Integration of Supply Management System in Auto Parts Company Using Web Services

Indra Gunawan¹, Wina Witanti², Faiza Renaldi³

Department of Informatics, Universitas Jenderal Achmad Yani, Cimahi – Indonesia

Email: ind.gunawan@student.unjani.ac.id¹, wina.witanti@lecture.unjani.ac.id², faiza.renaldi@unjani.ac.id³

Abstract. The web industry at this point is getting more and more aggressive to implement the concept of web services that provide message interoperability and data exchange interfaces. This is very useful in managing data from various websites where data integration might occur. The case study of this research was carried out on managing auto parts supply, using research methods collecting and analyzing data structures, architectural design data, building web service, building APIs, testing, and measurement. Background and research objectives can be formulated how to combine data from various information system applications, into a supply management system that is designed using web service technology by managing API data on the REST server so that it can be used by REST clients. This paper results in the creation of a web service through a testing phase with 4 trials with a total data received 18.31KB and a total response time of under 2000ms.

Keywords: Web Service, RESTful, Integration Data.

1. Introduction

Web services are included in today's web technology which is being widely presented to provide universal services. Service-oriented computing is usually published by service providers on the internet in the form of a web API (Application Programming Interface)[1]. On November 13, 2018, the world's largest online web service repository (ProgrammableWeb) gained more than 20,000 API services and 7000 mashups with more than 400 diverse web categories[2]. The web service concept provides message interoperability and data exchange interfaces[3]. It seems simple, but web services require a stand-alone connectivity or complex transaction chain to deal with in terms of security[4], reliability[5], version[6], and industrial activity[7]. The need for industrial activities that can apply the concept of web service is one of them in terms of supply. Supply is current assets in the form of equipment intended to support the operational activities of trading businesses and items provided for sale and / or surrender in the context of buying and selling. Supply management activities require input/output data[8], clarity of relevant information[9], supply data exchange support system[10], and others that are appropriate to the needs of supply. The need to manage the supply can be stated in a system that supports interaction between several different applications through the network by utilizing information technology based on web services[11].

One web-based information technology that can accommodate the system in terms of platform diversity[12], the ability to penetrate a firewall[13], and application integration with service-oriented architecture[14] is an advantage of the concept of web services. Web service is a data collection application (database), software, or piece of software that can be accessed remotely by various
devices with a particular intermediary. This means that web services can store information data in an XML (Extensible Markup Language) file[15] and JSON (JavaScript Object Notation)[16], so it can be accessed by other systems despite different platforms, operating systems or even different compiler languages[17]. The concept of web service emerged to support distributed systems running on different infrastructures[18], offers convenience to bridge the exchange of information used and supports the integration of various system platforms and applications[19], facilitate the exchange of data between different systems and between different platforms[20], integration modeling by presenting a new web platform and describing its functions[21], the use of features for monitoring access to the database on the server using one of the web service methods, namely REST (Representational State Transfer) in designing a web architecture[22], in this case, the REST method on the web service is superior to SOAP (Simple Object Access Protocol) because it does not require a lot of bandwidth so that access is faster[23].

Many studies are conducted in web service technology specifically to support the management of a company's organization. Some studies include discussing transaction technology[24], service provider for the price of items online[25], interoperability in credit card verification[26], and cache management as query optimization in the healthcare web community[27]. However, special research that is focused on managing supply is currently very minimal, even though this is an important matter with a high degree of urgency because it requires a service that can manage data between websites so that data integration occurs.

This paper aims to integrate data from 3 software applications into a unified system that is taken directly from the server and accessed by clients through web services. To support this research, a case study has been prepared using data parameters for auto parts distributors and workshops. The purpose and results of this software development can obtain data sourced from various kinds of software so that it can be used by REST client networks through REST server access that provides API data to be managed into a web service. Based on the above conditions, the authors conclude that there are gaps between the data from various software so that it can be resolved using the RESTful API web service technology.

2. Methods

The stages of making this web service using the RESTful API method that can be easily combined with other software and simulation software, through web services following the REST principle by exposing the service as a URI and displayed via HTTP messages[28]. The design of an auto parts supply management system involves a REST Server and a REST Client using the RESTful standard HTTP method, shown in Table 1[29].

| Method | CRUD       | Explanation          |
|--------|------------|----------------------|
| POST   | Create     | Creating data        |
| GET    | Retrieve/read | Showing data   |
| PUT    | Update     | Edit data            |
| DELETE | Delete     | Delete data          |

This framework consists of three main elements namely data structure analysis, data unification design architecture, and data access via API. The initial step in developing parts supply software is an analysis of the data structures found in the previous system. Then, proceed with the steps of unifying the data that has been analyzed on a database system, then the data API is formed into a JSON data format which can be accessed via the KEY API until it willinally enter the implementation phase. While managing, measuring, and monitoring the software will be discussed in the results and discussion section, the research method shown in Figure 1.
2.1. Data Structure Analysis

Analyzing a system that is running in an organization is one of the stages which aims to find out the pattern of the system and database architecture used. Analysis of data structures in the workshop only has parameters for storing auto parts data, including parameters id, image, time, vendor_id, and brand_id, shown in Figure 2.

![Figure 2: Data Structure](image)

At present, the system runs in three different organizations namely PT Laris Chandra, PT. Autochem Industry and PT. Dirgaputra Ekapratama. This organization is engaged in the industry that sells an auto parts product, the three distributors each have an online auto parts information system with the domains larischandra.com, autochem.id, and dirgaputra.co.id. The parameters that determine the three distributor information systems have the same characteristics, such as parts names, parts numbers, OEM types/numbers, parts descriptions, and models.

2.2. Data Integration Design Architecture

The design phase of data integration in one database is the process of establishing input and output parameters for each service. This stage explains what data specifications are used, shown in Table 2.

| Function                  | Message Type | Message Name     | Fill in the Message / Attribute                                                                 |
|---------------------------|--------------|------------------|--------------------------------------------------------------------------------------------------|
| Add Order                 | Input        | orderRequest     | id_autoparts (in the form of an array), id_workshop, code, type_sukucdgstock (in the form of an array) |
| See Order Status          | Output       | orderResponse    | id_order                                                                                         |
| See Parts Details         | Input        | statusRequest    | id_order, stts_order, id_order, date_order, description                                           |
|                           | Output       | statusResponse   | id_autoparts, name_autoparts, price_autoparts, image, code_autoparts, desc_autoparts, oem_autoparts |
| See Auto Parts Supply     | Input        | stockRequest     | id_autoparts                                                                                     |
|                           | Output       | stockResponse    | id_autoparts, name_autoparts, price_autoparts, stock_autoparts                                  |
2.3. Data Access API
The API is designed to be a RESTful web service, at this stage it explains the Web Service Description Language (WSDL) specifications and API URL design. To create a RESTful web service, the HTTP method used must comply with standards including GET, POST, PUT and DELETE[29].

2.3.1. Web Service Specifications Description Language
This stage is the design resulting from the service interface then implemented in the form of WSDL which aims to design the service interface format that is built.

WSDL: Service Interface Add Order is a service to show added orders ordered by the workshop, Shown in Table 3.

| No. | Web Service Description Language |
|-----|----------------------------------|
| 1.  | “Order”: {                     |
| 2.  | “status”: “”,                   |
| 3.  | “data”: {                      |
| 4.  | Id Order : “”                   |
| 5.  | }                               |
| 6.  |                                 |

Table 3. Service Interface Add Order.

WSDL: Service Interface See Order Status is a service to show how the web service responds to requests from workshops or distributors, Shown in Table 4.

| No. | Web Service Description Language |
|-----|----------------------------------|
| 1.  | “Order Status”: {               |
| 2.  | “status”: “”,                   |
| 3.  | “data”: {                       |
| 4.  | Status Order: “”                |
| 5.  | Id Order: “”                    |
| 6.  | Date Order: YYYY-mm-dd          |
| 7.  | Description: “”                 |
| 8.  | }                               |
| 9.  |                                 |

Table 4. Service Interface See Order Status.

WSDL: Service Interface See Parts Details is a service from the web service to show information related to selected parts and appear in the parameters of parts such as the id of the auto parts, the names of the parts, pictures of the parts, etc. Details of these parts appear in an array, Shown in Table 5.

| No. | Web Service Description Language |
|-----|----------------------------------|
| 1.  | “Detail Auto Parts”: {          |
| 2.  | “status”: “”,                   |
| 3.  | “data”: {                       |
| 4.  | Id Auto Parts: “”                |
| 5.  | Name Auto Parts: “”              |
| 6.  | Image Auto                      |

Table 5. Service Interface See Parts Details.
7. Description
   Auto Parts: ""
8 }
9 }

WSDL: Service Interface See Auto Parts Stock Supply is a service to show information related to auto parts stock supply that appears in the form of arrays, auto parts supply parameters such as auto parts id, spare parts stock, price of auto parts, etc., Shown at Table 6.

| No. | Web Service Description Language |
|-----|----------------------------------|
| 1.  | “Stock Supply”: {               |
| 2.  | “status”: ""                   |
| 3.  | “data”: [{                     |
| 4.  | Id Auto                         |
|     | Parts: "",                     |
| 5.  | Name Auto                       |
|     | Parts: "",                     |
| 6.  | Stock Auto                      |
|     | Parts: "",                     |
| 7.  | Price Auto                      |
|     | Parts: ""                      |
| 8.  | Description Auto Parts: ""      |
| 9.  | }]                              |
| 10. |                                 |

2.3.2. List of API URL Design
The stage of accessing API (Application Programming Interface) methods provided by the RESTful API, a regular URL address is used to make it easier for others to use it, these URLs are called endpoints[30]. Every user accessing the RESTful API has been registered as a user through the built-in REST server, which will then get an API_KEY to access all the services available in the RESTful API through the REST Client. List of URLs to access API data shown in Table 7.

| Num. API | URL                  | Method | Parameter | Description |
|----------|----------------------|--------|-----------|-------------|
| API1     | /listStockparts      | GET    | API_KEY   | The method used to display data for all parts |
| API2     | /addStockparts       | POST   | no_stockparts, nama, API_KEY | The method used to add auto parts data |
| API3     | /modifyStockparts/:no_stockparts | PUT | no_stockparts, nama, API_KEY | The method used to edit auto parts data is based on the no_stockparts parameter |
| API4     | /removeStockparts/:no_stockparts | DELETE | no_stockparts, API_KEY | The method used to delete parts data is based on the no_stockparts parameter |

3. Result and Discussions
In this research, web service is built with a REST service associated with each object in question including services for distributors and workshops in coordinating business processes, which are described in architecture, shown in Figure 3. Web service architecture in this case study, there are 3
main entities, namely service requester/client, service provider/web service, and service registry/service provider. Web service serves to provide services and process a registry so that these services can be available / access. The registry service or service provider functions as a central location that describes all the services needed in the form of an API. Service requester / client is a service requestor who seeks and finds needed services and uses the service.

3.1. Web Service
API data is retrieved and displayed on the REST Client application that has been made, before accessing the API data the authorization process is required to access the REST Server, input a valid Username and Password and the URL of the intended REST Server, if the token has not been entered then access to the REST Server cannot and was rejected with a 403: Forbidden status code. After the authorization, is valid, the next step is to access the token with key and value. Access the token if valid will bring up the status code 200, which means HTTP OK and displays the entire API data, API data is hosted by REST Server in JSON format, API data can be requested by the REST Client and displayed in the form of a website called a web service, as shown at Figure 4.

3.2. Data Integration Testing
The results of testing data integration on the REST server using Postman succeeded in showing that the response of the data received was by the database, and here is a postman test with code script, shown in Table 8.
Table 8. Script Code Testing.

| Parameter          | Script Code                                                                 |
|--------------------|-----------------------------------------------------------------------------|
| Number             | pm.test("Integration Test Data Number", function () { var jsonData =       |
|                    |   pm.response.json(); pm.expect(jsonData.data[0].no).to.eql("1"); });       |
| Name               | pm.test("Integration Test Data Name", function () { var jsonData =         |
|                    |   pm.response.json(); pm.expect(jsonData.data[0].name).to.eql("Brake        |
|                    |   Linings"); });                                                           |
| Code               | pm.test("Integration Test Data Code", function () { var jsonData =         |
|                    |   pm.response.json(); pm.expect(jsonData.data[0].code).to.eql("BL-001");   |
| OEM                | pm.test("Integration Test Data OEM", function () { var jsonData =          |
|                    |   pm.response.json(); pm.expect(jsonData.data[0].oem).to.eql(""));          |
| Description        | pm.test("Integration Test Data Description", function () { var jsonData = |
|                    |   pm.response.json(); pm.expect(jsonData.data[0].description).to.eql("Mitsubishi PS100 FR/RR"); |
| Stock              | pm.test("Integration Test Data Stock", function () { var jsonData =        |
|                    |   pm.response.json(); pm.expect(jsonData.data[0].stock).to.eql("88"); })   |
| Price              | pm.test("Integration Test Data Price", function () { var jsonData =        |
|                    |   pm.response.json(); pm.expect(jsonData.data[0].price).to.eql("Rp 88,000"); |
| Image              | pm.test("Integration Test Data Image", function () { var jsonData =        |
|                    |   pm.response.json(); pm.expect(jsonData.data[0].image).to.eql("1.jpg"); })|

Result data were tested in 8 different parameters, shown in Figure 5.

Fig. 5. Data Integration Testing Results.
3.3. Execution Time Measurement

The results of the access speed and output accuracy tests performed on the REST server API data distributor's trial with GET 100 data resulted in a total execution time of 457ms, total data received 16.24KB and total data response of 260ms, the POST data method produces a total execution time of 1522ms, total data received 1.45KB and total data response of 1426ms, the PUT method produces 475ms of total execution time, 62B of total data received and 358ms of total data response, and the DELETE data method produces a total execution time of 1882ms, total data received is 0B and total data response is 1755ms, as shown in Figure 6.

Fig. 6. Execution Time Measurement Results.

Here are the results of the measurement test system response time using the RESTful API method using Newman (see Table 9).

| Test Case Number | Total Data Execution | Method | URL                  | Time Respond (millisecond) |
|------------------|-----------------------|--------|----------------------|---------------------------|
| Tes1             | 100                   | GET    | /listStockparts      | 260                       |
| Tes2             | 1                     | POST   | /addStockparts       | 1426                      |
| Tes3             | 1                     | PUT    | /modifyStockparts/:no_stockparts | 358                     |
| Tes4             | 1                     | DELETE | /removeStockparts/:no_stockparts | 1755                     |

4. Conclusion

This research presents integrated information and data that can be operated by other platforms based on the RESTful web service architecture. Web service serves as an effective and important service used for industry needs, because these services can integrate their data and develop their products collaboratively in responding quickly to customer requirements, with the use of RESTful web service technology from the data API, it can make the system more structured in the HTTP method, so that it can be accessed by clients easily using different platforms. The problem handled by web services in the case study of this research is integrating distributor data with the workshop to be able to provide fast services according to the information needs of users. The solution to this problem is realized and implemented in a REST client software that has the ability to shareability, reusability, and interoperability by adopting the JavaScript Object Notation (JSON) format and RESTful web service technology.

Tests conducted using the Postman and Newman applications in this study obtained results of data response speeds of less than 2000ms and these figures indicate that RESTful web service is feasible to
implement, it is demonstrated by the fast data execution, the consumption of memory resources that are not large enough, and its performance which is quite good.

This paper results in the creation of a web service that has been able to process data exchange and minimize gaps found in previous software to be more connected and integrated. In this paper, we also hope to contribute with a more in-depth analysis of the scalability of the quality attributes of the RESTful web service. For further research, this technology can be developed from the aspects of user authorization and URL transparency to improve security.

References
[1] B. Z. Shengye Pang, Guobing Zou, Yanglan Gan, Sen Niu, “Augmenting Labeled Probabilistic Topic Model for Web Service Classification,” Int. J. Web Serv. Res., vol. 16, pp. 93–113, 2019.
[2] Y. G. Guobing Zou, Zhen Qin, Qiang He, Pengwei Wang, Bofeng Zhang, “DeepWSC: A Novel Framework with Deep Neural Network for Web Service Clustering,” Proc. - 2019 IEEE Int. Conf. Web Serv. ICWS 2019 - Part 2019 IEEE World Congr. Serv., vol. 1, pp. 434–436, 2019.
[3] A. K. Kalpana Johari, “Interoperability Issues in Web Services,” ACM Int. Conf. Proceeding Ser., vol. 12, pp. 614–619, 2012.
[4] S. M. Ketki Joshi, Karuna Pande Joshi, “A Semantic Approach for Automating Knowledge in Policies of Cyber Insurance Services,” Proc. - 2019 IEEE Int. Conf. Web Serv. ICWS 2019 - Part 2019 IEEE World Congr. Serv., vol. 19, pp. 33–40, 2019.
[5] J. E. Marco Franceschetti, “Checking Temporal Service Level Agreements for Web Service Compositions with Temporal Parameters,” Proc. - 2019 IEEE Int. Conf. Web Serv. ICWS 2019 - Part 2019 IEEE World Congr. Serv., vol. 19, pp. 443–445, 2019.
[6] T. J. G. Negar Abbasi, Ali Moienei, “Web Service Candidate Identification Using the Firefly Algorithm,” Int. J. Web Serv. Res., vol. 15, pp. 45–60, 2018.
[7] A. T. Aram Alsedrani, “Web Service Composition Processes: A Comparative Study,” Int. J. Web Serv. Comput., vol. 7, pp. 1–21, 2016.
[8] C. B. Florin Leon, “An Optimization Web Service for a Freight Brokering System,” Serv. Sci., vol. 9, pp. 324–337, 2017.
[9] R. S. Gregory Dobson, “Web-based Simulations for Teaching Queueing, Little’s Law, and Inventory Management,” INFORMS Trans. Educ., vol. 7, pp. 106–124, 2006.
[10] U. R. Anindya Ghose, Tridas Mukhopadhyay, “The Impact of Internet Referral Services on a Supply Chain,” Inf. Syst. Res., vol. 18, pp. 300–319, 2007.
[11] M. H. K. Pessi, “Process Integration and Web Services: A Case of Evolutional Development in a Supply Chain,” Scand. J. Inf. Syst., vol. 16, pp. 117–144, 2004.
[12] D. L. LiYing Cui, Soundar Kumara, “Scenario Analysis of Web Service Composition based on Multi-Criteria Mathematical Goal Programming,” Serv. Sci., vol. 3, pp. 280–303, 2011.
[13] R. M. A. S. Rubens Matos, Paulo R.M. Maciel, “Sensitive GRASP: Combinatorial Optimization of Composite Web Services Guided by Sensitivity Analysis,” Int. J. Web Grid Serv., vol. 12, pp. 63–80, 2016.
[14] W. H. Jianpeng Hu, Linpeng Huang, Ying Fan, Lanxuan Tong, “Bandwidth Planning of Web Services in Changing Contexts Based on Network Simulation,” Proc. - 2019 IEEE Int. Conf. Web Serv. ICWS 2019 - Part 2019 IEEE World Congr. Serv., vol. 19, pp. 242–246, 2019.
[15] L. L. Rekkal Sara, Amранe Bakhta, “A New Approach to Gather Similar Operations Extracted from Web Services,” Int. J. Electr. Comput. Eng., vol. 9, pp. 1437–1444, 2019.
[16] A. W. Akino Archilles, “Vision: A Web Service for Face Recognition Using Convolutional Network,” Telkomnika (Telecommunication Comput. Electron. Control.), vol. 18, pp. 1389–1396, 2020.
[17] W. N. Hyunyoung Kil, Reeseo Cha, “Transaction History-Based Web Service Composition for Uncertain QoS,” Int. J. Web Grid Serv., vol. 12, pp. 42–62, 2016.
[18] S. K. Lalit Purohit, “Clustering based Approach for Web Service Selection Using
SkylineComputation," Proc. - 2019 IEEE Int. Conf. Web Serv. ICWS 2019 - Part 2019 IEEE World Congr. Serv., vol. 19, pp. 260–264, 2019.

[19] A. S. Najme Abbasi Tehrani, “Aspectual Patterns for Web Services Adaptation,” Int. J. Web Serv. Comput., vol. 6, pp. 1–10, 2015.

[20] Y. L. Yueshen Xu, Jianwei Yin, “A Collaborative Framework of Web Service Recommendation with Clustering-Extended Matrix Factorisation,” Int. J. Web Grid Serv., vol. 12, pp. 1–25, 2016.

[21] T. F. Getachew F. Belete, Alexey Voinov, Inaki Arto, Kishore Dhavala, Tatyana Bulavskaya, Leila Niamir, Saeed Moghayer, “Exploring Low Carbon Futures: A Web Service Approach to Linking Diverse Climate Energy Economy Models,” MDPI, vol. 13, pp. 1–24, 2019.

[22] Y. Z. Chengduo Luo, Zibin Zheng, Xiaorui Wu, Fei Yang, “Automated Structural Semantic Annotation for RESTful Services,” Int. J. Web Grid Serv., vol. 12, pp. 26–41, 2016.

[23] C. V. Jesus Bellido, Rosa Alarcon, Cesare Pautasso, “SAW-Q: A Dynamic Composition Approach of REST Services Based on Queue Model,” Int. J. Web Grid Serv., vol. 15, pp. 29–58, 2019.

[24] Y. Z. Benchaphon Limthanmaphon, “Web Service Composition Transaction Management,” Proc. 15th Australas. database Conf. - Vol. 27, vol. 27, pp. 171–179, 2004.

[25] N. A. D. Shahram Esmaeilsabzali, “Online Pricing for Web Service Providers,” EDSER ’06 Proc. 2006 Int. Work. Econ. driven Softw. Eng., vol. 6, pp. 37–42, 2006.

[26] A. K. Kalpana Johari, “Some Interoperability Issues in the Designing of Web Services: Case Study on Credit Card,” Int. J. Web Serv. Comput., vol. 4, pp. 11–20, 2013.

[27] J. A. Hela Limam, “Managing Web Services Communities: a Cache for Queries Optimisation,” Int. J. Web Serv. Comput., vol. 1, pp. 1–19, 2010.

[28] G. W. Khaldoon Al-Zoubi, “Using REST Web-Services Architecture for Distributed Simulation,” Proc. - Work. Princ. Adv. Distrib. Simulation, PADS, vol. 23, pp. 114–121, 2009.

[29] D. R. M. S. Davis John, “A Framework for the Description, Discovery and Composition of RESTful Semantic Web Services,” ACM Int. Conf. Proceeding Ser., vol. 12, pp. 88–93, 2012.

[30] M. B. B. Yi Liu, Xuanzhe Liu, Yun Ma, Yunxin Liu, Zibin Zheng, Gang Huang, “Characterizing RESTful Web Services Usage on Smartphones: A Tale of Native Apps and Web Apps,” Proc. - 2015 IEEE Int. Conf. Web Serv. ICWS 2015, vol. 15, pp. 337–344, 2015.