Design of Restaurant Billing System (E Bill Resto) by Applying Synchronization of Data Billing in Branch Companies to Main Companies Based on Rest API

1stM Mahaputra Hidayat
Dept. of Engineering
University of Bhayangkara Surabaya
Surabaya, Indonesia
mahaputra@ubhara.ac.id

2ndR Dimas Adityo
Dept. of Engineering
University of Bhayangkara Surabaya
Surabaya, Indonesia
dimas@ubhara.ac.id

3rdAlek Siswanto
Dept. of Information System
University of Islamic Madura
Pamekasan, Indonesia
alexsiswanto@uim.ac.id

Abstract—**Culinary business is a business opportunity that is most in-demand, E Bill Resto is a restaurant billing system that was developed by involving several selling places/restaurants with the name of a brand that is connected to the parent company by a database server. With an integrated system, all revenue from restaurant sales can be monitored in realtime. The system design is made by implementing the RESTFul API architecture with security access tokens. The Master Application as a provider of Embedded Data Service Web resources on 3 Restaurant Information Systems, It does synchronization of 3 Web Service Clients, Data From the Master Slave Side was obtained by testing 3 data sampling, where both applications are tested QoS (Quality of Service) with 3 new data samples, from the INDOSAT Internet Provider which showed an average test result of Throughput of 170.3 bps, Packet Loss of 18.851% and Delay (latency) of 78.4 ms. Whereas using the TELKOM Internet Provider with an average throughput of 259.5 bps, Packet Loss of 14.28% and delay (latency) of 83.8 ms. Then it can be concluded that testing based on TIPHON Throughput and Delay (latency) from INDOSAT 4G and TELKOM signal Internet Providers showed the category of "Very Good" while the Packet Loss test obtained the "Bad" category.**

Keywords—restaurant billing, RESTFul API, web service, QoS

I. Introduction

With the increasing population in the current era of globalization, the potential for data growth is also increasing. With the growth of these data, the business world is formed, information technology is also demanded to develop and innovate so that it can run following the progress of the business world including data exchange and management of information on an increasingly complex Client-Server. The problem that occurs is the data on Information Systems in a main company and clients for business branches, especially for restaurant applications. In research applications in the client billing are still stored only in each local database, making it difficult to process and develop into the Web Content of Information System integrated, so that information management, data exchange until making management of two databases is still very limited. In addition to the limited management of information, the security of the data storage is also mandatory. So from the description, to overcome data exchange, management and security in each Web Content of Information System, then design a web service database storage using the RESTful API architecture. The design and integrated database system is Data Synchronization of the main company and client information system in the form of a POS (Point of Sales) application of 5 Bill Resto Information Systems as a Master Database Web Service. Second, data exchange is secured using the JSON Web Token (JWT) security access HS256 algorithm with a static data retrieval method. Data exchange mechanism that occurs is, where in general the RESTful API architecture there are two Server and Client applications to serve embedded database data exchange in 5 Client Information Systems as Servers and 1 Information System as Server Systems for a Main Company.

II. Theory

A. Data Synchronization

Synchronization is a process where the processes are concurrent and sharing data. Synchronization is important because it can avoid something inconsistent due to inaccurate data access. So in synchronizing, it must be done at the same time. In the synchronization process, the data sequence of certain activities that occur is based on the order and applicable provisions, so that data integrity is maintained [1]. According to Naveen Malhotra and Anjali, with a clear concept but different from the previous opinion saying that synchronizing data is the need to store multiple copies of a set of data related to each other, [2] means that data synchronization is a process of establishing consistency between data from a storage source to another storage target or vice versa so that data harmonization can be formed continuously over time. All databases must be consistent between one database and another [3].

B. REST (REpresentational State Transfer)

REST (REpresentational State Transfer) is a web-based communication architecture standard that is often applied in developing website services. Generally, use HTTP (Hypertext Transfer Protocol) is a protocol for data communication. In the REST architecture, the REST client provides resources (resources/data) and the REST server accesses and displays these resources for future use. Each resource is identified by URIs (Universal Resource Identifiers) or global IDs. These resources are represented in text format, JSON or XML. There are some HTTP Methods on REST Webservice such as GET, PUT, POST, DELETE, OPTIONS. GET Used for data access. PUT Used to create new resources, POST Used to update a resource, DELETE is used to delete a resource, OPTIONS Used to get operations that are supported on resouce. Whereas Web service is a standard used to exchange data between system applications because it can be written in different programming languages or run on different platforms. Web services based on the REST architecture became known as the RESTful API Web Services. This web service uses the HTTP method to implement the REST architecture concept [4].

C. JSON Web Token (JWT)

It is a type of token access that can contain several claims in it. Verified and trustworthy because JWT is digitally signed. When creating a computer application, the client needs to access resources (resources) and the resources will protect the taken access. The client needs permission to access it. If permitted, a permit information exchange tool (like a driver's license) will be given, this tool is usually called token access. Token authentication is used when a user logs in, it was applied to protect access of information data on JWT. Digital signatures provide confidence in the content that is loaded by the client / sender. JSON Web Token contains three parts, separated by a point (.), These 3 parts are:

a) Header

The header component of JWT contains information about how the JWT signature must be calculated. A header is a JSON object in the following format:

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

In JSON, the "typ" key-value determines that the object is JWT, and the value of the "alg" key determines the hashing algorithm used to create the JWT signature component. In the example, the HMAC-SHA256 algorithm is a hashing algorithm that uses a secret key to calculate signatures.

b) Payload

The charge component of JWT is the data stored in JWT (this data is also referred to as "claim" from JWT). In our example, the authentication server creates JWT with user information stored on it, specifically the user ID.

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

Keep in mind that the size of the data will affect the overall size of the JWT, this is generally not a problem but having a JWT that is too large can negatively impact performance and cause latency.

c) Signature

Signature is calculated using pseudo code as follows:

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

What this algorithm does is it encodes base64url for the headers and payloads that have been created. The algorithm then combines the encoded string together with the point (.) between them. In our pseudo code, this concatenated string is assigned to data. The data string is hashed with a secret key using the hashing algorithm specified in the JWT header. The resulting hash data is assigned to hashedData. This hash data is then encoded base64url to generate JWT signatures.

Then the header encoded base64url, Payload, and Signature by combining components with periods (.) The formula is as follows:

Header.Payload.Signature

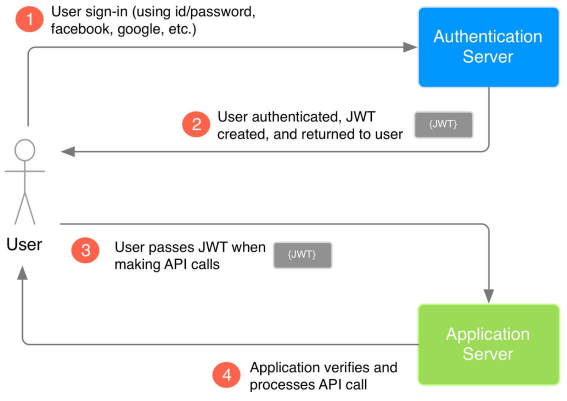

1
User sign-in (using id/password, facebook, google, etc.)
Authentication Server
2
User authenticated, JWT created, and returned to user
{JWT}
User
3
User passes JWT when making API calls
{JWT}
Application Server
4
Application verifies and processes API call

Fig. 1. JWT Access

For example 3 the simple entity, the application uses JWT which is made with the signature HS256 algorithm where only the authentication server and application server know the secret key. The application server receives a secret key from the authentication server when the application prepares the authentication process. Because the application knows the secret key, when the user makes a JWT API call attached to the application, the application can perform the same signature algorithm on JWT. The application can then verify that the signature obtained from its hashing operation matches the signature on the JWT itself (ie matches the JWT signature created by the authentication server). If the signature matches, then that means a valid JWT that indicates that the API call came from an authentic source.

JSON Web Token (JWT) Authentication:

eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIiOiIx
MjM0NTY3ODkwIiwibmFtZSI6IkpvaG4gRG9lIiwiYWRt
aW4iOnRydWV9.TJVA95OrM7E2cBab30
RMHrHDcEfxjoYZgeFONFh7HgQ

if the signature does not match, then the JWT received is invalid, which might be an indicator of potential attacks on the application. So by verifying JWT, the application adds a layer of trust between himself and the user [5].

III. Design System & Analysis

A. System Requirements Analysis

Data stored at this time is still not synchronized between the master database against the Slave Database, so the Master database on the Head Office web information system to the Slave Database Client web information system E Bill Resto is still invalid. So to design the system until the results obtained will be described in the form of flowchart and use case.

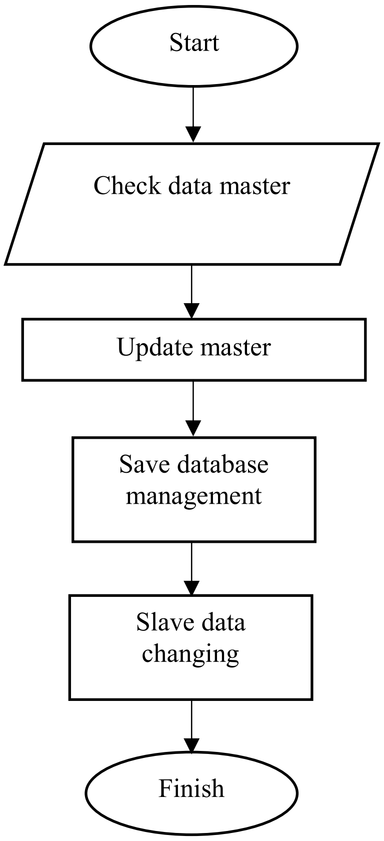


Fig. 2. Flowchart of Main System

B. Use case Diagram

The E-Bill Resto Officer is the User who will welcome Guest to come and fill in the Restaurant Transaction data, while the Administrator is the User who can view the Guest Transaction Web Service Data and the Synchronization Process in the Database Management, Master or Slave Data Applications.

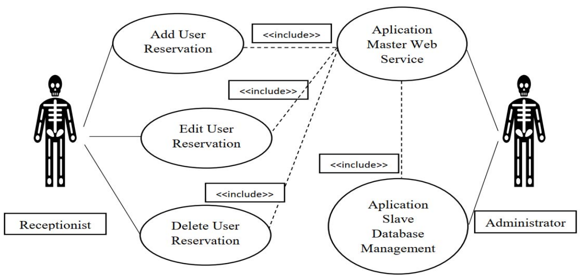


Fig. 3. Use case Diagram of Main System

C. System Design

The design of the system to be made refers to the amount of data to be synchronized and the selection of the AngularJS programming language as the base of the Synchronous Runtime Engine synchronization to build the User Interface Application and display the synchronization process.

The following ERD diagram explains the relationship of data with databases based on objects with relations. For example, Resto Transaction Data that will be used in the Resto Taxpayer Information System as a Web Service Master Data and Slave Management Database Application as Slave data with the same attributes between the two synchronization table applications. Some features of the application are made in such a way that they resemble the Original Features of the E Bill Resto Information System which is already installed and only for testing the data synchronization process.

IV. System Implementation

A. Hardware Specifications

Hardware used in this application implementation is as follows:

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

B. Software Specifications

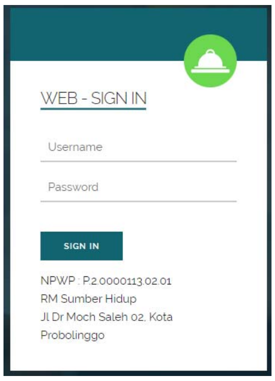
The software used in running this application is as follows:

a) GNU Linux Operating System Ubuntu 64 Bit release 17.10 (Artful Aardvark)
b) LAMPP v5.6 Web Server
c) Google Chrome browser

C. Application Implementation

Three applications are made, the Restaurant Information System Master Application which replaces the existing application, the Web Service Master Application and the Slave Management Database Application. Then the implementation was made GUI (Graphic User Interface) to facilitate users. The following application and database design that has been made.

D. Restaurant Information System Master Application



NPWP : P.2.0000113.02.01
RM Sumber Hidup
Jl.Dr Moch Saleh 02, Kota Probolinggo

Fig. 4. E-Bill Resto Login Form

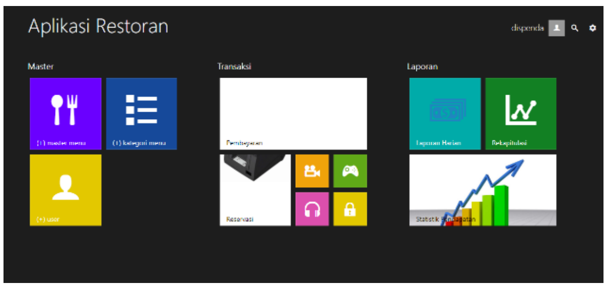


Fig. 5. E-Bill Resto Application Menu

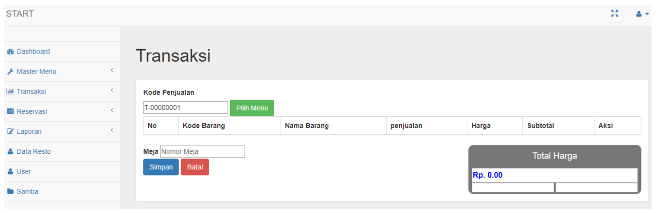
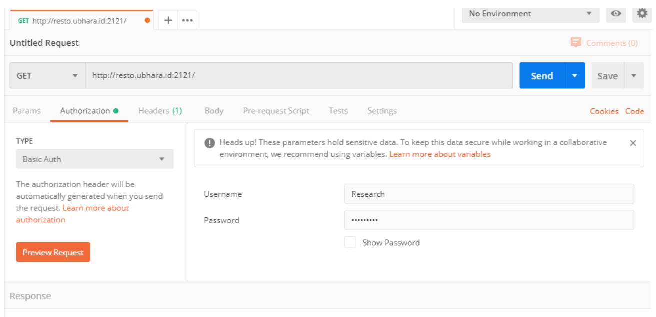


Fig. 6. E-Bill Resto Transaction Menu

In the same 3 Master Applications, each application provides a source of Web Service via token access with the HS256 Signature Algorithm and User Login. Web Service data is taken from 3 Database tables of E-Bill Resto Information System Applications. Web Service Testing using the Postman application. Sample usage The application used is the First Master E-Bill Resto Application. Here's how to use it. Generate a new Token, select the POST Method, point the URL to the controller/JsonWebToken/generate. Example of User Login that is used by E-Bill Resto, select Header Body, select application/x-www-form-urlencoded, fill in the admin username and username password value.



GET http://resto.ubhara.id:2121/

TYPE: Basic Auth

The authorization header will be automatically generated when you send the request. Learn more about authorization

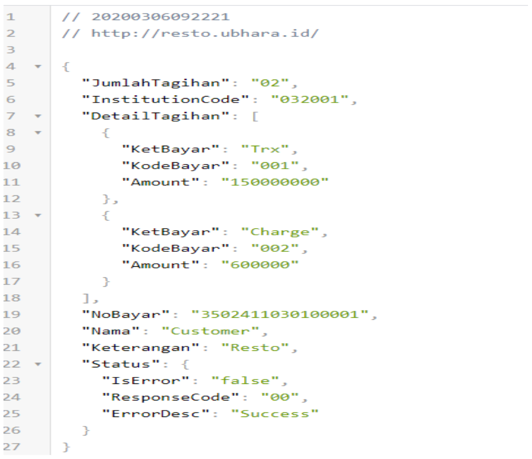
Heads up! These parameters hold sensitive data. To keep this data secure while working in a collaborative environment, we recommend using variables. Learn more about variables

Username: Research

Password: ·········

Fig. 7. Web Service Login

Select the GET Method, point the URL to the controller/Api. Select Headers contents Key Content-Type application value/x-www-form-urlencoded and Key Authorization value of the Generated Token as shown in Figure 7. If successful, then the Master Application will provide Web Service data as shown in Figure 8.



```
// 20200306092221
// http://resto.ubhara.id/

{
  "JumlahTagihan": "02",
  "InstitutionCode": "032001",
  "DetailTagihan": [
    {
      "KetBayar": "Trx",
      "KodeBayar": "001",
      "Amount": "150000000"
    },
    {
      "KetBayar": "Charge",
      "KodeBayar": "002",
      "Amount": "600000"
    }
  ],
  "NoBayar": "3502411030100001",
  "Nama": "Customer",
  "Keterangan": "Resto",
  "Status": {
    "IsError": "false",
    "ResponseCode": "00",
    "ErrorDesc": "Success"
  }
}
```

Fig. 8. Web Service Access

V. Results and Discussion

A. System Testing

The architecture of data exchange architecture based on RESTful API, the Web Service Master Application and Slave Application have been running well. For the application to be user-friendly, a QoS (Quality of Service) Analysis application test is performed to measure throughput, delay (latency) and Packet Loss in an IP (Internet Protocol) network. Based on QoS (Quality of Service) test results according to TIPHON by using the Wireshark Application to produce information in the form of:

a) The time required by a data packet is calculated from the time of transmission by the transmitter to when it is received by the receiver (throughput) with the following equation:

Throughput = Amount of Data Sent / Data Sending Time (1)

Table I. Throughput Category

Throughput Category	Throughput	Index
Very Good	100	4
Good	75	3
Medium	50	2
Bad	< 25	1

b) The difference in arrival intervals between packages at the destination terminal (delay/latency) with the following equation:

Delay = When the Package Is Received - When The Package Is Sent (2)

Table II. Latency Category

Latency Category	Delay (ms)	Index
Very Good	<150	4
Good	150 – 300	3
Medium	300 – 450	2
Bad	>=450	1

c) The number of packets lost during the process of transmission to the destination (packet loss) with the following equation:

Packet Loss = (Package Sent - Package Received / Package Shipped) x 100% (3)

Table III. Packet Loss Category

Degradation Category	Packet Loss (%)	Index
Very Good	0	4
Good	3	3
Medium	15	2
Bad	25	1

B. Value Index of QoS (Quality of Service)

The recapitulation of QoS (Quality of Service) value with the TIPHON version can be concluded that by using 2 ISPs (Internet Service Providers) that have been tested on 3 Master Application Servers that the average category obtained is "Good", where the quality of each Service Provider has Limitation bandwidth, number of users, access location and internet traffic conditions at certain hours, so these factors can affect the quality of QoS (Quality of Service).

Table IV. QoS Parameter Index

Parameter QoS	**Indosat**	**Telkom**	**Indosat**	**Telkom**	**Indosat**	**Telkom**
Throughput *(bps)*	183,5	470,8	52,1	415,3	169,75	251,25
Delay/ Latency *(ms)*	77,5	125	100	100	57,7	26,4
Packet Loss *(%)*	22,223	14,285	33,33	14,28	33,33	14,28

VI. CONCLUSION

Based on the System Implementation, functional testing, error handling testing and QoS (Quality of Service) test results in the previous chapter, it can be concluded that the cloud master web service and slave management database applications are running as expected. QoS (Quality of Service) Test results using INDOSAT Internet Providers which show an average Throughput value of 170.3 bps, Packet Loss of 18.851%, Delay (latency) of 78.4 ms, while using the TELKOM Internet Provider with average test results throughput of 259.5 bps, Packet Loss of 14.28% and Delay (latency) of 83.8 ms. Then it can be concluded that the test based on TIPHON Throughput and Delay (latency) from INDOSAT 4g and TELKOM Internet signal providers shows the category of "Very Good" while the Packet Loss test is obtained by the "Bad" category. Thus implementing a multi-database duplication data exchange architecture style based on RESTful API using JSON Web Token (JWT), data and information exchange becomes easier, the management and exchange of data information problems can be resolved so that Transaction data in the E-Bill Resto Information System not dependent on local databases.

REFERENCES

[1] Putra, R. E., Izzati, B. M., and F. Dewi, "Point Of Sale (POS) Performance Optimization With Database Synchronization Implementation Using Middleware". Informatika Mulawarman : Jurnal Ilmiah Ilmu Komputer, 12(2), 123, 2017.

[2] Malhotra, N., & Anjali, C., "Implementation of Database Synchronization Technique between Client and Server". International Journal of Engineering Science and Innovative Technology (IJESIT), 3(4), 460–465, 2014.

[3] Lin, Y., Kemme, B., Patiño-Martínez, M., & Jiménez-Peris, R., "Middleware based Data Replication Providing Snapshot Isolation". In Proceedings of the 2005 ACM SIGMOD International Conference on Management of Data - SIGMOD '05 (419), 2005.

[4] Feridi, "Get to know RESTful Web Services" - CodePolitan.com. www.codepolitan.com/mengenal-restful-web-services, 2019.

[5] Warda, A., Putra, P., Bhawiyuga, A., and Data, M., "Implementation of JSON Web Token (JWT) Authentication as MQTT Protocol Authentication Mechanism in NodeMCU Devices", J-Ptiik, 2(2), 584–593, 2018.