Design and Implementation of REST API for Academic Information System

IOP Conf. Series: Materials Science and Engineering **875** (2020) 012047 doi:10.1088/1757-899X/875/1/012047

Design and Implementation of REST API for Academic Information System

A A Prayogi[1]*, M Niswar[1], Indrabayu[1], and M Rijal[1]

[1]Department of Informatics Engineering, Engineering Faculty, Hasanuddin University, Makassar, Indonesia

*Email: aisprayogi@unhas.ac.id

Abstract. With the increased number of information systems used in an organization, there is also an increased importance of data exchange between these systems. This research deals with prototype development and performance analysis of Rest API for academic information system. Rest API was developed using two different server technologies, NodeJS and PHP. The prototype was developed on top of a database server using one sample table that represents employee in a higher education institution. For each of the Rest API developed, there were 2 types of endpoint created. Experiment was set with one database containing a single table and utilized Apache Jmeter to simulate up to 1000 concurrent requests. The results of the experiment show NodeJS implementation of REST API consistently has better performance compared to PHP based REST API implementation. NodeJS implementation reached 100% throughput for up to 1000 concurrent requests, while PHP reached 48.70% throughput when serving the same number of concurrent requests.

1. Introduction

The use of information system especially web-based information system in the academic field especially in a higher education institution has become the norm. Information system holds important role in the various business processes in a higher education institution, ranging from new student registration process, course registration at the beginning of an academic period, end of academic period evaluation process to graduation and alumnae information management process [1].

To accommodate and ensure the flow of these business process runs smoothly, there are also multiple and diverse information systems set up. Some more common information systems used by a higher education institution including Learning Management System (LMS), Content Management System (CMS), Asset Management System, Student Registration System and Course Management System. In most institution, these different type of information systems are developed either in-house (by information-technology work unit), outsourced to a third-party vendor or customized from open-source community-based project. The maintenance of an application based on proprietary solution comes from the vendor itself. The process to develop new feature or to fix certain bugs need a lengthy administrative process that lags behind the user time schedule. Customized open-sourced based solution also faces the same problem, in which each bugs or new features must be registered first before it can be processed. These circumstances then lead to new features or bug fixing fall short of user expectation and needs [2].

Within the institution there is a need to have a common and uniform data communication between each information systems. The purpose of this uniform data communication is to prevent data duplication and data mismatch stored within each information systems. With the uniform data

communication format, new applications can be built easier and faster by using data stored in the previously established systems. The number of smartphone usage among internet users has been increased steadily every year. Smartphone users in Indonesia has reached 83.5 million users in 2018. Although most information system is easily accessed from mobile web browser, using mobile native apps has certain advantages over mobile website [3]. Mobile native apps need to communicate with the existing information system server via Application Programming Interface (API).

There are many types of architecture can be chosen to build data communication between information systems or applications. One of the options is using REST (Representational State Transfer Protocol) API architecture. With REST architecture, web service can be built using simple methods and underlying HTTP architecture. A prototype of REST API architecture for academic information system built with two types of framework technologies. The prototypes then analyzed for performances with three parameters (response time, throughput and packet loss) under multiple concurrent connections.

2. REST API

REST (Representational state transfer) is an architectural API (Application Programming Interface) which provides client-server communications for Web Applications over HTTP protocol, making it easily implemented since it is not bound to any transfer protocol. The three main design principles of REST are addressability, uniform interface, and statelessness [4]. REST addresses acceptability by defining endpoints in a directory structure [5] via different URI for extracting the data. The API works on the principle of CRUD (Create, Read, Update, Delete), which correspond to the most popular functions [4] INSERT, SELECT, UPDATE, and DELETE, in persistent data-storages such as SQL.

Calling a RESTful service over the HTTP protocol can be done in multiple programming languages. In jQuery, a framework of Javascript, a REST server can be called from an Ajax query. A common response of data is in the form of JSON (Javascript Object Notation) data. Typical JSON response can be seen in Figure 1. Data can be returned as well in the form of XML (eXtensible Markup Language) [4].

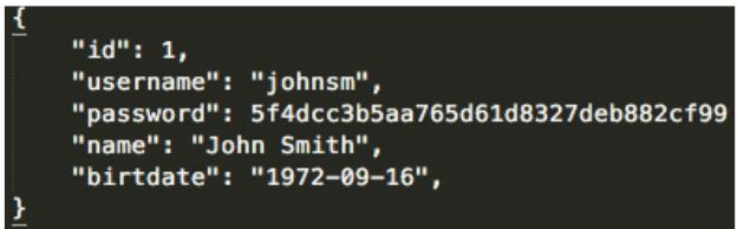

Figure 1. Examples of JSON response

3. Server Side Technologies

Developing web services with REST API architecture currently can be done with various server technologies. It is decided two server technologies most used, which are NodeJS and PHP.

3.1. NodeJS

Node JS is a web server technology based on JavaScript. It used V8 Javascript Engine with integrated module add-ons. NodeJS used event-driven architecture to handle request from clients. With the event-driven architecture, all requests from clients will be put to event queue. NodeJS single-threaded event loop then pick one of the requests to check if the request require blocking I/O operation or complex computation tasks. If it is, the request will be moved to different thread which taken from NodeJS Internal Pool Thread [6]. After thread finished processing the request either blocking I/O operation or complex computation tasks, it will send the response to back to event queue to be processed by the NodeJS single-threaded event loop later. This process is shown Figure 2.

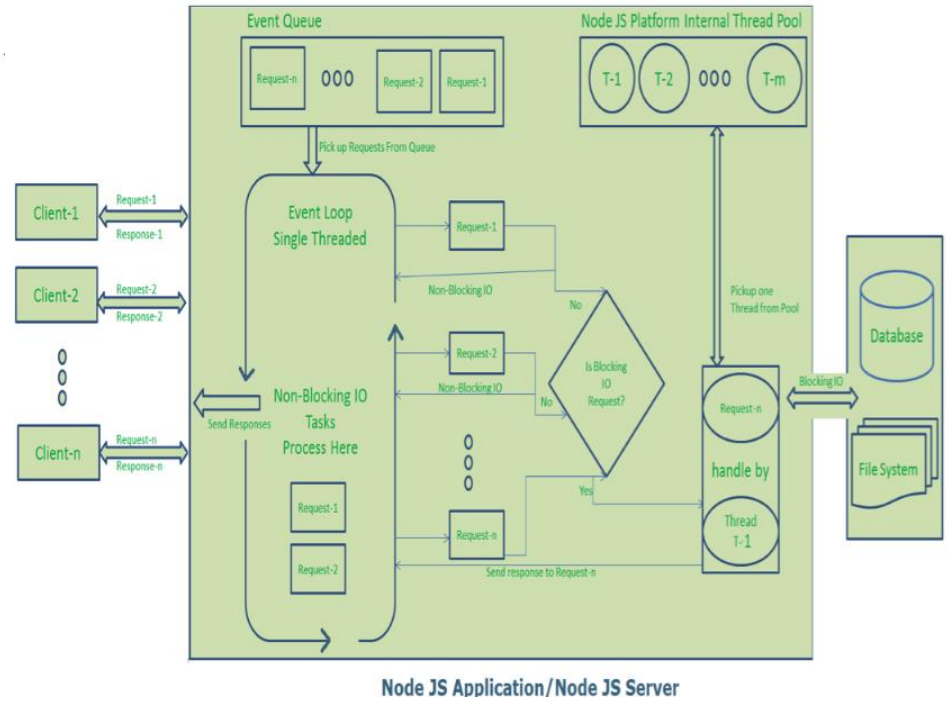

Figure 2. NodeJS architecture [6]

3.2. PHP

PHP also know as PHP:Hypertext Preprocessor is a script language developed to add dynamism to the static HTML page. It was first developed in 1994 by Rasmus Lerdorf. PHP script is processed by a certain process called PHP interpreter in a web server to generate HTML code, or binary image data. PHP evolved from procedural paradigm programming to object-oriented type programming in its latest iteration (PHP 7).

4. Experiment

The experiment was designed to show how the two server technologies used to implement REST API performed under multiple concurrent connections. There are 2 types of performances measurement used in this experiment including request time and throughput. Request time measures elapsed time between the time client send the request until the client received responses from server. Throughput is total number of processed requests in certain amount of fixed time. Packet loss is measured as percentage of packet lost with respects to packets sent.

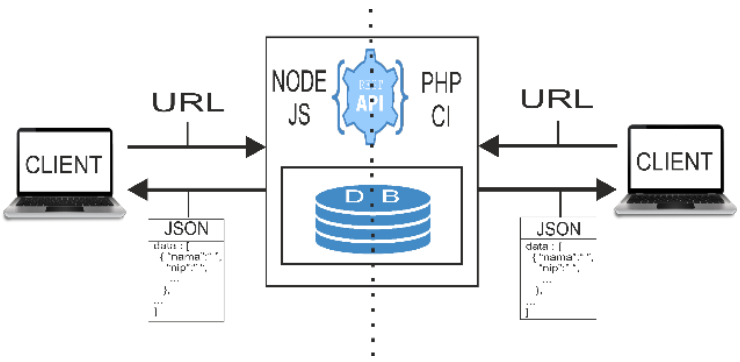

Figure 3. Prototype architecture

The prototype built as shown in figure 3. Clients send requests by accessing certain URLs provided by the server. Server was setup with two server technologies, NodeJS and PHP with Apache as web server. Every time request received either by NodeJS or PHP, the request processed based on the parameter and embedded data by the respected web server technology then return response in JSON (Javascript Object Notation) format.

A database was setup within the server to serve as data repository for both NodeJS and PHP REST API implementation. The database contains one table, called *profilpegawai* table. The table contains 5 fields and 3920 rows of data, copied from the current employee information management system implanted in Hasanuddin University. The detail structure of the table can be seen in Figure 4.

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
1	nama	varchar(70)	latin1_swedish_ci		No	None			Change Drop More
2	nip	varchar(18)	latin1_swedish_ci		No	None			Change Drop More
3	gender	varchar(25)	latin1_swedish_ci		No	None			Change Drop More
4	unitkerja	varchar(30)	latin1_swedish_ci		No	None			Change Drop More
5	posisi	tinyint(1)			No	None			Change Drop More

Figure 4. Table *profilpegawai* structure details

For this experiment, two types of REST API endpoints was created as shown in table 1. The first endpoint is used when a client wants to request all of rows, in this case employee data, from *profilpegawai* table. The second endpoint is used for a client to request certain employee data which matched the parameter NIP. Both endpoint responses are returned in JSON format.

Table 1. REST API endpoints

Parameter	URLs	
	NodeJS	PHP
GET ALL	http://localhost:3000/profilpegawai/	http://localhost:8080/rest-api/rest-ci/api/profilpegawai
GET PARAMS	http://localhost:3000/profilpegawai/:nip	http://localhost:8080/rest-api/rest-ci/api/profilpegawai?nip=NIP

After setting up database and API endpoints, next step was to prepare simulation for testing scenario. Apache JMeter was used to simulate requests from clients. To measure response time, for each NodeJS and PHP implementation, 10 requests was made to each endpoint. Each request repeated in 30 iterations and then the average of response time was calculated. Throughput was measured by simulating requests from 1, 100, 200, 300, 400, 500, 600, 700, 800, 900 and 1000 requests for each endpoint in each NodeJS and PHP implementation.

5. Result and Discussion

Response time for NodeJS REST API implementation shows significantly better result than those of PHP implementation. When testing GET ALL endpoint, average response time for NodeJS was ranging from 138ms to 156ms, while PHP implementation ranging from 244ms to 266ms. Complete results of response time measurement for GET ALL endpoint are shown in table 2. Similar result was also shown for the GET PARAMETER endpoint, in which NodeJS shows better results than its PHP counterpart.

Table 2. Response time results for GET ALL endpoint

Request	PHP			NODEJS		
	max (ms)	min (ms)	average (ms)	max (ms)	min (ms)	average (ms)
1	894	207	266	188	113	138
2	446	206	253	371	107	146
3	369	208	244	503	113	156
4	795	211	257	212	116	141
5	410	207	248	159	110	136

During throughput measurement, NodeJS implementation perform better compared to PHP implementation. When using GET ALL endpoint, NodeJS implementation consistently able to handle all requests from 1 to 1000 requests concurrently. PHP implementation on the other hand shows decreasing in throughput performance when the number of concurrent requests reached 300 requests as shown in table 3. When handling 1000 requests, PHP implementation only reached 48.70% throughput. NodeJS also show better performance in throughput value when testing with GET PARAMETER endpoint.

Table 3. Throughput measurement results for GET ALL endpoint

# of concurrent requests	THROUGHPUT	
	PHP	NodeJS
1	100%	100%
100	100%	100%
200	100%	100%
300	90%	100%
400	93,25%	100%
500	90,20%	100%
600	85%	100%
700	71,43%	100%
800	57,63%	100%
900	50,11%	100%
1000	48,70%	100%

The experiment result shown NodeJS implementation have better performance when asked to handle multiple concurrent requests. This type of situation was common real-world use case, in which a web service should handle multiple request in almost the same time. NodeJS architecture with non-blocking I/O single thread was crucial in handling simultaneous request while making sure all the requests is served in the least amount of time.

6. Conclusion

In our prototype of REST API implementation for academic information system with two server technologies, we explore the performance with two parameters, response time and throughput. Our prototype with NodeJS and PHP with two endpoints. NodeJS implementation tend to show better performance for both measurements in both endpoints. Further experiments need to be conducted to find out how the NodeJS and PHP implementation of REST API perform under more complex database setup and with more complex endpoints including I/O extensive operations.

Acknowledgement
This work has been supported by Faculty of Engineering, Universitas Hasanuddin under 2019 Labo-Based Education (LBE) grant.

References
[1] Aswati S, Mulyani N, Siagian Y and Zikra Syah A 2015 Peranan Sistem Informasi dalam Perguruan Tinggi *Jurnal Teknologi dan Sistem Informasi* **1** 79-86.
[2] Barata, R., Silva, S., Martinho, D., Cruz, L., & Guerra e Silva, L 2014 Open APIs in Information Systems for Higher Education. *Umea*.
[3] Turner-McGrievy GM, Hales SB, Schoffman DE 2017 et al Choosing between responsive-design websites versus mobile apps for your mobile behavioral intervention: presenting four case studies. *Transl Behav Med.* **7(2)** 224–232.
[4] Belqasmi, F., Singh, J., Melhem, S.Y.B., Glitho, R.H., 2012 SOAP-Based vs. RESTful Web Services: A Case Study for Multimedia Conferencing *IEEE Internet Comput.* **16** 54–63.
[5] Choi M. 2012 A Performance Analysis of RESTful Open API Information System *Future Generation Information Technology*, *Lecture Notes in Computer Science* Springer Berlin Heidelberg 59–64.
[6] Posa R. 2015 Node JS Architecture – Single Threaded Event Loop Retrieved from https://www.journaldev.com/7462/node-js-architecture-single-threaded-event-loop.