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. 3 (2020) 591 - 600 ISSN Media Elektronik: 2580-0760

Implementasi Web Service pada Perusahaan Logistik menggunakan JSON Web Token dan Algoritma Kriptografi RC4

Mochammad Rizky Royani[1], Arief Wibowo[2]
[1,2]Program Studi Teknik Informatika, Fakultas Teknologi Informasi, Universitas Budi Luhur
[1]mrizkyroyani@gmail.com, [2]arief.wibowo@budiluhur.ac.id

Abstract

The development of e-commerce in Indonesia in the last five years has significantly increased the growth for logistics service companies. The Indonesian Logistics and Forwarders Association (ALFI) has predicted the growth potential of the logistics business in Indonesia to reach more than 30% by 2020. One of the efforts of logistics business companies to improve services in the logistics services business competition is to implement web service technology on mobile platforms, to easy access to services for customers. This research aims to build a web service with a RESTful approach. The REST architecture has limitations in the form of no authentication mechanism, so users can access and modify data. To improve its services, JSON Web Token (JWT) technology is needed in the authentication process and security of access rights. In terms of data storage and transmission security, a cryptographic algorithm is also needed to encrypt and maintain confidentiality in the database. RC4 algorithm is a cryptographic algorithm that is famous for its speed in the encoding process. RC4 encryption results are processed with the Base64 Algorithm so that encrypted messages can be stored in a database. The combination of the RC4 method with the Base64 method has strengthened aspects of database security. This research resulted in a prototype application that was built with a combination of web service methods, JWT and cryptographic techniques. The test results show that the web service application at the logistics service company that was created can run well with relatively fast access time, which is an average of 176 ms. With this access time, the process of managing data and information becomes more efficient because before making this application the process of handling a transaction takes up to 20 minutes.

Keywords: Web Service, Cryptography, Restful, Json Web Token, RC4

Abstrak

Perkembangan e-commerce di Indonesia dalam lima tahun terakhir telah meningkatkan pertumbuhan bagi perusahaan jasa logistik secara signifikan. Asosiasi Logistik dan Forwarder Indonesia (ALFI) telah memprediksi potensi pertumbuhan bisnis logistik di Indonesia bisa mencapai lebih dari 30% pada tahun 2020. Salah satu upaya perusahaan bisnis logistik untuk meningkatkan pelayanan dalam persaingan bisnis jasa logistik adalah menerapkan teknologi web service pada platform mobile, untuk kemudahan akses layanan bagi pelanggan. Penelitian ini bertujuan untuk membangun web service dengan pendekatan RESTful. Arsitektur REST memiliki keterbatasan berupa tidak ada mekanisme autentikasi, sehingga pengguna dapat mengakses dan memodifikasi data. Untuk menyempurnakan layanannya, diperlukan teknologi JSON Web Token (JWT) dalam proses autentikasi dan keamanan hak akses. Dalam sisi keamanan penyimpanan dan transmisi data, juga diperlukan sebuah algoritma kriptografi untuk mengenkripsi dan menjaga kerahasiaan pada basis data. Algoritma RC4 merupakan salah satu algoritma kriptografi yang terkenal dengan kecepatan dalam proses penyandian. Hasil enkripsi RC4 diproses dengan Algoritma Base64 agar pesan yang dienkrip dapat disimpan pada basis data. Kombinasi metode RC4 dengan metode Base64, telah memperkuat aspek keamanan basis data. Penelitian ini menghasilkan prototipe aplikasi yang dibangun dengan kombinasi metode web service, JWT maupun teknik-teknik kriptografi. Hasil pengujian menunjukkan bahwa aplikasi web service pada perusahaan jasa logistik yang diciptakan dapat berjalan dengan baik dengan waktu akses yang relatif cepat, yaitu rata-rata 176 ms. Dengan waktu akses ini maka proses pengelolaan data dan informasi menjadi lebih efisien karena sebelum dibuatnya aplikasi ini proses penanganan sebuah transaksi memakan waktu hingga 20 menit.

Kata kunci: *Web Service*, Kriptografi, Restful, *Json Web Token*, RC4

1. Pendahuluan

Perusahaan logistik berkembang cepat di Indonesia, Industri pengiriman barang tumbuh positif tiap tahunnya seiring perkembangan *e-commerce* yang tercatat 500% dalam empat tahun terakhir. Asosiasi Logistik dan Forwarder Indonesia (ALFI) memprediksi potensi

Diterima Redaksi : 13-03-2020 | Selesai Revisi : 09-06-2020 | Diterbitkan Online : 20-06-2020

pertumbuhan bisnis logistik di Tanah Air bisa mencapai lebih dari 30% pada 2020 [1]. Bila dihitung secara rinci, estimasi pertumbuhan sektor ini secara menyeluruh bisa mencapai 40 triliun rupiah atau lebih per tahun. Dengan pertumbuhan atau penetrasi tersebut, maka perusahaan logistik perlu meningkatkan layanan. Salah satu usaha meningkatkan layanan adalah penyediaan infrastruktur berbasis teknologi web. Teknologi yang dapat dikembangkan seperti, sistem *tracking* dan pemutakhiran data.

Jasa pengiriman adalah bisnis yang mengutamakan kepercayaan pada kiriman yang harus tiba di tujuan dengan cepat dan aman. Pada bisnis jasa pengiriman, beberapa permasalahan dalam pengelolaan informasi, misalnya sering terjadi perbedaan data antar bagian, barang yang dikirim jumlahnya tidak sesuai atau barang kiriman mengalami kehilangan, serta kesalahan input pada sistem yang diakibatkan oleh kesalahan manusia, Hal tersebut dikarenakan penginputan data dilakukan dengan manual *form* yang relatif sederhana dan memakan waktu yang relatif lambat, yaitu kurang lebih dua puluh menit untuk satu penanganan proses order atau permintaan layanan pengiriman.

Teknologi *web service* menyediakan layanan-layanan pada suatu web, berupa *object* dan *method* yang memiliki tujuan untuk memfasilitasi sistem lain agar dapat berkomunikasi tanpa batasan platform serta bahasa pemrograman. Dengan penggunaan *web service*, memungkinkan sistem dengan platform atau dengan bahasa pemrograman yang berbeda dapat saling berinteraksi. Untuk mendukung integritas tersebut maka diperlukan teknologi RESTful *web service* sebagai solusi dari pertukaran data. Dengan RESTful *web service* maka setiap perangkat yang menggunakan jaringan internet dapat mengakses layanan data.

RESTful merupakan metode yang sedang tren, dikarenakan lebih mudah dalam penerapan, membutuhkan lebih sedikit *bandwith* dan sumber daya bila dibandingkan dengan SOAP. Berdasarkan hasil perbandingan SOAP dan REST, REST memiliki performa yang lebih bagus dibandingkan dengan SOAP untuk *request* dan *response* pada *web service* [2]. Teknologi *web service* memerlukan sisi keamanan yang baik. Hal itu bertujuan untuk menjaga kerahasiaan data dari transaksi yang ada. Keamanan data tersebut sangat penting untuk mencegah pencurian data oleh pihak yang tidak bertanggung jawab, yang akan membuat hilangnya kepercayaan pelanggan dengan jasa yang diberikan.

Web service adalah sebuah software yang dibuat untuk mendukung interopabilitas interaksi mesin ke mesin melalui sebuah jaringan [3]. *Web service* berbasis REST relatif rentan hal keamanan, namun dengan menggunakan *Json Web Token* (JWT) dapat memberikan fitur autentikasi kepada *user* dalam memberikan hak akses data *RESTful web service*. Studi terdahulu yang menggunakan *web service* antara lain perbandingan menggunakan SOAP dan RESTful [2], penambahan keamanan hak ases user pada *web service* menggunakan JSON Web Token (JWT) [4], serta penambahan keamanan enkripsi database menggunakan algoritma kriptografi RC-4 yang dinilai sederhana dan mudah untuk dikembangkan pada sistem yang kita buat [5].

Pada studi lainnya, *web service* tanpa JWT memungkinkan semua pengguna dapat melakukan *request* karena belum adanya validasi status akses [6]. Dalam sisi keamanan dalam penyimpanan serta transmisi data, diperlukan sebuah algoritma kriptografi untuk mengenkripsi data untuk melindungi pencurian oleh pihak yang tidak bertanggung jawab. Algoritma kriptografi RC4 merupakan salah satu algoritma yang popular dengan kecepatan dan kesederhanaan sehingga mudah untuk dikembangkan dan diimplementasikan pada sistem yang dikembangkan [5]. Hasil dari proses enkripsi algoritma RC4 adalah berupa *ciphertext* yang terdiri dari simbol-simbol. Agar dapat disimpan ke dalam basis data maka digunakan algoritma Base64 dalam proses dekripsi, sehingga seluruh data tersimpan dengan aman pada basis data.

Untuk menjawab permasalahan yang terjadi, maka penelitian ini membangun sebuah aplikasi *web service* menggunakan *RESTful* yang diimplementasikan pada berbagai platform. Pada aspek keamanan database peneliti menggunakan algoritma kriptografi *Rivest Code 4* (RC4) agar proses pengolahan data berlangsung lebih aman, cepat dan akurat. Pembangunan aplikasi *web service* dikombinasikan dengan algoritma enkripsi diharapkan dapat membuat proses pengelolaan data informasi menjadi lebih efisien dan mengurangi rasa kekhawatiran terkait dengan keamanan data dari tindakan pencurian maupun penyalahgunaan informasi.

2. Metodologi Penelitian

Untuk dapat memecahkan permasalahan maka penelitian ini mengimplementasikan *web cervice*, dengan tahap-tahap sebagai terlihat pada Gambar 1.

Gambar 1. Tahap Penelitian.

Pada Gambar 1, menjelaskan metode penelitian yang dilakukan. Studi literatur berupa penelitian-penelitian sebelumnya untuk memperoleh informasi dengan mempelajari metode-metode yang digunakan seperti *RESTful, JWT*, kriptografi, yang berhubungan dengan penelitian. Analisis permasalahan dilakukan untuk

menentukan kebutuhan sistem. Penerapan metode-metode seperti, Json Web Token, Algoritma kriptografi *RC-4*, dan algoritma *Base64* dalam *RESTful web service*. Perancangan sistem android dengan menggunakan model sistem yang sudah dibuat. Evaluasi sistem dengan melakukan uji coba untuk mengetahui stabilitas dan performa sistem yang telah dibuat.

REST merupakan gaya arsitektural dengan aturan seperti antar muka yang serupa, dengan kesamaan antar muka maka penerapan *web service* dapat memaksimalkan kinerja web terutama pada performa, skalabilitas, dan kemudahan untuk dimodifikasi. Pada arsitektur *REST*, data dan fungsi dianggap sebagai sumber daya dan diakses melalui *Uniform Resource Identifier* (URI), biasanya berupa tautan pada web [8]. Cara kerja RESTful *web service* berawal dari *Client* yang melakukan *request* dengan metode HTTP, dan server akan menerima serta memproses data. Selanjutnya server akan memberikan respon berupa hasil pemrosesan data kepada pengguna [4].

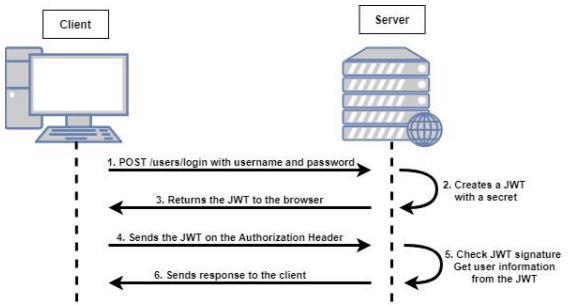
Cara kerja RESTful terdiri dari beberapa bagian, pertama adalah Verb, metode HTTP yang digunakan di antaranya adalah GET yaitu melihat tampilan dari resource, POST digunakan untuk menambah atau memperbarui data, dan DELETE digunakan untuk menghapus suatu data dari resource. Kedua adalah Uniform Resource Identifier (URI) memungkinkan untuk melakukan identifikasi lokasi atau sumber data pada server. Ketiga adalah HTTP Version yaitu untuk memperlihatkan versi dari HTTP yang digunakan. Selanjutnya yaitu Request Header, berisi tentang metadata yang digunakan pada HTTP Request seperti format body, cache, type client, dll. Terakhir adalah Request Body yang berisi konten dari data.

Pada penelitian ini, REST dirancang dengan arsitektur yang bekerja memanipulasi data pada sistem dengan menggunakan metode HTTP seperti GET, PUT, POST, DELETE, dll. Dalam arsitekturnya, REST memberikan unsur pengembalian data berupa format JSON, XML, HTML, atau format lainnya. REST disebut sebagai stateless, dikarenakan tidak dapat menyimpan sebuah penanda dari *client* pada *server*. Sehingga *web service* dapat melayani masing-masing *request* secara independen. Daftar web service yang diimplementasikan pada penelitian ini terlihat pada Tabel 1.

Tabel 1. Daftar Layanan *Web Service*

Service Name	Method	Path	Parameter
Login to application	POST	/login	Username, password
User data list	GET	/user/user_data	
User entry	POST	/user/user_entry	Name, username, password, password2, level
Delete user from list	POST	/user/user_delete	id
Item list	GET	/item/item_data	
Item entry	POST	/item /item_entry	name, cat, unit, weight,
Delete item from list	POST	/item /delete_item	Item code
Category list	GET	/category/category_data	
Category entry	POST	/category/category_entry	category
Delete category from list	POST	/category/category_delete	code
Shipment data list	GET	/shipment/shipment_data	
Shipment entry	POST	/shipment/shipment_entry	sender, receiver, shipment_detail
Delete shipment from the list	POST	/shipment/shipment_delete	Shipment_code
Shipment edit	POST	/shipment/shipment_edit	Shipment_code, sender, receiver
Shipment detail edit	POST	/shipment/shipment_detail_edit	Shipment_code, item_id, qty
Client data list	GET	/client/client_data	
Client entry	POST	/client/client_entry	Name, address, phone
Delete client from the list	POST	/client/client_delete	id

Pada Tabel 1, terlihat daftar layanan *web service* yang diterapkan pada aplikasi ini. Berisi nama layanan yang diberikan, metode, *path*, dan parameter yang digunakan. *JSON Web Token* adalah sebuah token yang berbentuk *string* panjang dan random, terdiri dari informasi mandiri yang berguna untuk melakukan autentikasi user dan untuk bertukar informasi. *JWT* berukuran kecil sehingga dapat dikirim melalui *URL*, *HTTP POST*, atau pada *Header HTTP*. Karena *JWT* memiliki ukuran yang kecil, akan mempercepat proses transmisi data. Token yang dihasilkan berisi informasi mandiri dari pengguna, sehingga proses *query* ke *database* hanya perlu dilakukan sekali [4].

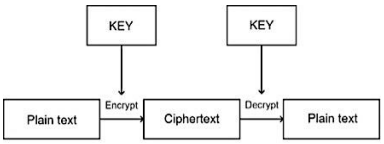


Gambar 2. Mekanisme JSON Web Token [9].

Pada Gambar 2, terlihat bahwa *JSON Web Token (JWT)* bekerja dengan menggunakan *token* pada *password*. Ketika user berhasil melakukan *login* maka server akan

memberikan sebuah *token* yang selanjutnya akan disimpan oleh pengguna pada *local storage* atau *cookies browser*. Bila pengguna ingin mengakses halaman-halaman tertentu maka diperlukan token tersebut. Pengguna akan mengirim balik token yang diberikan di awal sebagai bukti bila pengguna telah melakukan *login*. Struktur dasar *JWT* terdiri dari tiga bagian. Pertama adalah *Header*, lalu kedua adalah bagian *payload* atau data, dan yang ketiga adalah bagian *signature* [10].

Kriptografi *(Cryptography)* berasal dari Bahasa Yunani, terdiri dari dua suku kata yaitu *'kripto'* dan *'graphia'*. *Kripto* artinya menyembunyikan, sedangkan *graphia* artinya tulisan. Kriptografi merupakan suatu ilmu yang yang berhubungan dengan aspek keamanan informasi, seperti keabsahan data, integritas data, kerahasian data, dan autentikasi data. Namun demikian, tidak semua aspek keamanan informasi dapat diselesaikan dengan kriptografi [11]. Alur kriptografi terlihat pada Gambar 3.



Gambar 3. Alur Kriptografi

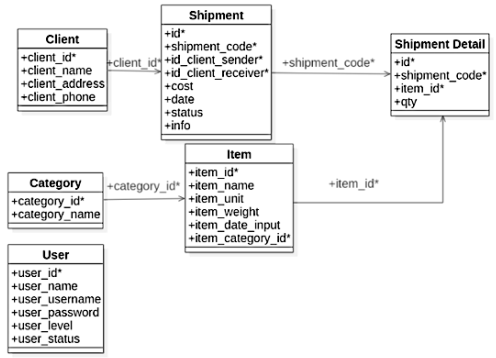
Terlihat pada Gambar 3, bahwa algoritma kriptografi terdiri dari tiga fungsi dasar yang pertama adalah Enkripsi. Enkripsi merupakan istilah lain dari proses menyandikan data penting kedalam bentuk simbol-simbol yang tidak dapat dimengerti lagi oleh pihak lain sehinggan keaslian dan keamanan data dapat terjaga. Fungsi kedua adalah Dekripsi, yaitu proses untuk merubah atau mengembalikan data tersandi kebentuk aslinya agar arti data dapat dimengerti oleh penerima. Fungsi ketiga adalah Kunci, merupakan elemen yang paling penting dalam mengimplementasikan proses enkripsi dan dekripsi. Keamanan kunci di dalam kriptografi menjadi prioritas karena serumit apapun algoritma yang digunakan akan dapat dipecahkan bila kunci yang digunakan berhasil ditemukan. Kunci terbagi menjadi dua bagian, kunci rahasia *(private key)* dan kunci umum *(public key)* [12].

Algoritma RC4 merupakan salah satu algoritma kunci simetris yang berbentuk *stream cipher*, yaitu memproses unit atau input data pada satu saat. Unit atau data pada umumnya sebuah *byte* atau kadang-kadang bit. Algoritma ini tidak harus menunggu sejumlah input data tertentu sebelum diproses, atau menambahkan byte tambahan untuk mengenkripsi. Algoritma ini ditemukan pada tahun 1987 oleh Ronald Rivest dan menjadi simbol keamanan RSA. RC4 merupakan enkripsi *stream simetric proprietary* yang dibuat oleh RSA Data Security Inc. (RSADSI) [13].

Transformasi *Base64* merupakan salah satu algoritma untuk *Encoding* dan *Decoding* suatu data ke dalam format ASCII, yang didasarkan pada bilangan dasar 64 atau bisa dikatakan sebagai salah satu metode yang digunakan untuk melakukan *encoding* (penyandian) terhadap data *binary*. Karakter yang dihasilkan pada transformasi Base64 ini terdiri dari A..Z, a..z dan 0..9, serta ditambah dengan dua karakter terakhir yang bersimbol yaitu + dan / serta satu buah karakter sama dengan (=) yang digunakan untuk penyesuaian dan menggenapkan data *binary* atau istilahnya disebut sebagai pengisi *pad*. Karakter simbol yang akan dihasilkan akan tergantung dari proses algoritma yang berjalan. Kriptografi Transformasi *Base64* banyak digunakan di dunia internet sebagai media data format untuk mengirimkan data, ini dikarenakan hasil dari Base64 berupa *plaintext*, maka data ini akan jauh lebih mudah dikirim dan disimpan kedalam database [14].

3. Hasil dan Pembahasan

Dalam aspek database, kebutuhan tabel basis data yang dibuat dalam bentuk *Logical Record Structure* (LRS) untuk aplikasi *web service* yang dibangun untuk perusahaan logistik, terlihat pada Gambar 4.



Gambar 4. Rancangan LRS *(Logical Record Struktur)*

Pada Gambar 4, terlihat struktur dari basis data yang digunakan untuk membuat aplikasi *web service* ini. Terdiri dari, tabel *user, client, item, shipment, shipment detail, dan category.* Basis data yang dirancang digunakan untuk menyimpan seluruh hasil pemodelan dan implementasi JWT pada penelitian ini, disertai dengan teknik pengamanan keamanan berbasis kriptografi.

Proses pembangunan aplikasi pada studi ini menggunakan beberapa tahap implementasi atau penerapan sistem *web service* dan algoritma kriptografi yang sudah dibuat. Tahapan implementasi terdiri dari dua fase, pertama adalah implementasi JSON Web Token (JWT) dan kedua adalah implementasi proses enkripsi dan dekripsi menggunakan algoritma RC-4 dan Algoritma Base64.

Fase pertama implementasi adalah penerapan JSON Web Token yang dilakukan untuk memastikan bahwa setiap user yang login akan divalidasi dengan benar. Implementasi JWT dilakukan dengan eksperimen akses masuk aplikasi dengan dua user, yaitu admin dan user non-admin. Pada proses login dengan user name admin,

status berhasil dalam waktu 69 ms, menghasilkan token dengan bentuk sebagai berikut: "eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJ1c2VyX2lkIjoiMSIsInVzZXJfdXNlcm5hbWUiOiJhZG1pbiIsImlhdCI6MTU4OTA0MjY5OCwiZXhwIjoxNTg5MzAxODk4fQ.jQs9JdFHQLvGIg7eUvD6nFOHYKiKT7f1EZBi3Xk6Xjo".

Pada proses ujicoba menggunakan login non-admin, dibuat satu akun pengguna “rizkyroyani” dengan status berhasil dalam waktu 63 ms, dengan token yang terbentuk adalah sebagai berikut: "eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJ1c2VyX2lkIjoiMiIsInVzZXJfdXNlcm5hbWUiOiJyaXpreXJveWFuaSIsImlhdCI6MTU4OTA0MjYxOCwiZXhwIjoxNTg5MzAxODE4fQ.TTL58Ily126Y9_ufFZTjiJTpsFwflFAXBPxbt1bTmVE"

Pada implementasi di atas, *login* dilakukan dengan user name yang digunakan untuk masuk kedalam sistem, apabila *login* berhasil dilakukan akan muncul pemberitahuan berikut dengan hak akses dan token yang diberikan. Untuk dapat melakukan proses *login*, *admin* diharuskan menambah *user* terlebih dahulu.

Fase kedua dalam implementasi adalah penerapan enkripsi dan dekripsi menggunakan algoritma kriptografi RC-4 dan algoritma Base64. Pada fase ini, implementasi enkripsi dan dekripsi yang dilakukan dengan memasukkan satu jenis proses input layanan pengiriman barang dari satu pelanggan.

Tabel 2. Pengujian Enkripsi dan Dekripsi

Plaintext	RC-4	RC-4 + Base64	ET (sec)	DT (sec)
Client name: “Mochammad Rizky Royani”	ì›q¾ÿZ â!«)nÃ ¨:<ÀoI ÂÏ*	7Jtxvv9a4iGrK W7DHag6PMB vScLPKg==	4.4107	4.5061
Client address: “Jakarta”	ë•y·iCî	65V5t+xD7g==	4.1007	3.9815
Client Phone: “08128409243 4”	‘Ì#ä¦— ¿yý=ž	kcwj5KYDv3n9 PQ+e	4.1009	4.1007
Item name: “Australian Oranges”	à• a¢ì Vã)®g å¢-{÷s	4IFhouxW4ymu ZxzlFaIte/dz	4.4107	4.2915
Item unit: “Kilogram”	ê• ~¹ù Eî-	6p1+uflF7i0=	4.0054	4.1007
Item qty: “100”	• Ä\"	kMQi	4.0054	4.0054
		Rate time	4.1723	4.1643

Pada Tabel 2, terlihat bahwa penerapan metode enkripsi dan dekripsi dengan algoritma RC-4 dan Base64 berjalan dengan baik. Terlihat pada kolom pertama adalah *plaintext* yang dimasukan, akan berubah menjadi *ciphertext* pada kolom kedua dan ketiga, dan pada kolom keempat dan kelima diperlihatkan waktu yang dibutuhkan untuk melakukan proses enkripsi dan dekripsi. Dengan rata-rata waktu yang dibutuhkan untuk proses enkripsi adalah 4,1723 detik dan 4,1643 detik untuk proses dekripsi.

penerapan metode *POST /auth/login* untuk masuk ke dalam sistem telah menghasilkan sebuah token "eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJ1c2VyX2lkIjoiMSIsInVzZXJfdXNlcm5hbWUiOiJhZG1pbiIsImlhdCI6MTU4OTE4MjAwOSwiZXhwIjoxNTg5NDQxMjA5fQ.ZV4Kd5sURbniZfvsip8t6DwjK6RdN0gy3l8mnEJk5IM" dan pesan “Berhasil Login”. Pengujian sukses dilakukan dengan waktu 73 ms, dan menghasilkan ukuran file sebesar 827 B.


```
{
    "status": 200,
    "error": false,
    "pengguna": [
        {
            "user_nama": "Admin",
            "user_username": "admin",
            "user_level": "1",
            "user_status": "1"
        },
```


Gambar 5. Penerapan Metode GET User Data

Pada Gambar 5 terlihat penerapan metode *GET /user/user_data* untuk melihat daftar dari user yang tersedia. Pengujian berhasil dilakukan dengan waktu 61 ms, dan menghasilkan ukuran file sebesar 1.09 KB.


```
{
    "status": 200,
    "error": false,
    "message": "Berhasil."
}
```


Gambar 6. Penerapan Metode POST User Entry

Pada Gambar 6 terlihat penerapan metode *POST /user/user_entry* untuk menambahkan user yang dapat digunakan. Pengujian berhasil dilakukan dengan waktu 207 ms, dan menghasilkan ukuran file sebesar 648 B.


```
{
    "status": 200,
    "error": false,
    "message": "Berhasil."
}
```

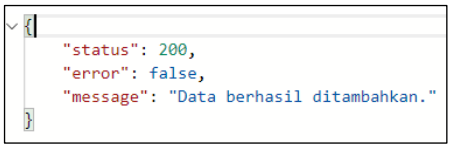

Gambar 7. Penerapan Metode POST Delete User

Pada Gambar 7 terlihat penerapan metode *POST /user/user_delete* untuk menghapus user pada daftar. Pengujian berhasil dilakukan dengan waktu 151 ms, dan menghasilkan ukuran file sebesar 648 B.


```
{
    "status": 200,
    "error": false,
    "barang": [
        {
            "barang_id": "BR000001",
            "barang_nama": "Kompor",
            "barang_satuan": "Unit",
            "barang_berat": "45",
            "barang_kategori_id": "1"
        },
```


Gambar 8. Penerapan Metode GET Item Data

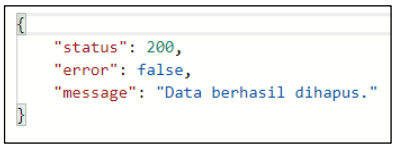
Pada Gambar 8 terlihat penerapan metode *GET /item/item_data* untuk melihat daftar barang yang ada. Pengujian berhasil dilakukan dengan waktu 53 ms, dan menghasilkan ukuran file sebesar 981 B.



```
{
    "status": 200,
    "error": false,
    "message": "Data berhasil ditambahkan."
}
```

Gambar 9. Penerapan Metode POST Item Entry

Pada Gambar 9 terlihat penerapan metode *POST /item/item_entry* untuk menambah data barang yang akan dikirim. Pengujian berhasil dilakukan dengan waktu 661 ms, dan menghasilkan ukuran file sebesar 665 B.



```
{
    "status": 200,
    "error": false,
    "message": "Data berhasil dihapus."
}
```

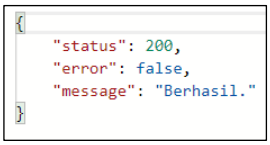
Gambar 10. Penerapan Metode POST Delete Item

Gambar 10 menunjukkan penerapan metode *POST /item/delete_item* untuk menghapus barang dari daftar. Pengujian berhasil dilakukan dengan waktu 173 ms, dan menghasilkan ukuran file sebesar 660 B.

```
{
    "status": 200,
    "error": false,
    "kategori": [
        {
            "kategori_id": "10",
            "kategori_nama": "Dangerous Goods"
        },
        {
            "kategori_id": "9",
            "kategori_nama": "Diplomatic Pouch (DIP)"
        },
```

Gambar 11. Penerapan Metode GET Category Data

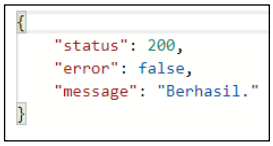
Gambar 11 menunjukkan penerapan metode *GET /category/category_data* untuk melihat daftar kategori barang yang tersedia. Pengujian berhasil dilakukan dengan waktu 125 ms, dan menghasilkan ukuran file sebesar 1.31 KB.



```
{
    "status": 200,
    "error": false,
    "message": "Berhasil."
}
```

Gambar 12. Penerapan Metode POST Category Entry

Gambar 12 menunjukkan penerapan metode *POST /category/category_entry* untuk menambah kategori. Pengujian berhasil dilakukan dengan waktu 154 ms, dan menghasilkan ukuran file sebesar 648 B.



```
{
    "status": 200,
    "error": false,
    "message": "Berhasil."
}
```

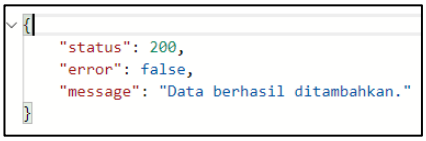
Gambar 13. Penerapan Metode POST Delete Category

Gambar 13 menunjukkan penerapan metode *POST /category/category_delete*. Pengujian berhasil dilakukan dengan waktu 157 ms, dan menghasilkan ukuran file sebesar 648 B.

```
{
    "status": 200,
    "error": false,
    "pengiriman": {
        "SHPMNT-5EB8FD78B": {
            "id_client_pengirim": "3",
            "id_client_penerima": "2",
            "biaya": "Rp. 480000",
            "tanggal": "2020-05-11 14:23:36",
            "status": "0",
            "keterangan": "",
            "detail_pengiriman": [
                {
                    "barang_id": "BR000002",
                    "barang_nama": "Gas",
                    "barang_berat": "3 kg",
                    "qty": "100"
                },
```

Gambar 14. Penerapan Metode GET Shipment Data

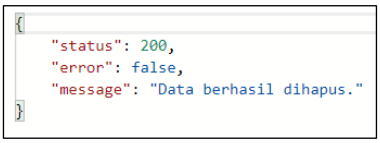
Gambar 14 menunjukkan implementasi dengan metode *GET /shipment/shipment_data* untuk melihat daftar transaksi pengiriman. Pengujian berhasil dilakukan dengan waktu 89 ms, dan menghasilkan ukuran file sebesar 978 B.



```
{
    "status": 200,
    "error": false,
    "message": "Data berhasil ditambahkan."
}
```

Gambar 15. Penerapan Metode POST Shipment Entry

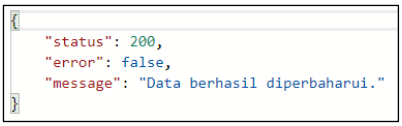
Gambar 15 menunjukkan implementasi dengan metode *POST /shipment/shipment_entry* untuk menambah transaksi. Pengujian berhasil dilakukan dengan waktu 340 ms, dan menghasilkan ukuran file sebesar 793 B.



```
{
    "status": 200,
    "error": false,
    "message": "Data berhasil dihapus."
}
```

Gambar 16. Penerapan Metode POST Delete Shipment

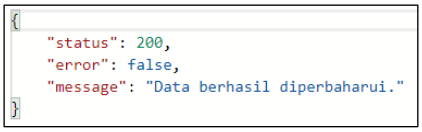
Gambar 16 menunjukkan penerapan *POST /shipment/shipment_delete* untuk menghapus transaksi pada daftar. Pengujian berhasil dilakukan dengan waktu 223 ms, dan menghasilkan ukuran file sebesar 661 B.



```
{
    "status": 200,
    "error": false,
    "message": "Data berhasil diperbaharui."
}
```

Gambar 17. Penerapan Metode POST Shipment Edit

Gambar 17 menunjukkan penerapan metode *POST /shipment/shipment_edit* untuk mengubah data pengiriman. Pengujian berhasil dilakukan dengan waktu 66 ms, dan menghasilkan ukuran file sebesar 978 B.



```
{
    "status": 200,
    "error": false,
    "message": "Data berhasil diperbaharui."
}
```

Gambar 18. Penerapan Metode POST Shipment Detail Edit

Gambar 18 menunjukkan penerapan metode *POST /shipment/shipment_detail_edit* untuk mengubah detil

pengiriman. Pengujian berhasil dilakukan dengan waktu 102 ms, dan menghasilkan ukuran file sebesar 666 Byte.


```
{
    "status": 200,
    "error": false,
    "client": [
        {
            "client_id": "4",
            "client_nama": "Elbarca El Swalatha",
            "client_alamat": "Jl. Veteran Raya No.5, Jakarta
            "client_notelp": "0896765564323"
        },
```


Gambar 19. Penerapan Metode GET Client Data

Gambar 19 menunjukkan penerapan metode *GET /client/client_data* untuk melihat daftar klien yang ada. Pengujian berhasil dilakukan dengan waktu 120 ms, dan menghasilkan ukuran file sebesar 1.16 KB.


```
{
    "status": 200,
    "error": false,
    "message": "Berhasil."
}
```


Gambar 20. Penerapan Metode POST Entry Client

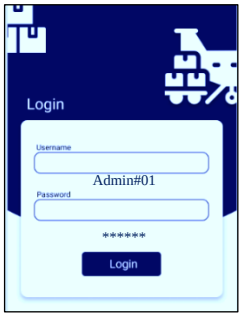
Gambar 20 menunjukkan penerapan metode *POST /client/client_entry* untuk menambahkan klien baru. Pengujian berhasil dilakukan dengan waktu 130 ms, dan menghasilkan ukuran file sebesar 648 B.


```
{
    "status": 200,
    "error": false,
    "message": "Berhasil."
}
```


Gambar 21. Penerapan Metode POST Delete Client

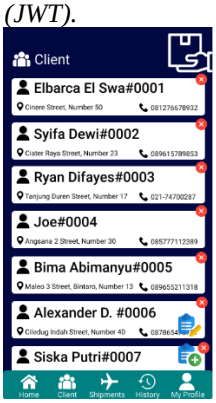
Gambar 21 menunjukkan penerapan metode *POST /client/client_delete* untuk menghapus data klien dari daftar. Pengujian berhasil dilakukan dengan waktu 184 ms, dan menghasilkan ukuran file sebesar 648 B.

Tahap akhir studi ini adalah melakukan implementasi pada aplikasi mobile Android. Berikut adalah tangkapan layar pada OS Android untuk aplikasi *web service* logistik yang sudah dibuat, pada fitur-fitur utama yang menerapkan implementasi *web service* dengan JWT dan kriptografi.



Gambar 22. Tangkapan Layar Login Aplikasi

Gambar 22, merupakan tangkapan layar dari form login dari aplikasi. *User* yang ingin mendapatkan akun, harus didaftarkan melalui *admin* pemegang sistem. Pada fitur ini, merupakan tampilan dari hasil implementasi dengan metode *POST /auth/login*. *User* yang sudah *login* akan divalidasi mengenai hak akses data dengan metode *JSON Web Token (JWT).*



Gambar 23. Tangkapan Layar Daftar Klien

Gambar 23 berisi tentang daftar klien yang ada, dari nama, alamat, nomor telepon. User dapat melakukan perintah tambah klien, edit klien, dan hapus klien. Pada fitur ini, data klien akan disimpan dalam metode enkripsi meliputi, nama klien, alamat klien, dan nomor telepon. Fitur ini merupakan tampilan implementasi dari metode *GET /client/client_data*, data yang ada disimpan dengan metode enkripsi dan ditampilkan dengan proses dekripsi.

Gambar 24. Tangkapan Layar Form Tambah Shipment

Pada Gambar 24, merupakan tangkapan layar dari form untuk menambah transaksi pengiriman. Didalamnya berisi kode pengiriman, pengirim, penerima, barang, jumlah barang, keterangan, dan biaya yang harus dikeluarkan. Fitur ini merupakan tampilan layar dari metode *POST /shipment/shipment_entry* yang dimana data akan disimpan dengan proses enkripsi dan dekripsi.

Gambar 25. Tampilan Info Detil Pengiriman

Pada Gambar 25, merupakan tangkapan layar dari info detil setiap pengiriman. Berisi kode pengiriman, pengirim, penerima, barang, jumlah barang, kategori, biaya pengiriman, keterangan, tanggal pengiriman, dan status pengiriman. Pada fitur ini, data disimpan dengan metode enkripsi dan dekripsi menggunakan kriptografi RC-4 dan algoritma Base64.

3.3. Evaluasi Program Aplikasi

Setelah dilakukan tahap pengujian aplikasi dengan metode *Blackbox*, ditemukan hasil berupa satuan waktu dalam proses autentifikasi hingga seluruh fitur dan proses. Hasil evaluasi ditunjukkan pada Tabel 3.

Tabel 3. Evaluasi Proses Pada Aplikasi

No	Proses	Deskripsi	Waktu
1.	*GET /user/user_data*	Melihat daftar pengguna	61 ms
2.	*POST /user/user_entry*	Menambah pengguna	207 ms
3.	*POST /user/user_delete*	Menghapus pengguna	151 ms
4.	*GET /item/item_data*	Melihat daftar barang	53 ms
5.	*POST /item/item_entry*	Menmbah data barang kiriman	661 ms
6.	*POST /item/delete_item*	Menghapus data barang	173 ms
7.	*GET /category/category_data*	Melihat daftar kategori	125 ms
8.	*POST /category/category_entry*	Menambah kategori	154 ms
9.	*POST /category/category_delete*	Menghapus kategori	157 ms
10.	*GET /shipment/shipment_data*	Melihat daftar pengiriman	89 ms
11.	*POST /shipment/shipment_entry*	Menambah data pengiriman	340 ms
12.	*POST /shipment/shipment_delete*	Menghapus data pengiriman	223 ms
13.	*POST /shipment/shipment_edit*	Mengubah data pengiriman	66 ms
14.	*POST /shipment/shipment_detail_edit*	Mengubah data detail pengiriman	102 ms
15.	*GET /client/client_data*	Melihat daftar klien	120 ms
16.	*POST /client/client_entry*	Menambah data klien	130 ms
17.	*POST /client/client_delete*	Menghapus data klien	184 ms
		Rata-rata Waktu	176 ms

Seluruh layanan *web service* dapat berfungsi dengan baik, dengan proses autentikasi pengguna dapat diselesaikan dalam waktu rata-rata 71ms. Proses enkripsi adalah 4,1723 detik sementara proses dekripsi membutuhkan waktu rata-rata 4,1643 detik. Penggunaan algoritma Base64, membuat hasil enkrip dari algoritma kriptografi RC-4 dapat disimpan pada database dengan sempurna dalam waktu 4,1723 detik. Sebagaimana terlihat pada Tabel 3, seluruh metode GET maupun POST yang terdiri dari tujuh belas proses, memakan waktu proses rata-rata selama 176 ms. Selain pengujian waktu pemrosesan, dilakukan pengujian fungsi aplikasi yang dilakukan oleh lima orang pengguna di perusahaan, dengan hasil terlihat pada Tabel 4.

Tabel 4. Evaluasi pengujian *Black Box* untuk fungsi aplikasi

Aktifitas	Input	Output	Status
User melakukan *login*	Username, password	Jika berhasil maka akan masuk kedalam dashboard, JWT akan memberikan akses token sebagai autentikasi hak akses user Jika username atau password salah akan muncul peringatan bahwa username/password salah atau belum tersedia	Valid
Menambah jumlah *user*	Name, username, password, password2, level	Jika berhasil akan muncul pemberitahuan "Data Berhasil ditambahkan" Jika password pertama dan kedua berbeda akan muncul peringatan password berbeda	Valid
Melihat daftar pengguna	-	Muncul tampilan daftar pengguna	Valid
Menghapus pengguna	id	Jika id benar, maka muncul pemberitahuan "Data berhasil dihapus" Jika id belum terdaftar, muncul pemberitahuan "id belum terdaftar"	Valid
Menambah data barang	name, cat, unit, weight,	Jika kategori tersedia maka muncul pemberitahuan "Data berhasil ditambah" Jika kategori tidak tersedia akan muncul peringatan "Kategori tidak tersedia"	Valid
Melihat data barang	-	Muncul tampilan daftar barang	Valid
Menghapus data barang	Item code	Jika kode benar, maka muncul pemberitahuan "Data berhasil dihapus" Jika kode belum terdaftar, muncul pemberitahuan "Kode belum terdaftar"	Valid

Aktifitas	Input	Output	Status
Menambah data kategori	category	Muncul pemberitahuan "Data berhasil ditambahkan"	Valid
Melihat data kategori	-	Muncul tampilan daftar kategori yang tersedia	Valid
Menghapus data kategori	Category code	Jika kode benar, maka muncul pemberitahuan "Data berhasil dihapus" Jika kode belum terdaftar, muncul pemberitahuan "Kode belum terdaftar"	Valid
Menambah data klien	Name, address, phone	Muncul pemberitahuan "Data berhasil ditambahkan"	Valid
Melihat data klien	-	Muncul daftar klien	Valid
Menghapus data klien	id	Jika id benar, maka muncul pemberitahuan "Data berhasil dihapus" Jika id belum terdaftar, muncul pemberitahuan "id belum terdaftar"	Valid
Menambah transaksi pengiriman	sender, receiver, shipment_detail	Jika id pengirim dan penerima benar, maka muncul pemberitahuan "Transaksi berhasil dibuat" Jika id pengirim atau penerima tidak terdaftar, maka muncul peringatan bahwa pengirim atau penerima belum terdaftar	Valid
Melihat daftar transaksi pengiriman	-	Muncul tampilan daftar transaksi pengiriman	Valid
Menghapus transaksi pengiriman	Shipment_code	Jika kode benar, maka muncul pemberitahuan "Transaksi berhasil dihapus" Jika kode belum terdaftar, muncul pemberitahuan "Kode transaksi belum terdaftar"	Valid
Mengubah data transaksi	Shipment_code, sender, receiver	Jika kode, id pengirim, dan penerima benar, maka muncul tampilan data transaksi Jika kode, id pengirim, dan penerima belum terdaftar, muncul pemberitahuan "Transaksi belum terdaftar"	Valid
Mengubah data detil pengiriman	Shipment_code, item_id, qty	Jika kode benar, maka muncul pemberitahuan "Data berhasil diubah" Jika kode belum terdaftar, muncul pemberitahuan "Kode transaksi belum terdaftar"	Valid

Pada Tabel 4, terlihat dengan jelas bahwa hasil evaluasi menunjukkan bahwa aplikasi yang dikembangkan memiliki fungsi yang berjalan dengan baik dan lancar.

4. Kesimpulan

Berdasarkan pembuatan aplikasi, pengujian metode, dan analisa program dari aplikasi ini, maka dapat diambil kesimpulan, bahwa seluruh layanan *web service* yang diimplementasikan pada aplikasi android dapat berjalan dengan baik. Implementasi RESTful web service pada aplikasi ini telah terbukti membuat perusahaan lebih efisien dalam proses penginputan transaksi, diukur dari sisi waktu pemrosesan yang rata-rata hanya 176 ms. Sebelum adanya aplikasi ini, proses transaksi yang berjalan memakan waktu sekitar 20 menit untuk satu proses layanan.

Penerapan JSON Web Token pada proses autentikasi user berfungsi dengan baik dalam membatasi hak akses user pada aplikasi. Metode enkripsi menggunakan algoritma RC4 dikombinasikan dengan algoritma Base64 berjalan dengan baik, sehingga sisi keamanan pada basis data dapat terjaga dengan aman menghindari pencurian dan penyalahgunaan data informasi.

Namun demikian, agar penelitian ini dapat dikembangkan di masa mendatang, perlu mengatasi batasan penelitian saat ini yang belum memiliki fitur tracking maps berbasis *location based*, untuk melihat posisi terkini pengiriman. Selain itu, dimungkinkan untuk melakukan pengembangan aplikasi pada sistem operasi non-android. Selain itu, dimungkinkan untuk melakukan pengembangan aplikasi menggunakan metode *mobile hybrid application*. Metode ini dapat digunakan pengembang aplikasi untuk menjembatani setiap *platform smartphone (Android dan IOS)*. Pembuatan dapat dilakukan dengan framework cordova, *framework* ini dapat membangun aplikasi dengan menggunakan *Javascript, HTML5, dan CSS*.

Daftar Rujukan

[1] F. Christian, "2020, Potensi Pertumbuhan Bisnis Logistik Lebih dari 30%," *Sindo News*, no. November 2019, pp. 1–4, Dec. 17, 2020.

[2] M. G. L. Putra and M. I. A. Putera, "Analisis Perbandingan Metode Soap Dan Rest Yang Digunakan Pada Framework Flask Untuk Membangun Web Service," *SCAN - J. Teknol. Inf. dan Komun.*, vol. 14, no. 2, pp. 1–7, 2019, doi: 10.33005/scan.v14i2.1480.

[3] K. Arianto, Mukhammad Agus; Munir, Sirojul; Khotimah, "Analisis dan Perancangan Representational State Transfer (REST) Web Service Sistem Informasi Akademik STT Terpadu Nurul Fikri Menggunakan YII Framework," *J. Teknol. Terpadu*, vol. 2, no. 2, pp. 1–8, 2016.

[4] A. Rahmatulloh, H. Sulastri, and R. Nugroho, "Keamanan RESTful Web Service Menggunakan JSON Web Token (JWT) HMAC SHA-512," *J. Nas. Tek. Elektro dan Teknol. Inf.*, vol. 7, no. 2, 2018, doi: 10.22146/jnteti.v7i2.417.

[5] G. A. Sekarsari, B. Nurhadiyono, and Y. Rahayu, "Analisis Algoritma Kriptografi Rc4 Pada Enkripsi," *Techno.COM*, vol. 14, no. 4, pp. 250–254, 2015.

[6] T. Hartanto, T. Informatika, F. T. Informasi, U. B. Luhur, P. Utara, and K. Lama, "Implementasi Web Service Berbasis Rest Menggunakan Algoritma AES 128 dan Affine Cipher," vol. 1, no. 3, pp. 1130–1136, 2018.

[7] R. Kurniawati, "Pengembangan Sistem Informasi Kependudukan Berbasis Mobile Dan Restful Web Service," *Semin. Nas. Teknol. Inf. dan Komun.*, vol. 2016, no. SENTIKA, pp. 605–609, 2016.

[8] E. Kurniawan, "Implementasi Rest Web Service Untuk Sales Order Dan Sales Tracking Berbasis Mobile," *J. EKSIS*, vol. 07, pp. 1–12, 2014.

[9] B. Satria, A. Kusyanti, and W. Yahya, "Implementasi Algoritme Blake2s pada JSON Web Token (JWT) sebagai Algoritme Hashing untuk Mekanisme Autentikasi Layanan REST-API," *J. Pengemb. Teknol. Inf. dan Ilmu Komput. Univ. Brawijaya*, vol. 2, no. 12, pp. 6269–6276, 2018.

[10] Edy, Ferdiansyah, W. Pramusinto, and S. Waluyo, "Pengamanan Restful API menggunakan JWT untuk Aplikasi Sales Order," vol. 1, no. 1, pp. 19–25, 2017.

[11] M. M. Amin, "Implementasi Kriptografi Klasik Pada Komunikasi Berbasis Teks," *Pseudocode*, vol. 3, no. 2, pp. 129–136, 2017, doi: 10.33369/pseudocode.3.2.129-136.

[12] N. R. Yanti, A. Alimah, and D. A. Ritonga, "Implementasi Algoritma Data Encryption Standard Pada Penyandian Record Database," *J-SAKTI (Jurnal Sains Komput. dan Inform.*, vol. 2, no. 1, p. 23, 2018, doi: 10.30645/j-sakti.v2i1.53.

[13] R. Sulaiman and B. Isnanto, "'Peningkatan Keamanan Pesan Dengan Kriptografi RC4 dan Steganografi LSB Pada File JPEG,'" *Konf. Nas. Sist. Inf. 2018*, pp. 8–9, 2018.

[14] R. Minarni, "Implementasi Algoritma Base64 untuk Mengamakan SMS p ada Smartphone," *Build. Informatics, Technol. Sci.*, vol. 1, no. 1, pp. 28–33, 2019.