Regular Article

Ford Lumban Gaol*, Steven Santoso and Tokuro Matsuo

Design and development of the application monitoring the use of server resources for server maintenance

https://doi.org/10.1515/eng-2022-0055
received November 06, 2021; accepted July 06, 2022

Abstract: Websites must assure web server availability in light of the worldwide increase in internet users. Web server monitoring is the process of examining web application server utilization by showing data in the form of statistics or graphs. Data about web server use will help server owners make decisions about server availability. The purpose of this study is to create a web monitoring program capable of retrieving and storing data from Java application server resources. The research methodology utilized in this study is divided into two phases: data gathering and software development. The data gathering step includes doing a literature study, conducting a survey, distributing questionnaires, and analyzing comparable apps. The development phase employs the waterfall methodology. The outcome of this study is a web application that can monitor the Java application server's resource use and send email alerts when resource usage becomes excessive. To summarize, web application server monitoring may be utilized to alleviate developer workload.

Keywords: web application, monitoring, java

1 Introduction

The evolution of technology, information, and the internet has fundamentally altered how people use the internet for accessing information, entertainment, and e-commerce, among other things. As a result, many individuals believe that we are currently living in the age of Industry 4.0. With the breakout of corona at the end of 2019 that developed into a pandemic, more people rely on the internet since activities that were previously performed physically must now be performed remotely, such as working or studying from home, or buying and selling items through the internet. According to Nielsen, Online's WORLD INTERNET USE AND POPULATION STATISTICS 2020 report, internet users worldwide are increasing year after year [1]. The table below contains statistics about WORLD INTERNET USE AND POPULATION STATISTICS 2020 (Table 1).

According to the above statistics, growth from 2000 to 2020 is approximately 1.187%, and penetration rate is 59.6% in 2020. This is also true in Indonesia, where penetration is already at 73.7% in 2019, up from about 64.8% a year ago, with year over year growth of 8.9% [2].

With the global rise in internet users, websites must ensure web server availability. Server availability is critical, especially if the web server receives a high volume of traffic, because if a website or server is unavailable for 90% of the year, it means that the website will be unavailable for approximately 87 h and 40 min, causing users to visit another website and resulting in a loss of revenue. Using web server monitoring is one way to improve and maintain a web server's high availability.

Web server monitoring is the practice of analyzing the usage of web application servers by displaying data in the form of statistics or graphs. Data about web server use will assist server owners in making choices regarding server availability. Thus, the author of this research envisions developing a monitoring program that would assist server administrators in making decisions. The following are some examples of how server monitoring will be used:

1. Confirm the webapp server's availability at the time of processing.

* **Corresponding author: Ford Lumban Gaol,** Computer Science Department, BINUS Graduate Program – Doctor of Computer Science, Binus University Jakarta, 11480, Indonesia, e-mail: fgaol@binus.edu
Steven Santoso: Computer Science Department, Binus Graduate Program, Binus University, Jakarta, 11480, Indonesia, e-mail: steven.santoso001@binus.ac.id
Tokuro Matsuo: Advanced Research Center for Service Science and Artificial Intelligence, Advanced Institute of Industrial Technology, Tokyo, Japan, e-mail: matsuo@aiit.ac.jp

Table 1: World internet usage and population statistics 2020

	Population (2020 Est)	Population % of world	Internet users 31 May 2020	Penetration rate (% Pop.)	Growth 2000–2020	Internet world %
Total	7,796,949,710	100%	4,648,228,067	59.6%	1.187%	100%

2. Indicate the current resource consumption of the webapp server.
3. Facilitate the server owner's ability to make the best choice possible based on the facts in the database.
4. Notify the owner via email if the webapp server goes down or if the server's resource consumption exceeds the limit.

The rest of the article is organized as follows: Section 2 discusses other works related to this study and Section 3 gives the system design. Section 4 presents the results of web server monitoring app and conclusion is drawn in Section 5.

2 Related works

According to Ray [3], the study objective is to develop a server monitoring solution that can be accessed through a mobile web application from any location. This program may be used to monitor the connection server's status, port availability, CPU and memory utilization, network state logging, reboot the host, restart the service, and send email notifications regarding network condition.

According to Kusuma's study [4], the researchers designed a monitoring system based on Simple Network Management Protocol (SNMP) and used syslog from the application monitoring. The purpose of this study is to learn about the SNMP protocol's usage in monitoring network conditions.

As Prabawati [5] mentioned, the server monitoring system is built using PHP, SNMP, and a Short Message Service (SMS) gateway. As a result, administrators may monitor the health of existing servers even when they are not physically present in the IT room.

Wijayanto's [6] research at Diponegoro University's Faculty of Mathematical Sciences examines how network equipment like routers, switches, servers, and access points are managed. Due to the large number of devices controlled by the administrator, the procedure is carried out manually, depending on client complaints in the event of network difficulties.

The aim of the research performed by Fanggidae et al. [7] is to create server monitoring applications utilizing the programming languages Python, PHP, and shell scripts in conjunction with the SNMP protocol connection. The program is designed to automate the process of restarting a down server.

Ismaidi's study [8] developed a monitoring system web application that makes use of the SNMP to help administrators in real time by storing and analyzing the results of server monitoring.

Mathapati's study [9] demonstrated how to easily combine numerous sensor inputs and perform human system administrator analysis to get correct results. Often, human system administrators monitor just a subset of accessible system data, including server load, network load, I/O load, and intrusion detection data. By integrating this data, a human analyst may determine whether to examine the behavior of a particular system/server in more detail. The resource data analysis system is a technology that enables the rapid integration of sensor resources and the use of custom-built models to identify the evidence of the observed activities.

Vusvyta's study [10] focuses on developing a website interface with the assistance of a database, network mapping, and an early warning system (such as SMSs).

Susilawati et al. [11] demonstrated that the purpose of developing this application is to create a network monitoring application that can assist network administrators in monitoring the flow of data on the server via a SMS server application running on a GSM network, thereby simplifying network administration.

As a result of Luan's research [12], a web-based network monitoring application with email notifications was developed to assist network administrators in not only retrieving values but also processing and storing them in a database system in order to display information reports including the availability of devices connected to a computer network.

3 Materials and methods

The study methodology comprises six phases: data gathering, data analysis, current state, application design, application development utilizing the Kanban technique,

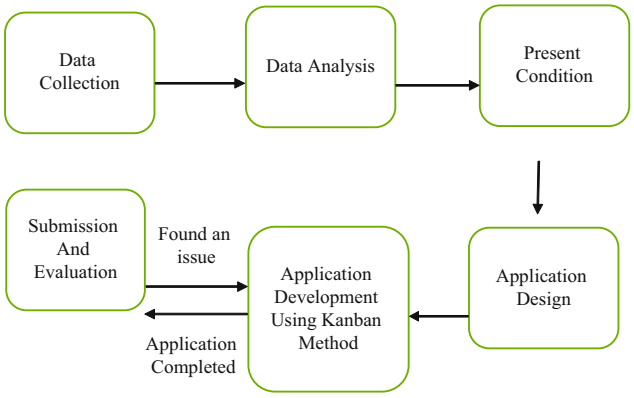


Figure 1: Research methodology.

submission and assessment. As shown in Figure 1, the first step of data collection will include conducting interviews with members of the mobile division. In stage two, we analyze the data collected in step one to conduct preliminary research and decide which features may be incorporated. In stage three, we assess the current state and decide what can be done to alleviate the developer's workload associated with server maintenance. After determining which features to be included and determining how we can assist the developer, we can create a system planning diagram such as a use case diagram, flowchart, activity diagram, and entity relationship diagram. Then, using the Kanban technique, we build the monitoring application. Finally, in step six, we submit our application for user review. If there is a problem, they will create Kanban cards to assist us in resolving it. If there are no issues, the application is complete.

3.1 Data collection

The first step is an interview with the head of the production department. After obtaining the data that the writer needs, the writer develops web application monitoring utilizing a waterfall timeline and Kanban technique. The rationale for adopting the Kanban method of development is that it has 14 reasons to be chosen for software development [13].

1. Panoramic perspective
2. Establishing a bottleneck
3. Self-managed teams
4. The sequence in which features are released
5. Complete focus on the critical
6. Concentration on job
7. Adaptability
8. The capacity to know everything
9. The absence of the necessity to evaluate characteristics
10. Less chit-chat and more action
11. Cohesion
12. More frequent errors
13. Increased flow
14. Concentrate on a single job.

3.2 Data analysis

Once the need is obtained, the following step is to decide which features may be implemented and which tools can be utilized. Following discussion, JavaMelody is already deployed on all client servers. JavaMelody is a free and open-source monitoring solution for JavaEE applications. JavaMelody's purpose is to monitor Java or Java EE applications in quality assurance and production settings. It is not a tool for simulating user requests; rather, it is a tool for measuring and calculating statistics on an application's real-world functioning based on actual use [14].

The writer then does research on other technologies for graph display in web applications and discovered Grafana's ability to create graphs. Grafana is a free and open-source visualization and analysis tool. This enables querying, visualization, alerting, and exploration of metrics stored in any location [15]. The issue here is that Grafana cannot directly access JavaMelody, necessitating the use of an intermediary application called Prometheus.

Prometheus is a free and open source monitoring and alerting system developed by SoundCloud. Since its debut in 2012, Prometheus has been embraced by a large number of businesses and organizations. Prometheus can read JavaMelody's API, while Grafana can read Prometheus's data.

3.3 Present condition

At the moment, the mobile production developer is required to perform monthly visits or maintenance to check the webapp server's health and to record the daily use of webapp server resources in the form of CPU, memory usage, active thread, transactions per minute, and garbage collector. At the moment, tracking daily webapp server resource consumption is still done manually by inspecting JavaMelody, a procedure that consumes between 1.5 and 2 h per server. Because data collection is still manual, it is prone to mistakes, and each developer has a unique interpretation of the webapp server resources graph.

3.4 Application design

Create a flowchart, an activity diagram, an entity connection diagram, and a narrative board using the data that have already been gathered. The chart shown in Figure 2 illustrates the web application Java application server monitoring flowchart.

Details of explanation of storyboard for web application monitoring can be seen in Table 2.

3.5 Application development using Kanban method

As the name implies, web application for server monitoring is developed using Kanban method. List of Kanban card and its each estimation (days) can be seen in Table 3.

Additionally, the writer used a simple waterfall chronology for the timeline as shown in Table 4. The aim of this

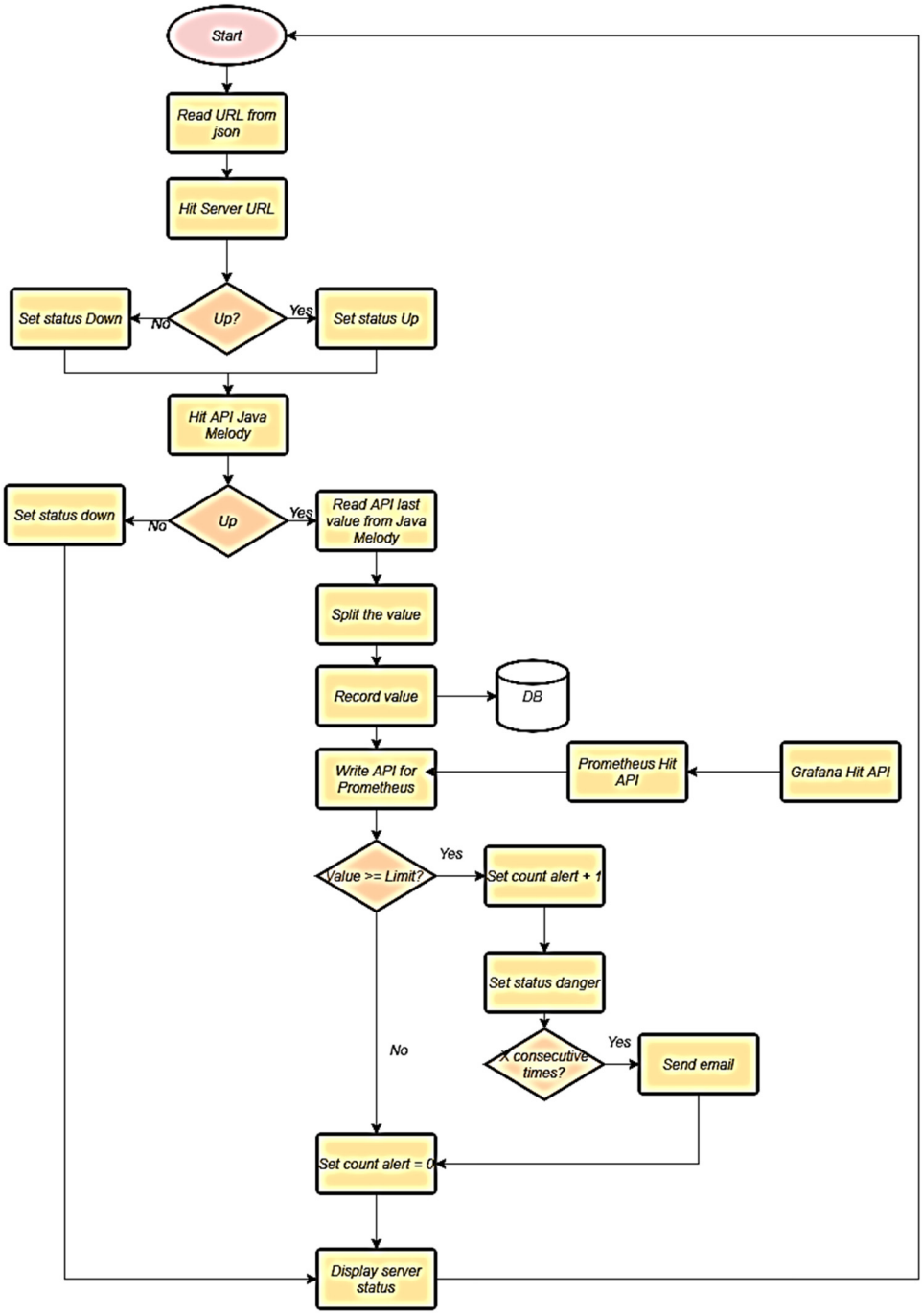
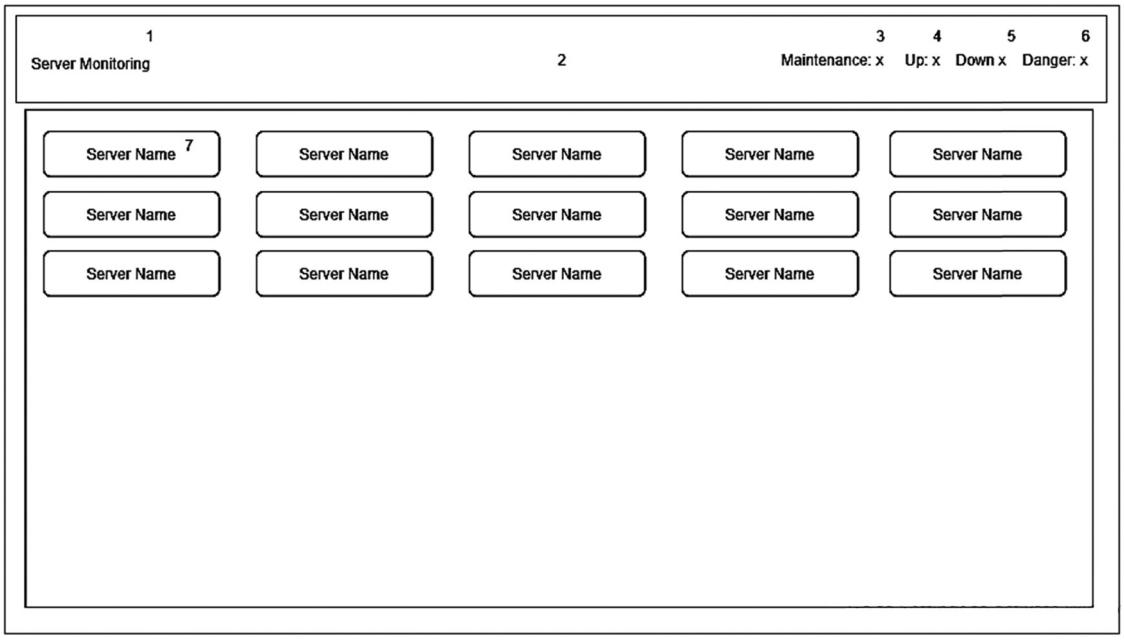


Figure 2: Flowchart web application Java application server monitoring.

Table 2: Storyboard

Storyboard	
Project: Monitoring server	Layer: 1 of 2
Screen ID: Home	

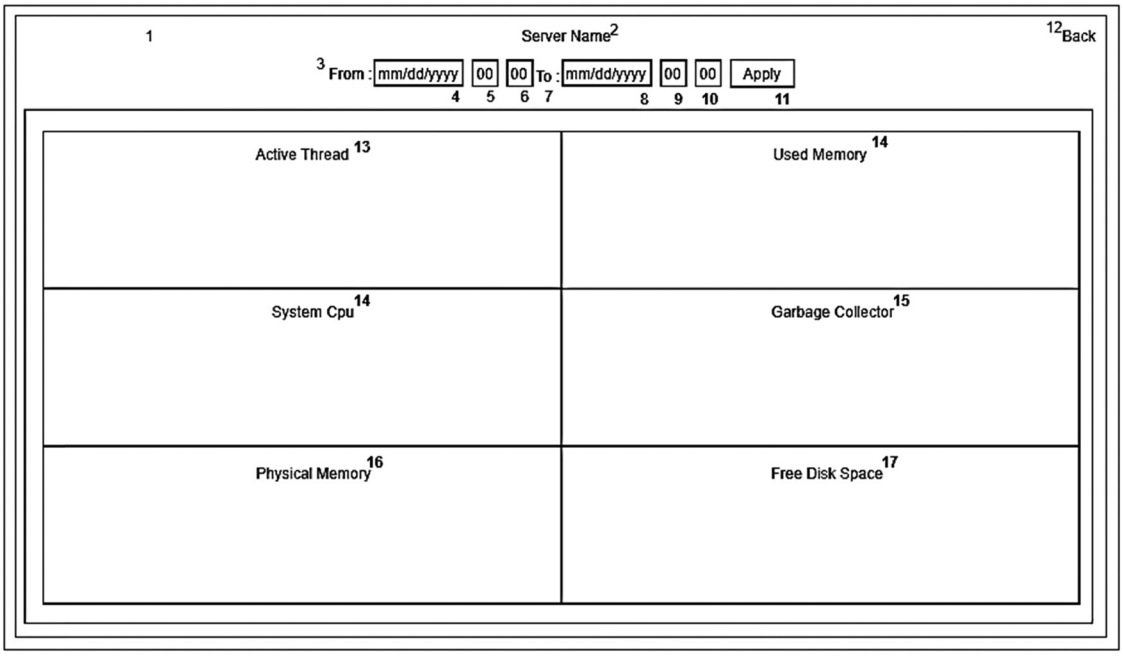


Description:
At homepage, there are several lists of webapp servers that can be clicked to open the server details page.
Link from screen ID: - Link to screen ID: Server details
Text attribute: 1: Arial, 2.75vw, color white #ffffff, background color black #000000
3: Arial, 1.75vw, color grey #dcdde1
4: Arial, 1.75vw, color green #2ecc71
5: Arial, 1.75vw, color red #FE0101
6: Arial, 1.75vw, color yellow #FFFF00
7: 2vw, color black #000000, border 2px solid #9392AE, border radius 90px
Background: 2: color black #000000

Storyboard
Project: Monitoring server
Screen ID: Server details

(*Continued*)

Table 2: *Continued*

Storyboard


Description:
At Server detail page, there is a feature to search the date between two dates to display the webapp server resources usage graph on the screen. At the top right there is a text “Back”, if clicked then web application goes back to home page
Link from screen ID: Home Link to screen ID: -
Text attribute: 2: Arial, 2.64vw, color white #ffffff, background color black #000000
3,7: 1.5vw, color white #ffffff
4,8: input type date
5,6,9,10: input type number
11: input type submit, 1.2vw
12: 1.5vw
Background: 1: color black #000000

Table 3: Kanban card

ID	Kanban card	Estimation (days)
1	Research JavaMelody	2
2	Research format data that can be read by Prometheus API	2
3	JSON format to use	1.5
4	Set up Java environment and database	1.5
5	Develop homepage	2
6	Store data from API JavaMelody to database	7
7	Store heavy query in database	2.5
8	Send email alert or early warning system if webapp server usage resources are greater than limit	2.5
9	Develop detail server page	7
Total		28

Table 4: Timeline

Month	February 2021				March 2021				April 2021				May 2021				June 2021				July 2021			
Week	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4
Data collection																								
Company survey	X																							
Interview with *user*		X	X																					
Data analysis																								
Problem analysis and problem solving				X	X	X																		
Application design research							X	X																
Application design																								
UML creation									X	X														
ERD creation										X	X													
Application development																								
App design and bug fixing											X	X	X	X	X	X								
Internal testing																X	X	X						
Submission and development																								
Testing by user																		X	X	X	X			
Application usability questionnaire																						X	X	
Documentation																								X

timeline is to explain the amount of time that will be spent processing requirements from start to finish.

3.6 Submission and development

The results of black box testing for server monitoring application are shown in Table 5.

4 Results

4.1 Application implementation

To perform this webapp application monitoring, we need the following hardware and software configurations: Minimum hardware specifications allow for web server monitoring that watches about 35 web apps in 15–25 s; however, we increase the timeout to 20 s if the connection is sluggish. As a result, each loop takes about 35–45 s to complete. To ensure that the program runs smoothly, each loop must be completed in less than 1 min. As a result, we need the following minimum hardware requirements as shown in Table 6.

Software definition that is both minimal and optimal. The reason Prometheus and Grafana need the right version is because the automated dashboard required to execute this application is obsolete and will be removed in the future edition of Grafana. For that reason, as shown in Table 7, we utilize Grafana v6.0.2, which retains the programmed dashboard, and then Prometheus v2.8.0, which was published at the same time as Grafana v6.0.2.

4.2 Application guidelines

The following is a tutorial for running the application. On the home page, the top part displays language such as current server monitor and server status, which includes Maintenance, Up, and Down, as well as the total number of web servers associated with each of those statuses. Each status is explained as follows:

1. The grayed-out Maintenance Status shows that the webapp server is undergoing maintenance.
2. The Up Status, which is highlighted in green, shows that the webapp server is operational and accessible to users.
3. The Down Status, which is highlighted in red, shows that the webapp server is unavailable to users. This may also occur if the web server monitoring service does not get a response from the web application server after about 20 s (timeout).
4. The Danger Status, which is highlighted in yellow, shows that the webapp server's resource usage has exceeded the configured limit.

Figure 3 shows a list of web application servers, along with their current state. When we click on one of the servers, the system takes us to the server's details page.

Table 5: Black box testing

No.	Testing scenario	Expected result	Test result	Conclusion
1	Clicking the webapp server name will display server details page	Display server details page	Successfully displays the server details page	Success
2	If the internet connection is lost	Display text "*There is no internet connection*"	Successfully displaying text "*There is no internet connection*"	Success
3	Search range date	Displays the server details page according to the date and time in the specified range	Successfully displays detail server page according to the date and time in the specified range.	Success
4	Click text "*back*" on *server details page*	Display *home page*	Successfully displays home page	Success
5	Click server name on the server details page	Will reset date search range and display today's graph	Successfully displays today's graph	Success
6	If the webapp server is down, the webapp server will change color to red and the down status will increase by 1 on home page	*Webapp server will turn red and the down status will increase by 1 on home page*	Successfully changes the color to red and the down status increases by 1 on home page	Success
7	If the webapp server resources usage exceeds the limit, the server will turn yellow and the danger status will increase by 1 on home page	*Webapp server will change color to yellow and the danger status will increase by 1 on home page*	Successfully changes the color to yellow and the danger status increases by 1 on home page	Success
8	If webapp server monitor time is between maintenance start and maintenance end, the server will turn gray and the maintenance status will increase by 1 on home page	*Webapp server will turn gray and the maintenance status will increase by 1 on home page*	Successfully changed the color to gray and the maintenance status increased by 1 on home page	Success
9	If webapp server has crossed the alert limit several times in a row	*An email regarding the webapp server's resources usage will be sent to the registered email address*	Successfully sent an email alert to the registered email address	Success
10	If the use of the query execution time exceeds the specified max or mean limit	*The query is stored in the database*	Successfully stored the query in the database	Success

Table 6: Hardware specification

RAM	4 GB
Memory	100 MB
CPU	4 × 2.20

Table 7: Software Specification

Database	SQL SERVER 2012
JavaMelody	>v 1.77.0
Prometheus	v 2.8.0
Grafana	v 6.0.2

If there is no connection, system will not show anything but a text "There is no internet connection" as shown in Figure 4.

When the user clicks on one of the server's home page, the server details page will appear as shown in Figure 5. It includes the webapp's name in the upper middle area. The top right corner has a back button that takes us back to the main page. It includes an input area for the time filter underneath the name section.

Additionally, this page includes six graphs that illustrate the webapp server's resource consumption. The following is an explanation of the six graphs.

1. Active Threads. These are used to refer to the processes that are now being executed on the webapp server.
2. System CPU. The system CPU statistic indicates the server's CPU utilization in terms of percentage. The increased use suggests that we either need to optimize our code and queries to reduce CPU usage, or we need to update our hardware.
3. Musculoskeletal Memory. This is used to indicate the Operating System's (OS) memory consumption.
4. Utilized Memory. This refers to the amount of Random Access Memory (RAM) used by the server.
5. Waste Disposal. This is a procedure that is used to eliminate unwanted objects. This is part of the garbage collector that is a component of the Java programming language that runs automatically until the application is terminated. This graph is necessary to determine if there is a memory leak, which may result in an error out of memory.
6. Free Drive Space. The term "Free Disk Space" refers to the leftover unused space on the server's hard disk. The lower the graph becomes, the more likely it is that we need to delete files that are no longer needed, such as logs, or transfer the file to another disk.

We may display graph data for a certain time period by entering the date, hour, and minutes in the From and End sections and then clicking the Apply button. Here is an example of a graph over a 30 min period as shown on Figure 6.

4.3 Metrics page (API)

This page provides data about metrics in order for the Prometheus to access the API and create a graph that the Prometheus can read.

While the system CPU and garbage collector are measured in percentages, utilized memory, physical memory, and free disk space are measured in bytes. When data are received, it will measure using the current thread's real

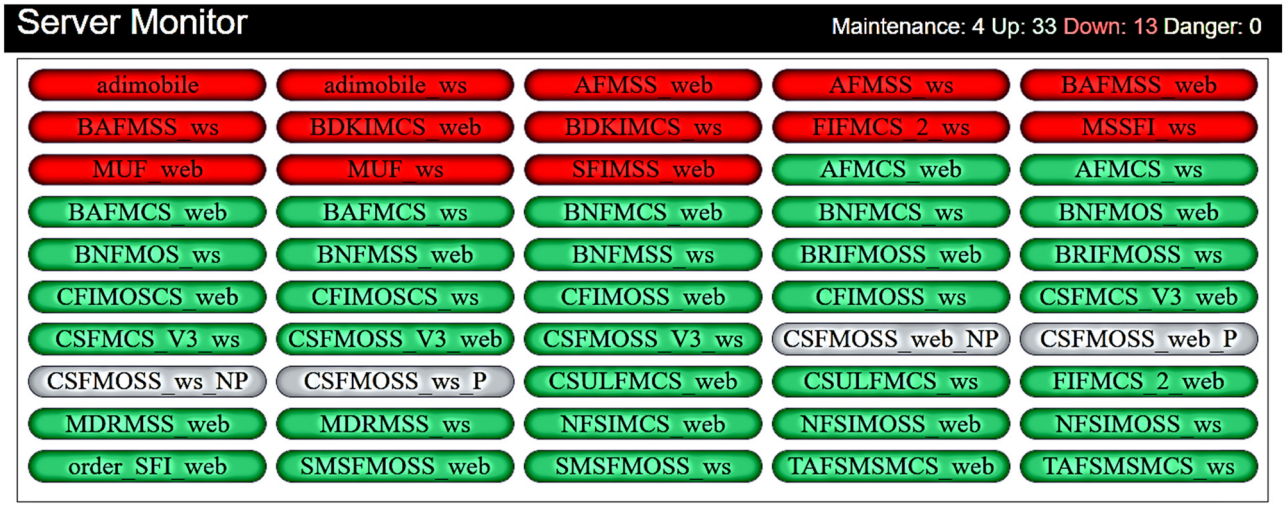


Figure 3: List of web application servers.

Figure 4: Home page with no internet connection.

Figure 5: Time filter period input example.

use thread. Grafana will utilize the Prometheus API to generate a graph that can be shown in web app server monitoring once the Prometheus app creates the graph.

Addition of a List Server

If the user wishes to add or delete a web application server, they may modify the JSON data that the app can access. This approach is preferred since it is more comprehensible and adaptable.

To add or delete a server from the home page, the user may modify the object data in the JSON file using the following syntax.

```
{
"Server": {
        "name": "adimobile",
          "url": "https://www.xxx.com/monitoring",
          "servertype": "webapp",
          "username": "username",
          "password": "password",
          "sqlmean": "6500",
          "sqlmax": "7500",
          "alertThreads": "10",
          "alertFreedisk": "150000000000",
          "alertGcmax": "5",
          "alertGcmean": "3",
          "alertSystemCPU": "50",
          "alertPhysicalMemory": "16000000000",
          "alerttime": "5",
          "alertDowntime": "120",
          "maintenancestart": "00.00",
          "maintenanceend": "00.00"
        }
}
```

Explanation of the above JSON.

Name

Name that will be displayed on the webapp server monitor.

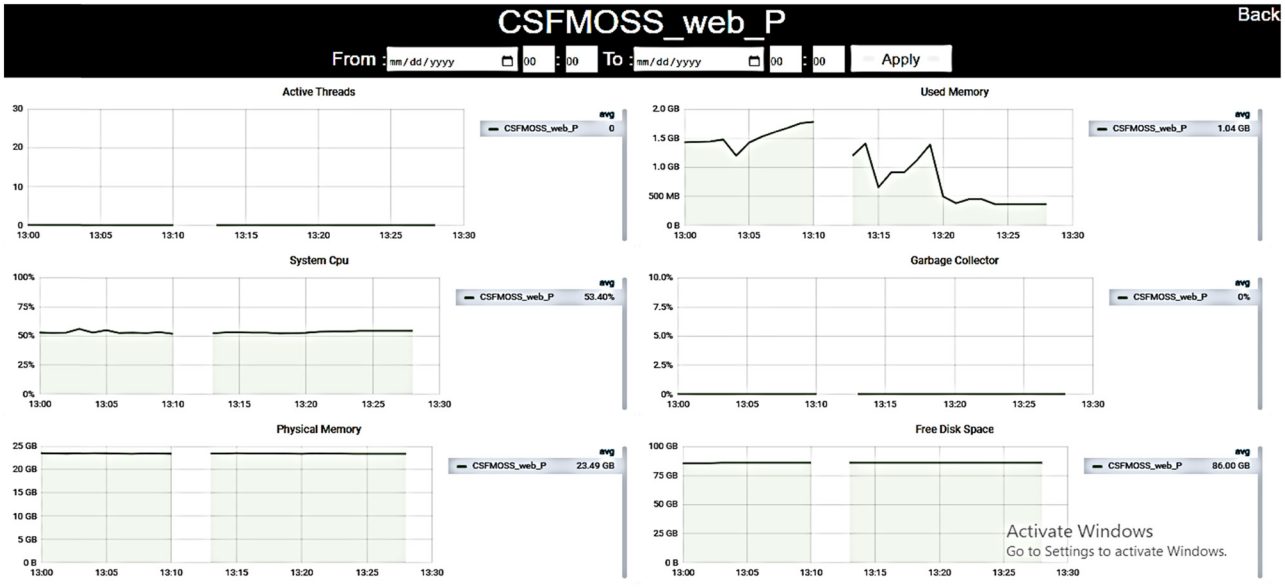


Figure 6: Server details page with a range of 30 min.

Url

App Url that will be monitored and has JavaMelody applied.

Servertype

Can be choose between *webapp*/*services*

Username

Username to access JavaMelody from the given url.

Password

Password to access JavaMelody from the given url.

Sqlmean

Average time (in seconds), which if there is a SQL query that runs above the limit of the sqlmean, then the system will record it in the database.

Sqlmax

Max time (in seconds), which if there is a SQL query that runs above the sqlmax, then the system will record it in the database.

AlertThreads

Maximum thread in which if there is thread usage that exceeds the limit of activeThreads, then the webapp status will change color to yellow or danger.

AlertFreedisk

If the current free disk (in bytes) is below the AlertFreeDiskValue, then the webapp status will change color to yellow or danger.

AlertGcmax

Maximum GC (in percentage). If GC usage volume exceeds alertGcmax, then the webapp status will change color to yellow or danger.

AlertGcmean

Average GC (in percentage), if GC average usage exceeds alertGcmean value, then the webapp status will change color to yellow or danger.

AlertSystemCPU

Maximum SystemCPU (in percentage), if SystemCPU usage exceeds alertSystemCPU value, then the web app status will change color to yellow or danger.

AlertPhysicalMemory

Maximum PhysicalMemory (in bytes), if PhysicalMemory usage exceeds alertPhysicalMemory value, then the webapp status will change color to yellow or danger.

Alerttime

if there is a webapp server that exceeds n number times alert in a row, then the system will send notification/email to the production team to check the server condition later.

AlertDowntime

If email has already been sent to the receiver, then even if it exceeds n number times alert, the email will not be sent, except if the time period from the last email sent to current time has exceeded AlertDownTime.

This will minimize spam email to the team if the server is continuously down and exceed alert limit

Maintenancestart

When the maintenance will start, if the current time is in the range of maintenancestart and maintenanceend, then the webapp status will change color to gray or maintenance.

Maintenanceend

When the maintenance will end, if the current time is in the range of maintenancestart and maintenanceend, then the webapp status will change color to gray or maintenance.

4.4 Email alert

This feature is used to notify the production team if there is a server which has status of down or danger for n times in a row.

1. Server Down Alert

   Email will be sent to the recipient when the webapp server cannot be accessed by server monitoring or experienced timeout as shown in Figure 7.
2. Query Alert

   Email will be sent to the recipient when there is a query that exceeds mean or max time as shown in Figure 8.
3. Alert Free Disk Space email will be sent to the recipient when the free disk space is below the settled parameter. *Alert GC Mean, GC Max, Physical Memory, System CPU,*

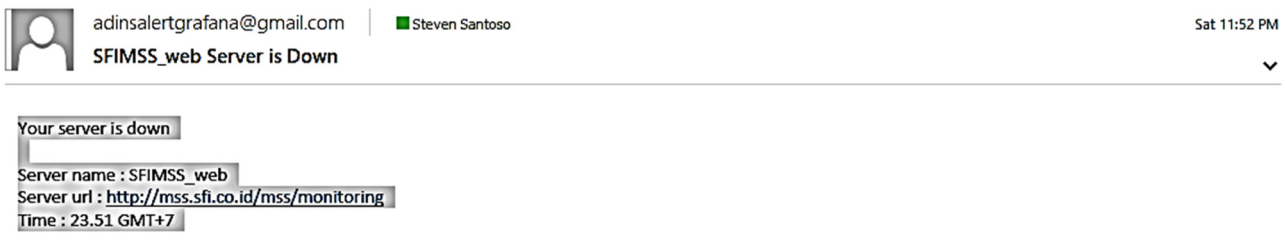


Figure 7: Alert server down example.

Your sql has some problems

Server name : BAFMCS_web
Server url : http://202.158.47.37:9200/bafmcs/monitoring

1.
Problem : Mean time query, Max time query,
Mean Time : 11476 ms
Mean Time Threshold : 3800 ms
Max Time : 84650 ms
Max Time Threshold : 3800 ms

Query :
SELECT UUID_TASK_H, CUSTOMER_NAME, AGREEMENT_NO, BRANCH_NAME, ASSIGN_DATE, SUBMIT_DATE, FULL_NAME, STATUS_TASK_DESC, SEND_DATE, FLAG FROM (SELECT a.UUID_TASK_H, a.CUSTOMER_NAME, a.AGREEMENT_NO, a.BRANCH_NAME, a.ASSIGN_DATE, a.SUBMIT_DATE, a.FULL_NAME, a.STATUS_TASK_DESC, SEND_DATE, FLAG, ROW_NUMBER() OVER(ORDER BY rownum) AS recnum FROM (SELECT b.UUID_TASK_H, b.CUSTOMER_NAME, b.AGREEMENT_NO, b.BRANCH_NAME, b.ASSIGN_DATE, b.SUBMIT_DATE, b.FULL_NAME, b.STATUS_TASK_DESC, b.SEND_DATE, b.FLAG, ? as FLAGORDER, ROW_NUMBER() OVER (ORDER BY b.ASSIGN_DATE DESC) as rownum FROM (SELECT trth.UUID_TASK_H, isnull(trth.CUSTOMER_NAME,'-') as CUSTOMER_NAME, isnull(trth.AGREEMENT_NO,'-') as

Figure 8: Alert query example.

and Threads have the same format as Alert Free Disk Space as shown in Figure 9.

Information
Time: 00:38 GMT + 7
Metrics: 76.79GB
Limit: 150GB

4.5 Database

In the database, there are a total of 6 tables used. Name and usage of each table is as follows.

☐ Email

Table is used to send email to recipients if there is a webapp server that experiences down or resource

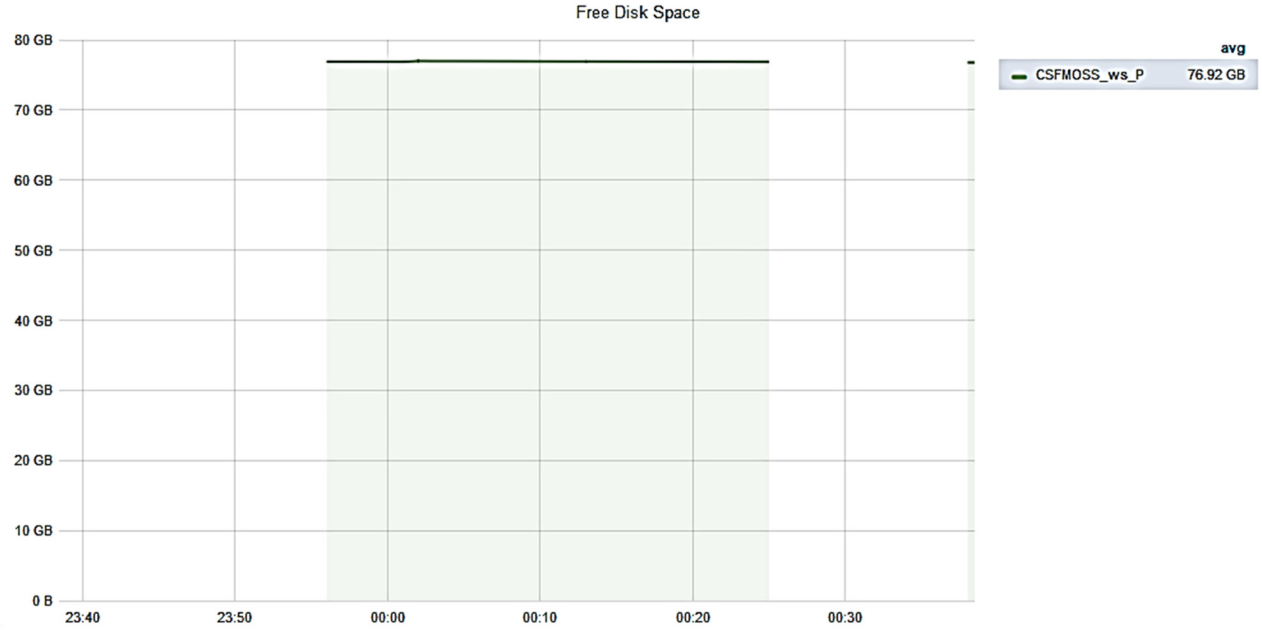


Figure 9: Alert free disk space email example.

usage that exceeds the limit. This table has 4 columns. which is:

1. Emailid
   Emailid is the *primary key* for the table.
2. Email_to
   Email_to is the email of the recipient. If there is more than 1 recipient, we can use semi colon (;) as a separator. For example if we want to send email to two recipient: aaa@aaa.com;bbb@bbbb.com
3. Email_cc
   Email_cc is the email of the recipient that will be given *carbon copy*. If there is more than 1 recipient, we can use semi colon (;) as a separator. For example if we want to send email to two recipient: aaa@aaa.com;bbb@bbbb.com
4. Is_active
   To mark if the email_to or email_cc is still active, if not, email will not be sent.

☐ Path data

PathData table has 2 columns, which is:

1. Directory
   Location in which the file will be read.
2. FileType
   There are 2 rows for this table containing JSON and img
   a. JSON: Path where JSON file is located.
   b. Img: Path where image is used for attachment in the email.

☐ Query data

This table contains 8 columns as follows:

1. QueryID (PK)
2. ClientCode
3. URL
4. QueryString: Query that is running in the database.
5. MeanTime: Average time needed to execute above QueryString.
6. MaxTime: Maximum time needed to execute QueryString above.
7. FlagQuery: Can be 0 or 1. 0 means there is no query LIKE.
8. TimeStamp: Exact time when QueryString taken.

☐ RV_ReportV2

On Table RV_ReportV2 there are 7 columns:

1. ID
2. *ClientCod*
3. *Max_Memory*
4. *Max_Thread*
5. *Max_TPM*
6. *Time_Stamp*
7. *Max_GC*

☐ RV_ReportSummary

RV_ReportSummary Table contains daily summary of resource usage from one webapp server that is created by stored procedure. There are eight columns of this table which is time data collected, name of server, avg memory, max memory, max thread, avg TPM, max TPM, and max GC on that day.

☐ Utilities

In *Utilities Table*, there are four columns as follows:

1. ID
   Primary key of Utilities Table.
2. Name
   There are 7 data for this column which is:
   Sql_start_check*:* start hour to check SQL query.
   Sql_pause_check: interval in hour between each check.
   Sql_end_check: end hour to check SQL query.
   Timeout: ping duration to the *webapp server* (ms)
   Send_sql_email: 1 or 0 (1 means email will be sent if there is SQL query that exceeds the limit)
   Send_email_down: 1 or 0 (1 means email will be sent if there is *webapp server* which experiences *down time*)
   Alert_count_threshold: number of times webapp server down is required before sending email.
3. Value
   Value of the *name*. To see the example of the value, one can refer to the name section above.
4. Description
   Description of the name utilities. To see the example of the value, one can refer to the name section above.

4.6 Comparison evaluation between JavaMelody and webapp server monitoring

Table 8 shows the comparison data between JavaMelody that is taken manually every day and web server monitoring for usage memory in Megabytes and percentage. For percentage we can use the formula as follows:

a. If Manual ≤ Webapp monitoring, then:

$$\text{Percentage} = \frac{\text{Manual}}{\text{Webapp monitoring}} \times 100\%$$

b. If Webapp Monitoring < Manual, then:

$$\text{Percentage} = \frac{\text{Webapp monitoring}}{\text{Manual}} \times \quad 100\%$$

Table 8: The comparison data between JavaMelody and webapp server monitoring

	18 June 2021				
	Avg memory	**Max memory**	**Max Thread**	**Avg TPM**	**Max TPM**
Server 1 Webapp					
Manual	2,500	5,900	8	388	1,643
Server monitoring	2,402	5,896	11	393	1,740
Diff percentage	96.08	99.94	72.73	98.73	94.43
19 June 2021					
Manual	2,000	3,900	9	194	1,329
Server monitoring	1,955	3,882	10	194	1,339
Diff percentage	97.74	99.54	90.00	100	99.25
20 June 2021					
Manual	2,100	5,000	10	293	1,272
Server monitoring	2,137	5,016	12	294	1,326
Diff percentage (%)	98.27	99.69	83.33	99.66	95.93
Server 1 Webservices					
18 June 2021					
Manual	1,500	5,000	24	92	574
Server monitoring	1308.534	5892.933	53	44	572
Diff percentage	87	85	45	48	100
19 June 2021					
Manual	1,200	3,800	26	69	316
Server monitoring	1114.839	3619.986	34	37	429
Diff percentage	93	95	76	54	74
20 June 2021					
Manual	1,200	4,000	21	94	474
Server monitoring	1053.902	5008.356	48	27	338
Diff percentage	88	80	44	29	71
Server 2 Webapp					
18 June 2021					
Manual	372	578	1	0	29
Server monitoring	371.6685	581.0834	1	0	32
Diff percentage	100	99	100	100	91
19 June 2021					
Manual	368	549	0	0	1
Server monitoring	365.5011	552.3373	0	0	1
Diff percentage	99	99	100	100	100
20 June 2021					
Manual	361	542	1	0	29
Server monitoring	361.4444	545.1581	1	0	33
Diff percentage	100	99	100	100	88
Server 2 Webservices					
18 June 2021					
Manual	372	568	15	6	182
Server monitoring	371.1034	570.1644	15	6	188
Diff percentage	100	100	100	100	97
19 June 2021					
Manual	367	550	14	7	184
Server monitoring	366.5338	553.7928	14	6	201
Diff percentage	100	99	100	86	92
20 June 2021					
Manual	361	544	14	3	141
Server monitoring	361.4459	547.7883	14	3	141
Diff percentage	100	99	100	100	100

5 Conclusion

Web application server monitoring has been made in accordance with the request and has been done as well as possible. Web applications are built by prioritizing data accuracy and ease of use of web applications. The web application server monitoring can be concluded as follows:

- Web application server monitoring has a simple user interface, making it easy to use and understand.
- Web application server monitoring can help ease the work of the production team when they want to do monthly web server maintenance.
- Web application server monitoring can provide information on web server power usage directly with good accuracy.

This monitoring web server application also has several shortcomings, namely:

- If the web server client is down or timeout, this will make accuracy worse because if the web server is down or got request timeout then the resource data entered into the database has a value of 0.
- Custom made User Interface (UI).
- There is no login feature so that it can be changed by anyone who can access the office's internal server.

For further research and development on the following topics, the authors have got some suggestions from users who have tried this application, including:

- If the client's web server is down, the data does not need to be stored in the database so it will improve accuracy.
- User Interface (UI) has been improved because currently on the home page there are only text and shapes.

Acknowledgments: This research is supported by Department of Information Systems Management, Bina Nusantara University.

Conflict of interest: The authors declare no conflicts of interest.

References

[1] Argaez Enrique De. World internet usage and population statistics; 2020. https://www.internetworldstats.com/stats.htm. (accessed on 29 June 2020).

[2] Asosiasi Penyelengara Jasa Internet Indonesia. Laporan Survei Internet APJII 2019 – 2020 [Q2]; 2020. https://apjii.or.id/survei. (accessed on 22 July 2021).

[3] Ray E. Pengembangan Aplikasi Monitoring Server Berbasis Mobile Web Dengan Sistem Notifikasi Email; 2015. http://repository.uinjkt.ac.id/dspace/bitstream/123456789/28103/1/ENDANG%20RAY-FST.pdf. (accessed on 07 March 2021).

[4] Kusuma FI. Perancangan Sistem Monitoring Perangkat Jaringan Berbasis SNMP; 2015. http://eprints.ums.ac.id/38600/27/Naskah-Publikasiku.pdf. (accessed on 07 March 2021).

[5] Prabawati A. Analisis Dan Perancangan Pemantau Jaringan Server Menggunakan PHP, SNMP Dan SMS Gateway Pada PT. PLN (Persero) Rayon Ponogoro; 2013. http://eprints.umpo.ac.id/495/2/KusumaHAL%20DEPAN%2C%20BAB%20I%2C%20DAFPUS%20anggar.pdf. (accessed on 07 March 2021).

[6] Wijayanto D. Aplikasi Monitoring Perangkat Dan Aktivitas Pengguna Pada Jaringan Menggunakan Protocol SNMP Dan Squid Proxy; 2016. https://core.ac.uk/download/151235843.pdf. (accessed on 19 March 2021).

[7] Fanggidae AM, Hermawan H, Pratiwi HI. Sistem Monitoring Server Dengan Menggunakan SNMP; 2019. http://www.ojs.upj.ac.id/index.php/journal_widya/article/view/218. (accessed on 20 July 2021).

[8] Ismaidi MA. Pengembangan Aplikasi monitoring performa server menggunakan simple network management Protocol; 2013. http://etd.repository.ugm.ac.id/penelitian/detail/65035. (accessed on 19 July 2021).

[9] Mathapati V. Performance analysis of system resources by server monitoring; 2013. http://www.ijirset.com/upload/july/30A_PERFORMANCE.pdf. (accessed on 19 July 2021).

[10] Vusvyta K. Design and implementation fast response system monitoring server using simple network management protocol (SNMP); 2017. http://edocs.ilkom.unsri.ac.id/1654/1/TASK2_Manajemen%20Jaringan_SNMP.pdf. (accessed on 07 March 2021).

[11] Susilawati S, Fitzgerald J, Renaldy C. Analisis dan Perancangan Aplikasi Traffic Monitoring Server Menggunakan SMS Pada PT. Anugrah Catur Abadi; 2008. http://eprints.binus.ac.id/2665/1/2008-2-00210-IF%20Abstrak.pdf. (accessed on 19 July 2021).

[12] Luan N. Aplikasi Monitoring Jaringan Berbasis Web Dengan Notifikasi Email; 2019. http://eprints.mercubuana-yogya.ac.id/5356/. (accessed on 19 July 2021).

[13] Ku P. 14 Reasons to Choose Kanban for Software Development; 2018. https://medium.com/hygger-io/14-reasons-to-choose-kanban-for-software-development-d3ddb420d273. (accessed on 28 March 2021).

[14] JavaMelody. JavaMelody: monitoring of JavaEE applications; 2019. https://github.com/javamelody/javamelody/wiki. (accessed on 09 August 2020).

[15] Grafana. What is Grafana. https://grafana.com/docs/grafana/latest/getting-started/what-is-grafana/. (accessed on 09 August 2020).