E-ISSN: 2798-4664

Application of Microservice Architecture in the Development of Website-Based Drainage System for Rumbai Area (Case Study: PUPR Service Pekanbaru)

Deanira Fadrinaldi[1)], Shumaya Resty Ramadhani[2)], and Retno Tri Wahyuni[3)]
[1,2,3] *Politeknik Caltex, Pekanbaru, Indonesia*

E-mail: *2)shumaya@pcr.ac.id

Abstract: Flood has always been an annual problem that hit most of the areas in Indonesia including Pekanbaru. According to the city's official website, there are 375 flood-prone points, and several roads have poor drainage systems. This is what triggers a large number of flood and damaged drainage reports by the public community per day. The management of the public's complaint data will need a significant amount of time and effort from the community and government staff. Additionally, errors and the risk of data loss in archiving public complaints reports are prone to occur because the flood complaint data is kept in hardcopy in the file cabinet. However, information regarding the details of the drainage system is still not fully available in the PUPR Office database. Hence, this research made a system with microservice architecture which is divided into 4 services: a backend service to handle user authentication, a backend service to manage public community's complaint data, a backend service to provide drainage monitoring data, and a frontend admin service. This website application will be named DRAINIT. This web application will only have one user level, admin, and will be managed by the Pekanbaru PUPR service officer. According to the test that has been conducted, namely unit testing, integration testing, and user acceptance testing, this web application is thought to have successfully met the requirements requested by Pekanbaru's PUPR service which are to records public complaints reports, see GIS-based drainage network, and see drainage monitoring history. This study performed load testing for the performance test, and the results showed that the application could handle 19.3 requests per second.

Keywords: Drainage system, microservice architecture, website

1. Introduction

The rapid development of information technology that is accelerated by the presence of the internet has encouraged various fields of life to utilize this technology as optimally as possible. The use of the internet in government aspects encourages the realization of e-government, which is expected to bring benefits in empowering the community through increasing access to information, improving government services to the community, and improving government management that is more efficient and transparent.

The development of e-government is an effort to develop electronic-based government administration in improving the quality of public services effectively and efficiently (Presidential Instruction No. 3/2003 concerning National Policies and Strategies). The application of technology, especially information systems, will assist officers in carrying out their work by reducing their limitations. The use of computer-based information systems is also expected to improve employee performance.

Pekanbaru City has an area of 632.26 km2 [1]. Quoted from the city's official website, as has happened in Pekanbaru, there are 375 flood-prone points, and several roads have poor drainage systems and are not running well. This is what affects the number of flood reports per day. Based on an interview with Mr. Joko Sutiardi, S.T., who serves as the Head of Water Resources of the Pekanbaru City PUPR Office, the number of flood reports received is around 10-20 reports per day. Many community complaints about flooding seem to have been ignored due to the unavailability of an effective connection between the community and the government in dealing with these problems and also because of the relatively long duration of work (about six months). Suppose complaints are submitted using current procedures, namely through word of mouth directly, via telephone, or via Whatsapp. In that case, it will take up a lot of time and effort from the community and government staff to manage the public's complaint data. In addition, errors, and the risk of data loss in archiving public complaints reports are prone to occur because the storage of complaint data related to the flood disaster is stored in the form of hardcopy in the file cabinet.

Based on interviews with the PUPR Office, information regarding the details of this drainage is still not fully available in the PUPR Office database. In order to make it easier for the PUPR Service Officer to find out

information about drainage conditions in Rumbai District, a Geographic Information System (GIS) that can map drainage networks and can provide information about drainage details such as width, depth, and type of drainage is needed.

In addition, as a measure to prevent flooding, drainage conditions must be monitored to maintain their proper function. Therefore, the system that will be created will also fetch data that has been stored in an IoT device's database so that it can provide an early warning system. The IoT tool has been carried out in previous research conducted by Politeknik Caltex Riau student Atilah Afif Al Ghifari, titled "Design of Multi-Node Sensors in Drainage Monitoring Systems" by placing 1 IoT device each at Politeknik Caltex Riau and Rumbai's Athlete's House. The results given are in the form of water level data in centimeters and also flow rate in ml/s units. Yet the data from IoT devices is still displayed on a separate platform, namely, google data studio which of course would make it difficult for PUPR officers to view the data.

Therefore, to overcome the problems that has been described above, which are Communities and PUPR officers have difficulty communicating directly about reports of broken drainage or flooded points, the storage of complaint data related to the flood disaster is stored in the form of hardcopy in the file cabinet which causes high possibility of errors and the risk of data loss, PUPR officers have difficulty getting information on broken drainage points and flood points in the city of Pekanbaru, PUPR officers have difficulty in monitoring drainage systems in the city of Pekanbaru manually, and the PUPR Service itself does not yet have a GIS-based drainage map, a website-based system that combines report recording systems, drainage network mapping using GIS, and fetch drainage monitoring data stored in IoT devices database placed in Politeknik Caltex Riau and Rumbai's Athlete's House is needed. This website application will be named DRAINIT.

Currently there are 2 kinds of architecture in application development [2], namely monolithic and microservice. Monolithic architecture is an application development architecture design where all services from applications, program code, databases, and program views will be combined into one. This type of software architecture has weaknesses, namely performance will decrease as the application gets bigger, it is difficult to adapt to new technologies, the application gets bigger and more complex so it is difficult to understand, the update process will affect the entire application, and cannot use different programming languages. These weaknesses can be solved by implementing a microservice architecture. Microservice architecture is a software architecture that breaks down an application into several smaller services. The service that has been divided runs according to its unique function. This collection of services will communicate so that it becomes one large system unit. The advantages of this microservice architecture are that it has a small complexity, can develop multi-platform applications, each service can stand alone, the update process only occurs on the service that needs be to updated.

DRAINIT applications have a lot of complexity in the application development process. Moreover, this website application will also be integrated with a mobile application for the community and a mobile application for the yellow officer. Based on these needs, the right solution for developing DRAINIT applications is to use a microservice architecture. The microservice architecture can overcome a major complexity because each requirement in the DRAINIT system will be divided into small parts. Besides that, this architecture allows developers to be able to develop cross-platform applications, and use different programming language according to the desired service needs.

2. Methods

The methodologies applied in this research are:

1) Interview

Interviews were conducted with Mr. Joko Sutiardi, S.T. as Head of the Water Resources Division of the Pekanbaru PUPR Office on November 5, 2021 to identify problems that occur in the process of handling community reports, the process of monitoring the drainage mechanism as well as analyzing and deciding what will be the solution and how the problem will be solved.

Based on the interview, it was found that the process of handling damaged drainage and flood is still done manually. The reporting process starts from the community must first report the incident of damaged drainage or flooding to the local RT / RW then forwarded to a higher functional position until it reaches the PUPR office. After receiving the report, PUPR will assign an officer to check the scene and do report updates by using WhatsApp group until it is finally done.

2) Literature Review

The collections of references in this research come from journals, books, e-books, and articles that are correlated with this research topic.

1. Related Research

Research related to the monitoring system that provides an actual update has previously been carried out by Wantoro et al., 2021 [3]. The research is entitled "Sistem Monitoring Perawatan dan Perbaikan Fasilitas PT PLN (Studi Kasus: Kota Metro Lampung)" and using the prototype method. The system can provide information

Mapping Repair and Maintenance Facilities PT. PLN Area Metro is used as information retrieval for distribution and damage to the electricity substation area, bridging the rayon office and warehouse in terms of requesting spare parts, arranging maintenance and repair schedules for distribution substations of PT. PLN Area Metro makes it easier for PT. PLN Area Metro to print reports on maintenance and repair activities, as well as giving customers access to report disturbances that occur related to disturbances that occur in the customer's area.

Another scope of this research is a website-based geographic information system application. For that scope, Enriko Subnanda (2018) [4] has done a research titled "Pembangunan Aplikasi Web Sistem Informasi Geografis Lokasi Bencana Alam Tanah Longsor Dan Banjir Kecamatan Tanjung Raya" at Information Technology Faculty of Universitas Andalas. This application aims to make the local governments easier to provide accurate and fast information to the community and help the community to access disaster locations and find out the number of losses caused. This application is implemented using PHP, Javascript, and JSON programming languages.

2. Related Literature

a. Microservice Architecture

Microservice is a technology that breaks down a large application into several smaller services [5].

b. API

API stands for Application Programming Interface, which allows developers to integrate two parts of an application or with different applications simultaneously [6].

c. PHP

PHP is a programming language that is geared towards web development [7]. PHP code is usually processed on a web server by a PHP interpreter implemented as a module, a daemon or as a Common Gateway Interface (CGI) executable. On a web server, the result of the interpreted and executed PHP code – which may be any type of data, such as generated HTML or binary image data – would form the whole or part of an HTTP response [8].

d. Laravel

Laravel is an open-source PHP Framework [9]. Laravel utilizes the MVC(Model-View-Controller) design pattern and is highly robust and easy to understand, it reuses existing components from other frameworks to build web applications. The web application that results from this design is more structured and pragmatic.

e. Go

Go is an open-source programming language developed by Google [10].

3) Software Development Method

This website will be developed using Extreme Programming (XP) development model. The Extreme Programming software development process starts with planning, and all iterations consist of four basic phases in its life cycle: designing, coding, testing, and listening.

3. Result and Discussion

The implementation of the Extreme Programming (XP) software development method in developing DRAINIT consists of several iterations that are applied to build features on the system. Each of these iterations serves to accommodate the changes that the user wants to the system being built by the developer.

1) Planning

This activity begins by listening to stories from users or in XP called user stories. This activity aims to gather needs that allow developers to understand the business context for the system that is being developed. In this research, listening to user stories is carried out by conducting interviews with Mr. Joko Sutiardi, S.T., who serves as the Head of Water Resources of the Pekanbaru City PUPR Office.

Based on the interview results with Mr. Joko Sutiardi on November 5, 2021, as a source person directly related to the PUPR Service of Pekanbaru, the results of a needs analysis regarding the development of the DRAINIT web application was obtained. The result includes identifying who the user is and what the user needs to use the system. The user of this web application is a PUPR officer who already knows which drainage is the responsibility of the Pekanbaru City PUPR Service.

Table 1. User Story

ID	Summary	Description	*Acceptance Criteria*
US01	Login/Logout	As a PUPR service officer, I want to access the application securely, to	Before using the main features of this web application there must be a login

ID	Summary	Description	*Acceptance Criteria*
		protect my information.	screen. After the PUPR service officer has logged in, there is a logout menu.
US02	Manage drainages	As a PUPR service officer, I want to have access in managing drainages data so that I can create if there is new drainage data, update if there is unmatched data or delete if the drainage is gone.	There is a drainage management menu where the PUPR service officer can add, edit, and delete the drainage data.
US03	Manage broken drainage points	As a PUPR service officer, I want to have access in managing broken drainage point data so that I can create if there is new broken drainage data, or update if there is unmatched data.	There is a broken drainage management menu where the PUPR service officer can add, and edit the broken drainage data.
US04	Manage flood points	As a PUPR service officer, I want to have access in managing flood points data so that I can create if there is new flood point data, or update if there is unmatched data.	There is a flood point management menu where the PUPR service officer can add, and edit the flood point data.
US05	Verify public community report	As a PUPR service officer, I want to review the report so that I can verify if the report is real and a part of the Pekanbaru PUPR service responsibility area.	On the detail of the public community report page, there is information such as picture, location, and description. There is also verify button and refute button to verify the report.
US06	Assign yellow officer	As a PUPR service officer, I want to assign a yellow officer to each verified public report so that the report can start to be handled by yellow officer.	On the detail page of a verified report, there is a component to enter a yellow officer.
US07	Drainage map	As a PUPR service Officer, I want to see drainage map so that I can see the drainage map along with the brief information of it.	There is a drainage map menu to navigate the web into the drainage map page.
US08	Filtered drainage map	As a PUPR service Officer, I want to see drainage with specific criteria so that I can see which drainages are matched with the criteria.	There is a filter menu by category so that the components displayed on the drainage map will be filtered based on the selected criteria.
US10	Water level history	As a PUPR service Officer, I want to see the water level history of the drainage so that I can monitor the water level.	On the drainage detail page, there is a menu that navigates the web into the drainage's water level history page.

At this stage, a use case diagram is also modeled to describe the functional requirements of the DRAINIT website application. The user of the application is described as an actor associated with the use case as a symbol of the functional requirements that the actor can perform. In the use case diagram shown in Figure 1, there will only be one user, namely an employee of the Pekanbaru City PUPR Service.

E-ISSN: 2798-4664

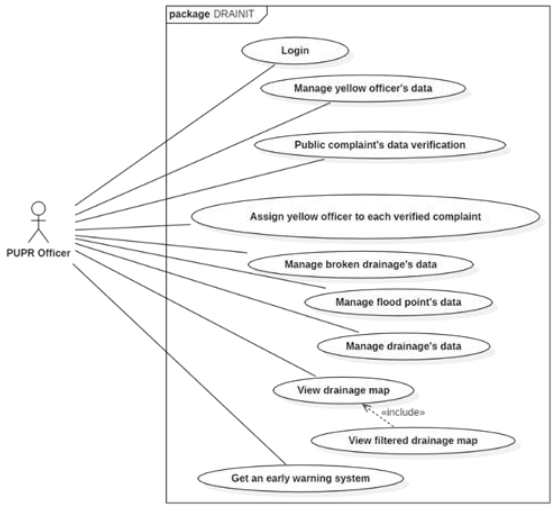

Figure 1. Use Case Diagram

The system that is going to be built in this research will use a microservice architecture. The microservice concept will divide the application into independent components (services) that have separate functions and data storage media according to their unique tasks. Therefore, after the use case is modeled, an analysis will be carried out first to break the application into several smaller services. Each service obtained from the results of the analysis will be built separately so that it can stand alone and be able to carry out their respective tasks.

Figure 2 shows the architectural design of the DRAINIT application. Each service has a different but integrated database. Based on XP, the construction of the entire DRAINIT system will be divided into several iterations according to the number of services to be built. The total iteration to be carried out is 4 iterations. The first iteration will build a UI design from the frontend admin service, the second iteration will build a user authentication service, the third iteration will build a complaint service, and the fourth iteration will build a drainage monitoring service and integrate the service in the previous iteration with the frontend admin service.

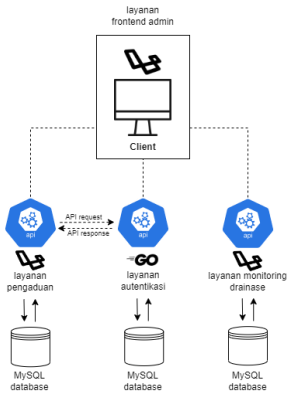

Figure 2. System's Architecture

2) Design

The design phase in the XP software development methodology advocated the use of class-responsibility-collaborator (CRC) cards as an effective mechanism for building object-oriented software. CRC cards identify and manage object-oriented classes relevant to the software development process per iteration.

Table 2 and 3 show the CRC cards of the user authentication service. Table 2 will contain the responsibilities and collaborators of the login_controller class. While table 3 will contain the responsibility and collaborator of the login_model class which is also a collaborator of the login_controller class.

Table 2. Login Controller's CRC Card

Class login_controller	
Responsibilities	**Collaborators**
Processing admin login **Processing yellow officer login** **Processing public community login**	Login_model

Table 3. Login Model's CRC Card

Class login_model	
Responsibilities	**Collaborators**
Check login admin **Check login yellow officer** **Check login public community**	

Tables 4, 5, and 6 show the CRC cards of the complaints service. Table 4 will contain the responsibilities and collaborators of the report_controller class. Table 5 will contain the responsibilities and collaborators of the handle controller class. Table 6 will contain the responsibilities and collaborators of the report updates controller class.

Table 4. Report Controller's CRC Card

Class report_controller	
Responsibilities	**Collaborators**
Fetch all reports **Verify a report** **Reject a report** **Finish a report**	

Table 5. Handle Controller CRC Card

Class handle_controller	
Responsibilities	**Collaborators**
Assign a yellow officer **Discharge a yellow officer**	

Table 6. Report Updates Controller CRC Card

Class report_updates_controller	
Responsibilities	**Collaborators**
Get report updates by report's id **Add a report update**	

3) Coding

Coding is the implementation stage from the previous stage into a language that is recognized by the computer. The coding stage in this research is done by creating a service using the Go programming language for backend services that handle user authentication, and PHP programming language for backend services that manage complaint data, backend services that provide drainage monitoring data, and frontend admin services. Coding begins by creating an authentication backend service, followed by a complaint backend service, then a drainage monitoring backend service and ends by integrating the backend service with the admin frontend.

The followings are the result of the frontend admin service that has been integrated with the backend services.

a. Login page

This page will do a request to the backend service that handles user authentication.

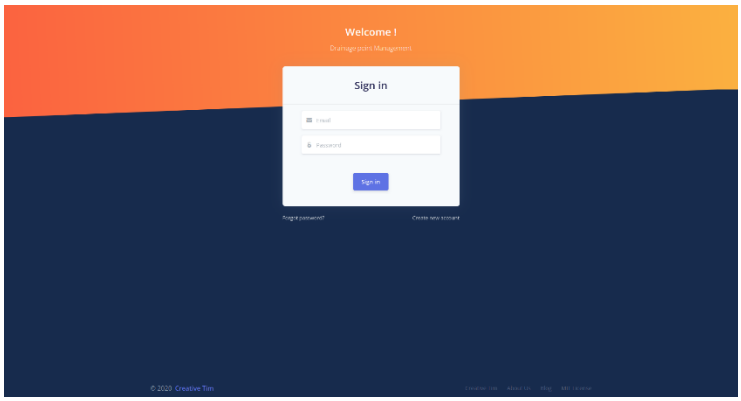

Figure 3. Login Page

b. List of report page
This page will do requests to the backend service that handles complaint data.

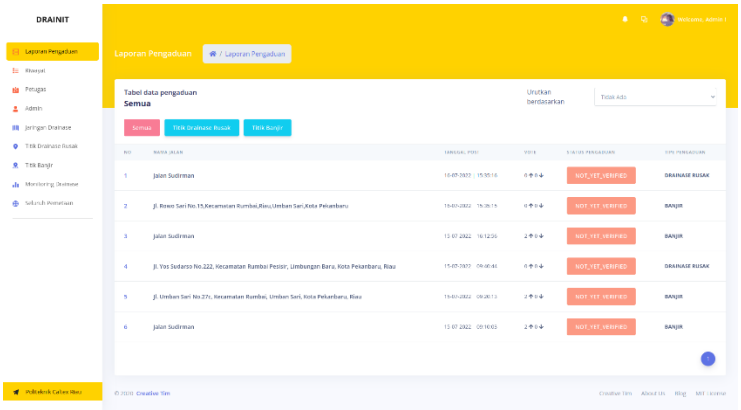

Figure 4. List of Report Page

c. Report's detail page
This page will do requests to the backend service that handles complaint data.

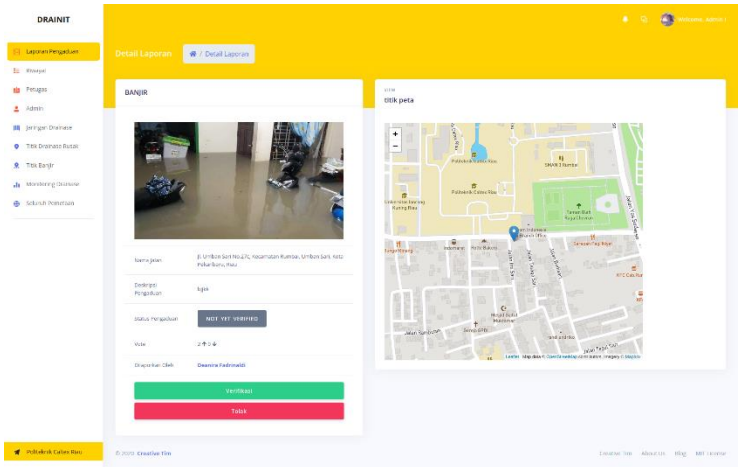

Figure 5. Report's Detail Page

d. Assign yellow officer page
This page will do requests to the backend service that handles complaint data.

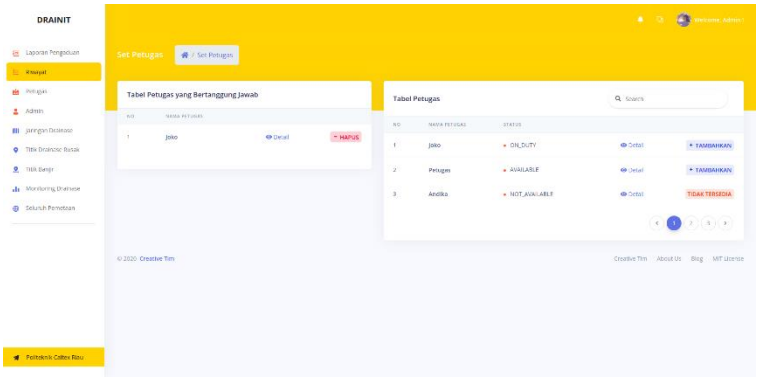

Figure 6. Assign Yellow Officer Page

e. Drainage monitoring history page
This page will do requests to the backend service that provide drainage monitoring data.

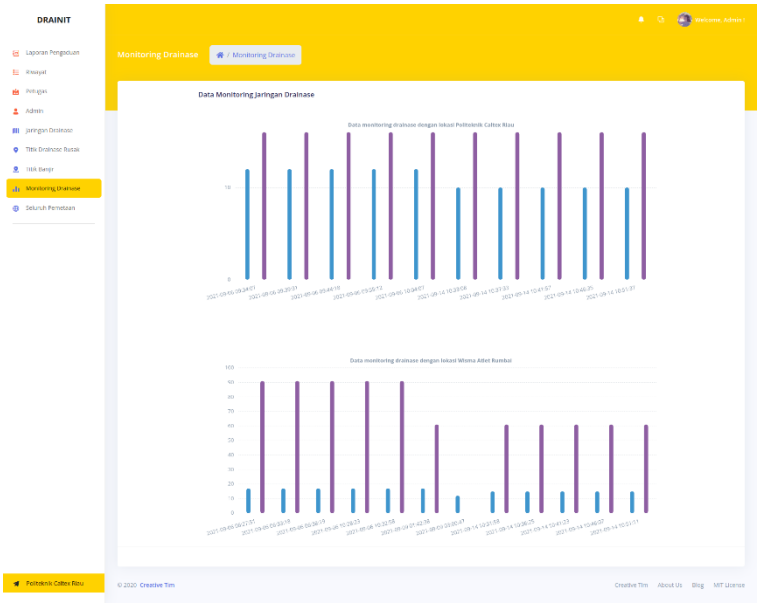

Figure 7. Drainage Monitoring History Page

4) Testing

Testing is the final stage of each iteration in XP. This study uses an approach of Unit Testing, User Acceptance Testing (UAT), and Performance Testing. Unit testing uses a white box testing technique that is used to test the behavior of each function that is being tested. As for the UAT, this study uses the black box testing method. This method is often categorized as functional testing which is done to analyze certain functions without letting the tester see the internal code structure of the software.

The implementation of unit testing in this study uses the white box method. White box testing is a testing method carried out to test the internal structure, design, function, and implementation details of an application.

Table 7. Unit Testing

Test Case ID	Test Case Description	Test Data	Expected Results	Actual Results	Result
UT22	Get All Reports	-	Status code: 200	Status code: 200	Pass
UT23	Create Report	Authorization: “Bearer [jwt token]” Body: ['nama_jalan' => $this->faker->city(), 'foto' => null, 'deskripsi' => 'deskripsi', 'geometry' => '{"type":	Status code: 201	Status code: 201	Pass

Test Case ID	Test Case Description	Test Data	Expected Results	Actual Results	Result
		"Point", "coordinates": [101.415468, 0.584255]}'];			
UT24	Verify Report	Authorization: "Bearer [jwt token]" Param: Str::random(10)	Status code: 200	Status code: 200	Pass
UT25	Reject Report	Authorization: "Bearer [jwt token]" Param: Str::random(10)	Status code: 200	Status code: 200	Pass
UT26	Assign Yellow Officer	Authorization: "Bearer [jwt token]" Body: ['vote' => true, 'id_pengaduan' => $pengaduan->id, 'id_petugas' => Str::random(10)];	Status code: 201	Status code: 201	Pass
UT27	Create Report Updates	Authorization: "Bearer [jwt token]" Body: ['id_pengaduan' => $pengaduan->id,'judul' => $this->faker->city(), 'deskripsi' => $this->faker->name()];	Status code: 201	Status code: 201	Pass

UAT is done by black box method. This test is carried out to determine whether the system has met user needs and can support all user business scenarios. UAT is performed by the client and end-user. At the end of the testing process, the user needs to conclude whether the test was successful so that the requirements were met or not. If an error occurs in the test, it is necessary to record and trace the repair.

This UAT was conducted by conducting interviews with Mrs. Retno Tri Wahyuni as a client, and also to Mr. Indra Pomi who served as Head of the PUPR Office of Riau Province as an end-user. Based on the UAT that has been carried out on the features of the DRAINIT website application, it is found that all system functionality can run and work well according to the test case. With no errors that occur in the functionality of the system, it shows that the whole system is running as it should and can produce output in accordance with the input given and provide a response according to requests from users. This test can be used as evidence that the DRAINIT website application is performing well and can be used.

Table 8. User Acceptance Testing

Test Case ID	Test Case	Result
UAT01	Name: Login Admin Testing Description: Verifying that the website can only be accessed by registered users Test Case: - Email: admin@gmail.com - Password: admin Expected Output: - If succeed, shows the list of reports page - If failed, shows an error message	Success
UAT02	Name: Verify a Report Testing Description: Verifying the status of the report has changed into "VERIFIED" Test Case: - Choose a report from the report table Expected Output: - If succeed, shows the report history's detail page, and the report status has changed to "VERIFIED" - If failed, shows an error message	Success
UAT03	Name: Reject a Report Testing Description: Verifying the status of the report has changed to "REJECTED" Test Case: - Choose a report from the report table	Success

Test Case ID	Test Case	Result
	Expected Output: - If succeed, shows the report history's detail page, and the report status has changed into "REJECTED" - If failed, shows an error message	
UAT04	Name: Assign Yellow Officers to a Report Testing Description: Verifying yellow officers has been added to a verified report Test Case: - Choose a report from the verified report history table Expected Output: - If succeed, shows yellow officers that have been assigned to the responsible yellow officer section - If failed, shows an error message	Success

Performance testing will be carried out using the load testing method. This load testing scenario is to test how many login requests can be processed by the DRAINIT website application. As seen in the code below, there are 7 stages of the request. The first stage is to simulate an increase in traffic from 1 to 60 users for 5 minutes. The second stage will keep the request at 60 users for 10 minutes. The third stage will increase the request to 100 users for 3 minutes to simulate the start of rush hour. The fourth stage will keep the request at 100 users for 2 minutes to simulate rush hour. The fifth stage will lower requests to 60 users for 3 minutes. The sixth stage will keep the request at 60 users for 10 minutes. The seventh stage will downgrade the request to 0 users. After this load testing scenario was tested on the DRAINIT admin website application, it turns out that this application can only process up to 19.3 requests per second. This happens because, with a microservice architecture, any communication between services will most likely occur via API calls. In a monolith architecture, these API calls are converted into code calls or function calls. In this sense, in the performance of applications using monolith architecture can be faster than applications using microservice architecture.

```
export const options = {
    setupTimeout: '10s',
    stages: [
      { duration: '5m', target: 60 }, // simulate ramp-up of traffic from 1 to 60 users over 5 minutes.
      { duration: '10m', target: 60 }, // stay at 60 users for 10 minutes
      { duration: '3m', target: 100 }, // ramp-up to 100 users over 3 minutes (peak hour starts)
      { duration: '2m', target: 100 }, // stay at 100 users for short amount of time (peak hour)
      { duration: '3m', target: 60 }, // ramp-down to 60 users over 3 minutes (peak hour ends)
      { duration: '10m', target: 60 }, // continue at 60 for additional 10 minutes
      { duration: '5m', target: 0 }, // ramp-down to 0 users
    ],
    thresholds: {
      http_req_duration: ['p(99)<1500'], // 99% of requests must complete below 1.5s
    },
  };

export default () => {
    let response = http.get("https://drainit-frontend-uq37z.ondigitalocean.app");

    response = response.submitForm({
      ...
    });

    sleep(1);
}
```

4. Conclusions

The conclusions obtained from the results of the study are as follows:

1. The DRAINIT website application has successfully implemented a list of features that have been requested by the Public Works Department of Pekanbaru City, namely recording public complaints reports, viewing GIS-based drainage networks, and viewing drainage monitoring data.
2. The handling of application functions is divided into several services according to the principles of the microservice architecture. These services are backend services to handle user authentication, backend services to manage complaint data, backend services to provide drainage monitoring data, and frontend admin services.

3. The results of the tests that have been carried out on this web application—unit testing, integration testing, and user acceptance testing—show that all the features in the DRAINIT application has been running well.
4. The result of performance testing shows that this web application is able to process up to 19.3 requests per second.

Acknowledgement

This research would not have been completed without the cooperation with the Pekanbaru's PUPR Service

Reference

[1] Pemerintah Kota Pekanbaru, "Mengenal Kota Pekanbaru," 2020, [Online]. Available: https://www.pekanbaru.go.id/p/menu/profil-kota/mengenal-kota-pekanbaru#:~:text=Nama Pekanbaru dahulunya dikenal dengan,terletak di muara Sungai Siak.

[2] D. K. Sasidharan and S. K. N, *Full Stack Development with JHipster: Build full stack applications and microservices with Spring Boot and modern JavaScript frameworks, 2nd Edition*. Packt Publishing, 2020. [Online]. Available: https://books.google.co.id/books?id=eBDMDwAAQBAJ

[3] A. Wantoro, S. Samsugi, and M. J. Suharyanto, "Sistem Monitoring Perawatan dan Perbaikan Fasilitas PT PLN (Studi Kasus : Kota Metro Lampung)," *Jurnal TEKNO KOMPAK*, vol. 15, no. 1, pp. 116–130, 2021.

[4] S. Enriko, "PEMBANGUNAN APLIKASI WEB SISTEM INFORMASI GEOGRAFIS LOKASI BENCANA ALAM TANAH LONGSOR DAN BANJIR KECAMATAN TANJUNG RAYA," 2018.

[5] F. I. M. S. Rd. Nuraini Siti Fatonah, *Penggunaan Digital Invoice Dan Settlement*. 2020. [Online]. Available: https://books.google.co.id/books?id=-JzzDwAAQBAJ

[6] M. Masse, *REST API Design Rulebook: Designing Consistent RESTful Web Service Interfaces*. O'Reilly Media, 2011. [Online]. Available: https://books.google.co.id/books?id=eABpzyTcJNIC

[7] "PHP: Hypertext Preprocessor." https://www.php.net/ (accessed Nov. 09, 2021).

[8] "PHP: What can PHP do? - Manual." https://www.php.net/manual/en/intro-whatcando.php (accessed Nov. 09, 2021).

[9] "What Is Laravel Framework and What is the Use Of Laravel ?" https://www.zworthkey.com/what-is-laravel-framework/ (accessed Nov. 09, 2021).

[10] "The Go Programming Language." https://go.dev/ (accessed Aug. 30, 2022).