleware RESTFul*. Dan setiap respon dari hasil *request* yang menggunakan format data standar yaitu format data *JSON*. *Client* akan diberikan *token* saat *client* tersebut berhasil melakukan *login* ke *server* seperti yang diilustrasikan pada gambar 16.



Gambar 16. Diagram alur kerja akses *RESTFul WS*.

Tipe konten '*application/x-www-form-URLencoded*' adalah jenis konten *default* komunikasi metode *HTTP* [13]. Semua karakter dikodekan sebelum dikirim (spasi dikonversi ke simbol "+", dan karakter khusus dikonversi ke nilai-nilai *ASCII HEX*) [31]. Hasil konversi yang dilakukan, dimuat dalam *HTTP Header*. *HTTP Header* terdiri dari beberapa entitas diantaranya metode *HTTP*, *URL*, *token*, *username, content type* dan panjang konten seperti yang diilustrasikan pada gambar 17. Sedangkan hasil request yang didapatkan merupakan format data *JSON* yang memiliki entitas data berbeda-beda berdasarkan jenis data yang direquest (gambar 18).

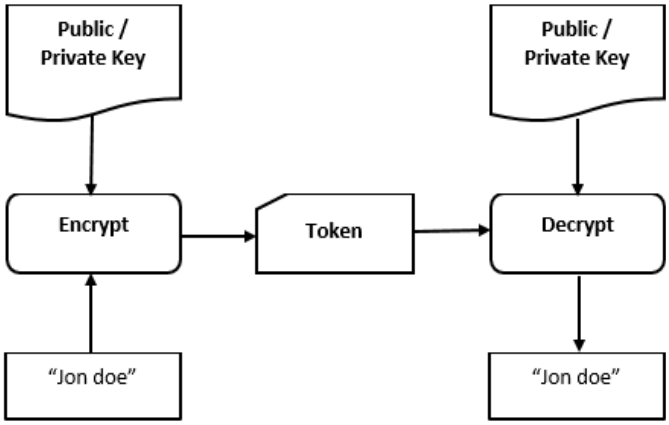
```
POST /thesis/ws/get_summary HTTP/1.1
Host: localhost
Accept: */*
X-TOKEN:
eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJkYXRlY3JlYXRlIjoxNDUzMjc5NjY3LCJkYXRl
ZXhwaXJlIjoxNDUzMjgxNDY3LCJpZCI6IjEiLCJpcCI6IjEyNy4wLjAuMSIsInVzZXJuYW1lIjoi
YWRtaW4ifQ.NxFqP_bnWYKgpoROUVa7TzmTpQRQdvZwfbJwd6wXHaY
X-USERNAME: admin
Content-Length: 139
Expect: 100-continue
Content-Type: application/x-www-form-urlencoded; boundary=----------
9c57132ebc2a6ce3
```

Gambar 17. *Request Header*


```
{
	"success":true,
	"title":"Report Summary 2016",
	"response":{
		"month":[{
			"id":"01","name":"Jan","y":4,"drilldown":"Jan"
		}],
		"drilldown":[{
			"id":"Jan",
			"name":"Jan",
			"data":[["Orders",2],["Process",1],["Finish",1]]
		}],
		"all_order":4,"orders":2,"process":1,"end":1
	}
}
```


Gambar 18. Hasil dokumen dalam format *JSON*.

Pada lapisan *middleware RESTFul WS*, terdapat dua ruter yang digunakan untuk membuat *JWT*. Diantaranya yaitu ruter "*/login*" dan sub ruter "*/WS*". Dalam proses enkripsi *JWT*, terdapat tiga bagian *JWT* yaitu *header*, *payload* dan *signature* [5]. Isi dari *JWT Header* yaitu: *type* dan *algorithm*. *Conten* dari *type* yaitu "*JWT*" dan *algorithm* yaitu "HS256". Sedangkan isi dari *JWT payload* yaitu *dateCreate* (berisikan waktu membuat *token*), *dateexpire* (berisikan waktu kadaluarsa *token* yaitu 15 menit), id (berisikan id *client*), *username* (berisikan *username client*) dan ip (berisikan *ipaddress client*). Sedangkan isi dari *signature* diantaranya hasil kombinasi dari enkripsi yang menggunakan base64uelencode *JWT header* dan *JWT payload*, kunci rahasia dan algoritma dari *JWT header*. Sedangkan pada proses validasi *JWT*, menggunakan *request header* dari *client* dan menggunakan kunci rahasia.



Gambar 19. *JWT encrypted* and *decrypted*.

Dalam melakukan proses enkripsi dan deskripsi *token* , digunakan kunci publik untuk melakukan proses tersebut. Pada penerapannya, jika data *JWT* dari *client* tidak valid, atau persyaratan konfirmasi tidak dapat terpenuhi, *server* harus menyusun suatu respon *error* [32]. *Token* yang dihasilkan untuk *client* harus berdasarkan persetujuan dari *server*. *Client* menggunakan *token* untuk mengakses sumber daya yang dilindungi oleh *Server* [6] seperti yang dituangkan pada gambar 19.

B. Pembahasan

Mengatasi setiap permintaan dari *client* dengan memanfaatkan *RESTFul WS* merupakan solusi yang tepat. Hal ini didasarkan pada *RESTFul WS* atau juga dikenal dengan *REST Application Programming Interface* (*REST* API), yang secara luas diterapkan dibanyak aplikasi besar seperti *Google Map*, *Google docs*, *Wikipedia*, *Facebook*^{TM} dan sebagainya [7]. Salah satu faktor pendukung yang menjadi keunggulan dari *RESTFul WS* yaitu *lightweight system* dan memudahkan *client* agar dapat mengakses sumberdaya sebagaimana adanya, atau memodifikasi sumberdaya tersebut [30]. Yang dimaksudkan dengan *lightweight system* (sistem yang ringan) karena jalur komunikasi yang digunakan untuk komunikasi *server* dan *client* yaitu menggunakan protokol *HTTP*.

Stateless dapat diartikan dengan tidak adanya *state* atau penanda dari *client*. Contohnya pada penggunaan *Session*. *Session* adalah cara untuk menyimpan informasi (dalam variabel) yang akan digunakan di beberapa halaman [31]. *Session* merupakan penanda dari *client* yang disimpan di *server* agar *client* dengan mudah dapat dikenali. Namun pada pendekatan *session* tidak diperbolehkan pada *RESTFul*.

Keuntungan dari *stateless* antaralain:

- *RESTFul WS* dapat merespon setiap *request* dari *client* secara independen dikarenakan, setiap sumberdaya dari *RESTFul WS* dialamatkan menggunakan *URI* yang unik.
- *RESTFul WS* tidak menggunakan *Session*. Dengan demikian *RESTFul WS* dapat digunakan pada aplikasi-aplikasi berskala besar.

JWT menyediakan cara bagi *client* untuk mengotentikasi setiap permintaan tanpa harus mempertahankan *session* atau melakukan *login* berulang kali ke *server*. Mekanisme otentikasi *stateless* sebagai penanda bagi *client* tidak pernah disimpan pada *memory server* [5]. *Server* akan melakukan validasi JWT yang digunakan oleh *client* dan jika JWT tersebut dinyatakn valid, client akan diijinkan untuk mengekspos data yang diinginkan.

RESTFul WS menggunakan protokol *HTTP* untuk berkomunikasi dengan *client*. Protokol *HTTP* merupakan jalur komunikasi yang *stateless*. Proses penggabungan *RESTFul WS* dan *JWT* sangat dimungkinkan. Faktor pendukung yang memungkinkan untuk penggabungan tersebut yaitu penggunaan format data yang sama yaitu format data *JSON* dan konsep *stateless* yang menjadi nilai tawar dari pendekatan *RESTFul* dan *JWT*.

V. KESIMPULAN

Membangun komunikasi dan menditribusikan data atau informasi antar sistem merupakan langkah yang dapat ambil untuk mengatasi permasalahan keterbatasan informasi. Langkah tersebut dapat dibuktikan dengan cara pembentukan arsitektur integrasi, penyediaan layanan yang aman, ketersediaan informasi dan proses yang memberikan kemampuan integrasi antar sistem yang dapat digunakan dalam menunjang setiap aktifitas dalam lingkungan kelompok maupun individu [29]. RESTFul WS merupakan suatu cara yang dapat digunakan dalam mengintegrasikan sistem dan mendistribusikan data untuk sistem yang berbeda-beda. Dalam hal pengamanan sumberdaya, langkah yang dapat diambil yaitu dengan cara menggabungkan *RESTFul WS* dan *JWT*. *JWT* merupakan sebuah konsep yang dapat digunakan untuk mengamankan data oleh karena *JWT* memanfaatkan *HMAC SHA256* dalam melakukan enkripsi dan dekripsi data yang komplex berdasarkan setiap bagian dari *JWT* itu sendiri.

Penelitian ini menghasilkan sebuah arsitektur *RESTFul WS* yang aman bagi PT. XYZ. Adapun *RESTFul WS* yang dibangun, menggunakan *JSON Web Token* (*JWT*) dalam mengamankan komunikasi yang terjadi. Dengan demikian, dapat disimpulkan bawa sangat memungkinkan bagi PT. XYZ dalam mengintegrasikan sumberdaya dari sistem dengan menggunakan aplikasi yang berbeda dan dapat diintegrasikan dengan jalur komunikasi yang aman dengan memanfaatkan *RESTFul WS*. Dikarenakan prosedur komunikasi yang di bangun dalam *RESTFul* yaitu setiap melakukan *request*, *client* harus menyertakan *token* yang didapat dari dari hasil *login* untuk di validasi sebelum *request client* di proses.

Keterbatasan penelitian ini yaitu hanya berfokus pada pemanfaatan arsitektur *RESTFul WS* dan pemanfaatan *JWT*. berdasarkan permasalahan tersebut, maka penting untuk dilakukan dilakukan kajian lebih lanjut mengenai

pemanfaatan *RESTFul WS* dan penggunaan algoritma pada *JWT*, terutama algoritma yang digunakan oleh *JWT* saat ini sangat umum digunakan, sehingga dapat menjadi ancaman tersandiri bagi keamanan *RESTFul WS*.

DAFTAR PUSTAKA

[1] M. I. Hussain and N. Dilber, "Restful web services security by using ASP . NET web API MVC based," vol. 12, no. 1, 2014.
[2] R. Sommermeier, A. Heil, and M. Gaedke, "Lightweight Data Integration using the WebComposition Data Grid Service," in *Proceedings of the First International Workshop on Lightweight Integration on the Web (ComposableWeb'09)*, 2009, pp. 30–38.
[3] J. Meng, S. Mei, and Z. Yan, "RESTful Web Services: A Solution for Distributed Data Integration," *Comput. Intell. Softw. Eng. 2009. CiSE 2009. Int. Conf.*, vol. 6, no. 4, pp. 1–4, 2009.
[4] K. V. Kanmani and P. S. Smitha, "Survey on Restful Web Services Using Open Authorization (Oauth) ," *IOSR J. Comput. Eng.*, vol. 15, no. 4, pp. 53–56, 2013.
[5] IETF, "JSON Web Token." [Online]. Available: https://jwt.io/. [Accessed: 16-Jan-2016].
[6] M. B. Jones, "The Emerging JSON-Based Identity Protocol Suite," *W3C Work. Identity Brows.*, pp. 1–3, 2011.
[7] C.-J. Su and C.-Y. Chiang, "Enabling successful Collaboration 2.0: A REST-based Web Service and Web 2.0 technology oriented information platform for collaborative product development," *Comput. Ind.*, vol. 63, no. 9, pp. 948–959, 2012.
[8] R. T. Fielding, "Architectural Styles and the Design of Network-based Software Architectures," *Building*, vol. 54, p. 162, 2000.
[9] M. Kariluoma, "A RESTFUL ARCHITECTURE FOR MULTIUSER VIRTUAL ENVIRONMENTS AND SIMULATIONS," North Dakota State University of Agriculture and Applied Science, 2014.
[10] D. Guinard, M. Mueller, and V. Trifa, "RESTifying Real-World Systems: A Practical Case Study in RFID," in *REST: From Research to Practice*, E. Wilde and C. Pautasso, Eds. Switzerland: Springer Science+Business Media, LLC, 233 Spring Street, New York, NY 10013, USA, 2015, pp. 359–379.
[11] A. J. Rettig, S. Khanna, and R. a. Beck, "Open source REST services for environmental sensor networking," *Appl. Geogr.*, vol. 60, pp. 294–300, 2015.
[12] J. Sandoval, *RESTFul Java Web Service*, vol. 53. Brimingham - Mumbai: Packt Publishing Ltd., 2009.
[13] www.w3.org, "WWW." [Online]. Available: https://www.w3.org/. [Accessed: 16-Feb-2016].
[14] C. Timmons, "Thin Client Distributed Simulation Of Discrete Event Models," Carleton University, 2013.
[15] T. Schreiber, "Amazon Web Services Security Analysis of S3 and SQS Interfaces," Ruhr-Universität Bochum, 2011.
[16] a Rodriquez, "RESTful Web services: The basics," no. February, pp. 1–11, 2008.
[17] S. Jamal and R. Deters, "Using a Cloud-Hosted Proxy to support Mobile Consumers of RESTful Services," *Procedia Comput. Sci.*, vol. 5, pp. 625–632, 2011.
[18] M. B. Jones, B. Campbell, and C. Mortimore, "JSON Web Token (JWT) Profile," *Internet Eng. Task Force*, no. 1, pp. 1–5, 2015.
[19] M. Dosé, "A schematic for comparing web backend application frameworks With regards to responsiveness and load scalability," Chalmers University Of Technology, 2015.
[20] J. Bradley, N. Sakimura, and M. Jones, "JSON Web Token (JWT)."
[21] M. B. Jones, "IDENTITY PROTOCOL SUITE RECOMMENDED PRACTICE : Establishing Suggested Practices Regarding Single Sign-On (ESPReSSO)," *Inf. Stand. Q.*, vol. 2, no. 3, p. 32, 2014.
[22] N. Birwadkar, P. Malone, J. Hange, K. Doyle, E. Robson, D. Conway, S. Ivanov, Ł. Radziwonowicz, and R. Kleinfeld, "Personal Cloudlets : Implementing a User-Centric Datastore with Privacy Aware Access Control for Cloud-based Data Platforms," *IJARCCE*, vol. 4, no. 12, pp. 570–574, 2015.
[23] S. Yusmantoro, E. Hermansyah, and R. Efendi, "PENGAMANAN KEASLIAN SURAT ELGAMAL DAN SECURE HASH ALGORITHM 256 STUDI KASUS : BADAN PELAYANAN PERIZINAN TERPADU (BPPT) KOTA BENGKULU," vol. 2, no. 1, pp. 28–36, 2014.
[24] J. Huff, D. Schlacht, A. Nadalin, J. Simmons, P. Rosenberg, P. Madsen, and T. Ace, "Online Multimedia Authorization Protocol An Industry Standard for Authorized Access to," 2012.
[25] T. Guenther, "Bachelor Thesis Automatic Recognition , Processing and Attacking of Single Sign-On Protocols with Burp Suite Erklärung," Ruhr-University Bochum, Germany, 2015.
[26] S. Price, P. a. Flach, S. Spiegler, C. Bailey, and N. Rogers, "SubSift web services and workflows for profiling and comparing scientists and their published works," *Futur. Gener. Comput. Syst.*, vol. 29, no. 2, pp. 569–581, 2013.
[27] I. Abdullahi, S. Arif, and S. Hassan, "Survey on caching approaches in Information Centric Networking," *J. Netw. Comput. Appl.*, vol. 56, no. 2015, pp. 48–59, 2015.
[28] K. Annapureddy, "Security Challenges in Hybrid Cloud Infrastructures," *Aalto Univ. Semin. Netw. Secur.*, p. T–110.5290, 2010.
[29] T. Kristanti, "Integrasi Enterprise (Studi Kasus: Yayasan Pendidikan 'X')," *J. Sist. Inf.*, vol. 4, p. 19, 2012.
[30] M. Athanasopoulos and K. Kontogiannis, "Extracting REST resource models from procedure-oriented service interfaces," *J. Syst. Softw.*, vol. 100, pp. 149–166, 2015.
[31] www.w3schools.com, "HTML form enctype Attribute." [Online]. Available: http://www.w3schools.com. [Accessed: 16-Feb-2016].
[32] M. Jones, B. Campbell, and C. Mortimore, "JSON Web Token (JWT) Bearer Token Profiles for OAuth 2.0," no. draft-jones-oauth-jwt-bearer-04.txt, 2012.