# BAB IV
# HASIL DAN ANALISIS PENELITIAN

## 4.1 Analisa Perancangan Sistem

### 4.1.1 Identifikasi Kebutuhan Sistem

Identifikasi kebutuhan sistem didapatkan berdasarkan hasil dari observasi, wawancara dan analisa dokumen yang telah dilakukan oleh penulis. Hasil Identifikasi tersebut berupa permasalahan yang terjadi didalam metode *monitoring* yang dilakukan oleh tim infrastruktur saat ini dan kendala apa saja yang dialami selama menggunakannya. Berikut merupakan uraian dari masalah yang terjadi didalam metode *monitoring* saat ini.

1. Dalam proses monitoring saat ini, staf infrastruktur diharuskan memeriksa setiap server milik PT. XYZ satu per satu. Staf tersebut harus melakukan aktivitas yang sama berulang kali untuk melakukan *monitoring* ke seluruh server yang dimiliki PT. XYZ. Meskipun metode ini dapat memberikan visibilitas mendalam terhadap masing-masing data server yang dibutuhkan, namun memakan waktu dan tidak praktis karena PT. XYZ memiliki banyak server.
2. Proses monitoring saat ini di PT. XYZ juga menghadapi masalah lain ketika diperlukan respons cepat. Metode monitoring yang dilakukan sesuai jadwal mingguan kurang responsif terhadap perubahan mendadak atau keadaan darurat, seperti lonjakan traffic tinggi yang menyebabkan aplikasi microservice mati. Oleh karena itu, dibutuhkan pemantauan real-time dan notifikasi instan agar tim dapat mengetahui keadaan darurat tersebut dan segera mengambil tindakan.

### 4.1.2 Analisa Kebutuhan Sistem

Setelah melakukan identifikasi kebutuhan sistem, selanjutnya analisa kebutuhan sistem ini dilakukan untuk meninjauan ulang kebutuhan yang diperlukan untuk mengatasi masalah tersebut. Dalam konteks penelitian ini, penulis memaparkan solusi dari permasalahan yang teridentifikasi di PT. XYZ, yaitu melalui Rancang Bangun Aplikasi *Monitoring Resource* Server dan *Microservice* Berbasis Web dengan Notifikasi Telegram Menggunakan Pendekatan Waterfall. Sistem ini dirancang untuk mempermudah Staff infrastruktur dalam melakukan *monitoring* terhadap setiap server yang dimiliki oleh PT. XYZ. Berikut merupakan beberapa fitur yang dibutuhkan berdasarkan hasil identifikasi oleh penulis diantaranya sebagai berikut :

1. Terdapat fitur yang dapat menarik data resource dan status aplikasi microservice dari setiap server. Fitur tersebut akan dipenuhi oleh Aplikasi lapor yang merupakan aplikasi yang berjalan secara background menggunakan scheduler, aplikasi tersebut akan dapat

mengirimkan data server berikut dengan resource server dan status aplikasi microservice sesuai dengan interval waktu yang telah ditetapkan.

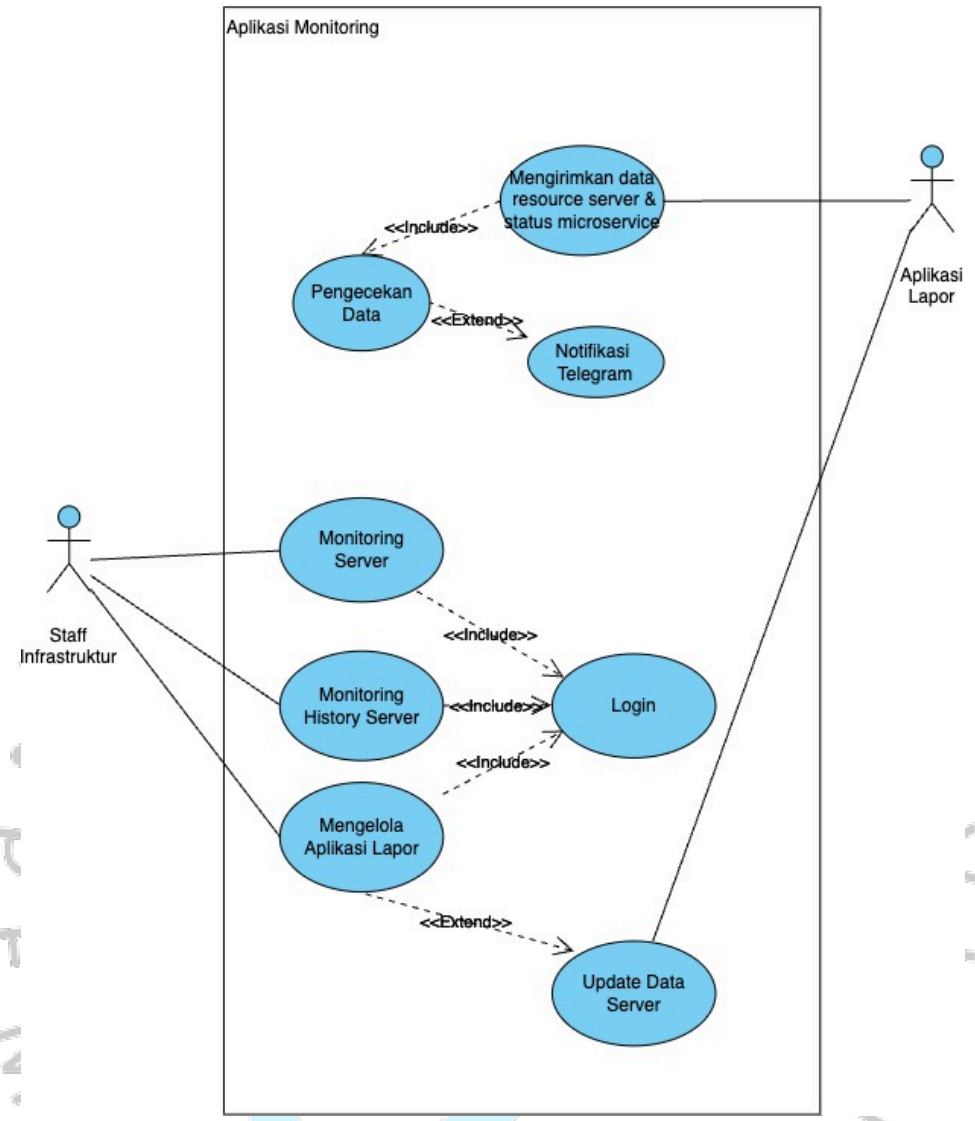
2. Staff infrastruktur dapat melakukan *monitoring* terhadap beberapa server didalam satu tampilan. Dengan adanya fitur tersebut Staff infrastruktur tidak perlu melakukan aktivitas berulang untuk memeriksa setiap server yang dimiliki oleh PT. XYZ.
3. Staff infrastruktur dapat melakukan *monitoring* terhadap *history* penggunaan *resource* dan status aplikasi *microservice* dari masing masing server supaya Staff infrastruktur dapat mengetahui di jam berapa aktivitas server sedang tinggi.
4. Staff infrastruktur dapat mengelola aplikasi lapor yang di *install* pada server yang dilakukan *monitoring.*
5. Staff infrastruktur mendapatkan notifikasi melalui telegram apabila *resource* server melebihi batas wajar dan status aplikasi *microservice* mati supaya tim dapat melakukan tindakan dengan cepat.

## 4.2 Perancagan Diagram Sistem Usulan

Dalam merancang dan membangun diagram untuk aplikasi *monitoring* yang diusulkan, penulis menggunakan pendekatan *Unified Modeling Language* (UML). Pendekatan ini melibatkan penerapan berbagai jenis diagram UML yang masing-masing memiliki peran penting dalam memodelkan sistem secara komprehensif. Penulis menggunakan *use case* diagram, *activity* diagram, *sequence* diagram dan *class* diagram, selain UML didalam perancangan ini penulis menggunakan DFD dan ERD sebagai diagram pendukung didalam merancang aplikasi monitoring. Dengan mengintegrasikan berbagai jenis diagram UML disertai DFD berikut dengan ERD, penulis dapat menciptakan gambaran yang jelas dan terstruktur tentang bagaimana aplikasi monitoring akan dibangun dan berfungsi sesuai dengan kebutuhan.

### 4.2.1 *Use Case* Diagram

Dalam proses perancangan *use case* diagram untuk aplikasi monitoring yang diusulkan, penulis mengidentifikasi dua aktor utama yang berinteraksi dengan sistem. Aktor pertama adalah Staff infrastruktur, yang berperan sebagai pengguna utama aplikasi ini. Aktor kedua adalah aplikasi lapor, yang berfungsi untuk mengirimkan data resource dan status *microservice* ke aplikasi monitoring. Berikut merupakan rancangan *use case* diagram untuk aplikasi *monitoring* di PT. XYZ.




Gambar 4. 1 *Use Case* Aplikasi Monitoring

Gambar 4.1 merupakan rancangan *use case* diagram untuk aplikasi *monitoring*. Selanjutnya penulis akan mendeskripsikan setiap *use case* dengan spesifikasi *use case*. Berikut merupakan spesifikasi *use case* dari setiap *use case* tersebut :

### *1. Use Case Description Login*

Tabel 4. 1. *Use Case Description Login*

| **Use Case Name :** | **ID :** | **Priority :** |
|---|---|---|
| Login | UC-SIM-001 | High |

| |
|---|
| **Description:**<br>Use case "Login" merupakan proses untuk memastikan bahwa hanya pengguna yang didaftarkan yang dapat mengakses aplikasi monitoring dengan melakukan login. |
| **Actor :**<br>1. Staff infrastruktur |
| **Trigger:**<br>Ketika user ingin mengakses menu yang ada didalam aplikasi monitoring, sebelum bisa masuk akan di arahkan ke halaman login untuk melakukan login terlebih dahulu. |
| **Pre-conditions:**<br>1. Pengguna harus sudah terdaftar dalam sistem dengan username dan password yang valid. |
| **Post-conditions:**<br>1. Sistem akan menampilkan halaman dashboard monitoring. |
| **Normal Flow:**<br>1. Sistem akan menampilkan halaman login yang terdapat form berisi username dan password.<br>2. User diminta memasukan username dan password dan klik login.<br>3. Sistem akan melakukan validasi terhadap username dan password yang di masukan.<br>4. Jika valid maka user berhasil masuk ke halaman dashboard monitoring. |
| **Subflows :**<br>- |
| **Exceptions Flow :**<br>1. Menampilkan notifikasi "Username atau Password Invalid" ketika proses login ke sistem gagal. |

Tabel diatas merupakan spesifikasi *use case Login* yang dapat dilakukan oleh Staff infrastruktur. *Use case Login* diawali dari *user* ketika ingin mengakses aplikasi *monitoring* yang kemudian sistem menampilkan halaman *login* dan diakhiri dengan *user* yang berhasil masuk kehalaman *login* jika username dan password yang dimasukan valid dengan *exception flow* menampilkan notifikasi apabila proses *login* gagal.

### 2. *Use Case* Description *Monitoring* Server

Tabel 4. 2. *Use Case Description Monitoring* Server

| **Use Case Name :** | **ID :** | **Priority :** |
|---|---|---|
| Monitoring Server | UC-SIM-002 | High |

| |
|---|
| **Description:**<br><br>Use case "Monitoring Server" memungkinkan Staff infrastruktur untuk memantau Resource server dan status aplikasi microservice dari berbagai server. |
| **Actor :**<br>Staff infrastruktur. |
| **Trigger:**<br>User menekan button menu Realtime Monitoring pada sidebar dashboard monitoring. |
| **Pre-conditions:**<br>User telah berhasil login dengan username dan password yang valid. |
| **Post-conditions:**<br>Sistem akan menampilkan halaman realtime monitoring. |
| **Normal Flow:**<br>1. User menekan tombol button menu Realtime Monitoring pada sidebar dashboard monitoring.<br>2. Sistem menampilkan halaman realtime monitoring dengan data resource dan status aplikasi microservice dari berbagai server. |
| **Subflows :**<br>- |
| **Exceptions Flow :**<br>- |

Tabel diatas merupakan spesifikasi *use case Monitoring* Server yang dapat dilakukan oleh Staff infrastruktur. *Use case Monitoring* Server diawali dari *user* ketika menekan tombol menu *Realtime Monitoring* pada sidebar *dashboard monitoring* yang kemudian diakhiri dengan sistem menampilkan halaman *realtime monitoring*.

### 3. *Use Case* Description *Monitoring History* Server

Tabel 4. 3. *Use Case Description Monitoring History Server*

| **Use Case Name :**<br><br>History Monitoring | **ID :**<br><br>UC-SIM-003 | **Priority :**<br><br>High |
|---|---|---|
| **Description:**<br>Use case "History Monitoring " memungkinkan Staff infrastruktur untuk melihat data history Resource server dan status aplikasi microservice dari server yang dipilih, data history yang ditampilkan yaitu 24 jam sebelumnya. | | |
| **Actor :**<br>Staff infrastruktur. | | |

| |
|---|
| **Trigger:**<br>User menekan button menu History Monitoring pada sidebar dashboard monitoring. |
| **Pre-conditions:**<br>User telah berhasil login dengan username dan password yang valid. |
| **Post-conditions:**<br>Sistem akan menampilkan halaman History monitoring. |
| **Normal Flow:**<br>1. User menekan tombol button menu History Monitoring pada sidebar dashboard monitoring.<br>2. Sistem menampilkan menu history monitoring dengan field dropdown yang terdapat list ip untuk dipilih oleh user.<br>3. User memilih ip dari server yang ingin dilihat.<br>**4.** Sistem akan menampilkan data resource server dan status aplikasi microservice dari ip yang dipilih. |
| **Subflows :**<br>- |
| **Exceptions Flow :**<br>- |

Tabel diatas merupakan spesifikasi *use case History Monitoring* yang dapat dilakukan oleh Staff infrastruktur. *Use case History Monitoring* diawali dari *user* ketika menekan tombol menu *History Monitoring* pada *sidebar dashboard monitoring* yang kemudian diakhiri dengan sistem menampilkan data *resource* server dan status aplikasi *microservice* dari ip yang dipilih.

### 4. *Use Case Description* Mengelola Aplikasi Lapor

Tabel 4. 4. *Use Case Description Mengelola Aplikasi Lapor*

| **Use Case Name :**<br><br>Mengelola Aplikasi Lapor | **ID :**<br><br>UC-SIM-004 | **Priority :**<br><br>High |
|---|---|---|
| **Description:**<br><br>Use case "Mengelola Aplikasi Lapor " memungkinkan Staff infrastruktur untuk mengelola data aplikasi lapor dengan mengedit data atau menghapus data. Data yang diubah selanjutnya akan di update juga ke aplikasi lapor | | |
| **Actor :**<br>Staff infrastruktur. | | |
| **Trigger:**<br>User menekan button menu Manajemen Server pada sidebar dashboard monitoring. | | |
| **Pre-conditions:** | | |

| |
|---|
| User telah berhasil login dengan username dan password yang valid. |
| **Post-conditions:**<br>Sistem akan menampilkan halaman Manajemen server. |
| **Normal Flow:**<br>1. User menekan tombol button menu Manajemen Server pada sidebar dashboard monitoring.<br>2. Sistem menampilkan menu Manajemen Server.<br>3. User dapat mencari data server yang terdapat aplikasi lapor untuk dilihat dengan mengetik ip atau nama server yang di cari di kolom pencarian.<br>4. User dapat mengelola data server seperti :<br>1) Mengedit data server, selanjutnya ke subflow MS-1 Edit Data Server.<br>2) Menghapus data server, selanjutnya ke subflow MS-2 Delete Data Server. |
| **Subflows :**<br>MS-1 Edit Data Server<br>1. User berada di halaman manajemen server.<br>2. User menekan button edit di data server yang diinginkan.<br>3. Sistem akan menampilkan halaman untuk mengedit data.<br>4. User melakukan edit data dan melakukan submit.<br>5. Sistem akan menampilkan pesan "apakah anda yakin ingin melakukan perubahan data?".<br>6. Sistem akan menyimpan data yang diubah dan melakukan update ke aplikasi lapor yang ada di server yang di edit datanya.<br>**7.** Sistem akan menampilkan pesan "Data berhasil diubah".<br><br>MS-2 Delete Data Server<br>1. User berada di halaman manajemen server.<br>2. User menekan button delete di data server yang diinginkan.<br>3. Sistem akan menampilkan pesan "apakah anda yakin ingin menghapus data?".<br>4. Sistem akan menghapus data dan melakukan update ke aplikasi lapor yang ada di server yang dihapus datanya.<br>5. Sistem akan menampilkan pesan "Data berhasil di hapus!". |
| **Exceptions Flow :**<br>- |

Tabel diatas merupakan spesifikasi *use case* Mengelola Aplikasi Lapor yang dapat dilakukan oleh Staff infrastruktur. *Use case* Mengelola Aplikasi Lapor diawali dari *user* ketika menekan tombol button menu Manajemen Server pada *sidebar dashboard monitoring* yang kemudian diakhiri dengan *user* yang dapat mengelola data server.

5. ***Use Case Description Update* Data Server**

Tabel 4. 5. *Use Case Description Update Data Server*

| **Use Case Name :**<br><br>Update Data Server | **ID :**<br><br>UC-SIM-005 | **Priority :**<br><br>High |
|---|---|---|
| **Description:**<br>Use case "Update Data Server" memungkinkan aplikasi lapor yang dipasang di server yang dimonitoring untuk mengirimkan data identitas server tersebut. | | |
| **Actor :**<br>Aplikasi Lapor. | | |
| **Trigger:**<br>Aplikasi lapor di jalankan di server yang dimonitoring. | | |
| **Pre-conditions:**<br>Aplikasi lapor telah diunggah ke server yang dimonitoring. | | |
| **Post-conditions:**<br>Sistem akan menyimpan data indentitas server. | | |
| **Normal Flow:**<br>1. Aplikas lapor dijalankan di server yang dimonitoring.<br>**2.** Aplikasi lapor mengirimkan data identitas server ke aplikasi monitoring. | | |
| **Subflows :**<br>- | | |
| **Exceptions Flow :**<br>- | | |

Tabel diatas merupakan spesifikasi *use case Update* Data Server yang dapat dilakukan oleh aplikasi lapor. *Use case Update* Data Server diawali dari aplikasi lapor yang dijalankan di server yang akan *dimonitoring* yang kemudian diakhiri dengan aplikasi lapor mengirimkan data ke aplikasi *monitoring*.

6. ***Use Case Description* Mengirimkan Data *Resource* Server dan Status *Microservice***

Tabel 4. 6. *Use Case Description* Kirim Data

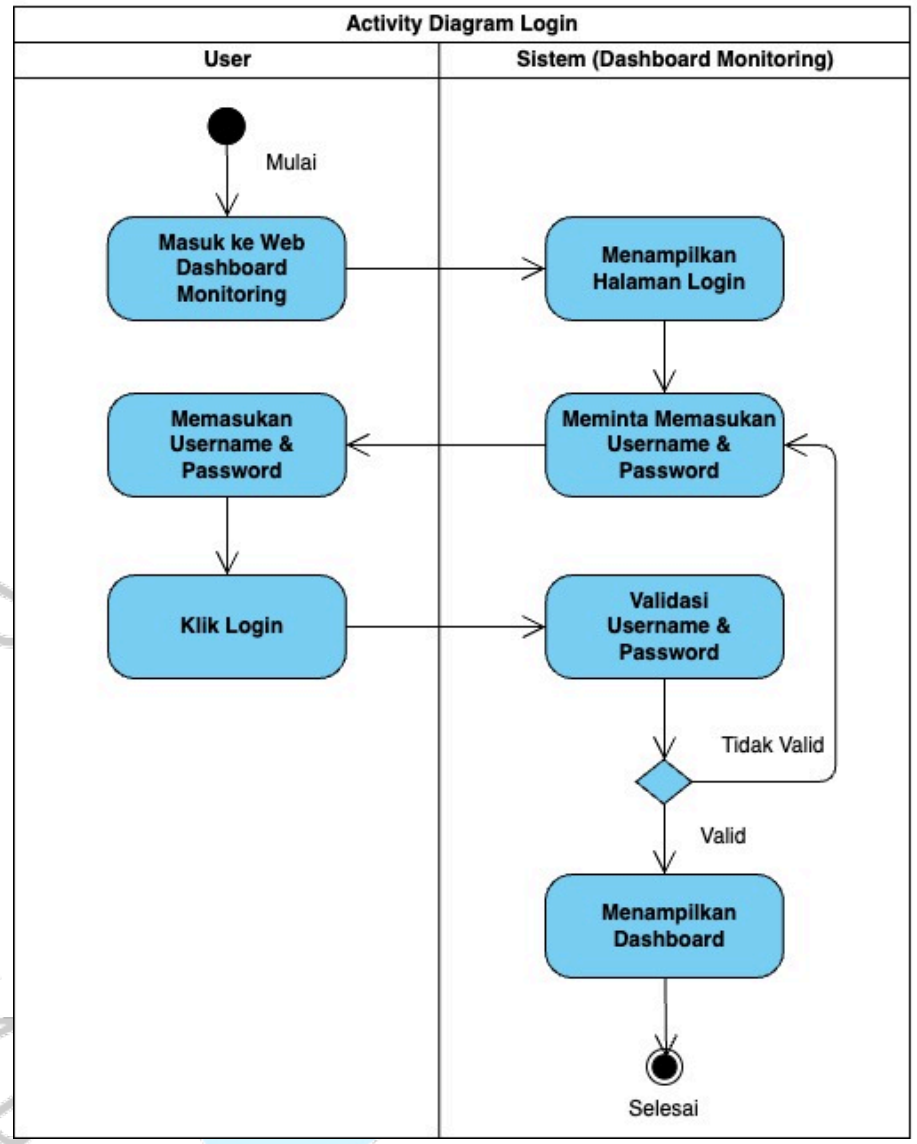
| **Use Case Name :**<br><br>Mengirimkan Data Resource Server dan Status Microservice | **ID :**<br><br>UC-SIM-006 | **Priority :**<br><br>High |
|---|---|---|

| |
|---|
| **Description:**<br>Use case "Mengirimkan Data Resource Server dan Status Microservice" memungkinkan aplikasi lapor yang dipasang di server yang dimonitoring untuk mengirimkan data resource server dan status aplikasi microservice ke aplikasi monitoring. Data yang dikirimkan akan dilakukan pengecekan, apabila data yang dikirimkan tidak lolos dalam pengecekan maka sistem akan mengirimkan notifikasi dan selanjutnya data akan di simpan. |
| **Actor :**<br>Aplikasi Lapor dan Sistem Monitoring |
| **Trigger:**<br>Scheduler yang berjalan setiap 5 menit. |
| **Pre-conditions:**<br>Aplikasi lapor berjalan dan aktif di server yang dilakukan monitoring. |
| **Post-conditions:**<br>1. Sistem akan melakukan pengecekan terhadap data yang dikirim.<br>2. Sistem akan mengirimkan notifikasi apabila data yang dikirim.<br>3. Sistem akan menyimpan data resource dan status aplikasi microservice. |
| **Normal Flow:**<br>1. Scheduler di aplikasi lapor aktif.<br>2. Aplikasi Lapor mengirimkan data resource server dan status aplikasi microservice ke sistem monitoring.<br>3. Sistem monitoring melakukan pengecekan data yang diterima.<br>4. Sistem monitoring menyimpan data resource dan status aplikasi microservice yang dikirimkan oleh aplikasi lapor. |
| **Subflows :** |
| **Exceptions Flow :**<br>Apabila data yang diterima tidak lolos dalam pengecekan maka sistem akan mengirimkan notifikasi ke telegram sesuai dengan data yang tidak lolos dalam pengecekan. |

### 4.2.2 *Activity* Diagram

*Activity* diagram merupakan alat yang penting dalam perancangan dan pengelolaan aplikasi *monitoring*, karena membantu mengoptimalkan alur kerja dan meningkatkan koordinasi serta efisiensi dalam penyelesaian proses bisnis. Berikut merupakan *activity* diagram yang dirancang sesuai dengan use case yang telah dibuat sebelumnya.

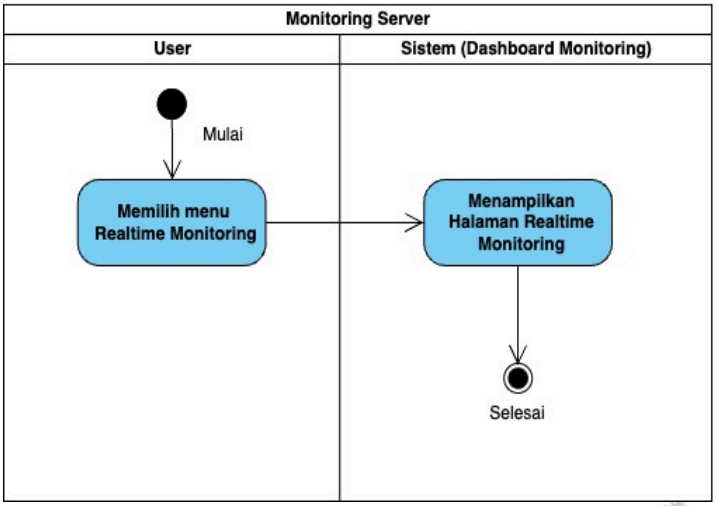
1. ***Activity* Diagram *Login***




Gambar 4. 2 *Activity* Diagram *Login*

Berdasarkan *Activity* Diagram *Login* diatas *User* memulai proses dengan masuk ke web *dashboard monitoring*. Sistem kemudian menampilkan halaman *login* yang meminta user untuk memasukkan *username* dan *password*. *User* kemudian memasukkan *username* dan *password* tersebut dan menekan tombol *login*. Sistem melakukan validasi terhadap *username* dan password yang dimasukkan. Jika validasi gagal, sistem akan kembali meminta *user* untuk memasukkan *username* dan *password* di halaman *login*. Namun, jika validasi berhasil, sistem akan menampilkan *dashboard* kepada *user*.

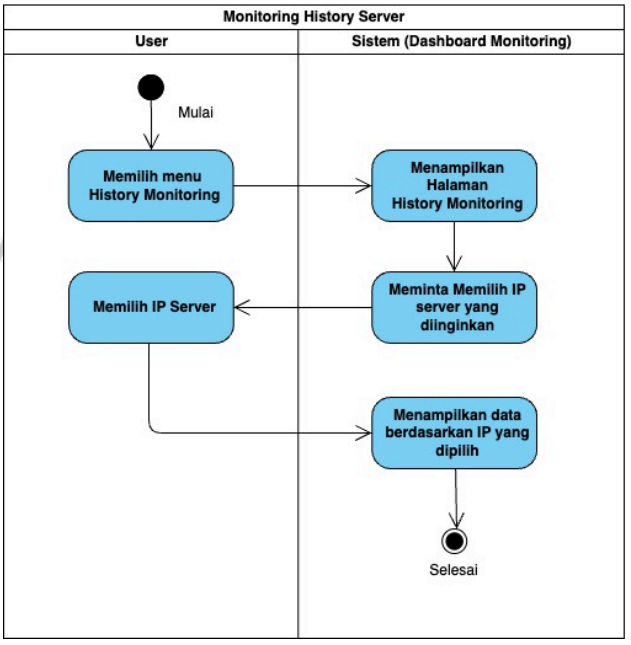
**2. *Activity* Diagram *Monitoring Server***




Gambar 4. 3 *Activity* Diagram *Monitoring Server*

Berdasarkan *Activity* Diagram *Monitoring* Server diatas dapat dijelaskan bahwa *User* memulai dengan memilih menu *realtime monitoring* pada *sidebar dashboard monitoring*. Setelah itu, sistem akan menampilkan halaman *realtime monitoring*. Pada halaman tersebut, sistem menyajikan data mengenai *resource* server dan status aplikasi *microservice* dari berbagai server, memberikan user informasi terkini dan detail tentang kondisi dan performa server serta aplikasi *microservice* yang *dimonitoring*.

**3. *Activity* Diagram *Monitoring History* Server**



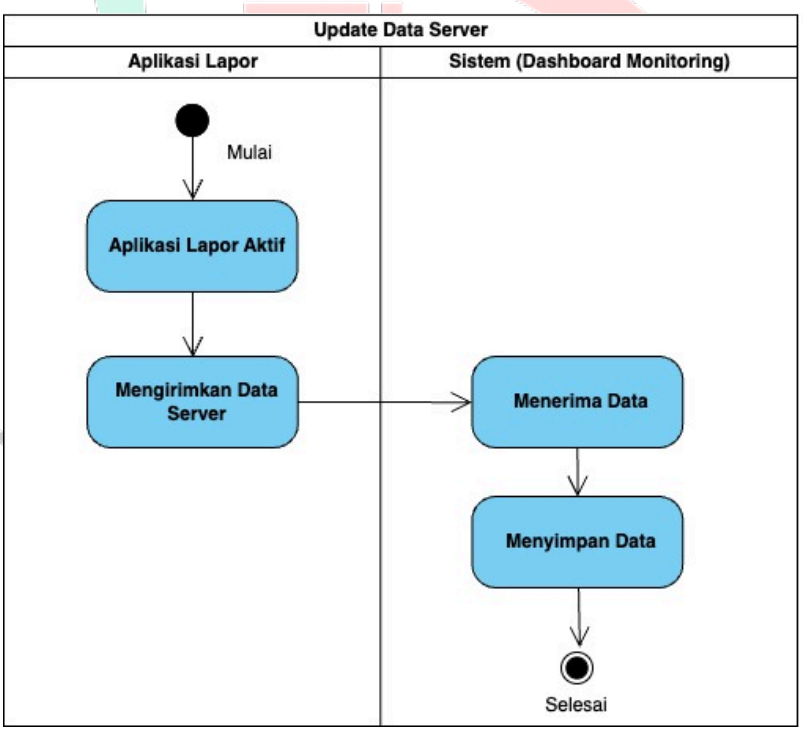

Gambar 4. 4 *Activity* Diagram *Monitoring History Server*

Berdasarkan *Activity* Diagram *Monitoring History* Server diatas dapat dijelaskan bahwa *User* mengakses *dashboard monitoring* dan memilih menu "*History Monitoring*" pada *sidebar dashboard monitoring*. Setelah memilih menu tersebut, sistem langsung mengarahkannya ke halaman "*History Monitoring*".

Di halaman "*History Monitoring*", *user* disajikan dengan *field dropdown* yang menampilkan daftar IP server yang tersedia. *Field dropdown* ini memungkinkan user untuk memilih IP server yang ingin di lakukan *monitoring* oleh *user*. Sistem meminta *user* untuk memilih IP server yang diinginkan dari daftar yang tersedia. *User* kemudian memilih IP server yang ingin dilakukan pengecekan.

Setelah *user* memilih IP server, sistem kemudian menampilkan data yang diminta berdasarkan IP server yang dipilih. Halaman "*History Monitoring*" selanjutnya menampilkan informasi terdahulu tentang *resource* server dan status aplikasi *microservice* yang terkait dengan IP server yang dipilih oleh user. Dengan informasi ini, user dapat memantau *history* dari kinerja server dan status aplikasi yang terkait dengan IP yang dipilihnya.

### 4. *Activity* Diagram *Update* Data Server

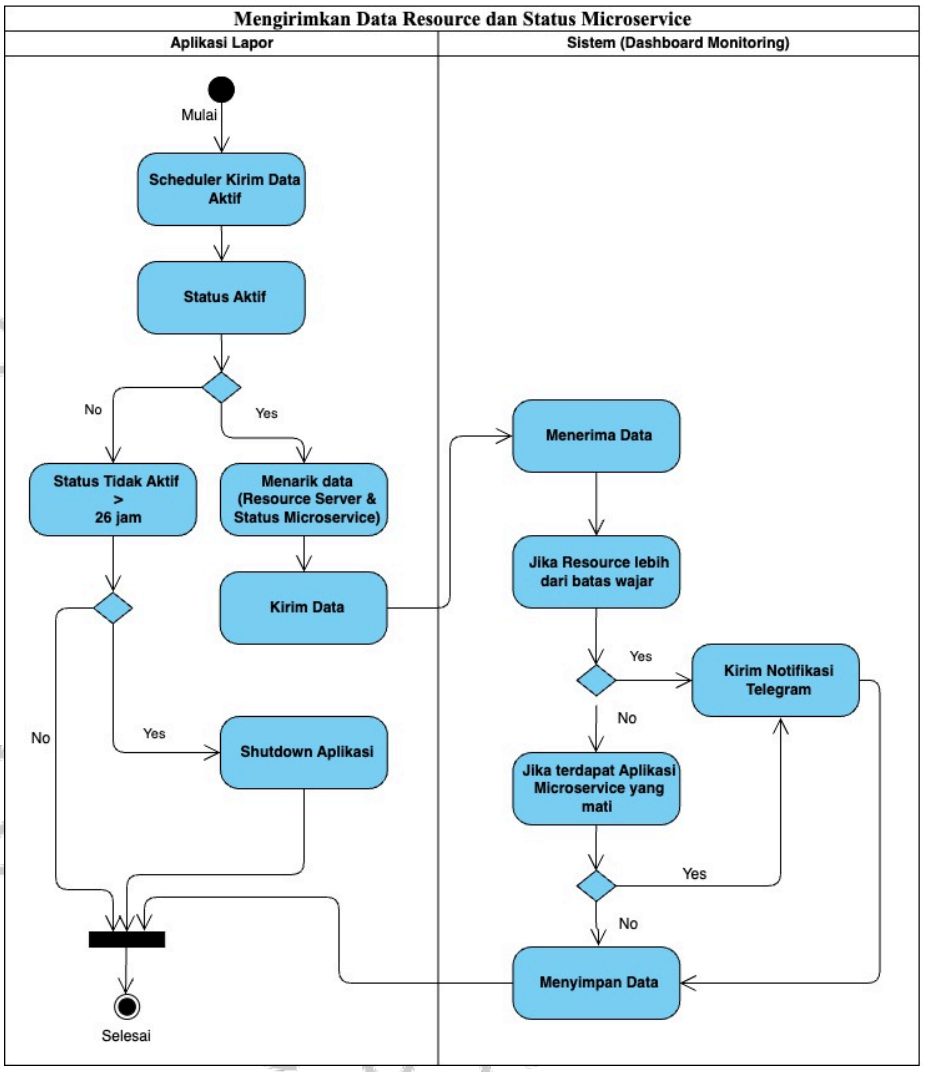



Gambar 4. 5 *Activity Diagram Update* Data

Berdasarkan *Activity* Diagram *Update* Data Server diatas dapat dijelaskan bahwa Ketika Aplikasi lapor aktif beroperasi di server yang sedang dilakukan *monitoring*, Aplikasi lapor akan menyiapkan data indentitas dari server yang sedang dilakukan *monitoring*. Kemudian

aplikasi lapor mengirimkan data terkait server yang dipantau ke dalam sistem *dashboard monitoring*. Selanjutnya data tersebut diterima dan disimpan ke dalam *database*.

### 5. *Activity* Diagram Mengirimkan Data *Resource* dan Status *Microservice*




Gambar 4. 6 *Activity* Diagram Kirim Data

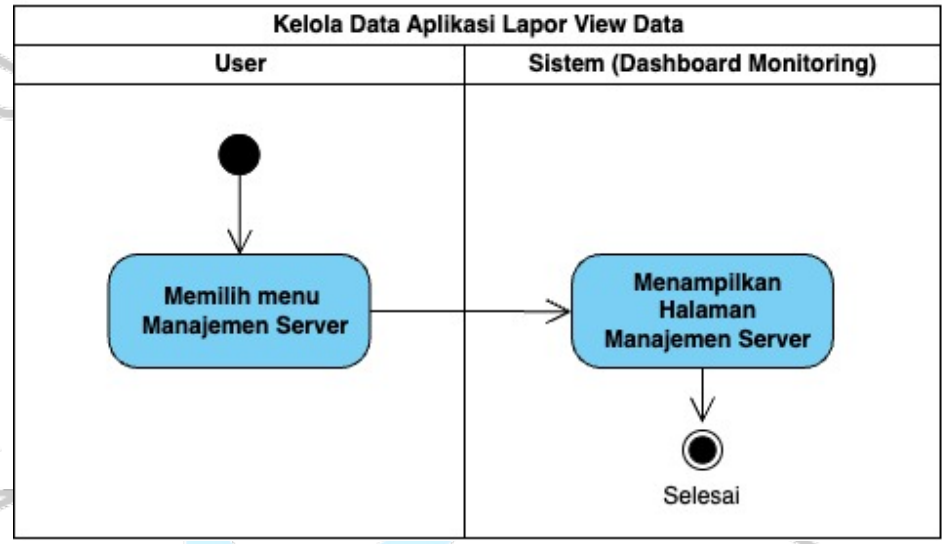
Berdasarkan *Activity* Diagram Mengirimkan Data *Resource* dan Status *Microservice* diatas dapat dijelaskan bahwa Ketika *Scheduler* dari aplikasi lapor aktif. Aplikasi lapor akan mengecek status yang ada di aplikasi tersebut. Ketika status aktif "*Yes*" terverifikasi, aplikasi lapor akan mengekstrak data yang berupa informasi mengenai *resource* server dan status aplikasi *microservice* dari server yang dilakukan *monitoring*. Data yang telah terkumpul tersebut kemudian dikirimkan ke sistem.

Setelah menerima data dari aplikasi lapor, sistem melakukan validasi untuk memastikan bahwa data resource server tidak melebihi batas yang telah ditentukan. Jika melebihi, sistem segera mengirimkan notifikasi melalui Telegram berdasarkan data yang melampaui batas tersebut, dan menyimpan informasi terkait ke dalam *database*.

Namun, jika data dalam batas wajar, sistem akan melanjutkan dengan memeriksa status aplikasi *microservice*. Jika ada yang mati, sistem akan memberikan notifikasi melalui Telegram dan juga menyimpan data ke dalam *database*.

Namun, jika tidak ada masalah, sistem akan langsung menyimpan data yang diterima ke dalam *database* dan memastikan integritas data yang tersimpan.

### 6. *Activity* Diagram Mengelola Aplikasi Lapor

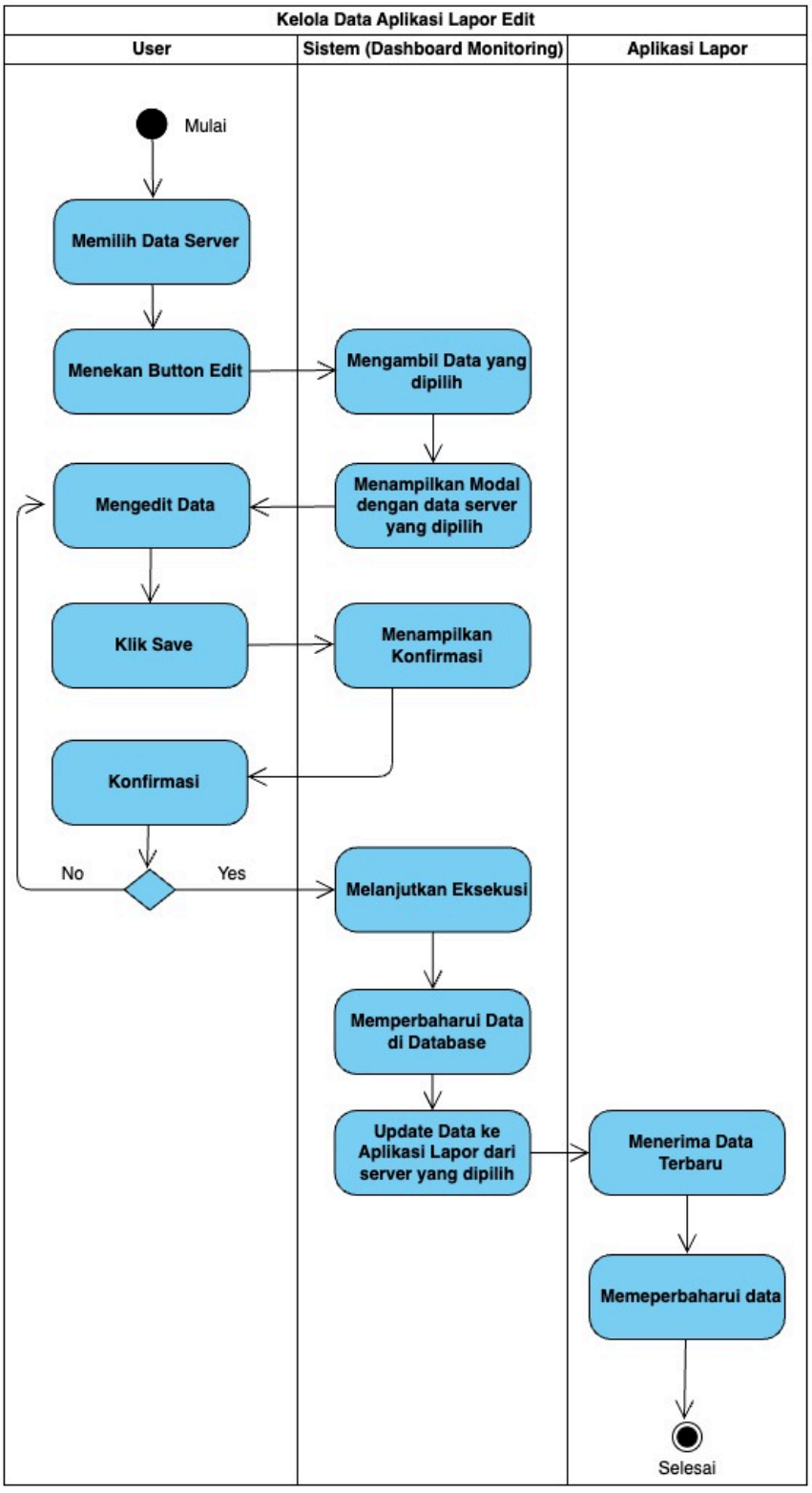



Gambar 4. 7 *Activity* Diagram Mengelola Aplikasi Lapor

Berdasarkan *Activity* Diagram Mengelola Aplikasi Lapor diatas dapat dijelaskan bahwa *User* memulai dengan memilih opsi "Manajemen Server" dari menu *sidebar* pada *dashboard monitoring*. Sistem kemudian menampilkan halaman "*History Monitoring*" yang berisi daftar data server yang di *monitoring*. Setelah menelusuri data server, user memilih satu data yang ingin diedit atau dihapus, kemudian menekan tombol action yang diinginkan.

Berikut merupakan activity diagram lanjutan yang merupakan detail dari proses edit atau hapus data didalam activity diagram mengelola aplikasi lapor.

1) *Edit* Data Server

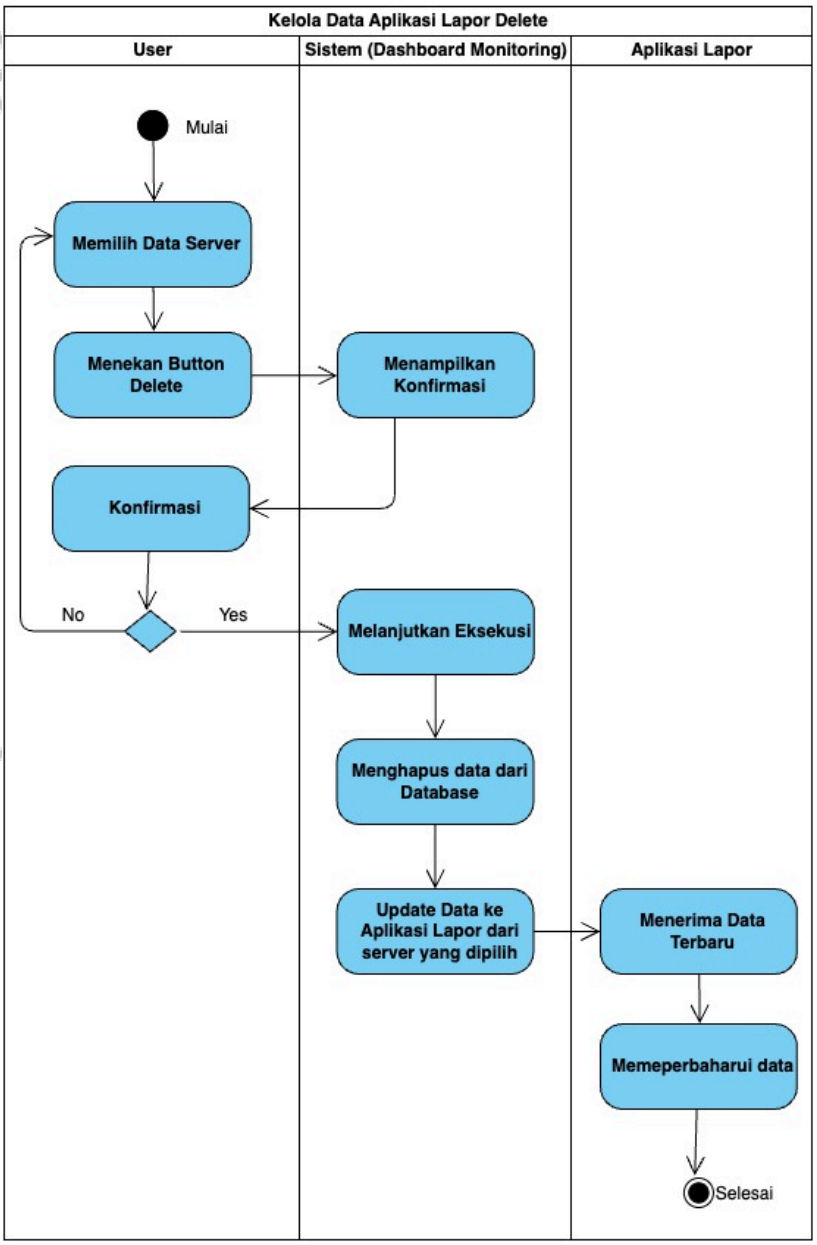



Gambar 4. 8 *Activity* Diagram Mengelola Aplikasi Lapor - *Edit*

Setelah user memilih data server dan menekan tombol "*Edit*", sistem akan mengambil data yang dipilih dan membuka sebuah modal *edit*.

Modal tersebut muncul dengan *field-field* yang sudah terisi oleh data yang dipilih sebelumnya, memungkinkan *user* untuk mengedit informasi yang diinginkan. Setelah *user* menyelesaikan pengeditan, *user* kemudian menekan tombol "*Save*" untuk menyimpan perubahan tersebut. Sistem akan menampilkan konfirmasi untuk memastikan bahwa *user* ingin menyimpan perubahan tersebut. Jika *user* menyetujui, sistem melanjutkan dengan memperbarui data di *database* sesuai dengan informasi yang telah dirubah. Selanjutnya, sistem melakukan *update* data ke aplikasi lapor dengan data yang telah diperbarui. Aplikasi lapor menerima pembaruan data tersebut dan mengintegrasikannya.

2) *Delete* Data Server



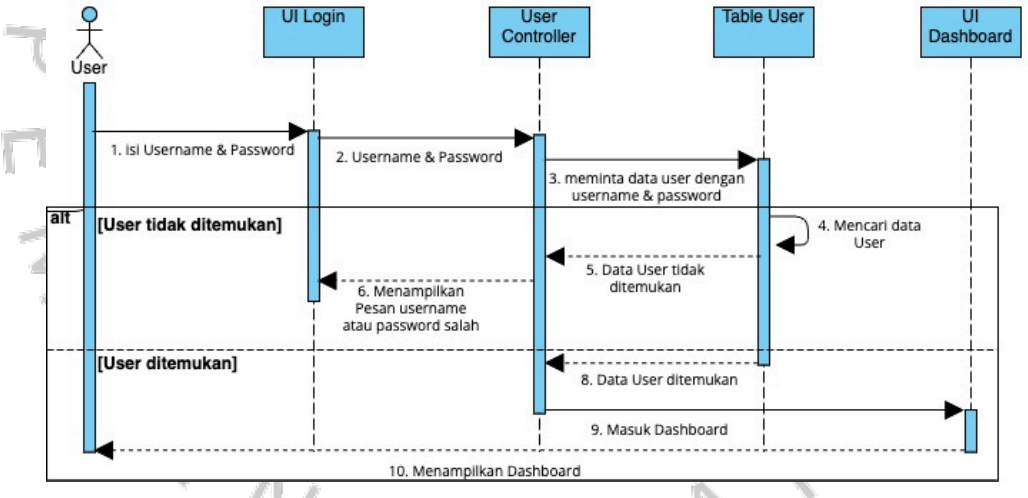

Gambar 4. 9 *Activity* Diagram Mengelola Aplikasi Lapor - *Delete*

Selain itu setelah user memilih data server dan menekan tombol "*Delete*", sistem memberikan konfirmasi terlebih dahulu untuk memastikan bahwa *user* benar-benar ingin menghapus data tersebut. Jika *user* menyetujui penghapusan, sistem melanjutkan dengan memperbarui status data di *database* bahwa data tersebut dihapus oleh *user*. Kemudian, seperti sebelumnya, sistem mengirimkan pembaruan data ke aplikasi lapor. Kemudian aplikasi lapor menerima dan memperbaharui data sesuai dengan perubahan yang dilakukan oleh *user*. Jika *user* memilih untuk tidak menghapus, sistem kembali menampilkan *dashboard* "Manajemen Server" untuk *user*.

### 4.2.3 *Sequence* Diagram

*Sequence* diagram merupakan alat penting untuk mengoptimalkan alur komunikasi dan koordinasi yang terjadi didalam aplikasi *monitoring*. Berikut adalah *sequence* diagram yang dirancang berdasarkan *use case* yang telah dibuat.

1. ***Sequence* Diagram *Login***




Gambar 4. 10 *Sequence* Digram *Login*

Gambar 4.10 menggambarkan diagram urutan (*sequence* diagram) untuk proses pengiriman *login*. Diagram ini melibatkan beberapa komponen utama, yaitu *User*, *User Controller*, Table *User* dan UI *Dashboard*.

Proses login dimulai ketika user memasukkan *username* dan *password* ke dalam antarmuka *login* (UI *Login*). Setelah pengguna memasukkan kredensial, UI *Login* mengirimkan informasi tersebut ke *User Controller* untuk dilakukan verifikasi.

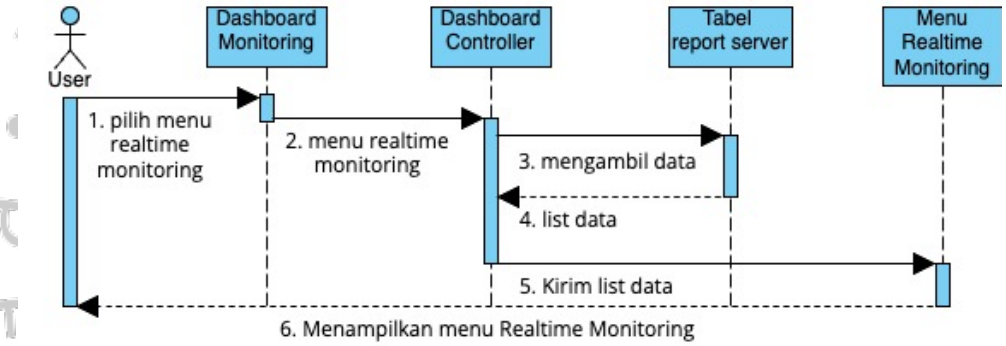
*User Controller* menerima *username* dan *password* dari UI *Login* dan mengirimkan permintaan ke Table *User* untuk memeriksa apakah data *user*

yang diberikan ada di dalam *database*. Table *User* kemudian mencari data user berdasarkan *username* dan *password* yang telah diberikan.

Jika Tabel *User* tidak menemukan data *user* yang cocok, maka akan dikirimkan pesan ke *User Controller* bahwa data *user* tidak ditemukan. *User Controller* kemudian mengirimkan pesan ke UI *Login* untuk menampilkan pesan kesalahan kepada *user* yaitu memberi tahu bahwa *username* atau *password* yang dimasukkan salah.

Namun, jika Tabel *User* menemukan data *user* yang cocok, maka data pengguna tersebut dikirimkan kembali ke *User Controller*. Setelah menerima data *user*, *User Controller* mengirimkan sinyal ke UI *Dashboard* untuk mempersiapkan tampilan *dashboard* bagi *user*. UI *Dashboard* kemudian ditampilkan kepada *user*, sehingga user dapat mengakses berbagai fitur dan data yang tersedia dalam aplikasi *monitoring*.

## 2. *Sequence* Diagram *Monitoring* Server



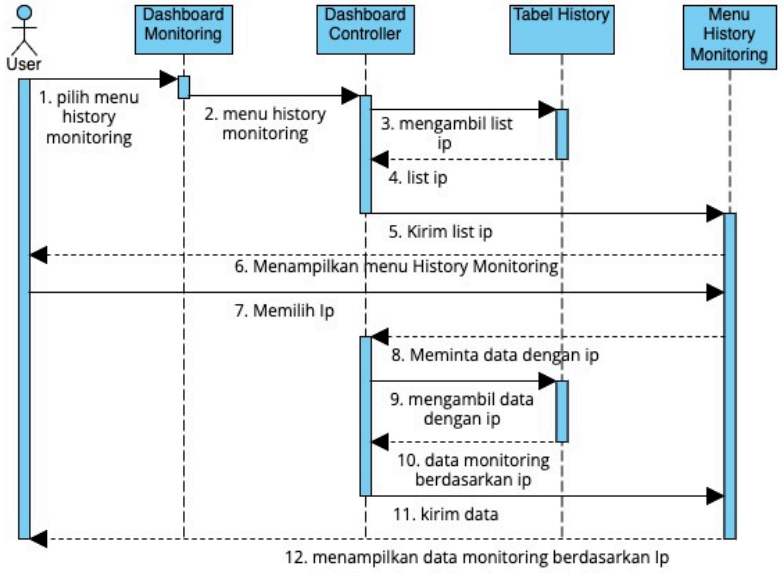

Gambar 4. 11 *Sequence* Diagram *Monitoring Server*

Gambar 4.11 menggambarkan diagram urutan (*sequence* diagram) untuk proses *monitoring* server. Diagram ini melibatkan beberapa komponen utama, yaitu *Dashboard Monitoring*, *Dashboard Controller*, Tabel *report server* dan Menu *Realtime Monitoring*.

Proses dimulai ketika *user* memilih menu "*Realtime Monitoring*" pada UI *dashboard monitoring*. Setelah *user* memilih menu *realtime monitoring*, permintaan tersebut dikirimkan dari ui *dashboard monitoring* ke *Dashboard Controller*. *Dashboard Controller*, yang bertanggung jawab memproses permintaan tersebut, kemudian mengirimkan permintaan *user* ke tabel *report* server untuk mengambil data yang diperlukan untuk ditampilkan yaitu berupa data *resource* server dan status aplikasi *microservice* dari berbagai server.

Tabel *report* server menerima permintaan dari *Dashboard Controller* dan mulai mengambil data yang diminta dari *database*. Setelah proses pengambilan data selesai, tabel *report* server mengirimkan data yang telah diambil ke *Dashboard Controller*. *Dashboard Controller* kemudian menerima data dari tabel *report* server dan mengirimkannya ke menu "*Realtime Monitoring*" untuk ditampilkan kepada *user*.

### 3. *Sequence* Diagram *Monitoring History* Server




Gambar 4. 12 *Sequence* Diagram *Monitoring History Server*

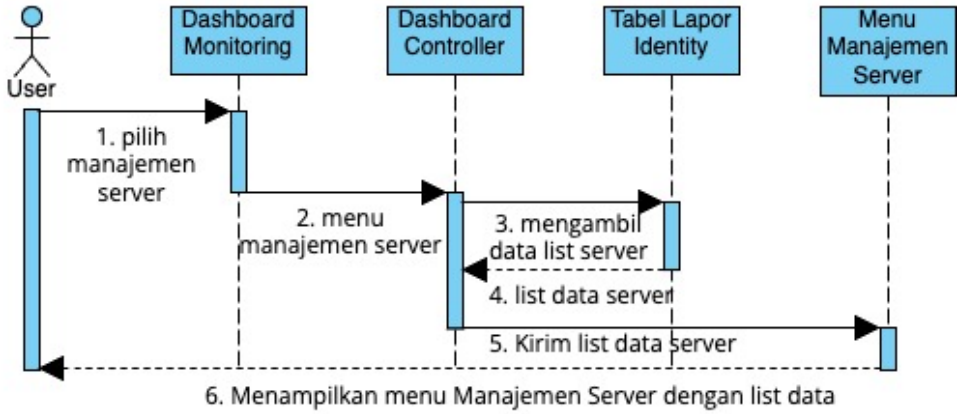
Gambar 4.12 menggambarkan diagram urutan (*sequence* diagram) untuk proses *monitoring history* server. Diagram ini melibatkan beberapa komponen utama, yaitu *Dashboard Monitoring*, *Dashboard Controller*, Tabel *Historing* dan Menu *History Monitoring*.

Proses dimulai ketika *user* memilih menu "*History Monitoring*" pada ui *dashboard monitoring*. Permintaan tersebut kemudian dikirimkan ke *Dashboard Controller* untuk diproses lebih lanjut. *Dashboard Controller* menerima permintaan tersebut dan mengirimkannya ke Table *Report* Server untuk mengambil daftar IP yang tersedia. Tabel *report* server memproses permintaan tersebut dengan mengambil daftar IP dari *database* dan mengirimkannya kembali ke *Dashboard Controller*.

Setelah menerima daftar IP dari tabel *report* server, *Dashboard Controller* mengirimkan daftar tersebut ke ui "*History Monitoring*". Menu *history Monitoring* kemudian menampilkan daftar IP kepada *user*. *User* kemudian memilih salah satu IP dari yang telah ditampilkan. Setelah *user* memilih IP, *Dashboard Controller* mengirimkan permintaan ke tabel *report* server untuk mengambil data *history monitoring* yang terkait dengan IP tersebut.

Tabel *history* menerima permintaan tersebut dan mengambil data *history monitoring* yang sesuai dengan ip tersebut dari *database*. Data *history monitoring* yang telah diambil kemudian dikirimkan kembali ke *Dashboard Controller*. *Dashboard Controller* menerima data *history monitoring* tersebut dan mengirimkannya ke menu "*History Monitoring*". Setelah itu, menu "*History Monitoring*" menampilkan data *history monitoring* yang terkait dengan IP yang dipilih kepada *user*.

**4. *Sequence* Diagram Mengelola Aplikasi Lapor**




Gambar 4. 13 *Sequence* Diagram *Mengelola Aplikasi Lapor*

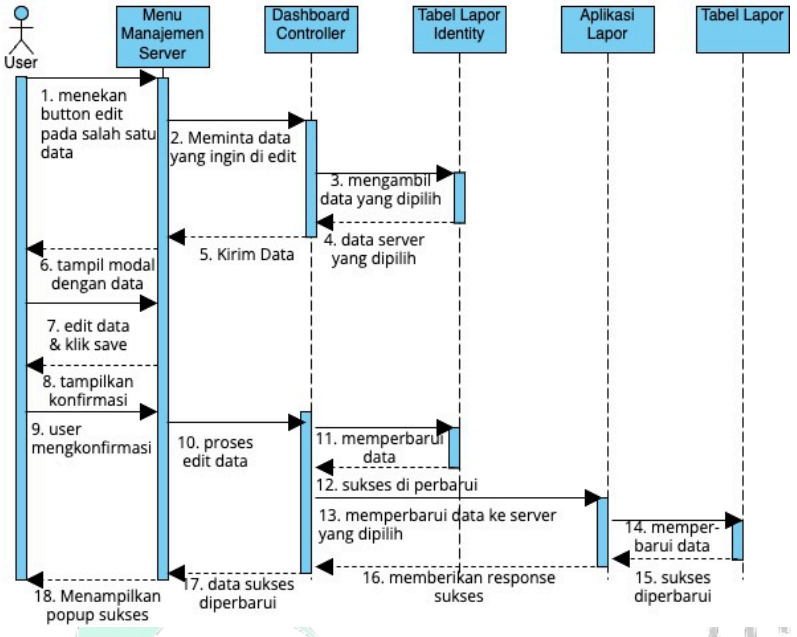
Gambar 4.13 menggambarkan diagram urutan (*sequence* diagram) untuk sistem manajemen server yang terdiri dari beberapa komponen, termasuk pengguna (*User*), *Dashboard Monitoring*, *Dashboard Controller*, Tabel Lapor Identity, Menu Manajemen Server, dan Aplikasi Lapor. Diagram ini memvisualisasikan alur kerja dari tiga proses utama: melihat daftar server, merubah data server, dan menghapus data server.

Pada proses pertama, *user* memilih menu manajemen server di *Dashboard Monitoring*. *Dashboard Controller* kemudian menampilkan daftar server dengan mengambil data dari tabel Lapor Identity dan mengirimkannya kembali ke *Controller Dashboard Monitoring*. Data tersebut kemudian akan dikirim ke menu Manajemen Server dan ditampilkan kepada *user*.

Setelah data ditampilkan, *user* akan dapat melihat list dari server yang telah dipasangkan aplikasi lapor. Didalam menut tersebut *user* dapat memilih salah satu data dari list data server tersebut. Setelah memilih data, *user* dapat melakukan *edit* atau *delete* terhadap data tersebut dengan menekan button *edit* atau *delete* yang ada.

Ketika *user* melakukan salah satu dari kedua action tersebut sistem juga akan melakukan perubahan data ke aplikasi lapor di server yang dipilih, adapun detail proses tersebut dapat dijelaskan melalui sequence diagram dibawah ini adalah sebagai berikut:

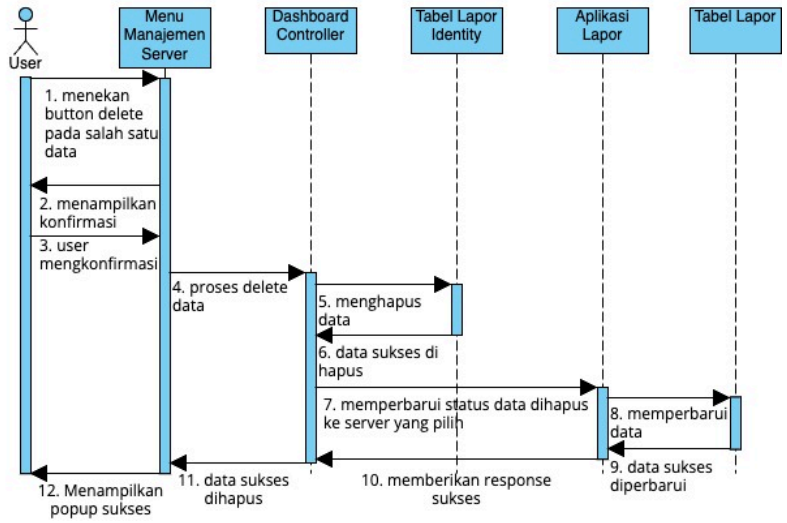
1) *Edit*




Gambar 4. 14 *Sequence* Diagram Mengelola Aplikasi Lapor - *Edit*

Untuk proses perubahan data server, *user* menekan tombol edit pada data server yang dipilih. Menu Manajemen Server kemudian meminta data server yang dipilih dari Tabel Lapor *Identity* dan menampilkan halaman modal untuk merubah data server tersebut. Setelah *user* merubah data dan menekan tombol simpan, sistem akan meminta konfirmasi dari *user*. Setelah konfirmasi diberikan, data server diperbarui di tabel Lapor *Identity* dan perubahan tersebut dikirim juga ke aplikasi lapor. Aplikasi Lapor menerima perubahan data tersebut dan melakukan perubahan juga ke tabel Lapor, Setelah itu aplikasi lapor mengkonfirmasi ke *dashboard controller* bahwa data telah di rubah.

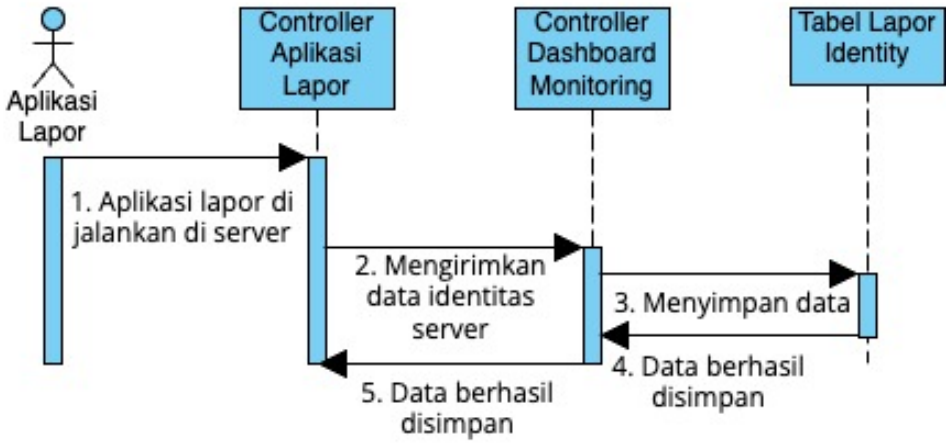
2) *Delete*




Gambar 4. 15 *Sequence* Diagram Mengelola Aplikasi Lapor - *Delete*

Pada proses penghapusan data server, pengguna menekan tombol *delete* pada data server yang dipilih. Menu Manajemen Server memproses permintaan penghapusan dengan meminta konfirmasi dari pengguna. Setelah pengguna memberikan konfirmasi, data server tersebut di perbarui ke tabel Lapor *Identitity* untuk dirubah statusnya bahwa data server tersebut telah dihapus dan perubahan ini dikirim juga ke aplikasi lapor. Aplikasi lapor pun menerima perubahan tersebut dan melakukan perubahan di tabel Lapor, Setelah itu aplikasi lapor mengkonfirmasi ke *dashboard controller* bahwa data telah di rubah.

**5. *Sequence* Diagram *Update* Data Server**

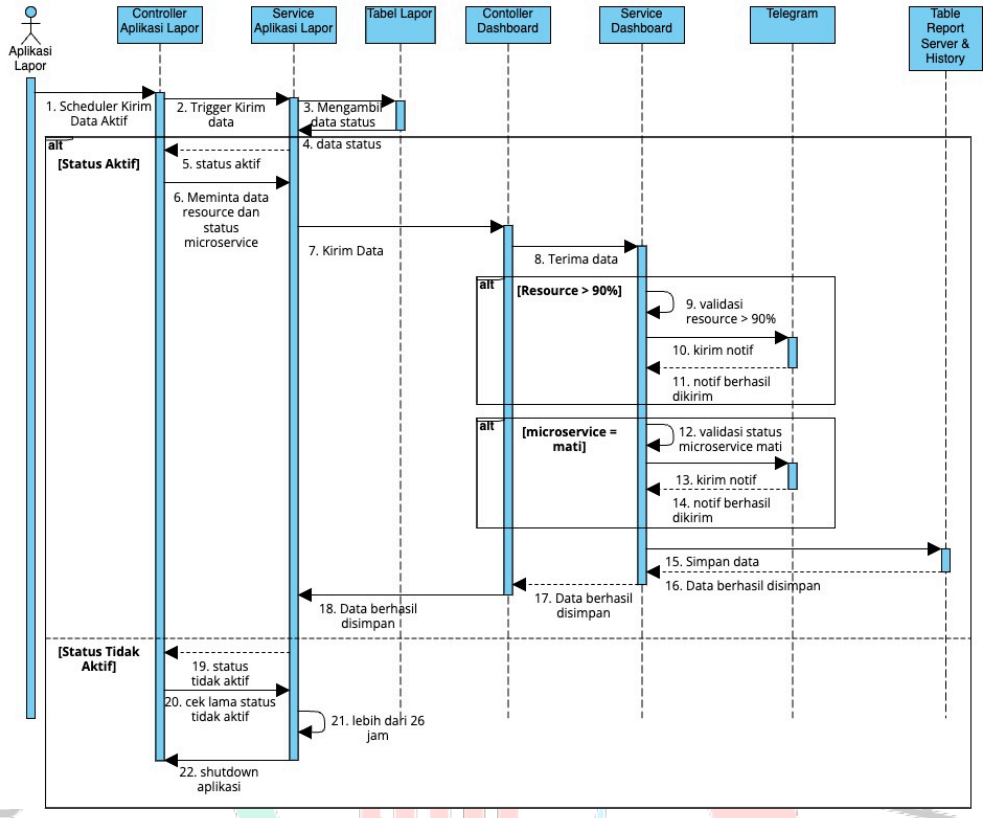



Gambar 4. 16 *Sequence* Diagram *Update Data*

Gambar 4.12 menunjukkan diagram urutan (*sequence* diagram) untuk proses pelaporan identitas server melalui Aplikasi Lapor. Diagram ini melibatkan beberapa komponen utama: Aplikasi Lapor, Controller Aplikasi Lapor, *Controller Dashboard Monitoring*, dan Tabel Lapor *Identity*.

Proses dimulai ketika Aplikasi Lapor dijalankan di server yang akan dilakukan *monitoring* yang kemudian melakukan *trigger* ke *Controller* Aplikasi Lapor agar dapat mengirimkan data identitas server ke *Controller Dashboard Monitoring*. Setelah menyiapkan data yang akan dikirim, *Controller* Aplikasi Lapor meneruskan data tersebut ke *Controller Dashboard Monitoring*. Kemudian *Dashboard Monitoring* menerima data tersebut dan menyimpannya ke tabel lapor *identity*, setelah data berhasil disimpan tabel lapor identity akan memberikan konfirmasi data berhasil disimpan sebagai *response* untuk *Controller Dashboard Monitoring*. Terakhir *Controller Dashboard Monitoring* menyampaikan pesan data berhasil disimpan ke *Controller* Aplikasi Lapor dan proses selesai.

## 6. *Sequence* Diagram Mengirimkan Data *Resource* dan Status *Microservice*




Gambar 4. 17 *Sequence* Diagram Kirim Data

Gambar 4.13 menggambarkan diagram urutan (*sequence* diagram) untuk proses pengiriman data dan pemantauan status dalam Aplikasi Lapor. Diagram ini melibatkan beberapa komponen utama, yaitu Aplikasi Lapor, *Controller* Aplikasi Lapor, *Service* Aplikasi Lapor, *Controller Dashboard*, *Service Dashboard*, Telegram, dan Table *Report* Server.

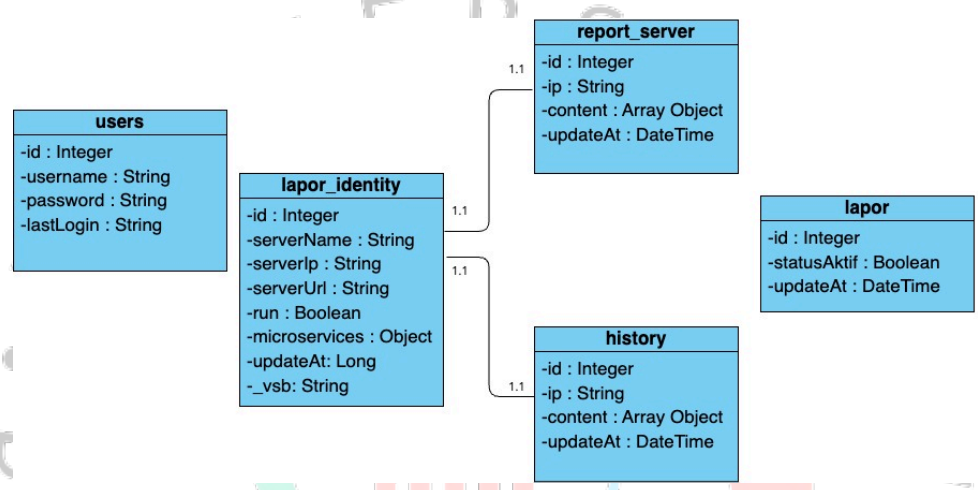
Proses dimulai dengan *Scheduler* Kirim Data Aktif di Aplikasi Lapor, yang memicu pengiriman data. *Controller* Aplikasi Lapor kemudian memeriksa status sistem dengan meminta data status dari *Service* Aplikasi Lapor yang berasal dari tabel lapor. Jika status dikonfirmasi aktif oleh controller aplikasi lapor, *controller* aplikasi lapor meminta data *resource* dan status aplikasi lapor ke *service* aplikasi lapor dan kemudian dikirim ke *Controller Dashboard*.

Setelah *Controller Dashboard* menerima data, dilakukan validasi terhadap dua kondisi utama: pertama, jika penggunaan sumber daya (*resource*) lebih dari 90%, maka sistem akan mengirim notifikasi melalui Telegram; kedua, jika terdapat aplikasi *microservice* yang mati, notifikasi juga akan dikirim melalui Telegram mengenai aplikasi yang mati tersebut. Setelah validasi dan pengiriman notifikasi, data disimpan ke dalam Tabel *Report* Server & *history*, dan konfirmasi penyimpanan data dikirim kembali ke Service Aplikasi Lapor.

Jika status dikonfirmasi tidak aktif oleh *controller* aplikasi lapor, *Controller* Aplikasi Lapor akan mengecek apabila status tidak aktif selama lebih dari 26 jam maka *service* aplikasi lapor akan melakukan *shutdown* pada Aplikasi Lapor.

### 4.2.4 *Class* Diagram

Berikut merupakan rancangan *class* diagram yang dibuat berdasarkan tabel database dari hasil penggambaran *sequence* diagram sebelumnya yang telah dibuat.




Gambar 4. 18 *Class* Diagram

Berdasarkan *class* diagram diatas dapat dirancang spesifikasi *database* yang sesuai dengan aplikasi *monitoring*.

1. Tabel *Users*

Tabel 4. 7. *Basisdata Tabel Users*

| Nama Field | Tipe Data | Keterangan |
|---|---|---|
| **_id** | Integer | ID pengguna (primary key) |
| **username** | String | Nama pengguna |
| **password** | String | Kata sandi pengguna |
| **lastlogin** | DateTime | Tanggal dan waktu terakhir kali login |

Tabel 4.7 merupakan tabel yang berfungsi untuk menyimpan data yang digunakan untuk *login* kedalam aplikasi *monitoring*.

2. Tabel Identitas Lapor

Tabel 4. 8. Basisdata Tabel Identitas Lapor

| Nama Field | Tipe Data | Keterangan |
|---|---|---|
| **_id** | Integer | ID (primary key) |
| **serverName** | String | Nama server |
| **serverIp** | String | IP Server |
| **serverUrl** | String | Endpoint aplikasi lapor yang dijalankan di server yang dimonitoring |
| **statusAktif** | Boolean | Status aktif aplikasi lapor |
| **microservices** | Object | url microservice yang ada di aplikasi lapor |
| **updateAt** | DateTime | Tanggal data di update |
| **_vsb** | String | Visibilitas data (aktif / tidak aktif) |

Tabel 4.8 merupakan tabel yang berfungsi untuk menyimpan data identitas server yang mana server tersebut tempat di pasangnya aplikasi lapor yang digunakan untuk melaporkan data untuk dilakukan *monitoring*.

3. Tabel *Report* Server dan *History*

Tabel 4. 9. Basisdata Tabel *Report* Server dan *History*

| Nama Field | Tipe Data | Keterangan |
|---|---|---|
| **_id** | Integer | ID (primary key) |
| **ip** | String | Nama pengguna |
| **content** | Array Object | Berisi data resource server dan status aplikasi microservice |
| **updateAt** | DateTime | Tanggal & waktu data di terima |

Tabel 4.9 merupakan tabel yang berfungsi untuk menyimpan data aktual (untuk tabel *report* server) dan juga data history (untuk tabel *history*) yang dikirimkan oleh aplikasi lapor yaitu berupa data *resource* server dan status aplikasi *microservice*. Data tersebut berada didalam *field content* yang merupakan tipe data *Array Object*, Mengenai detail dari field content diatas adalah sebagai berikut.

Tabel 4. 10. Tabel Detail *Content*

| Nama | Tipe Data | Keterangan |
|---|---|---|
| **type** | String | Tipe dari data yang dikirim (PERCENTAGE, STATUS) |

| | | |
|---|---|---|
| **name** | String | Nama data tersebut (STORAGE, MEMORY, CPU, MICROSERVICE) |
| **used** | Double | Angka Penggunaan resource |
| **usedPercentage** | Double | Persentase dari penggunaan resource |
| **free** | Double | Angka space kosong dari resource |
| **freePercentage** | Double | Persentase space kosong dari resource |
| **status** | Object | Berisi status microservice |

Tabel 4.10 merupakan tabel detail dari *field content* yang ada pada tabel *report* server dan *history*. Data dari tabel tersebut yang akan diolah oleh sistem untuk validasi ketika terdapat data *resource* yang melebihi batas yang di tentukan, maupun status *microservice* yang tidak berjalan di server yang mengirimkan data tersebut. Selain itu, data tersebut akan di olah juga oleh sistem agar dapat ditampilkan di halaman dashboard guna kepentingan *monitoring*.

4. Tabel Lapor

Tabel 4. 11. Tabel Lapor

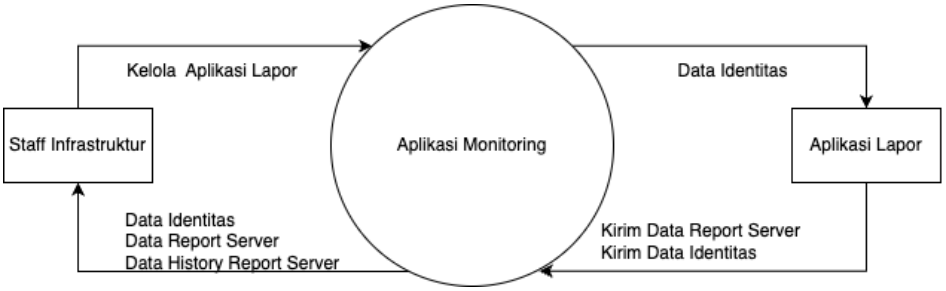
| Nama Field | Tipe Data | Keterangan |
|---|---|---|
| **_id** | ObjectId | ID (*primary key*) |
| **statusAktif** | Boolean | Status yang menjadi acuan aktif atau tidaknya aplikasi lapor |
| **updateAt** | DateTime | Tanggal & waktu data di ubah |

Tabel 4.11 merupakan tabel yang berfungsi untuk menyimpan data status aplikasi lapor sebagai acuan untuk setiap aksi yang dilakukan oleh aplikasi lapor didalam proses mengirimkan data *monitoring*.

### 4.2.5 Data Flow Diagram

Data Flow Diagram (DFD) memiliki beberapa tingkatan, yaitu diagram konteks (level 0), dan level 1. Berikut adalah DFD untuk aplikasi Monitoring dengan beberapa tingkatan tersebut.

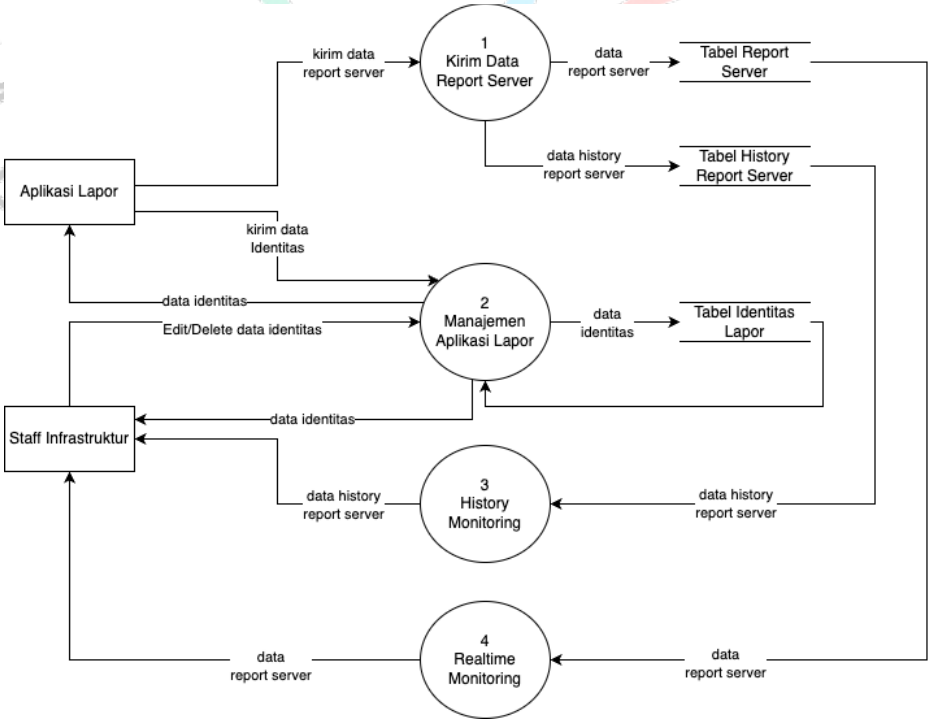
1. DFD Level 0




Gambar 4. 19 DFD Level 0 Aplikasi *Monitoring*

Gambar 4.19 diatas merupakan data flow diagram level 0 untuk aplikasi monitoring, diagram tersebut memiliki 2 entity yaitu Staff Infrastruktur dan Aplikasi Lapor, kemudian 1 proses yaitu Aplikasi Monitoring. Staff infrastruktur dapat melakukan monitoring dengan menerima data report server dan history report server, selain itu staff infrastruktur dapat mengelola data identitas dengan menerima data identitas dan melakukan perubahan data terhadap data identitas tersebut ke aplikasi lapor. Selain itu terdapat Aplikasi lapor yang melakukan pengiriman data report server dan juga data identitas ke aplikasi monitoring, kemudian aplikasi lapor dapat menerima data identitas apabila terdapat perubahan.

2. DFD Level 1




Gambar 4. 20 DFD Level 1 Aplikasi Monitoring

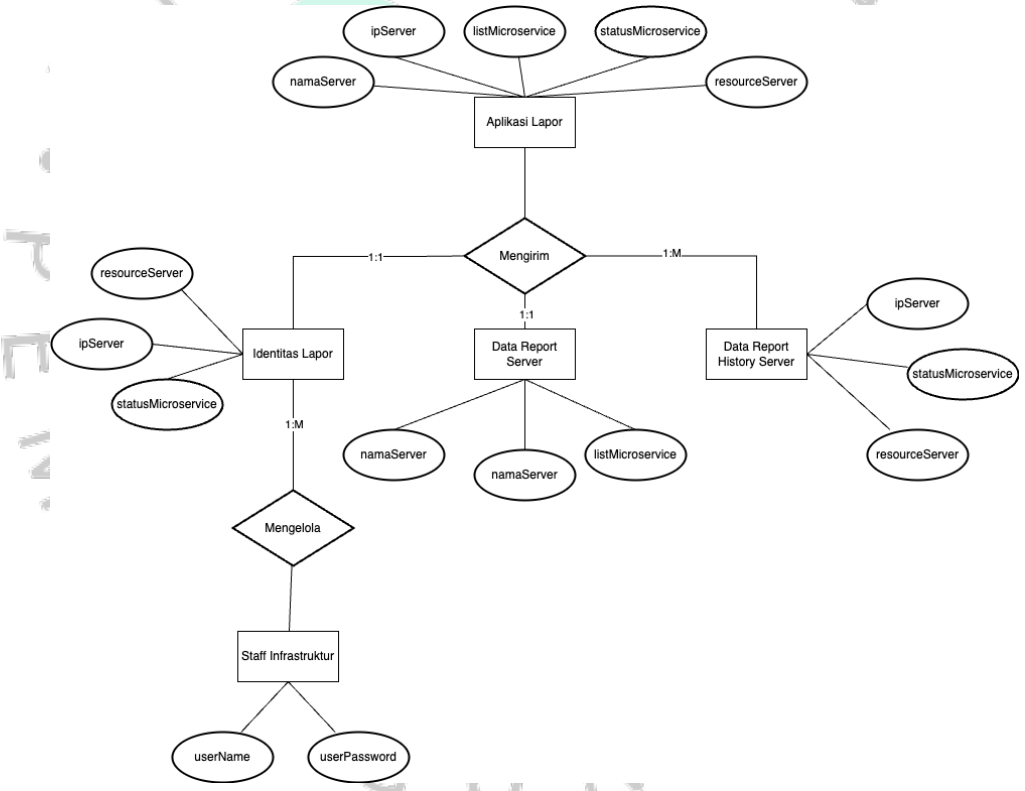
Gambar 4.20 diatas merupakan data flow diagram level 1 untuk aplikasi monitoring. Didalam diagram tersebut terdapat proses Kirim Data Server yang dilakukan oleh Aplikas Lapor, kemudian terdapat Manajemen Aplikasi Lapor yang dapat dilakukan oleh aplikasi lapor dan juga staff infrastruktur. Setelah itu terdapat History Monitoring dan Realtime Monitoring yang dapat dilakukan oleh Staff Infrastruktur. Mengenai detail dari proses DFD level 1 tersebut dapat dijelaskan pada tabel dibawah ini

Tabel 4. 12 Detail Proses DFD Level 1 Aplikasi *Monitoring*

| No | Proses | Input | Output | Deskripsi |
|---|---|---|---|---|
| 1 | Kirim Data Report Server | -ipServer<br>-resourceServer<br>-statusMicroservice | Data report server dan data history report server | Proses untuk mengirimkan data report server oleh aplikasi lapor yang kemudian data tersebut masuk ke tabel report server untuk dilakukan update dan masuk ke tabel history report server untuk disimpan |
| 2 | Manajemen Aplikasi Lapor | -namaServer<br>-ipServer | Data Identitas | Proses mengelola data identitas seperti melakukan melihat, update, delete dan bahkan menambahkan data oleh staff infrastruktur dan aplikasi lapor |
| 3 | History Monitoring | - | -ipServer<br>-resourceServer<br>-statusMicroservice | Proses menampilkan data resource server dan status microservice dari report |

| | | | | |
|---|---|---|---|---|
| | | | | server untuk staff infrastruktur |
| 4 | Realtime Monitoring | - | -ipServer<br>-resourceServer<br>-statusMicroservice | Proses menampilkan seluruh data resource server dan status microservice dari report history server untuk staff infrastruktur |

### 4.2.6 Entity Relationship Diagram



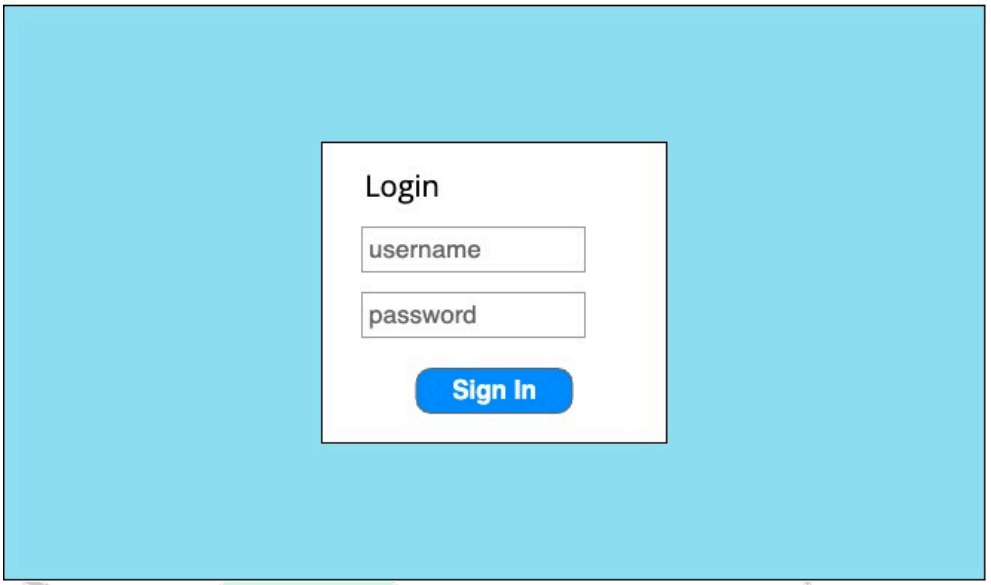

Gambar 4. 21 ERD Aplikasi Monitoring

Gambar 4.20 diatas merupakan entity relationship diagram untuk aplikasi monitoring. Diagram tersebut mencakup entitas, atribut dan proses yang terjadi saat user (Staff infrastruktur) dan aplikasi lapor berinteraksi dengan aplikasi monitoring. Berdasarkan diagram diatas terdapat aplikasi lapor yang mengirimkan data report server, identitas lapor dan report history server ke aplikasi monitoring, Kemudian terdapat user (staff infrastruktur) yang dapat mengelola data identitas lapor didalam aplikasi monitoring.

## 4.3 Perancangan Antar Muka Pengguna

Didalam merancang aplikasi *monitoring* diperlukan perancangan antar muka (*User Interface*) pengguna yang dirancang sesuai dengan kebutuhan dengan fokus untuk meningkatkan kenyamanan pengguna didalam memakai aplikasi *monitoring*. Berikut merupakan rancangan antar muka pengguna dari aplikasi *monitoring*.

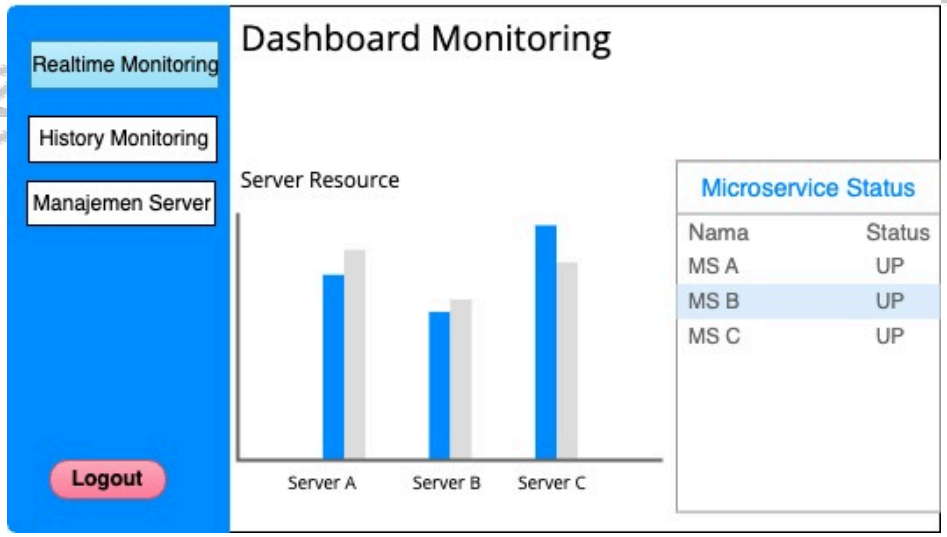
1. Antar Muka *Login*




Gambar 4. 22 Rancangan Halaman *Login*

Pada gambar 4.19 menampilkan antarmuka *login*, Fungsi dari halaman ini adalah untuk memungkinkan pengguna memasukkan nama pengguna (*username*) dan kata sandi (*password*) mereka untuk mengakses aplikasi *monitoring* setelah menekan tombol "*Sign In*".

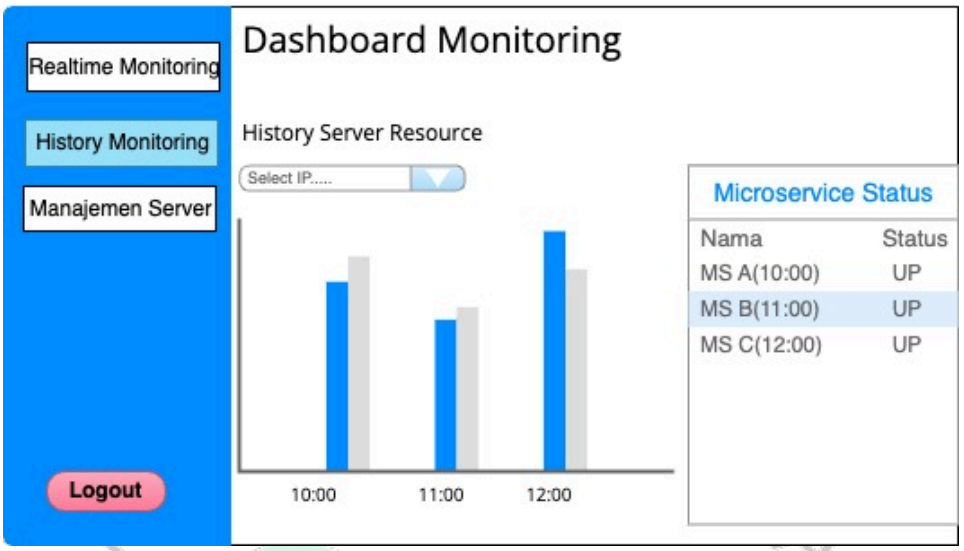
2. Antar Muka *Dashboard* / *Realtime Monitoring*




Gambar 4. 23 Rancangan Halaman *Realtime Monitoring*

Pada gambar 4.20 menampilkan antar muka menu *realtime monitoring*, yang mana menu ini memungkinkan pengguna untuk melihat data aktual mengenai data *resource* server dan status aplikasi *microservice* yang ada didalam server tersebut.

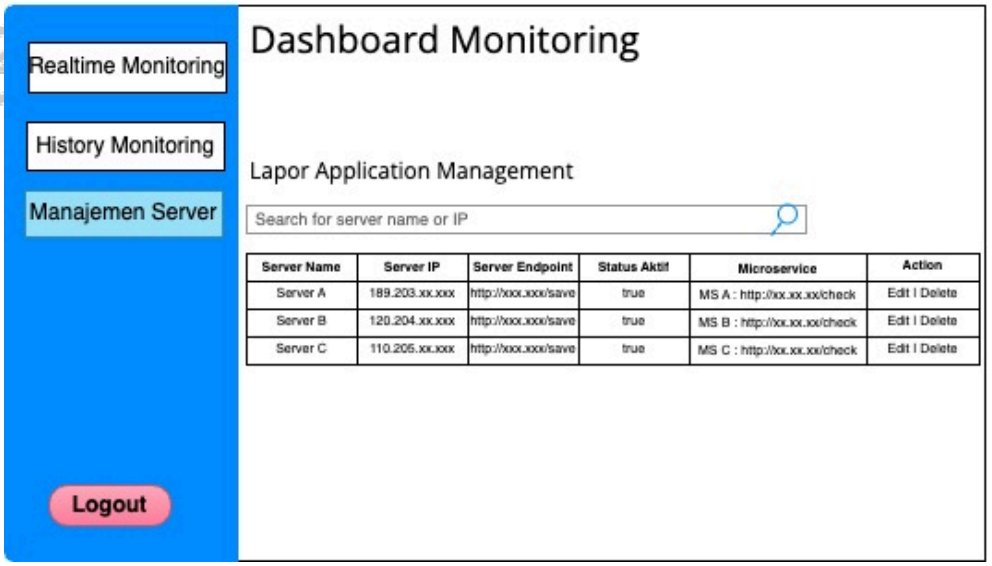
3. Antar Muka *History Monitoring*




Gambar 4. 24 Rancangan Halaman *History Monitoring*

Pada gambar 4.21 menampilkan antar muka menu *history monitoring*, yang mana menu ini memungkinkan pengguna untuk melihat data mengenai *resource* server dan status aplikasi *microservice* dari waktu yang telah lalu atau lebih tepatnya 24 jam terakhir. Untuk menampilkan data tersebut pengguna perlu untuk memilih IP yang diinginkan untuk menampilkan data tersebut.

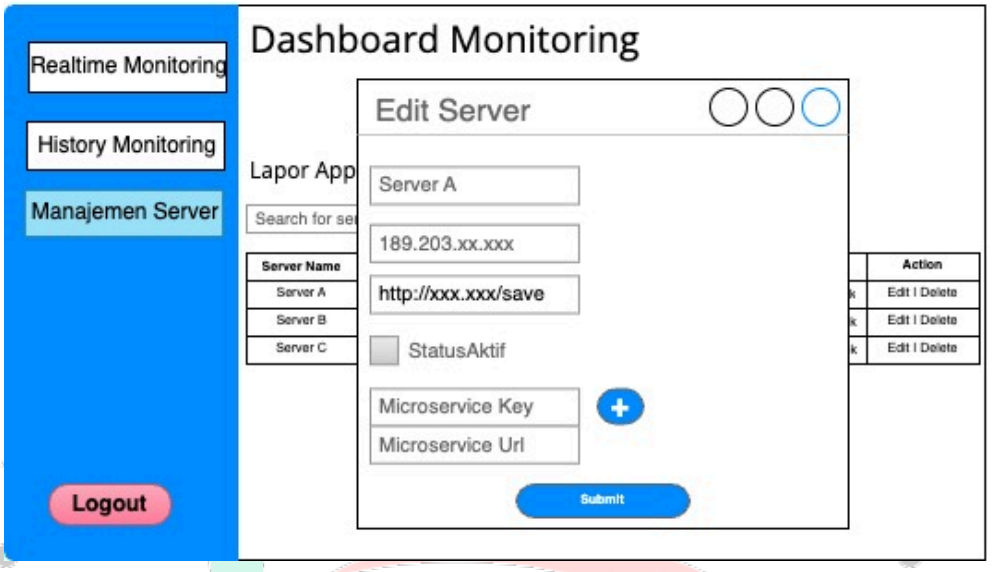
4. Antar Muka Manajemen Server




Gambar 4. 25 Rancangan Halaman Manajemen Server

Pada gambar 4.22 menampilkan antar muka menu majemen server, menu ini merupakan menu yang berfungsi untuk pengelolaan aplikasi lapor yang terpasang di server. Didalam menu ini pengguna dapat

mematikan aplikasi lapor tersebut dan menambahkan aplikasi *microservice* yang ingin dipantau oleh aplikasi monitoring dengan menekan tombol edit yang kemudian akan memunculkan modal seperti gambar 4.23 dibawah ini.




Gambar 4. 26 Rancangan Modal Edit

## 4.4 Perancangan Implementasi

### 4.4.1 Perencanaan Implementasi

Untuk mendukung Rancang Bangun Aplikasi *Monitoring Resource* Server dan *Microservice* Berbasis Web dengan Notifikasi Telegram Menggunakan Pendekatan Waterfall, persiapan alat-alat dalam kebutuhan perangkat keras (*hardware*) serta perangkat lunak (*software*) sangat penting. Berikut adalah rincian alat-alat yang perlu dipersiapkan:

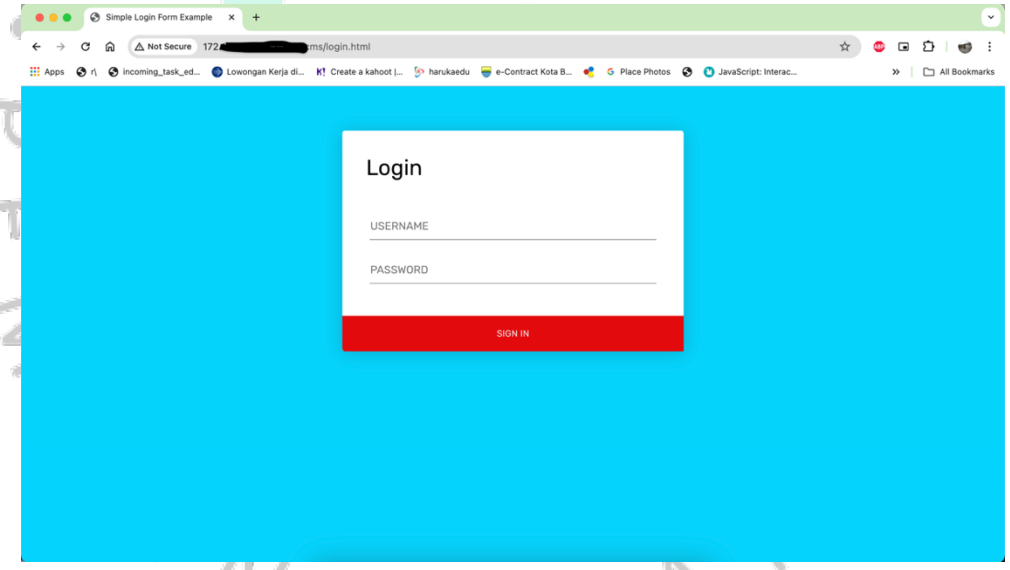
1. Kebutuhan Alat Perangkat Keras (*Hardware*)
   1) Laptop dengan Processor M1 dan RAM 8gb
   2) Harddisk minimal 50 gb untuk penyimpanan data
   3) Wifi dengan kecepatan minimal 10 mb/s
   4) Monitor 20 inch
   5) Kabel Konektor Monitor
2. Kebutuhan Alat Perangkat Lunak (*Software*)
   1) MacOs sebagai sistem operasi
   2) Ampps dengan Apache yang sudah terinstall
   3) Mongodb sebagai basisdata
   4) Intellij IDE dan Visual Studio sebagai text editor
   5) Java dengan versi 13 sebagai bahasa pemrograman
   6) Web Browser sebagai alat untuk melakukan uji coba tampilan
   7) Postman sebagai alat untuk melakukan uji coba API

Sistem akan dibangun dengan kombinasi dari aplikasi *server-side* (*server-side* application) dan aplikasi berbasis web. Aplikasi ini akan menggunakan teknologi-teknologi yang handal dan modern, termasuk Java sebagai bahasa pemrograman utamanya, dan Spring Boot sebagai *framework* yang mendukung pengembangan aplikasi yang kuat dan skalabel. Untuk tampilan antar muka pengguna, aplikasi ini akan menggunakan HTML dan CSS. Selain itu, MongoDB akan digunakan sebagai basis data.

### 4.4.2 Hasil Implementasi

Berikut merupakan hasil implementasi antarmuka berdasarkan perancangan antarmuka yang telah dibuat sebelumnya. Proses implementasi ini melibatkan beberapa tahapan penting yang bertujuan untuk memastikan antarmuka yang dihasilkan tidak hanya estetis, tetapi juga fungsional dan *user-friendly* sesuai dengan kebutuhan pengguna.

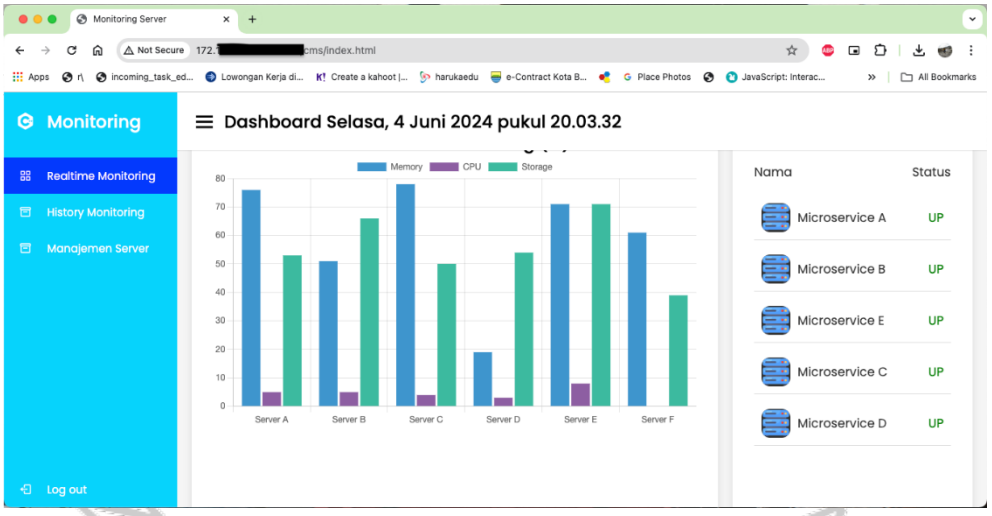
1. Login




Gambar 4. 27 Implementasi *Login*

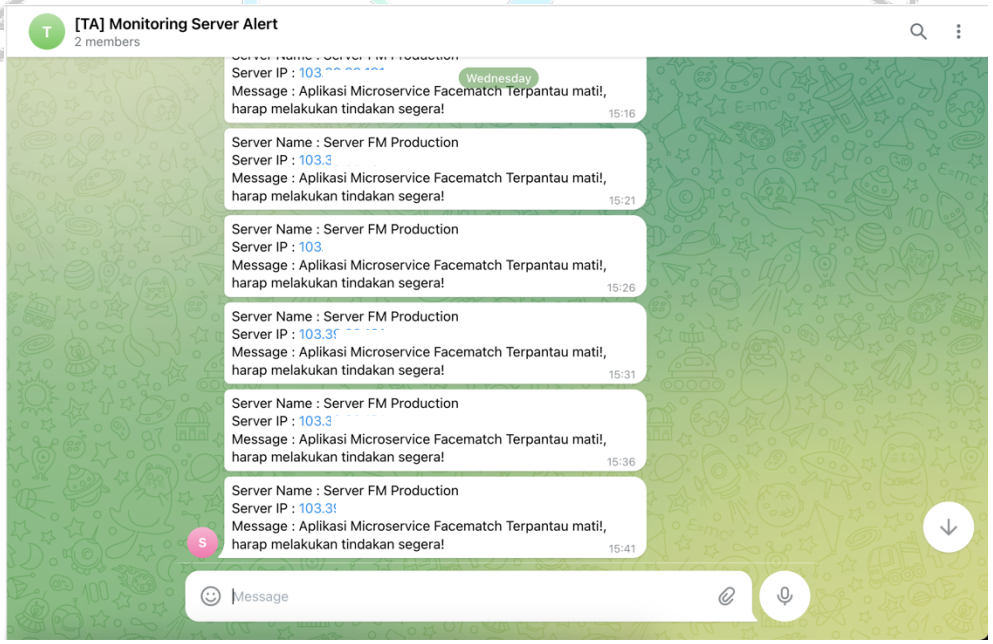
Pada gambar 4.24 merupakan hasil implementasi antarmuka *login* berdasarkan rancangan yang telah dibuat, didalam halaman ini pengguna akan diminta untuk memasukan *username* dan *password* yang telah didaftarkan agar bisa masuk ke halaman *dashboard monitoring*.

2. Realtime Monitoring



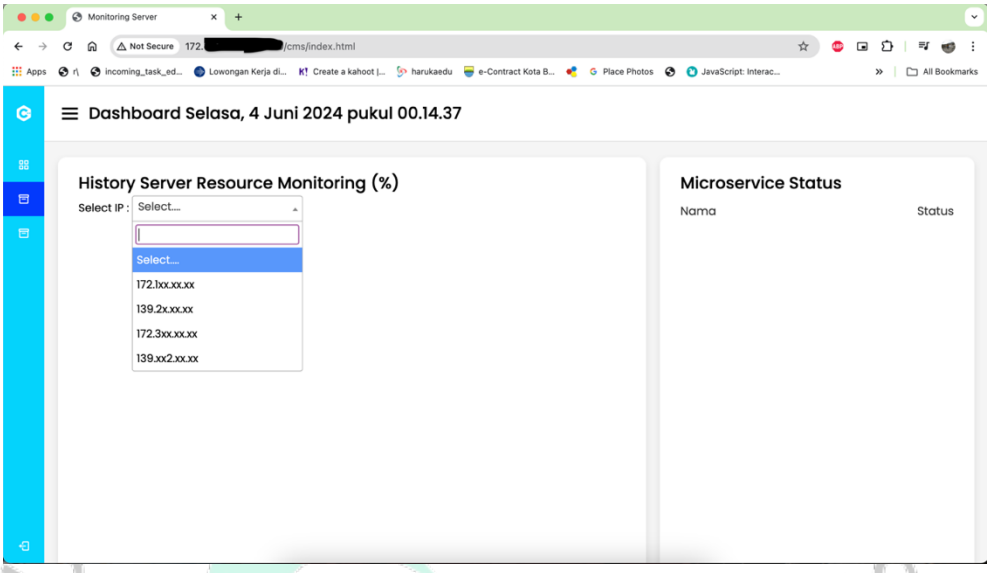

Gambar 4. 28 Implementasi *Realtime Monitoring*

Pada gambar 4.25 merupakan hasil implementasi antar muka untuk menu *Realtime Monitoring* berdasarkan rancangan yang telah dibuat, didalam halaman ini sistem akan menampilkan data terkini mengenai data *resource* dan status aplikasi *microservice* dari berbagai server yang terdaftar (yang terpasang aplikasi lapor). Apabila terdapat data resource baik itu CPU, Memory dan Storage yang melebihi batas wajar atau aplikasi microservice tidak berjalan, maka sistem akan mengirimkan notifikasi melalui telegram sesuai dengan data tersebut, berikut merupakan contoh dari notifikasi telegram yang dikirimkan oleh sistem.



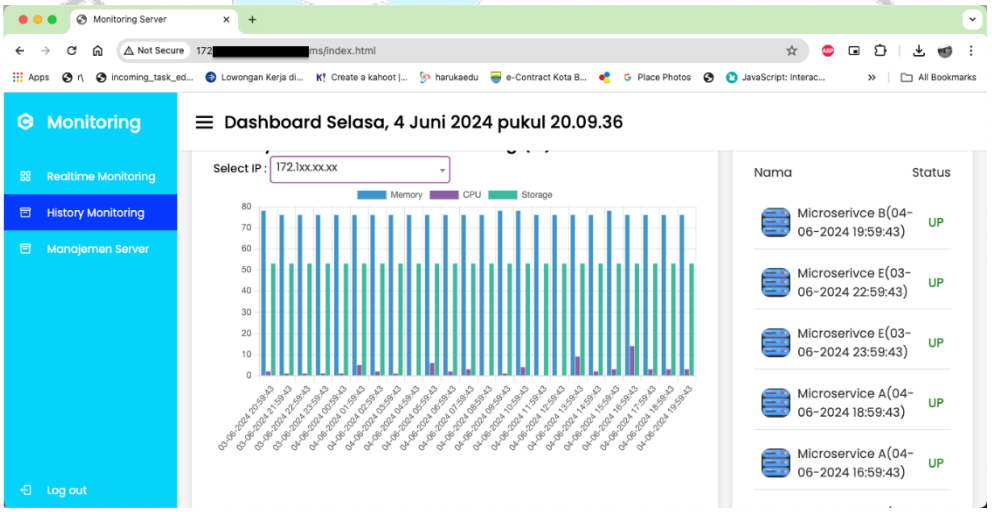

Gambar 4. 29 Notifikasi Telegram

3. History Monitoring



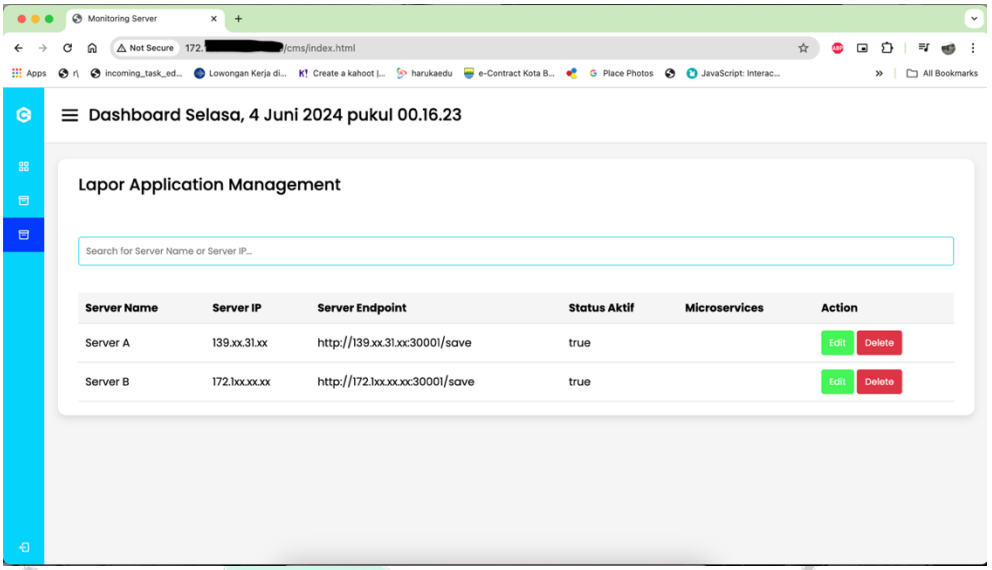

Gambar 4. 30 Implementasi *History Monitoring* A

Pada gambar 4.26 merupakan hasil implementasi antar muka untuk menu *History Monitoring* berdasarkan rancangan yang telah dibuat, didalam halaman tersebut ketika pengguna pertama kali membuka halaman tersebut, hanya akan terdapat field *dropdown* yang berisi list ip. Pengguna akan diminta untuk memilih salah satu IP yang diinginkan untuk melihat data *history resource* dan status aplikasi *microservice*. Setelah pengguna memilih IP, maka sistem akan mengambil data *resource* dan status aplikasi *microservice* berdasarkan IP yang telah dipilih selama 24 jam terakhir seperti pada gambar 4.27 dibawah.



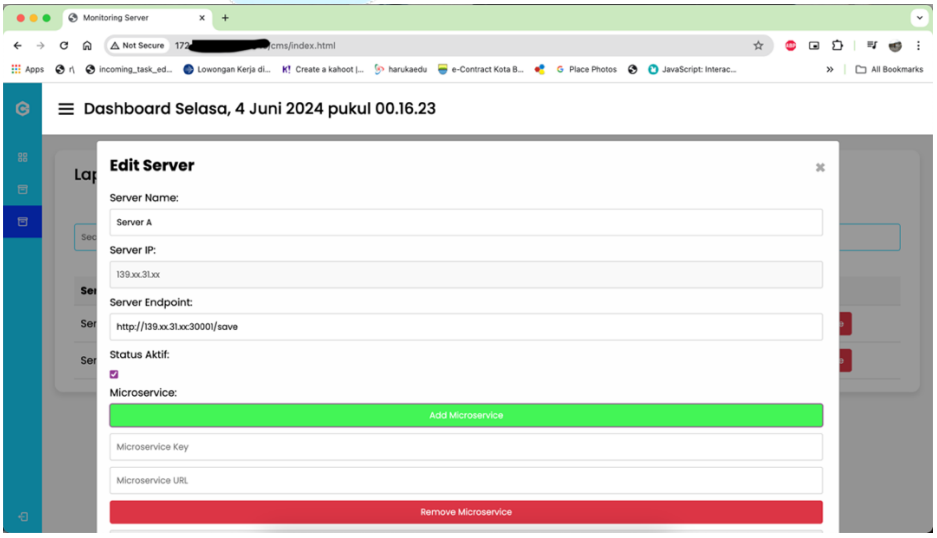

Gambar 4. 31 Implementasi *History Monitoring* B

4. Manajemen Server




Gambar 4. 32 Implementasi Manajemen Server

Pada gambar 4.28 merupakan hasil implementasi antar muka untuk menu Manajemen Server berdasarkan rancangan yang telah dibuat, didalam menu tersebut pengguna dapat mengelola data server yang terpasang aplikasi lapor didalam *dashboard monitoring* tersebut. Pengguna dapat mencari data server yang diinginkan dengan fitur pencarian yang terdapat diatas tabel. Selain itu pengguna dapat menghapus data server dengan menekan *button delete* yang ada di kolom Action pada data server yang diinginkan. Didalam kolom *Action* juga terdapat *button edit* yang memungkinkan pengguna dapat mengubah data server, ketika pengguna menekan button tersebut sistem akan mengeluarkan modal *edit* seperti gambar 4.29, untuk meminta pengguna mengubah data sesuai dengan yang diinginkan.




Gambar 4. 33 Implementasi Modal *Edit* Data Server

### 4.4.3 Pengujian Sistem

Pengujian sistem dilakukan bertujuan untuk memastikan bahwa seluruh komponen sistem yang telah dibuat berfungsi sesuai dengan spesifikasi dan kebutuhan yang telah ditetapkan. Pengujian dilakukan berdasarkan rancangan yang telah dibuat sebelumnya, pengujian sistem dilakukan dengan menggunakan metode *black box testing*.

Tabel 4. 13. *Black Box Testing*

| No | Test Case | Scenario Test | Result |
|---|---|---|---|
| 1. | *Login* dengan *username* dan *password* yang benar | 1. Mengakses halaman aplikasi *monitoring*<br>2. Mengisi *username* dan *password* yang telah terdaftar dan menekan tombol submit | *Login* Sukses |
| 2. | Login dengan *username* dan *password* yang salah | 1. Mengakses halaman aplikasi *monitoring*<br>2. Mengisi *username* dan *password* yang tidak terdaftar dan menekan tombol submit | *Login* Gagal |
| 3. | Menampilkan data *monitoring* terkini | *1. Login*<br>2. Menekan *Button Realtime Monitoring* pada *sidebar* | Menampilkan halaman *realtime monitoring* dengan data *resource* server dan status aplikasi *microservice* terkini |
| 4. | Menampilkan data history berdasarkan IP | *1. Login*<br>2. Menekan *Button History Monitoring* pada sidebar<br>3. Memilih IP | Menampilkan Halaman *History Monitoring* dengan data *history resource* server dan Status aplikasi *microservice* berdasarkan IP yang dipilih |
| 6. | Menambahkan data server | 1. Aplikasi lapor diaktifkan di server yang ingin di lakukan *monitoring*<br>2. Aplikasi lapor mengirimkan data server ke aplikasi *monitoring* | Data server terkirim |

| | | | |
|---|---|---|---|
| 7. | Menampilkan Data server | *1. Login*<br>2. Menekan *Button* Manajemen Server pada *sidebar* | Menampilkan halaman Manajemen Server dengan list data server |
| 8. | Melakukan *edit* Data Server dan Kirim ke aplikasi lapor | *1. Login*<br>2. Menekan *Button* Manajemen Server pada *sidebar*<br>3. Menekan tombol edit pada data server yang diinginkan<br>4. Melakukan *edit* data<br>5. Menekan tombol submit<br>6. Menampilkan konfirmasi pengubahan data<br>7. Klik button konfirmasi | Data Server berhasil diubah dan terkirim |
| 9. | Menghapus Data Server dan Kirim ke aplikasi lapor | *1. Login*<br>2. Menekan Button Manajemen Server pada *sidebar*<br>3. Menekan tombol *delete* pada data server yang diinginkan<br>4. Menampilkan konfirmasi penghapusan data<br>5. Klik button konfirmasi | Data Server berhasil di hapus dan terkirim |
| 10. | *Logout* | *1. Login*<br>2. Menekan Button *Logout* | Keluar dari halaman *monitoring* dan menuju ke halaman *login* |