Backend Server System Design Based on REST API for Cashless Payment System on Retail Community

1st Bob Maulana Adam
Department of Informatics and Computer Engineering
Politeknik Elektronika Negeri Surabaya
Surabaya, Indonesia
bobmaulanaadam8@gmail.com

2nd Adnan Rachmat Anom Besari
Department of Informatics and Computer Engineering
Politeknik Elektronika Negeri Surabaya
Surabaya, Indonesia
anom@pens.ac.id

3rd Mochamad Mobed Bachtiar
Department of Informatics and Computer Engineering
Politeknik Elektronika Negeri Surabaya
Surabaya, Indonesia
mobed@pens.ac.id

Abstract*— Backend is logical space with functional and operations from software applications or information system. One of its implementations is cashless system. The increasing of cashless and electronic payment, and then retail community which not implement it, they must have electronic data capture especially to fullify cashless technology. Through this study we build a system cashless technology as backend server based on REST API. this system can handle some backend process such as top up, and then all of its feature will served in one system which can be accessed from any software platform. In this study, already build data protocol which can be accessed by client application (front-end), backend service with login system, register, withdraw, top-up balance with virtual account, top-up from administrator, top-up with bank account, transactions between users, balance information, transaction log, logout and database configuration itself. The backend system is tested for its robustness with 100 API requests carried out in 1 second. The success rate for the entire system is 76.92% of the 13 features offered. Transaction features have a 45% success rate for the process of reducing buyer balance, a 65% success rate for the process of adding seller balance. The top up balance feature reaches 72% and for the withdraw feature has a success rate of 77%.

Keywords*— Backend Server, cashless, REST API.

I. INTRODUCTION

The development of information technology in the current era has experienced a lot of progress. The development of web technology tends to be divided into 3 main concentrations, one of which is the backend. Back-end refers to programs and scripts that work on servers behind the scenes. So, back-end can be interpreted as a container of the core functional logic and operation of software applications or information systems. The backend system ensures that the data or services requested and sent by the front-end system or application are delivered through programmed methods. Back-end consists of core application logic, database, data integration and application, API and other back-end processes. The implementation of back-end technology which, is often used is in cashless payments.

Cashless payment is a payment system without money (paper or metal). The special tools are needed to make a cashless payment, one of them is Electronic Data Capture (EDC). EDC is an electronic machine used on debit or credit cards to conduct cashless payment. EDC has terms and conditions that are divided into several groups, including individuals, business entities and foundations [1]. Where some retail communities cannot always meet these requirements. Therefore, a system is needed to help consumers and the retail community in conducting cashless payment.

With these problems, it is filled in the concept of inter-communal cashless payment in a small-scale place-based community such as schools or institutions, from the concept also explains the importance of local communities and institutions to need data where the data is transaction data and consumer behavior. Because the cashless payment system generally has, a centralized system and local communities or institutions do not necessarily know the data from consumers and transaction behavior. Furthermore, with this concept, the community can apply money transactions and turnover to only one community and turn on transactions between citizens from one community, such as schools or institutions.

In this study, a cashless payment system will be created in the form of a REST API-based backend server. In the non-cash transaction system that created, it can handle the process in the form of payments, top up balances and transaction logs, and then all these features will be packaged in a single system that can be accessed by various soft platform platforms. The data communication method used is HTTP Request using the REST API, where this method is widely used for application development because it can be used by many programming languages and many platforms.

This paper discussed the server and API for the cashless payment system. Section 1 discussing about introduction of cashless payment. Section 2 discussing the related works. Section 3 discussing about system design. Section 4 discusses the experimental result, and the last section is discussing about the conclusion.

II. RELATED WORKS

A. Cashless Payment Design

According to research conducted by T. Ma, H. Zhang (2015), entitled "The Design and Implementation of an Innovative Mobile Payment System Based on QR Bar Code [2]. In this research, infrastructure was describe as one way to implement the non-cash payment system.

The system in question is an interface with an online client application, a wireless network with servers, a server that supports cellular payment processes based on QR codes, then a database, storage, maintenance, and account handling. The scheme of the payment system by involving buyers, sellers, and payment system servers.

978-1-7281-4449-8/19/$31.00 ©2019 IEEE

B. *Server Based on REST API*

Research conducted by X. H. Huang with the title "A Token-Based User Authentication Mechanism for Data Exchange in RESTful API" [3]. This study discussed the use of RESTful API. REST is not a standard but a software architecture design pattern. REST is a practical approach to web application development where systems in development need to be improved or need simple ways to interact with independent systems. REST is stateless and data-oriented, everything in the REST architecture is data. Each request is independent, the server does not store any request status. An Application Programming Interface (API) that follows the REST Style is called a RESTful API. RESTful API uses a Uniform Resource Identifier (URI) to represent data. For operations on data, the GET method is used to obtain data, The POST method is used to create new data, The PUT method is used to update data with the resource id, and the DELETE method is used to delete data or data sets [4].

III. SYSTEM DESIGN

System design is the stage after analysis of the system development cycle. Defining the needs of functional requirements and preparation for design and implementation describes how a system is formed. The aim is to meet the needs of system users and provide a clear picture and complete design to developers and readers. The process for server and API for the cashless payment system shown in Fig 1.

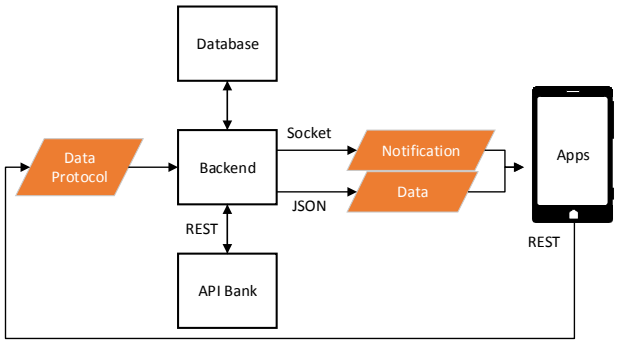

Fig 1. System Design

In this system design, it show that the backend server that created is a server to process the data received from the client application and send the results to the client application. There is a logical process from the non-cash payment system. The system accepts input from the front-end section or can be called an application. Then after input is processed and processed on the back end such as a list of accounts, account logins, payment transactions, API Bank access to request Top Up transfer codes, replies from API Bank in the form of transfer code and so on connection with database to store all logs and data in the payment system.

A. *Data Protocol*

The data protocol is a rule that defines functions that exist on the server and must be fulfilled by the client application to be able to use the function. The protocol is to design a Uniform Resource Identifier (URI) from the API along with the REST method in accessing the API [5]. When the server is active, it will run on the IP provided by the server system. The server used is a Virtual Private Server (VPS) and in this design is given a domain ezpy.advlop.com, which will facilitate the form of the URI and then the domain will connect to the server IP. The examples from the data protocol shown in Table 1.

TABLE I. PROTOCOL DATA

Login	
Method :	POST
URI :	http://ezpy.advlop.com/api/v1/user/loginApi

Body (JSON Object) :	{
	"email": "<email>",
	"password": "<pass>"
	}

Response (JSON)	{
	"success": "Berhasil Login",
	"token": "<jwt_token>",
	"Nama": "<nama>",
	"Role": "<angka>"
	}

B. *Database*

Designing data storage is an important part of designing a server system [6]. To design a server on an application, a list needed as shown in the following Fig 2.

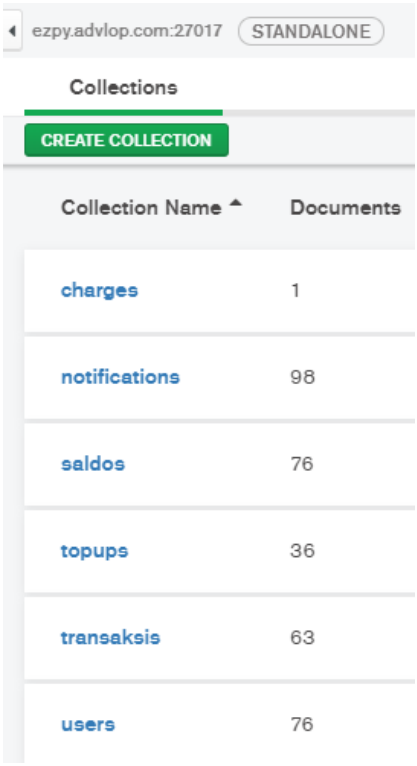

Fig 2. Collections Database Server on MongoDB

From the list of collections database servers, then each document model of the collections is as follows:

1. Charge
   Charge is a collection for storing documents from bank transfer billing responses with a virtual account for each user account.
2. Notification
   Notification are collections for storing notification documents of successful transactions and top up balances of each user account
3. Balance
   Balances are collections for storing balance documents from each user account.

4. Top up

   Top up are collections for storing top up virtual account notification documents from payment service providers, top up balances with bank transfer manual validation and top up balances in the admin for each user account.
5. Transactions

   Transactions are collections for storing transaction logs from each user account.
6. User

   Users are collections for storing user account data from each user.

C. *Backend Design*

When the data protocol is accessed or requested by the client application, the API (data protocol) that is accessed will be routing, in this section there are two path options, namely the API path and the Display path (view), this API path is used to access the features of the server , then for the display path used to access the display in the form of a web page from the server for the admin section. There is an outline of the workflow to explain the flow of input is routing, and produce an output that matches the input shown in Fig 3.

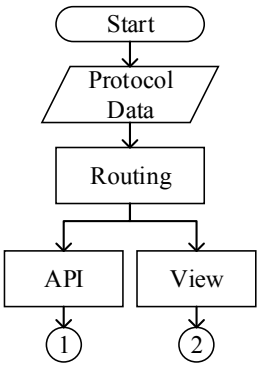

Fig 3. Routing path

After passing through the routing path, then next is routing to the features or API that is own, routing for each API shown in Fig 4.

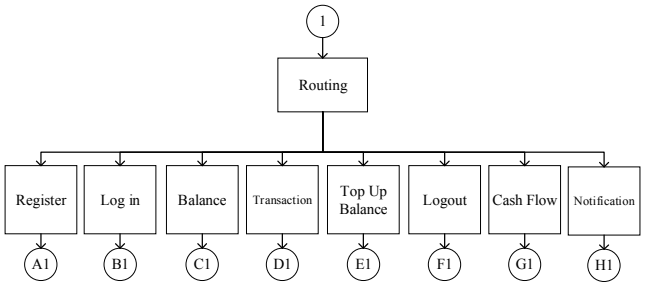

Fig 4. Routing API

After routing, the next process is enter the controller that is needed. In the process according to the API call (data protocol) requested, after the process is complete, there is output data according to the requested API in the form of JSON. However, several systems are carried out first to check tokens carried by clients when accessing data protocols. The token is obtained when the API login is successfully accessed. In figure 5, is the verification section of the token, the system will check the token received and verified whether it is appropriate or not. The token is checked whether it matches the key of the system and whether the token is still active because when making tokens the active period of the token is given, after that it can use the requested system API. The token used here is from JSON Web Token (JWT) [7]. Figure 5 is the token check code line. There are other parts of the backend server system is create a new account, login, balance, transaction, withdraw, top up admin, top up virtual account, transfer bank account, log top up, log notification, cash flow, logout.

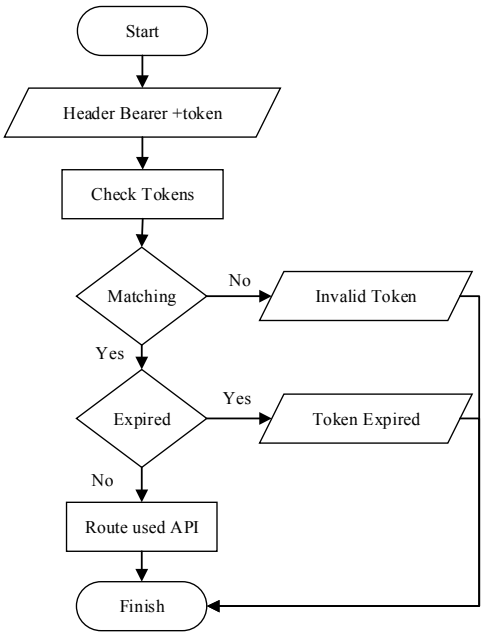

Fig 5. Token Checking

D. *Connect to API Bank*

Fig 6 is the flow of top up requests through the API Bank at the back end:

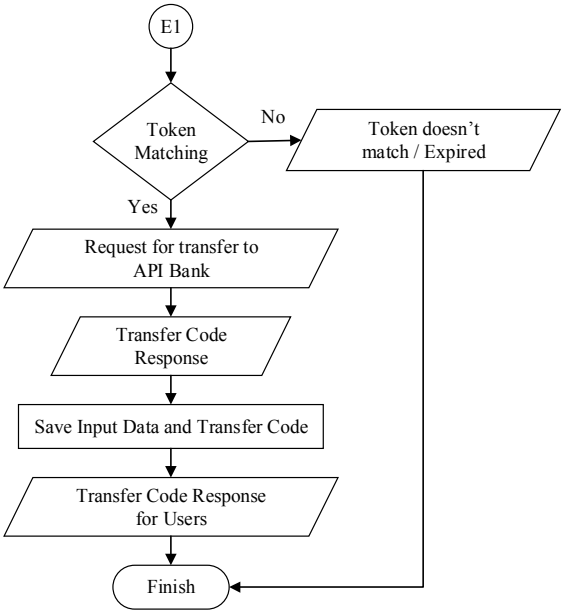

Fig 6. Flow of Top Up requests on API Bank

The buyer's user first gives the desired number of top up, then the system checks the token from the user, when appropriate, and then proceed to the top up the process and

when it does not match until its finished. Then when it matches the token, then the user top up requests according to the data format and protocol data needed by the API Bank, then the system will get a virtual account number reply that will be forwarded to the buyer's user and save it to the database.

After the buyer user transfers via ATM with the given virtual account number, the back end receives a reply in the form of POST Notification from the payment gateway MIDTRANS server when the transfer has been completed. Fig 7 is the flow of API Bank replies to the back end.

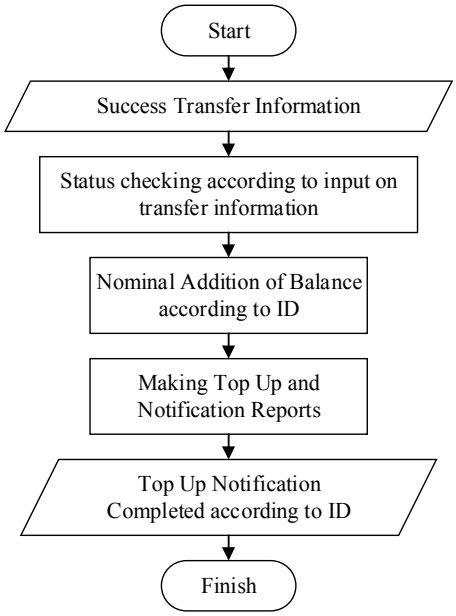

Fig 7. Top Up Flow Successful on Bank API

The data is compared with top up request data according to the user ID requesting top up to differentiate from top up requests with other users. Then after comparison, the addition of the nominal balance according to the user ID top up and the making of top up reports and at the same time save the database.

There is a database of top up information in the form of notifications by storing it in the notification collection and then giving the bank top-up notification successfully to the buyer's account via socket.io.

IV. Experimental Setup

System testing is done by running a program on a Virtual Private Server (VPS) with certain specifications. The VPS specifications, operating systems [8], and programming tools for this testing process are those mentioned in Table 2

TABLE II. PROGRAMMING TOOLS

No	Description	Specification
1	*Processor*	1 vCPU
2	*RAM*	1 GB
3	*Harddisk*	25 GB Disk
4	*OS type*	Ubuntu 16.04.5 64 bit
5	*Database*	MongoDB
6	*Framework*	NodeJS
7	*Application*	Postman, IntelliJ and JMeter

A. Token for security system

Incorrect token testing is done by using an inappropriate token or several lines of tokens that are tried to delete and check the token checking system, this test can be done with the entire backend system that is on the login and user list. The results issued by the system are invalid signatures in figure 8.

Fig 8. Incorrect Token

Testing of expired tokens is done by using tokens that expire from more than 24 hours after being created, this test can be carried out throughout the existing backend system except for the login and user list. The results released by the system are Jwt Expired in figure 9.

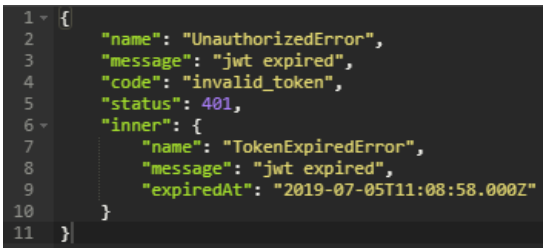

Fig 9. The token has expired

B. Top Up Via API BANK

Testing the top up virtual account through a server using the PUT data protocol http://ezpy.advlop.com/api/v1/charge/midtrans/<name>/<email>/<nominal_topup>. The process are required name, email user and nominal that will added to the balance. The data protocol is accessed and successful, the system will issue data as in table 3, namely the response obtained from the MIDTRANS payment gateway server is one such as the virtual account number (bill key) and company number (biller code) with the transaction status is pending, for banks selected in this mode only PT Bank Mandiri (Persero) Tbk.

TABLE III. TESTING TOP UP VIRTUAL ACCOUNT

Postman Top Up Virtual Account	
Method :	PUT
URI :	http://ezpy.advlop.com/api/v1/charge/midtrans/Bob Maul

Token :	eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJlbWFpbCI6ImJvYm1hdWxhbmFhZGFtO

Response (JSON)	{
	"status_code": "201",
	"status_message": "OK, Mandiri Bill transaction is successful",
	"transaction_id": "42e0bb91-175c-4025-bb75-f82c2184f3b6",
	"order_id": "A-1562297525033",
	"gross_amount": "10000.00",
	"currency": "IDR",
	"payment_type": "echannel",
	"transaction_time": "2019-07-05 10:32:05",
	"transaction_status": "pending",
	"fraud_status": "accept",
	"bill_key": "875710356796",
	"biller_code": "70012"
	}

In the payment test, there is a web page from MIDTRANS to collect top up payments when accessing the API of top up virtual account, what is needed is a virtual account number (bill_key) and company number (biller_code) in response to table 3 then simulating payments by accessing web pages https://simulator.sandbox.midtrans.com/mandiri/bill/index.

Once paid, the server gets a notification from the link that has been provided for MIDTRANS in the form of POST notification with the form of the data protocol is http: // ezpy .advlop.com/api/v1/notifMidtrans. In figure 10, received notifications will be stacked on top up database but only need to require transaction status data responses and stacked transaction times by relying on virtual account number searches from top up databases when billing according to the virtual account numbers received at POST notifications.

Fig 10. Notifications received from Midtrans

It can be seen in Figure 10 for the process that occurs on the server when receiving notifications from MIDTRANS. Notifications in figure 11 will appear in the buyer's console log account that is owned by the client application, in this test exemplified by a web socket with a web display.

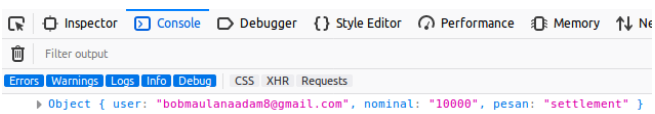

Fig 11. Top up virtual account notifications with Web Socket

Changes the transaction status to be paid or "settlement" then there is a process of adding balance according to the user who made the top up a virtual account.

C. *Testing for robustness in data traffic*

In this test, the endurance test is carried out on the server, the tests performed are API requests by many users in time, so that the system can be seen as resilient in dealing with data traffic that can change over time. Tests are carried out on each backend system. This test is done by using a third-party application, namely Jmeter and done on a Virtual Private Server (VPS). JMeter is a Java-based open-source application that can be used for performance tests [9]. For a QA JMeter Engineer can be used to load/stress testing Web Application, FTP Application, and Database server tests.

In this test, the system will be accessed by 100 users done in 1 second [10][11]. Table IV are the test results:

TABLE IV. TESTING ALL SYSTEM

Name	Success Rate	Latency (ms)	Connect Time (ms)	Load Time (ms)
Sign Up	100 %	5188,1	5,6	5188,2
Login	100 %	5238,12	41,45	5238,2
Balance	100 %	71,81	5,94	71,87
Transaction	Buyer 45% Seller 46%	339,93	59,7	340,12
Top Up	72 %	891,41	16,01	891,43
Log Transaction	100 %	1275,55	9,99	1746,7
VA Charge	100 %	1196,09	9,66	1196,1
Transfer Bank Charge	100 %	77,26	6,77	77,28
Log Top Up	100 %	1645,58	47,49	7169,8
Log Notification	100 %	837,27	6,03	1292,1
Withdraw	77 %	84,07	5,64	84,11
Cash Flow	100 %	837,27	6,03	1292,1
Logout	100 %	759,59	297,3	756,68

V. CONCLUSION

The backend system which designed in this paper involves account management systems such as login, register and logout. Beside that is balance system, admin top-up information, log notification, account transfer and cash flow report. The protocol designed using REST architecture with HTTP Request there are GET, POST and PUT and also involved security system with JWT token. The result of this research are backend testing with its robustness on 100 API request carried out in 1 second with success rate for the entire system is 76.92% of the 13 features offered. Transaction features have a 45% success rate for the process of reducing

buyer balance, a 46% success rate for the process of adding seller balance. The top-up balance feature reaches 72% and for the withdraw feature has a success rate of 77%.

Based on the results of the research as far as the author did, some things must be added for further development is the server specifications, it needs to be improved for the processor and memory so that all requests that enter the server can be processed by the system. After that, change the data update method in the database to support the success of the data processing system. After that, additional permission from the MIDTRANS payment gateway is needed to be able to use real money transactions.

REFERENCES

[1] "EDCBCA",HTTPs://www.bca.co.id/id/Bisnis/Produk-dan-Layanan/E-Banking/edc, Accessed on 2 July 2018.

[2] T. Ma, H. Zhang, J. Qian, X. Hu and Y. Tian, "The Design and Implementation of an Innovative Mobile Payment System Based on QR Bar Code," 2015 International Conference on Network and Information Systems for Computers, Wuhan, 2015, pp. 435-440.

[3] Xiang-Wen Huang, Chin-Yun Hsieh, Cheng Hao Wu and Yu Chin Cheng, "A Token-Based *User* Authentication Mechanism for Data Exchange in *REST*ful *API*," 2015 18th International Conference on Network-Based Information Systems, Taipei, 2015, pp. 601-606.

[4] Z. Niu, C. Yang and Y. Zhang, "A design of cross-terminal web system based on JSON and REST," 2014 IEEE 5th International Conference on Software Engineering and Service Science, Beijing, 2014, pp. 904-907.

[5] A. Agocs and J. L. Goff, "A web service based on RESTful API and JSON Schema/JSON Meta Schema to construct knowledge graphs," 2018 International Conference on Computer, Information and Telecommunication Systems (CITS), Colmar, 2018, pp. 1-5.

[6] D. Ramesh, E. Khosla and S. N. Bhukya, "Inclusion of e-commerce workflow with *NoSQL* DBMS: MongoDB document store," 2016 IEEE International Conference on Computational Intelligence and Computing Research (ICCIC), Chennai, 2016, pp. 1-5.

[7] M. Haekal and Eliyani, "Token-based authentication using JSON Web Token on SIKASIR RESTful Web Service," 2016 International Conference on Informatics and Computing (ICIC), Mataram, 2016, pp. 175-179.

[8] A.J. Poulter, S.J. johnston and S. J. Cox, "Using the MEAN stack to implement a *RESTful* service for an Internet of Things application," 2015 IEEE 2nd Forum on Internet of Things (WF-IoT), Milan, 2015, pp.

[9] S. Kiran, A. Mohapatra and R. Swamy, "Experiences in performance testing of web applications with Unified Authentication platform using Jmeter," 2015 International Symposium on Technology Management and Emerging Technologies (ISTMET), Langkawai Island, 2015, pp. 74-78.

[10] I.Y. Andhica and D. Irwan, "Performa Kinerja Web Server Berbasis Ubuntu Linux Dan Turnkey Linux" 2017 Jurnal Penelitian Ilmu Komputer, Sistem Embedded & Logic 5(2) : 68-78 (2017)

[11] Dani, Rahmad & Suryawan, Fajar. (2017). Perancangan dan Pengujian Load Balancing dan Failover Menggunakan NginX. Khazanah Informatika: Jurnal Ilmu Komputer dan Informatika. 3. 43. 10.23917/khif.v3i1.2939.