RANCANG BANGUN SISTEM BILLING RESTORAN (EBILL RESTO) DENGAN MENERAPKAN SINKRONISASI DATA BILLING PADA CABANG PERUSAHAAN KE INDUK PERUSAHAAN BERBASIS REST API.

M. Mahaputra Hidayat[1], R Dimas Adityo[2], Irwan Kurnia Adianto[3]
[1,2,3] Jurusan Teknik Informatika, Fakultas Teknik,Universitas Bhayangkara Surabaya
Jl. A Yani 114 Surabaya
Tlp. 031-8541506
mahaputra@ubhara.ac.id, dimas@ubhara.ac.id,Irwan@ubhara.ac.id

ABSTRAK

Usaha kuliner merupakan sebuah peluang usaha yang paling banyak dimintai, *E Bill Resto* merupakan Sistem Billing restoran yang dikembangkan dengan melibatkan beberapa tempat penjualan / restoran dengan nama sebuah Merk / *Brands* yang terhubung di induk perusahaan oleh server database. Dengan sistem terintegrasi maka seluruh pendapatan dari penjualan restoran dapat terpantau secara realtime, Rancangan Sistem dibuat dengan menerapkan arsitektur RESTFul API dengan akses keamanan Token, merujuk pada Pemodelan DFD (Data Flow Diagram) dengan pemanfaatan Web Service untuk itegrasi dan Pengelolahan data pada masing- masing Sistem Informasi. Aplikasi Master sebagai penyedia resource Web Service Data Embeded pada 5 Sistem Informasi Restoran, dan Aplikasi Slave Management Database melakukan sinkronisansi berjalan satu arah secara Statis terhadap 5 cloud Aplikasi Master Web Service, dimana kedua Aplikasi tersebut diuji Qos (Quality of Service) dengan 5 sampel data baru, dari Internet Provider INDOSAT yang menunjukkan rata-rata hasil pengujian nilai Throughput sebesar 170,3 bps, Packet Loss sebesar 144,4 %, Delay (latency) sebesar 78,4 ms, sedangkan menggunakan Internet Provider TELKOM dengan rata-rata hasil pengujian nilai Throughput sebesar 259,5 bps, Packet Loss sebesar 42,8 % dan Delay (latency) sebesar 83,8 ms. Maka dapat disimpulkan bahwa pengujian berdasar TIPHON Throughput dan Delay (latency) dari Internet Provider INDOSAT signal 4g dan TELKOM menunjukkan kategori "Sangat Bagus" sedangkan pengujian Packet Loss diperoleh kategori "Jelek".

.

Kata Kunci: *Billing Restoran, RESTFul API, Web Service, Qos*

I. PENDAHULUAN

Semakin bertambahnya penduduk di era globalisasi saat ini, maka potensi pertumbuhan data juga semakin bertambah. Dengan bertumbuhnya data-data tersebut, terbentuklah dunia bisnis, teknologi informasi pun dituntut untuk berkembang dan berinovasi agar dapat berjalan sesuai kemajuan dunia bisnis termasuk pertukaran data dan pengelolahan Informasi pada Client Server yang semakin kompleks. Permasalahan yang terjadi adalah, data pada Sistem Informasi di sebuah induk perusahaan dan Client untuk cabang usaha terutama untuk aplikasi restoran, Dalam penelitian Aplikasi di Billing Client masih disimpan hanya pada masing-masing lokal database, sehingga sulit untuk diolah dan dikembangkan kedalam Content Web Sistem Informasi yang terintegrasi, sehingga Pengelolahan Informasi, Pertukaran data hingga Pembuatan pengelolaan terhadap kedua database tersebut masih sangat terbatas. Selain terbatasnya Pengelolahan Informasi, keamanan terhadap penyimpanan data-data tersebut juga wajib dilakukan. Maka dari uraian tersebut, untuk mengatasi Pertukaran, Pengelolahan dan keamanan data pada masing-masing Content Web Sistem Informasi, maka di rancanglah sebuah penyimpanan database Web Service menggunakan arsitektur RESTful API. Sistem database yang dirancang dan diintegrasikan berupa data Sinkronansi Sistem Informasi Induk Perusahaan dan Client berupa aplikasi POS (Point Of Sales) dari 5 Sistem Informasi Bill Resto sebagai Master Database Web Service. Kedua. pertukaran data tersebut diamankan menggunakan akses keamanan JSON Web Token (JWT) Algoritme HS256 dengan Metode pengambilan data statis. Mekanisme pertukaran data yang terjadi adalah, dimana pada umunya arsitektur RESTful API terdapat dua aplikasi Server dan Client untuk melayani pertukaran data embedded database di 5 Sistem Informasi Client Sebagai Server dan 1 Sistem Informasi Sebagai System Server Sebuah Induk Perusahaan

II. : LANDASAN TEORI

2.1 Sinkronisasi Data

Sinkronisasi adalah suatu proses dimana proses tersebut saling bersamaan dan saling berbagi data bersama. Sinkronisasi menjadi penting karena bisa menghindari sesuatu yang tidak konsisten akibat data akses yang kurang akurat. Jadi dalam melakukan sinkronisasi ini dilakukan dalam waktu

yang bersamaan. Dalam proses sinkronisasi, data urutan aktivitas tertentu yang terjadi didasarkan pada urutan dan ketentuan yang berlaku sehingga integritas data akan tetap terjaga (Putra, Izzati, & Dewi, 2017). Menurut Naveen Malhotra dan Anjali, dengan konsep yang jelas namun berbeda dengan pendapat sebelumnya mengatakan bahwa sinkronisasi data merupakan kebutuhan untuk menyimpan beberapa salinan dari satu set data yang terkait satu sama lain, (Malhotra & Anjali, 2014) artinya sinkronisasi data merupakan proses pembentukan konsistensi antara data dari sumber penyimpanan kepada sebuah target penyimpanan yang lain maupun sebaliknya, sehingga tercipta harmonisasi data secara terus menerus dari waktu ke waktu. Seluruh database harus konsistensi antara satu database dengan database lainnya (Lin, Kemme, Patiño-Martínez, & Jiménez-Peris, 2005)

2.2 REST (REpresentational State Transfer)

REST (REpresentational State Transfer) merupakan standar arsitektur komunikasi berbasis web yang sering diterapkan dalam pengembangan layanan website. Umumnya menggunakan HTTP (Hypertext Transfer Protocol) sebagai protocol untuk komunikasi data. Pada arsitektur REST, REST client menyediakan resources (sumber daya/data) dan REST server mengakses dan menampilkan resource tersebut untuk penggunaan selanjutnya. Setiap resource diidentifikasi oleh URIs (Universal Resource Identifiers) atau global ID. Resource tersebut direpresentasikan dalam bentuk format teks, JSON atau XML. Metode HTTP yang umum digunakan dalam arsitektur berbasis REST yaitu GET, Menyediakan hanya akses baca pada resources, PUT, digunakan untuk menciptakan resource baru, DELETE, digunakan untuk menghapus resource, POST, digunakan untuk memperbarui resource yang ada atau membuat resource baru, OPTIONS, digunakan untuk mendapatkan operasi yang disupport pada resource, sedangkan Web service adalah standar yang digunakan untuk melakukan pertukaran data antar aplikasi sistem, karena aplikasi yang melakukan pertukaran data bisa ditulis dengan bahasa pemrograman yang berbeda atau berjalan pada platform yang berbeda. Web service yang berbasis arsitektur REST kemudian dikenal sebagai RESTful API Web Services. Layanan web ini menggunakan metode HTTP untuk menerapkan konsep arsitektur REST (Feridi, 2019)

2.3 JSON Web Token (JWT)

Merupakan tipe akses token yang mampu memuat sejumlah claim di dalamnya. Terverifikasi dan dapat dipercaya karena JWT ditanda tangani secara digital. Saat membuat aplikasi komputer, maka client perlu mengakses sumber daya (resource) dan sumber daya yang diambil aksesnya dilindungi. Client perlu izin untuk mengaksesnya. Jika diizinkan, maka akan diberikan alat pertukaran informasi izin (layaknya surat izin mengemudi), alat ini biasanya disebut akses token. Otentikasi Token digunakan ketika user login, user akan mendapatkan token jwt, token ini digunakan untuk mengakses layanan yang dilindungi dan pertukaran Informasi pada JWT memuat informasi dan dapat ditanda tangani secara digital untuk memberikan kepercayaan bahwa konten yang dimuat asli dari client/pengirim. JSON Web Token berisi tiga bagian, yang dipisahkan oleh titik (.), 3 bagian tersebut adalah :

a) Header

Komponen header dari JWT berisi informasi tentang bagaimana tanda tangan JWT harus dihitung. Header adalah objek JSON dalam format berikut:

{ "typ": "JWT", "alg": "HS256" }

Dalam JSON, nilai kunci "typ" menentukan bahwa objek adalah JWT, dan nilai dari tombol "alg" menentukan algoritma hashing yang digunakan untuk membuat komponen tanda tangan JWT. Dalam contoh algoritma HMAC-SHA256, algoritma hashing yang menggunakan kunci rahasia, untuk menghitung signature.

b) Payload

Komponen muatan dari JWT adalah data yang disimpan di dalam JWT (data ini juga disebut sebagai "klaim" dari JWT). Dalam contoh kami, server autentikasi membuat JWT dengan informasi pengguna yang disimpan di dalamnya, khususnya ID pengguna.

{"userId":"b08f86af-35da-48f2-8fab-cef3904660bd"}

Perlu diingat bahwa ukuran data akan mempengaruhi ukuran keseluruhan JWT, ini umumnya bukan masalah tetapi memiliki JWT terlalu besar dapat berdampak negatif terhadap kinerja dan menyebabkan latensi.

c) Signature

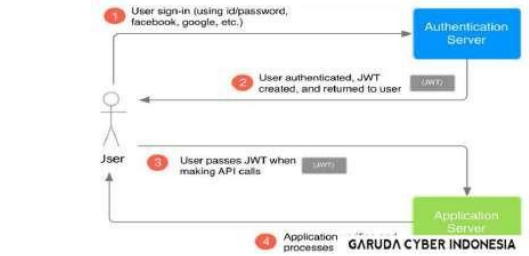
Signature di hitung menggunakan pseudo kode seperti berikut :

data = base64urlEncode (header) + "." + base64urlEncode (payload), hashedData = hash (data, rahasia), signature = base64urlEncode (hashedData)

Yang dilakukan oleh algoritma ini adalah mengkodekan base64url untuk header dan payload yang dibuat . Algoritma kemudian menggabungkan string yang dikodekan bersama dengan periode (.) Di antara keduanya. Dalam kode pseudo kami, string yang digabungkan ini ditugaskan ke data. String data di-hash dengan kunci rahasia menggunakan algoritma hashing yang ditentukan dalam header JWT. Data hash yang dihasilkan ditugaskan ke hashedData. Data hash ini kemudian dienkode base64url untuk menghasilkan tanda tangan JWT.

Maka header yang dienkode base64url, Payload, dan Signature dengan menggabungkan komponen dengan titik (.) Rumusnya seperti berikut :

Header.Payload.Signature



Gambar 2.1 Akses *JWT*

Dalam contoh 3 entitas sederhana, aplikasi menggunakan JWT yang dibuat dengan signature algoritma HS256 di mana hanya server autentikasi dan server aplikasi yang tahu kunci rahasia. Server aplikasi menerima kunci rahasia dari server autentikasi ketika aplikasi menyiapkan proses autentikasi. Karena aplikasi mengetahui kunci rahasia, ketika pengguna membuat panggilan API JWT terlampir ke aplikasi, aplikasi dapat melakukan algoritma tanda tangan yang sama pada JWT. Aplikasi kemudian dapat memverifikasi bahwa tanda tangan yang diperoleh dari operasi hashing miliknya sendiri cocok dengan tanda tangan pada JWT itu sendiri (yaitu cocok dengan tanda tangan JWT yang dibuat oleh server autentikasi). Jika tanda tangan cocok, maka itu berarti JWT valid yang menunjukkan bahwa panggilan API berasal dari sumber yang autentik.

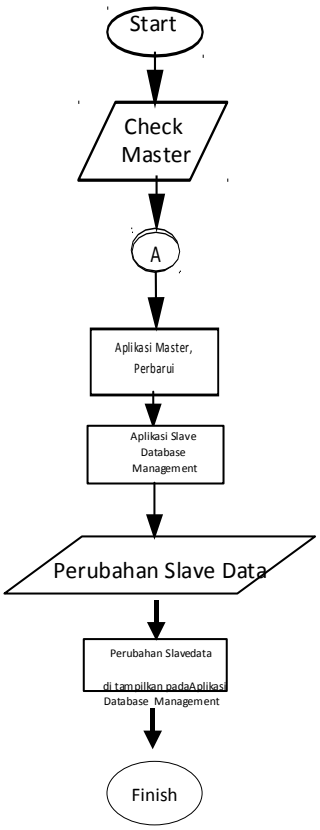
Autentikasi JSON Web Token (JWT) :
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIiOiIx
MjM0NTY3ODkwIiwibmFtZSI6IkpvaG4gRG9lIiwiYWRt aW4iOnRydWV9.TJVA95OrM7E2cBab30
RMHrHDcEfxjoYZgeFONFh7HgQ

jika tanda tangan tidak cocok, berarti JWT yang diterima tidak valid, yang mungkin merupakan indikator serangan potensial pada aplikasi. Jadi dengan memverifikasi JWT, aplikasi menambahkan lapisan kepercayaan antara dirinya dan pengguna (Warda, Putra, Bhawiyuga, & Data, 2018)

III. ANALISA & DESAIN SISTEM

3.1 Analisa Kebutuhan Sistem

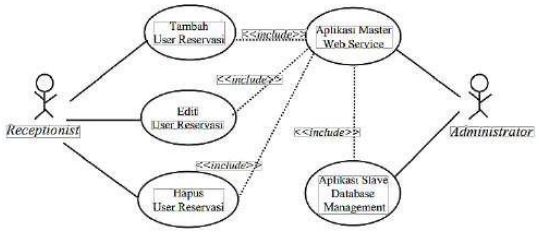
Data yang tersimpan saat ini masih belum tersinkronisasi antara database master terhadap Slave Database, sehingga database Master pada Web Sistem Informasi Kantor Pusat terhadap Slave Database Web Sistem Informasi Client E Bill Resto masih belum valid, maka untuk merancang sistem hingga hasil yang didapat, akan dijabarkan dalam bentuk Flowchart dan Usecase



Gambar 3.1 Flowchart

3.2 Usecase Diagram

Petugas E-Bill Resto adalah User yang akan menyambut Tamu datang dan mengisi data Transaksi Restoran, sedangkan Administrator adalah User yang dapat melihat data Web Service Transaksi Tamu dan Proses Sinkronansi pada Aplikasi Management Database, data Master maupun Slave.

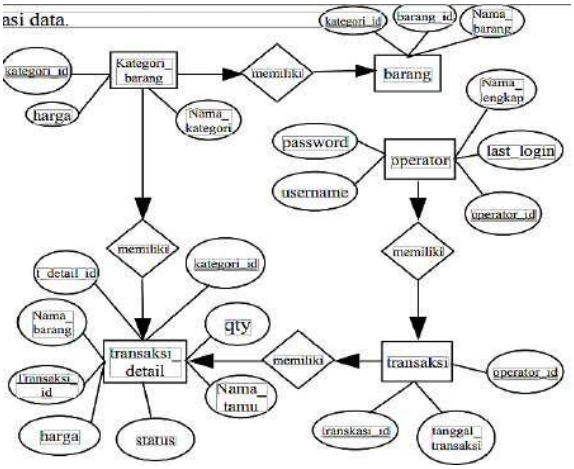


Gambar 3.2 Usecase Diagram

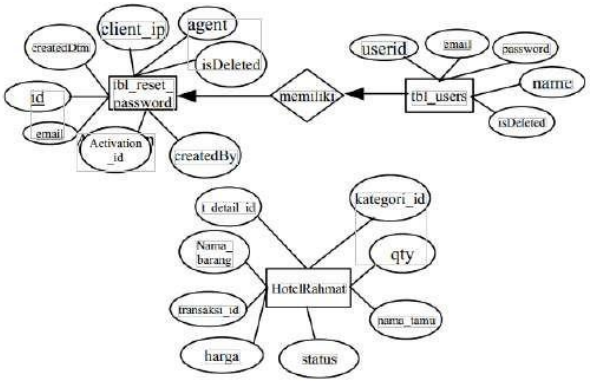
3.3 Rancang Desain Sistem

Perancangan dan Desain sistem yang akan dibuat mengacu pada jumlah data yang akan disinkronkan dan pemilihan bahasa pemrograman AngularJS sebagai base Runtime Engine sinkronisasi secara Asychronous untuk membangun Aplikasi User Interface dan menampilkan proses sinkronisasi.

Diagram ERD berikut menjelaskan hubungan data dengan database berdasar objek-objek dengan relasi. Sebagai contoh Data Transkasi Resto yang akan digunakan Pada Sistem Informasi Wajib Pajak Resto sebagai data Master Web Service dan Aplikasi Slave Management Database sebagai data Slave dengan atribut yang sama antar kedua aplikasi tabel sinkronisasi. Beberapa fitur pada aplikasi dibuat sedemikian menyerupai Fitur Asli pada Sistem Informasi E Bill Resto yang sudah terinstall dan hanya untuk menguji proses sinkronisasi data



Gambar 3.3 Diagram ERD Sistem Informasi E-Bill Resto



Gambar 3.4 Diagram ERD Aplikasi Slave Database Management

IV. IMPLEMENTASI SISTEM

4.1 Spesifikasi Perangkat Keras

Perangkat Keras yang digunakan dalam Implementasi aplikasi ini adalah sebagai berikut :

a) Asus A456U
b) Intel Core i5-6200U, CPU @ 2.30GHz × 4
c) Intel® HD Graphics 520 (Skylake GT2)
d) VGA NVIDIA GeForce 930M
e) Hardisk HDD 500GB
f) Ram ddr3L 8GB

4.2 Spesifikasi Perangkat Lunak

Perangkat lunak yang digunakan dalam menjalankan aplikasi ini adalah sebagai berikut :

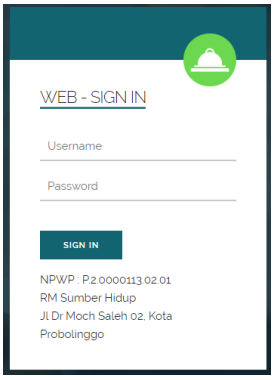
a) Sistem Operasi GNU Linux Ubuntu 64 Bit release 17.10 (Artful Aardvark)
b) Web Server LAMPP v5.6
c) Browser Google Chrome

4.3 Implementasi Aplikasi

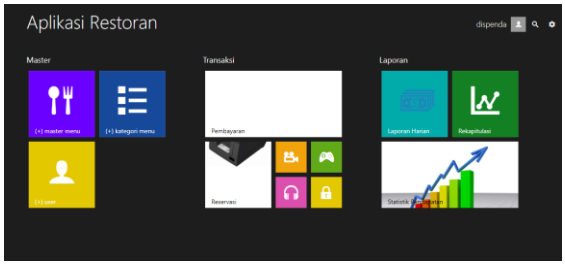
Tiga Aplikasi yang dibuat yaitu Aplikasi Master Sistem Informasi Restoran sebagai pengganti Aplikasi yang sudah ada, Aplikasi Master Web Service dan Apilkasi Slave Management Database. Maka Impementasinya dibuatlah GUI (Graphic User Interface) untuk memudahkan pengguna. Berikut Aplikasi beserta desain Database yang sudah dibuat.

4.4 Aplikasi Master Sistem Informasi Restoran

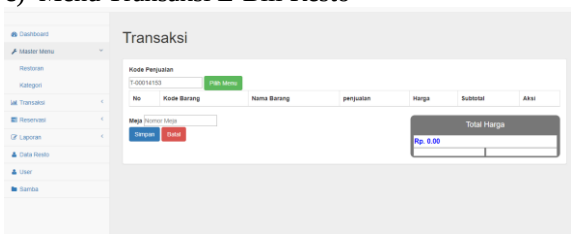
a) Form Login E-Bill Resto



b) Menu Aplikasi E-Bill Resto



c) Menu Transaksi E-Bill Resto



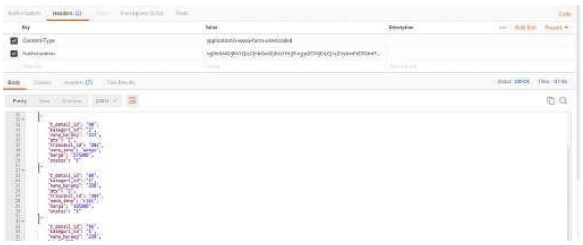
a.) Login Web Service

Pada 5 Aplikasi Master yang sama, masing-masing aplikasi memberikan source Web Service melalui akses token dengan Signature Algoritme HS256 dan Login User. Data Web Service diambil dari 5 tabel Database dari Aplikasi Sistem Informasi E-Bill Resto. Pengujian Web Service menggunakan Aplikasi Postman. Penggunaan sampel Aplikasi yang digunakan adalah Aplikasi Master E-Bill Resto Pertama. Berikut cara menggunakannya, Generate Token baru, pilih Method POST, arahkan url ke controller /JsonWebToken/generate. Contoh Login User yang digunakan E-Bill Resto pilih Header Body pilih application/x-www-form- urlencoded, isi Key Username value admin dan Key Password value

4.4 Login ke Web Service

b) Akses Web Service

Pilih Method GET, arahkan url ke controller /Api, pilih Headers isi Key Content-Type value application/x- www-form-urlencoded dan Key Authorization value isi token yang sudah di Generate seperti pada gambar 4.5 , jika berhasil, maka Aplikasi Master akan memberikan data Web Service seperti pada gambar



Gambar 4.5 Akses Web Service

V. HASIL DAN PEMBAHASAN

5.1 Pengujian Sistem

Rancangan Arsitektur pertukaran data berbasis RESTful API, pada Aplikasi Master Web Service dan Aplikasi Slave telah berjalan dengan baik. Agar Aplikasi menjadi user friendly, maka dilakukan uji aplikasi Analisa QoS (Quality of Service) untuk mengukur troughtput, delay (latency) dan Packet Loss dalam jaringan IP (Internet Protokol). Berdasar hasil uji QoS (Quality of Service) menurut TIPHON dengan menggunakan Aplikasi Wireshark untuk menghasilkan informasi berupa :

a) Waktu yang dibutuhkan oleh sebuah paket data terhitung dari saat pengiriman oleh transmitter sampai saat diterima oleh receiver (throughput) dengan persamaan sebagai berikut :

Troughtput = Jumlah Data yang Dikirim / Waktu Pengiriman Data

Tabel 5.1 Kategori Troughput

Kategori Troughtput	Troughtput	Indeks
Sangat Bagus	100	4
Bagus	75	3
Sedang	50	2
Jelek	< 25	1

b) Perbedaan selang waktu kedatangan antar paket diterminal tujuan (delay/latency) dengan persamaan sebagai berikut :

Delay = Waktu Paket Diterima – Waktu Paket Dikirim

Tabel 5.2 Kategori Delay / Latency

Kategori Latensi	Besar Delay (ms)	Indeks
Sangat Bagus	< 150 ms	4
Bagus	150 ms s/d 300 ms	3
Sedang	300 ms s/d 450 ms	2
Jelek	> 450 ms	1

c) Banyaknya paket yang hilang selama proses transmisi ke tujuan (packet loss) dengan persamaan sebagai berikut :

Packet Loss = (Paket Dikirim – Paket Diterima / Paket Dikirim) x 100%

Tabel 5.3 Kategory *Packet Loss*

Kategori Degredasi	Packet Loss (%)	Indeks
Sangat Bagus	0	4
Bagus	3	3
Sedang	15	2
Jelek	25	1

5.2 Indeks Nilai QoS (Quality of Service)

Rekapitulasi nilai *QoS (Quality of Service)* dengan versi TIPHON dapat disimpulkan bahwa dengan menggunakan 2 *ISP (Internet Service Provider)* yang telah diujikan terhadap 5 Server Aplikasi Master bahwa rata-rata kategori yang didapat "Bagus", dimana kualitas pada setiap Service Provider memiliki Batasan *bandwidth*, jumlah User, lokasi akses dan kondisi *traffic internet* di jam tertentu, sehingga faktor tersebut yang dapat mempengaruhi kualitas *QoS (Quality of Service).*

Tabel 5.4 Tabel Indeks Parameter QOS

No	Parameter QoS	INDOSAT	TELKOM	INDOSAT	TELKOM	INDOSAT	TELKOM
1	*Troughtput*	*183,5 bps*	*470,8 bps*	*52,1 bps*	*415,3 bps*	*169,75 bps*	*251,25 bps*
2	*Delay/ Latency*	*0,00334 s*	*0,0001 s*	*0,0003 s*	*0,0847 s*	*0,0001 s*	*0,155 s*
3	*Packet Loss*	*22,223 %*	*14,285 %*	*33,33%*	*14,28 %*	*33,33%*	*14,28 %*

VI PENUTUP

Berdasar Implementasi Sistem, hasil uji functional testing, error handling testing dan Pengujian terhadap *QoS (Quality of Service)* pada Bab sebelumnya, dapat disimpulkan bahwa Aplikasi cloud Master Web Service dan Aplikasi Slave Management Database sudah berjalan sesuai yang diharapkan. Hasil Pengujian *QoS (Quality of Service)* menggunakan Internet Provider INDOSAT yang menunjukkan rata-rata hasil nilai *Throughput* sebesar 170,3 bps, *Packet Loss* sebesar 144,4 %, *Delay (latency)* sebesar 78,4 ms, sedangkan menggunakan Internet Provider TELKOM dengan rata-rata hasil pengujian nilai *Throughput* sebesar 259,5 bps, *Packet Loss* sebesar 42,8 % dan *Delay (latency)* sebesar 83,8 ms. Maka dapat disimpulkan bahwa pengujian berdasar TIPHON *Throughput* dan *Delay (latency)* dari Internet Provider INDOSAT signal 4g dan TELKOM menunjukkan kategori "Sangat Bagus" sedangkan pengujian Packet Loss diperoleh kategori "Jelek". Dengan demikian menerapkan gaya arsitektur pertukaran data duplikasi Multi database berbasis RESTful API menggunakan JSON Web Token (JWT), pertukaran data dan informasi menjadi lebih mudah dilakukan, masalah pengelolahan dan pertukaran informasi data sudah dapat teratasi, sehingga data Transaksi pada Sistem Informasi E-Bill Resto tidak seutuhnya bergantung pada database lokal

REFERENSI

Putra, R. E., Izzati, B. M., & Dewi, F. (2017). Optimasi Kinerja Point Of Sale (POS) Dengan Penerapan Sinkronisasi Database Menggunakan Middleware. *Informatika Mulawarman : Jurnal Ilmiah Ilmu Komputer*, *12*(2), 123. https://doi.org/10.30872/jim.v12i2.654

Lin, Y., Kemme, B., Patiño-Martínez, M., & Jiménez-Peris, R. (2005). Middleware based data replication providing snapshot isolation. In *Proceedings of the 2005 ACM SIGMOD international conference on Management of data - SIGMOD '05* (hal. 419). New York, New York, USA: ACM Press. https://doi.org/10.1145/1066157.1066205

Malhotra, N., & Anjali, C. (2014). Implementation of Database Synchronization Technique between Client and Server. *International Journal of Engineering Science and Innovative Technology (IJESIT)*, *3*(4), 460–465. Diambil dari http://www.ijesit.com/Volume 3/Issue 4/IJESIT201404_63.pdf

Warda, A., Putra, P., Bhawiyuga, A., & Data, M. (2018). Implementasi Autentikasi JSON Web Token (JWT) Sebagai Mekanisme Autentikasi Protokol MQTT Pada Perangkat NodeMCU. *J-Ptiik*, *2*(2), 584–593.

Feridi. (2019). Mengenal RESTful Web Services - CodePolitan.com. Diambil 25 September 2019, dari https://www.codepolitan.com/mengenal-restful-web-services