# Back-End Website Development for Iot-Based Automated Water Dissolved Oxygen Control

**Habib Irfan Mahaasin[1], Purba Daru Kusuma[2], Faisal Candrasyah Hasibuan[3]**
[1,2,3]Department of Computer Engineering, School of Electrical Engineering, Telkom University, Indonesia

**Article Info**

***Article history:***

Received April 14, 2025
Revised April 21, 2025
Accepted April 21, 2025

***Keywords:***

Communciation Data
HTTP API
Internet of Things
Waterfall Method
Webhook

## ABSTRACT

The process of fish farming will never be separated from dissolved oxygen level control devices such as aerators so that the cultured fish can grow and develop optimally. This is quite troublesome for fish farmers, especially for people who often leave their cultivation sites, so they cannot monitor directly. Therefore, a platform for remote monitoring is needed. This journal focuses on designing and developing a back-end system, one of which functions as a data communication medium from the IOT antares platform to the website built so that the website can be used to monitor and control IOT devices using the waterfall development method and HTTP communication protocol. this method and communication protocol were chosen because they are currently quite popular and very flexible to use. The results of this research show that data communication carried out by the server with Antares using the GO programming language can run well and quite efficiently. The system built is also able to accommodate several requests from users simultaneously with an average response time of 60ms, while the exchange of IOT device data with the website has an average response time of 3 seconds.

***Corresponding Author:***

Habib Irfan Mahaasin
Department of Computer Engineering
School of Electrical Engineering, Telkom University
Bandung, Indonesia
Email: habibmahaasin@student.telkomuniversity.ac.id

## 1. INTRODUCTION

The development of technology today is very rapid. Various tools/systems in supporting human work have now begun to be integrated with the internet, commonly known as the internet of things. The Internet of Things (IoT) is a framework that gives every object a digital identity and online presence. It aspires to provide new services and applications that connect the real and virtual worlds, with machine-to-machine (M2M) connections serving as the foundational form of communication for interactions between devices and cloud-based apps [1]. One application of the IOT concept is in the field of fisheries, where IOT can be applied as a supporting tool for monitoring and controlling various things, one of which is dissolved oxygen levels. Using the sensor, the system continually checks the dissolved oxygen levels and feeds back to the controller. The controller adjusts the oxygen supply as necessary if the measured values depart from the required range [2]. To support the system, a platform is needed so that monitoring and controlling can be done remotely. one of the flexible platforms to do this is the website.

In making a website or a IOT system, one of the most important things is the back end. A website's back end controls servers, databases, and procedures that are invisible to users. Data, automation, scalability, security, and other issues are all handled. In order to provide information effectively, components including servers, databases, and APIs must cooperate. The code that enables hidden site components is created by back-end development, also known as server-side development, and is overseen by backend engineers [3]. In the scope of IoT System, IOT backend service providers run the gateway infrastructures that facilitate IoT device functionality. They make it easier for IoT devices, internal systems, and perhaps other platforms to share data [4]. In this research, the backend is used to facilitate communication between the system built and the IOT Platform used, where the IOT platform is Antares.

As a result, the goal of this project is to create a back end system that attempts to facilitate data interchange between IoT devices and servers using Antares as an IOT platform/middleware, either by implementing the HTTP API or HTTP Webhook communication protocol. As a result of the data communication, the website will have features including those for controlling IOT devices and monitoring their data. The implementation of Antares as an IOT Platform/Middleware also requires to allow the data produced by the IOT device may be stored independently in the server database and make data processing on the website to be constructed more simple.

## 2. METHOD

The research topic focuses on the development stage of data communication on the IOT Platform Antares with a server using HTTP as a communication protocol. The method used is the waterfall system development method. Waterfall is one of the software development methods or also known as the Software Development Life Cycle. It is called waterfall because the development model is analogous to a waterfall, where each stage is done sequentially from top to bottom. The waterfall model has a number of steps, including modeling, modeling, communication, and construction. Before going on to the next step, each one must be finished [5].

### 2.1. System Requirements Analysis

The system built consists of two parts. The IOT architecture is the first component of the system that has been constructed. Sensing Layer, Network Layer, Middleware, and Application Layer are the four main layers of the IOT architecture [6], while Model, Repository, Service, and Handler are the four main layers of the back end architecture, which follows Uncle Bob's concept of Clean Architecture [7]. in the development of the back end, the programming language used is the GO programming language with the GIN framework and the GORM library. The server software communication protocol with Antares IOT Platform used is a combination of HTTP API for POST or sending data by the server to Antares. While the process of sending data from Antares to the server will use HTTP Webhook through the created endpoint. The process can be seen in Figure 1.

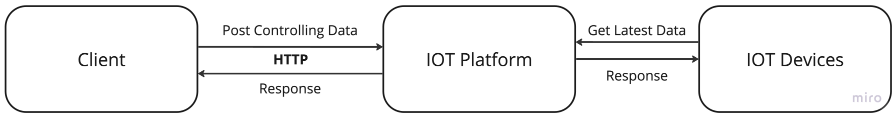


Figure 1. Overview of How the System Works

### 2.2. System Design

The next stage in the waterfall method is the system design process. Design is done to help provide a complete picture of what to do, make it flow, look and be in. in system design there are three main designs that are built, such as IOT system design, back-end system design and database design. the first system design is the main picture of the IOT system being built. in Figure 2 is the design of the IOT system that was built and is divided into several layers.

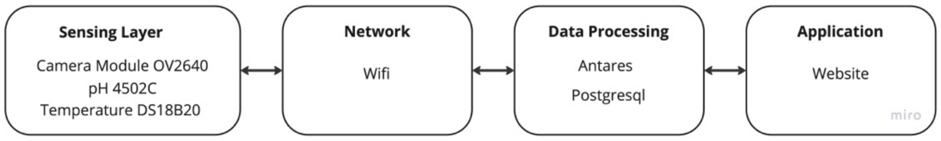


Figure 2. System architecture IOT

The next design system is the main description of the back end of the system which is divided into 2 processes, namely the implementation of the HTTP API as a medium for server communication with the IOT platform, the second is the implementation of HTTP Webhook for communication from the IOT Platform to the server. Figures 3 and 4 are the process flow diagrams of the two processes.

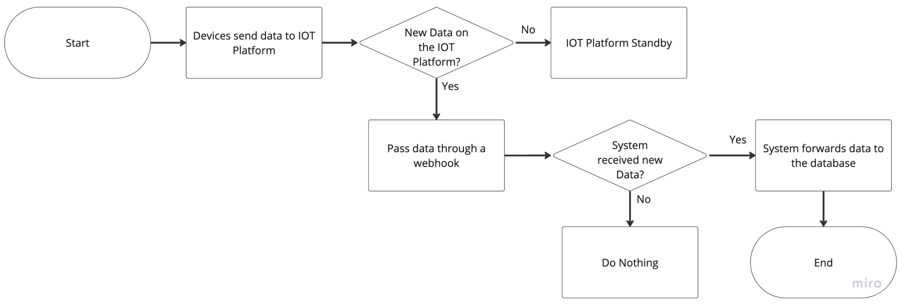

Start
Devices send data to IOT Platform
New Data on the IOT Platform?
No
IOT Platform Standby
Yes
Pass data through a webhook
System received new Data?
Yes
System forwards data to the database
No
Do Nothing
End
miro

Figure 3. Flowchart Webhook

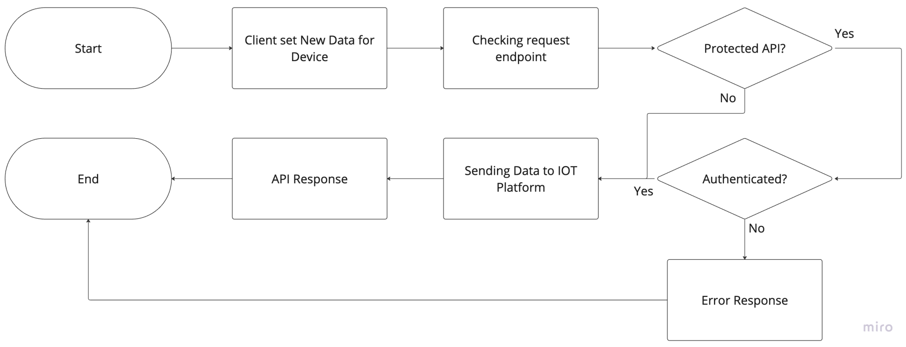

Start
Client set New Data for Device
Checking request endpoint
Protected API?
Yes
No
Authenticated?
Yes
No
Sending Data to IOT Platform
API Response
End
Error Response
miro

Figure 4. Flowchart POST API

The final step is database design, or ERD (Entity Relationship Diagram). In its application, there are many tables that are built, but the role of the back end in exchanging data between the systemand the IOT Platform only involves two tables, namely the history table and the device table. in Figure 5 is the ERD design that was made.

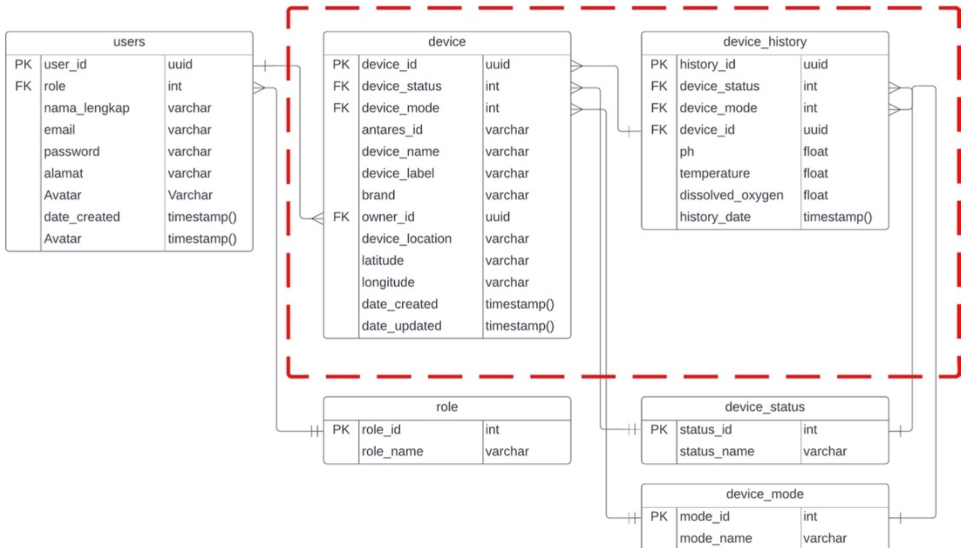
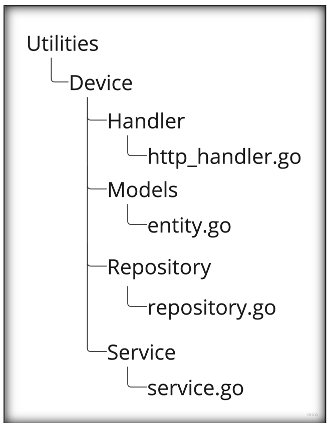


Figure 5. Entity Relationship Diagram

## 2.3. Implementation

In implementing the back end system, the first implementation is to implement the planned architecture using 4 layers that are interconnected and have their respective functions with the results of the directory arrangement as shown in Figure 6 below.



```
Utilities
  └─Device
      ├─Handler
      │   └─http_handler.go
      ├─Models
      │   └─entity.go
      ├─Repository
      │   └─repository.go
      └─Service
          └─service.go
```

Figure 6. Work Directory

The second implementation is the implementation of the POST process in sending data to the IOT Platform which can be seen in table 1. This function is one of the functions that is useful for connecting client communication with the IOT platform server to get the desired communication results before being forwarded by the IOT platform to the IOT device. The data format used is JSON. Where JSON itself is a format that is widely preferred because it is easy to understand, lightweight, concise, and shows structured data based on JavaScript object syntax [8]. However, Antares has its own request and response format so that it must adjust the format provided so that the data sent can be conveyed properly. The format can be seen in table 2.

Table 1. Source code POST API Antares

| Source Code |
|---|

```
func (r *repository) PostControlAntares(InputData string) error {
   data := "{json format data}"

   timeout := time.Duration(5 * time.Second)
   client := http.Client{
      Timeout: timeout,
   }

   req, err := http.NewRequest("POST", r.conf.App.Antares_url+"/cnt-"+antares_id, bytes.NewBuffer([]byte(data)))

   req.Header.Set("X-M2M-Origin", token)
   req.Header.Set("Content-Type", "application/json;ty=4")
   req.Header.Set("Accept", "application/json")

   resp, err := client.Do(req)
   _, err = ioutil.ReadAll(resp.Body)

   return nil
}
```

Table 2. API Requirement

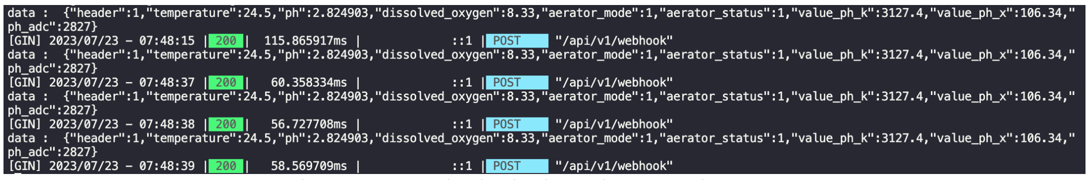
| Variabel | Description |
|---|---|
| Endpoint | https://{application_url}/control/:data |
| Method | POST |
| Header | Token/Authentication, Content-Type, Accept format |
| Input Format | {<br> "m2m:cin": {<br>  "con": "{\"key1\":\"integer-value\", \"keyN\":\"string-value\"}"<br>  }<br>} |
| Response | {<br>  "m2m:cin": {<br>    "rn": "cin_893238184",<br>    ….<br>    "con": "{\"key1\":integer-value, \"keyN\":\"valueN\"}"<br>  }<br>} |

The next implementation is webhook implementation. This method was chosen because the Antares IOT platform already supports the use of webhook subscriptions which will basically make software communication more efficient. What needs to be done is to prepare a function/endpoint that will be used as a data sending/notification address when there is new data or data changes on the IOT Platform as in table 3 below.

Table 3. Create End Point Golang

| Source Code |
|---|
| api := router.Group("/api/v1")<br>api.POST("/webhook", deviceHandlerV1.SubscribeWebhook) |

If there is new data/notification on the created endpoint, the system response on the terminal will look like figure 7 below with JSON data format.



```
data :  {"header":1,"temperature":24.5,"ph":2.824903,"dissolved_oxygen":8.33,"aerator_mode":1,"aerator_status":1,"value_ph_k":3127.4,"value_ph_x":106.34,"
ph_adc":2827}
[GIN] 2023/07/23 - 07:48:15 | 200 |  115.865917ms |             ::1 | POST     "/api/v1/webhook"
data :  {"header":1,"temperature":24.5,"ph":2.824903,"dissolved_oxygen":8.33,"aerator_mode":1,"aerator_status":1,"value_ph_k":3127.4,"value_ph_x":106.34,"
ph_adc":2827}
[GIN] 2023/07/23 - 07:48:37 | 200 |   60.358334ms |             ::1 | POST     "/api/v1/webhook"
data :  {"header":1,"temperature":24.5,"ph":2.824903,"dissolved_oxygen":8.33,"aerator_mode":1,"aerator_status":1,"value_ph_k":3127.4,"value_ph_x":106.34,"
ph_adc":2827}
[GIN] 2023/07/23 - 07:48:38 | 200 |   56.727708ms |             ::1 | POST     "/api/v1/webhook"
data :  {"header":1,"temperature":24.5,"ph":2.824903,"dissolved_oxygen":8.33,"aerator_mode":1,"aerator_status":1,"value_ph_k":3127.4,"value_ph_x":106.34,"
ph_adc":2827}
[GIN] 2023/07/23 - 07:48:39 | 200 |   58.569709ms |             ::1 | POST     "/api/v1/webhook"
```

Figure 7. Terminal Display when Receives Data

However, because Antares itself has its own JSON structure, it is necessary to break down the JSON response by doing nested structs in models that are quite complicated in the GO programming language, in order to receive the results according to what is sent by the sensor to the IOT Platform. With the final result of the HTTP webhook function as in the table 4 below:

Table 4. Webhook Details

| Variabel | Description |
|---|---|
| Endpoint | https://{application_url}/api/v1/webhook |
| Method | POST |
| Header | None |
| Input Format | JSON with Antares format |

### 2.4. Testing

The testing method that will be used to test the functions that have been made is by conducting direct testing, unit testing, and load testing. These methods need to be done so that they can be used as a reference if an error occurs either in the development process or during deployment. Some of the functions of testing are so that it can be used as verification of a function if the function is 100% running, to find out and improve performance, and make things that are built safer.

### 2.5. Maintenance

The stage in the waterfall method is the maintenance stage. In terms of ease of maintenance, the system made will be easier to maintain because the main architecture of the system already uses clean architecture, where if an error occurs it can be easily tracked so that it makes it easier for developers to solve the problems that occur. Maintenance can be done every month so that the performance of software communication can be seen properly.

## 3. RESULTS AND DISCUSSION

After doing research and implementing the planned things, to find out the results of the implementation, testing needs to be done. Because the Back end is one part of this system, testing can be done in many ways. Starting from the innermost testing, namely unit testing to the outer part, namely load testing. Unit testing is a software testing method used by testing the smallest unit of code. The purpose of unit testing is to minimize errors or bugs in the system [9]. While load testing is a performance testing technique where the system response is measured in various load conditions. This test helps determine how the software handles when several users access the function simultaneously [10].

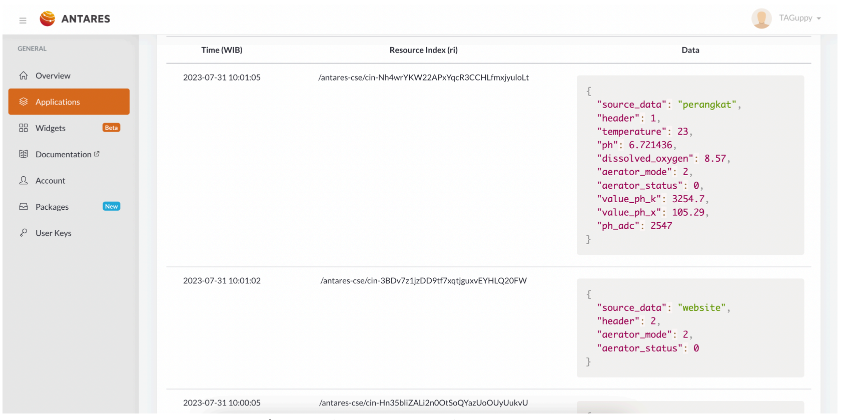
### 3.1. Result Code Coverage

Table 5. Code coverage

| No | Layers | Code Coverage |
|---|---|---|
| 1 | Handler | 100% |
| 2 | Service | 100% |
| 3 | Repository | 100% |

Based on table 5 above, the results of the back end system development have code coverage of 100% on each layer, so that all functions made can function or have the value as expected. These results were obtained after conducting unit testing using package testing from Golang.

### 3.2. Result Controlling IOT Devices



| Time (WIB) | Resource Index (ri) | Data |
|---|---|---|
| 2023-07-31 10:01:05 | /antares-cse/cin-Nh4wrYKW22APxYqcR3CCHLfmxjyuloLt | { "source_data": "perangkat", "header": 1, "temperature": 23, "ph": 6.721436, "dissolved_oxygen": 8.57, "aerator_mode": 2, "aerator_status": 0, "value_ph_k": 3254.7, "value_ph_x": 105.29, "ph_adc": 2547 } |
| 2023-07-31 10:01:02 | /antares-cse/cin-3BDv7z1jzDD9tf7xqtjguxvEYHLQ20FW | { "source_data": "website", "header": 2, "aerator_mode": 2, "aerator_status": 0 } |
| 2023-07-31 10:00:05 | /antares-cse/cin-Hn35bliZALi2n0OtSoQYazUoOUyUukvU | |

Figure 8. Data on Antares

Figure 8 is the result of the Antares dashboard when successfully receiving data from the website or device. Based on this figure, Antares is able or successful in receiving data from the website and the website successfully sends data with the contents of source_data, header, mode, and device status. So the Antares controlling or POST data function can function as expected.

### 3.3. Result Webhook Testing

Table 6. Result Webhook Testing

| No | Test details | Input | Expected Result | Actual | Status |
|---|---|---|---|---|---|
| 1 | Testing with input following the Antares format | JSON with Antares format | success | { "code": 200, "status": "success" } | Pass |
| 2 | Testing with input that doesn't comply with Antares format | blank | error | { "code": 220, "status": "error" } | Pass |

In table 6 are the results obtained after testing the webhook created. There are 2 test cases used, the first is to input JSON with a format in accordance with the Antares rules according to the documentation [11]. with expected result success and actual result success, which means the test was successfully carried out. For the second, the input is not in accordance with the format in the Antares documentation which results in expected result error and actual result also error. This means that this function is as expected.

### 3.4. Result Load Testing POST API and Webhook

The next testing is load testing. Load testing will be run on a local client so that there are no server differences by using a number of threads 20, and ramp up 2 second. which means that every 1 second there are 10 inputs entering simultaneously. The results of the testing scenario are as follows:

Table 7. Result Load Testing

| Label | Samples | Avg | Min | Max | Std. Dev. | Err | Throughput | Received kb/s | Sent kb/s | Avg Bytes |
|---|---|---|---|---|---|---|---|---|---|---|
| POST API | 20 | 61 | 0 | 67 | 2.95 | 0% | 10.2kb/s | 3.66 | 8.44 | 367 |
| Webhook | 20 | 58 | 0 | 64 | 2.12 | 0% | 10.2kb/s | 3.17 | 3.54 | 317 |

The test results with 20 threads that can be seen in table 7 above do not make the POST API or webhook function become an error, but the difference is seen in the process of sending data, incoming data and the average obtained. The POST API process is slightly slower because the process is carried out by the server against the IOT platform API, while the webhook is faster because the server used during testing is a local server. However, both have a difference that is not so far with an average response time of 60ms which means the performance is quite good.

### 3.5. Response Time Test Results

Table 8. Response time devices

| Time | Testing | Response Time (Second) |
|---|---|---|
| 02.58.03 | Testing 1 | 4.46 s |
| 02.59.33 | Testing 2 | 3.39 s |
| 03.01.23 | Testing 3 | 2.3 s |
| 03.02.33 | Testing 4 | 2.07 s |
| 03.04.13 | Testing 5 | 4.36 s |

Table 8 shows the results of tests carried out directly for several trials to find out how much the response time of the IOT device when controlling via the website with an average result of 3.31 seconds. These results are quite fluctuating because the response time obtained is strongly influenced by the signal on the IOT device and the website.

## 4. CONCLUSION

Based on the results of the backend system development process, especially in the communication of the Antares IOT platform with the server made with the HTTP communication protocol, it can be concluded that the use of the HTTP protocol in the IOT scope is still very relevant, combined with the support of the IOT platform as middleware making the combination of software communication protocols used can vary, considering that the IOT platform can connect with more than one device. This is shown in the response time test results with an average API response time of 60ms and an IOT device response of 3.31 seconds. In addition, the results of testing functionality such as POST API and webhook with several test scenarios show that the GO programming language is able to attend the developed data communication with 100% code coverage. In the future, this research can be further developed by comparing other protocols such as MQTT with HTTP in the scope of IOT, especially from the services provided by the Antares IOT platform.

## REFERENCES

[1] R. A. Mouha, "Internet of Things (IoT)," *Journal of Data Analysis and Information Processing,* no. 9, pp. 77-101, 2021.

[2] Kaufman MH, Ghosh RN, Grate J, Shooltz DD, Freeman MJ, Ball TM, et al, "Dissolved oxygen sensor in an automated hyporheic sampling system reveals biogeochemical dynamics," *PLOS Water,* pp. 1-18, 2022.

[3] Coursera, "What Does a Back-End Developer Do?," Coursera, 16 June 2023. [Online]. Available: https://www.coursera.org/articles/back-end-developer. [Acesso em 12 August 2023].

[4] S. J. Saidi, S. Matic e O. Gasser, "Deep Dive into the IoT Backend Ecosystem," *Proceedings of the 22nd ACM Internet Measurement Conference,* pp. 1-16, 2022.

[5] A. Muntaheen, "The Introduction of the Waterfall Model," *American Journal of Computer Science and Engineering Survey,* vol. 9, no. 4, 2021.

[6] S. Jena, "Architecture of Internet of Things (IoT)," geeksforgeeks, [Online]. Available: https://www.geeksforgeeks.org/architecture-of-internet-of-things-iot/. [Acesso em 12 August 2023].

[7] Y. N. Nugroho, D. S. Kusumo e M. J. Alibasa, "Clean Architecture Implementation Impacts on Maintainability Aspect for Backend System Code Base," *2022 10th International Conference on Information and Communication Technology (ICoICT),* pp. 134-139, 2022.

[8] T. Lv, P. Yan e W. He, "Survey on JSON Data Modelling," *Journal of Physics: Conference Series,* 2018.

[9] A. Tosun, M. Ahmed, B. Turhan e N. Juristo, "On the Effectiveness of Unit Tests in Test-driven Development," *International Conference on Software and System Processes,* 2018.

[10] H. M. AlGhamdi, C.-P. Bezemer, W. Shang, A. E. Hassan e P. Flora, "Towards reducing the time needed for load testing," *Journal of Software: Evolution and Process,* 2020.

[11] Antares, "Data of Device HTTP," Antares, [Online]. Available: https://docs.antares.id/api-or-http/data-of-device. [Acesso em 12 August 2023].