Laman web jurnal: https://ejournal.unama.ac.id/index.php/processor

Processor: Jurnal Ilmiah Sistem Informasi, Teknologi Informasi dan Sistem Komputer

P-ISSN: 1907-6738 | E-ISSN: 2528-0082

Implementasi Mikrotik-API pada Filter Rule Mikrotik OS Menggunakan PHP Native untuk UPT Lab Universitas Amikom Yogyakarta

Muhammad Firdaus Ilhamy[1], Andika Agus Slameto[2]

[1,2]Informatika Universitas Amikom Yogyakarta, Jl. Ring Road Utara, Condong Catur, Depok, Sleman, Yogyakarta 55281, Indonesia

Abstrak−*Mikrotik Routeros* merupakan sistem operasi jaringan yang populer digunakan oleh administrator jaringan. Dilengkapi dengan fitur *Firewall*, pengelolaan *Filter rule* dalam mengontrol akses jaringan menjadi penting. Namun, antarmuka *WEB* atau *CLI* dapat sulit dan memakan waktu. Salah satu alternatif pengelolaan adalah menggunakan *Mikrotik API* yang memungkinkan akses melalui program atau aplikasi terintegrasi.Winbox adalah aplikasi GUI umum untuk mengontrol akses jaringan, tetapi memiliki beberapa masalah. Pengguna dapat menyebabkan kesalahan tidak disengaja, dan aplikasi ini hanya tersedia untuk *Windows*, bukan untuk *Mac os* atau *Linux*. Alternatif lain seperti *CLI* dan *API* dapat diakses dari berbagai *platform*. Di Universitas Amikom Yogyakarta, masalah muncul di UPT (Unit Pelaksana Teknis) Lab karena masih menggunakan Winbox untuk mengelola izin jaringan, terutama pada *Filter rule*. Oleh karena itu, proyek penelitian ini bertujuan untuk mengurangi kesalahan pengguna dengan menggunakan *Mikrotik API*. Penggunaan *API* ini memungkinkan pengelolaan *Filter rule* yang lebih efisien dan efektif melalui program atau aplikasi yang sesuai kebutuhan, meningkatkan keamanan jaringan secara keseluruhan.

Kata Kunci: *Routeros*; *Mikrotik API*; *Filter rule*; Antarmuka *WEB*; Winbox; *PHP*; Keamanan jaringan.

Abstract−Mikrotik Routeros is a popular network operating System used by network administrators. Equipped with a Firewall feature, managing Filter rules in controlling network access is important. However, the WEB interface or CLI can be difficult and time-consuming. One management alternative is to use the Mikrotik API which allows access through integrated programs or applications. Winbox is a common GUI application for controlling network access, but it has some problems. Users may cause accidental errors, and this app is only available for Windows, not for Mac os or Linux. Other alternatives such as CLI and API can be accessed from various platforms. At Amikom University, Yogyakarta, problems arOSe at the UPT (Technical Implementation Unit) Lab because they were still using Winbox to manage network permissions, especially on Filter rules. Therefore, this research project aims to reduce user errors by using the Mikrotik API. The use of this API enables more efficient and effective management of Filter rules through programs or applications as needed, increasing overall network security.

Keywords: Routeros; Mikrotik API; Filter rule; WEB interface; Winbox; PHP; Network security.

1. PENDAHULUAN

Mikrotik Routeros merupakan salah satu sistem operasi jaringan yang sangat populer digunakan oleh para administrator jaringan. *Routeros* dilengkapi dengan berbagai fitur untuk mengelola dan memantau jaringan, termasuk fitur *Firewall* untuk mengontrol akses jaringan. Salah satu cara untuk mengelola *Firewall* pada *Mikrotik Routeros* adalah dengan menggunakan *Mikrotik API*. *API Mikrotik* memungkinkan pengguna untuk mengakses dan mengelola fitur pada *Routeros* melalui program atau aplikasi yang diintegrasikan. Dalam penggunaan *Firewall* pada *Mikrotik Routeros*, *Filter rule* memainkan peran penting dalam mengontrol akses jaringan. *Filter rule* digunakan untuk memfilter paket yang melewati *Router* dan menentukan apakah paket tersebut diizinkan atau tidak. Namun, pengelolaan *Filter rule* pada *Mikrotik Routeros* melalui antarmuka *WEB* atau *CLI* dapat menjadi sulit dan memakan waktu.

Terdapat aplikasi GUI yang biasa digunakan dalam mengontrol akses jaringan yaitu Winbox, akan tetapi meskipun winbox sangat berguna dan mudah digunakan, namun ada beberapa masalah yang dapat yang paling utama adalah setiap pengguna dapat menyebabkan kesalahan yang tidak disengaja ketika mengatur jaringan seperti tidak sengaja menghapus settingan *Router* yang sudah dibuat sebelumnya, lalu masalah lain adalah winbox hanya tersedia untuk sistem operasi *Windows*, dan tidak tersedia untuk sistem operasi lain seperti *Mac os* atau *Linux*. Hal ini membuat pengguna yang menggunakan sistem operasi lain harus mencari alternatif

https://doi.org/10.33998/processor.2024.19.1.1641

Submitted: 02 Maret 2024; Reviewed: 04 Maret 2024; Accepted; 02 April 2024; Published: 30 April 2024

lain untuk mengelola perangkat *Mikrotik*. Oleh karena itu, meskipun Winbox merupakan aplikasi manajemen jaringan yang berguna, pengguna harus mempertimbangkan masalah-masalah ini sebelum memutuskan untuk menggunakannya. Ada alternatif lain yang dapat digunakan untuk mengelola perangkat *Mikrotik*, seperti *CLI* (*Command Line Interface*) dan *API* (*Application Programming Interface*), yang dapat diakses dari berbagai 2 *platform*.

Permasalahan yang ada di Universitas Amikom Yogyakarta khususnya di UPT (Unit Pelaksana Teknis) Lab yaitu para pengelola dan staff saat mengelola izin jaringan yang terhubung dengan internet, pihak UPT (Unit Pelaksana Teknis) masih menggunakan *Platform* Winbox untuk mengontrol jaringan terutama pada *Filter rule*, oleh karena itu, proyek penelitian ini bertujuan untuk mengurangi kemungkinan kesalahan pengguna saat memproses *Filter rule*. Dengan menggunakan *Mikrotik* API (*Application Programming Interface*), pengelola jaringan dapat mengelola *Filter rule* dengan lebih efisien dan efektif melalui program atau aplikasi yang dirancang sesuai kebutuhan. Hal ini akan mempercepat proses pengelolaan *Filter rule* dan meningkatkan keamanan jaringan secara keseluruhan. Maka dari itu, penelitian tentang penggunaan *Mikrotik API* (*Application Programming Interface*) untuk mengelola *Filter rule* pada *Mikrotik Routeros* sangat penting untuk meningkatkan efisiensi dan efektivitas pengelolaan jaringan.

2. TINJAUAN PUSTAKA

Berbagai penelitian telah mencoba menjelaskan tentang *Mikrotik API* yang memungkinkan pengguna untuk mengakses dan mengelola fitur pada *Routeros* melalui program atau aplikasi yang diintegrasikan. Dengan adanya penelitian terdahulu bisa membuat suatu referensi untuk membuat penelitian ini serta berguna sebagai perbandingan dengan penelitian sebelumnya.

Menurut Penelitian Andriansyah Zakaria dkk, (2019) menemukan bahwa metode *API* bisa digunakan untuk membuat Sistem pendaftaran mandiri dan aktivasi akun pengguna *hotspot* dengan adanya teknologi *API* dan di sempurnakan dengan bahasa pemograman *PHP* dan Mysql sehingga layanan internet *hotspot* dapat berjalan lebih efektif, efisien dan aman.[1]

Selain itu, Skenario yang sama digunakan oleh Yusran Said dkk, (2019) dan Kadek Juni Arta dkk, (2020) Melihat dari hasil penelitian mereka, penulis sama-sama membuat sebuah aplikasi manajemen *hotspot* sebuah *API* yang berbasis *WEB* sehingga mempermudah untuk pengguna *hotspot* serta manajemen *hotspot* untuk menganggulangi permasalahan pada *hotspot*. [2], [3]

Menurut Heru Kurniawan dkk, (2020) menggunakan metode lain yaitu *Squid proxy* yang terintegrasi dengan *Mikrotik* pada sistem operasi ubuntu juga dapat membuat suatu sistem untuk me*monitoring traffic* jaringan yang berbasis *Website* dengan menggunakan bahasa pemograman *PHP* dan Mysql sebagai *database*. Penelitian tersebut bertujuan untuk menjaga kestabilan dan kecepatan dalam akses internet.[4]

Sedangkan penelitian Rinanza Zulmy Alhamri dkk, (2021) juga mengembangkan aplikasi *monitoring* jaringan berbasis *Android* untuk Administrator dapat mengaktifkan aplikasi, melakukan *login*, dan melihat *log* aktivasi *Router*, melihat *traffic* pada *interface Router* dan melihat kondisi internet.[5]

Belakangan ini, Ruri Ashari Dalimunthe dan Herman Saputra, (2022) memiliki hasil penelitian yang sama dengan penelitian sebelumnya yaitu membuat sistem *monitoring*. Tetapi penelitian kali ini menggunakan metode *Mikrotik API* untuk menampilkan data *monitoring* yang ada di *Mikrotik* dan disimpan kedalam *database* guna dijadikan sebagai bahan evaluasi infrastruktur jaringan. [6]

3. METODOLOGI PENELITIAN

Penelitian ini bertujuan untuk mengembangkan solusi yang dapat meningkatkan efisiensi, mengurangi kesalahan pengguna, dan meningkatkan keamanan sistem jaringan di UPT Lab Universitas Amikom Yogyakarta. Solusi ini mencakup pengimplementasian antarmuka pengguna yang lebih intuitif, validasi input, dan mekanisme otomatisasi menggunakan *API Mikrotik*.

Penelitian ini dimulai dengan menyusun tujuan dan kebutuhan yang ingin dicapai. Tujuan meliputi peningkatan efisiensi pengelolaan izin jaringan, pengurangan kesalahan pengguna, dan peningkatan keamanan sistem. Kebutuhan khusus yang perlu dipenuhi termasuk antarmuka pengguna yang intuitif, validasi input, tampilan visual yang jelas, dan petunjuk pengguna yang mudah diikuti. Selanjutnya, dilakukan identifikasi jenis *Filter rule* yang diperlukan berdasarkan kebutuhan yang telah diidentifikasi sebelumnya. Contoh *Filter rule* yang dipertimbangkan mencakup pemblokiran lalu lintas berdasarkan alamat IP, protokol, *port*, atau jenis layanan.

Selanjutnya, penelitian ini melibatkan koneksi ke *API Mikrotik*. Menggunakan *Mikrotik API*, aplikasi atau Software yang akan berinteraksi dengan *Filter rule* terhubung dengan perangkat *Mikrotik*. Penulis Pastikan terdapat akses ke perangkat *Mikrotik* melalui *API* dan autentikasi yang sesuai untuk melakukan operasi yang diperlukan, seperti mengubah nilai *Filter rule*.

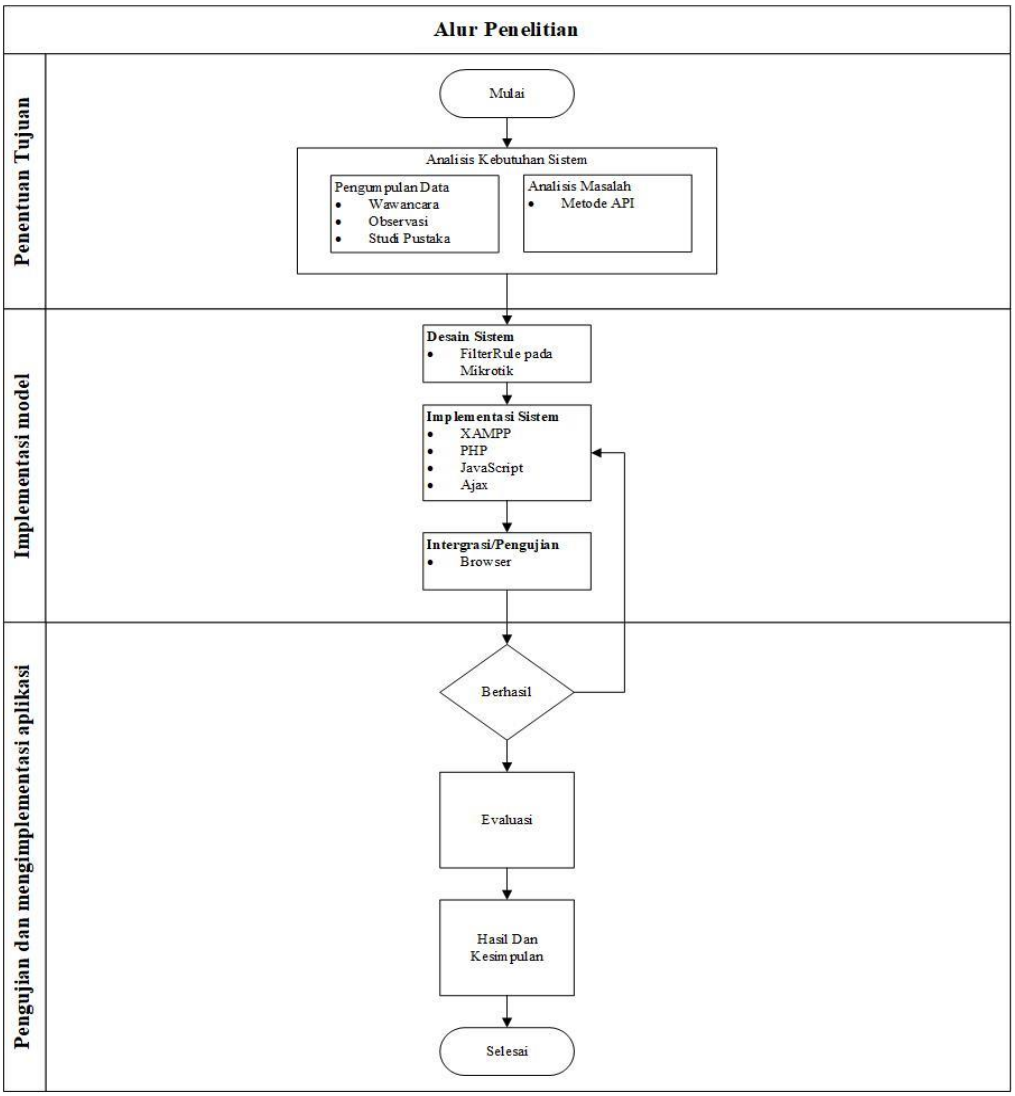
https://doi.org/10.33998/processor.2024.19.1.1641
Submitted: 02 Maret 2024; Reviewed: 04 Maret 2024; Accepted; 02 April 2024; Published: 30 April 2024

Setelah itu, dilakukan implementasi *Filter rule* dengan menggunakan *Mikrotik API*. Hal ini bertujuan untuk mencapai pengelolaan izin jaringan yang lebih efisien, mengurangi kesalahan pengguna, dan meningkatkan keamanan sistem.

Untuk menguji rancangan solusi ini, tujuan pengujian ditetapkan. Tujuan pengujian meliputi memastikan bahwa implementasi *Mikrotik API* pada *Filter rule Mikrotik* OS menggunakan *PHP* berfungsi dengan benar, serta memverifikasi kinerja dan stabilitas sistem setelah penerapan implementasi *Mikrotik API*. Pengujian dilakukan dalam lingkungan yang melibatkan sistem operasi *Mikrotik* OS, bahasa pemrograman *PHP*, *server WEB Apache*, dan komputer/*server* yang menjalankan *Mikrotik* OS dan *PHP* dengan *server WEB*.

Desain pengujian terdiri dari beberapa langkah. Pertama, dilakukan persiapan lingkungan pengujian, termasuk instalasi *Mikrotik* OS, *PHP*, dan *server WEB*, serta konfigurasi koneksi jaringan antara komputer/*server* dan perangkat *Mikrotik* OS. Selanjutnya, implementasi *script PHP* dengan menggunakan *Mikrotik API* dilakukan untuk mengakses dan mengelola *Filter rule* pada perangkat *Mikrotik* OS. Pengujian dilakukan dengan mengubah *Filter rule*, dan beberapa parameter relevan seperti Ip *address* dan *port* perangkat *Mikrotik* OS, kredensial (*username* dan *password*), informasi mengenai *Filter rule* yang akan diubah, dan pengukuran waktu yang diperlukan untuk operasi dan respons dari perangkat *Mikrotik* OS.

Langkah-langkah pengujian dicatat dan hasilnya dianalisis untuk mengidentifikasi keberhasilan implementasi *Mikrotik API*, kinerja sistem, dan stabilitas. Jumlah pengujian dapat ditentukan berdasarkan kompleksitas skenario dan ketersediaan sumber daya, dan melakukan beberapa pengujian dengan skenario penggunaan yang berbeda untuk memastikan keberhasilan dan kinerja implementasi. Setelah selesai melakukan pengujian, hasil pengujian divalidasi dengan membandingkan hasil yang diharapkan dengan hasil yang diperoleh. Jika hasil validasi sesuai dengan harapan, dapat disimpulkan bahwa implementasi *Mikrotik API* pada *Filter rule Mikrotik* OS menggunakan *PHP* untuk UPT Lab Universitas Amikom Yogyakarta berhasil.



Gambar 1. Alur Penelitian

https://doi.org/10.33998/processor.2024.19.1.1641
Submitted: 02 Maret 2024; Reviewed: 04 Maret 2024; Accepted; 02 April 2024; Published: 30 April 2024

Berdasarkan Gambar 1, terdapat tahap yang paling utama dengan melakukan analisis kebutuhan sistem atau penentuan tujuan terkait masalah metode *API*. Tahap selanjutnya melakukan persiapan perangkat keras yang digunakan penelitian sebagai berikut :

1. Laptop *Windows* 10 64bit
2. *Router Board* RB450Gx4 dan *hAP lite*
3. Kabel UTP CAT 5

Selanjutnya melakukan persiapan perancangan struktur dan fitur-fitur sistem, termasuk bagaimana *FilterRule* akan diimplementasikan pada perangkat *Mikrotik*. Lalu membangun aplikasi menggunakan teknologi XAMPP (*WEB server stack*), *PHP* (bahasa pemrograman *server-side*), Java*Script*, dan Ajax (teknik pengiriman data asinkron). Setelah implementasi, selanjutnya mengintegrasikan komponen-komponen yang telah dibangun dan melakukan pengujian fungsional melalui *browser WEB*.

Setelah pengujian, jika aplikasi berhasil memenuhi persyaratan dan kriteria, selanjutnya dapat melanjutkan ke tahap evaluasi. Namun, jika ada kegagalan, harus kembali ke tahap implementasi untuk memperbaiki masalah tersebut. Tahap selanjutnya mengevaluasi kinerja dan efektivitas sistem atau aplikasi berdasarkan tujuan dan persyaratan yang telah ditetapkan sebelumnya.

Terakhir menentukan hasil dan kesimpulan menganalisis hasil evaluasi dan merumuskan kesimpulan berdasarkan performa dan keberhasilan sistem atau aplikasi yang telah dikembangkan. Dalam alur penelitian ini, terlihat bahwa menggabungkan elemen-elemen pengembangan sistem, implementasi aplikasi, serta pengujian dan evaluasi untuk memastikan bahwa solusi yang dihasilkan efektif dan sesuai dengan tujuan yang telah ditetapkan.

4. HASIL DAN PEMBAHASAN

4.1. Perancangan

4.1.1.Rancangan Sistem

Pada tahap awal, dilakukan analisis kebutuhan untuk memahami secara mendalam persyaratan dari sistem yang akan dikembangkan. Komunikasi intensif dengan pihak UPT Lab Universitas AMIKOM Yogyakarta dilakukan untuk mengklarifikasi fitur-fitur yang diinginkan, khususnya terkait pengelolaan *Filter rule* pada perangkat *Mikrotik* melalui antarmuka *WEB*.

Setelah kebutuhan terdefinisi dengan jelas, dilakukan perancangan antarmuka *WEB* yang akan menjadi jembatan antara pengguna dan perangkat *Mikrotik*. *Interface* ini akan memberikan kemudahan dalam manajemen *Filter rule* serta pengaturan aturan-aturan yang diterapkan pada jaringan.

Implementasi *MIKROTIK-API* menjadi bagian krusial dalam sistem ini, karena akan menjadi penghubung antara aplikasi *WEB* dan perangkat *Mikrotik*. Melalui fitur ini, data dan konfigurasi *Filter rule* dapat diterapkan, serta informasi dari perangkat *Mikrotik* dapat diambil dan ditampilkan melalui aplikasi *WEB*.

Dengan menggunakan *Visual Studio Code* dan *platform server* XAMPP, pengembangan aplikasi *WEB* dapat berjalan dengan efisien dan dapat diuji secara lokal.

4.1.2.Rancangan Proses

Rancangan Proses dilakukan untuk menggambarkan langkah-langkah yang akan diambil untuk merancang, mengembangkan, menguji, dan mengimplementasikan solusi atau sistem yang diusulkan. Rancangan proses membantu mengatur dan mengarahkan jalannya proyek secara terstruktur sehingga tujuan dapat dicapai dengan lebih efisien dan efektif.

1. *HTML* dan *Javascript* Libraries:
   a. Pada bagian <head>, terdapat beberapa *Library* yang di-*load* menggunakan <*script*> dan <link> tags. *Library* ini adalah dependensi yang digunakan dalam halaman *WEB* ini.
   b. jQuery.min.js: *Library jQuery* digunakan untuk mengakses dan memanipulasi elemen *HTML* lebih mudah.
2. *Bootstrap*.min.CSS dan *Bootstrap*.min.js: *Library Bootstrap* adalah Framework CSS dan *Javascript* untuk tampilan dan interaksi elemen tampilan yang responsif.
3. *Navbar*:
   Di dalam <*body*>, terdapat sebuah navigasi (*navbar*) yang menggunakan *Bootstrap*. *Navbar* ini berisi judul, logo, dan tombol *Logout*.
4. Modal *Confirmation*:

https://doi.org/10.33998/processor.2024.19.1.1641
Submitted: 02 Maret 2024; Reviewed: 04 Maret 2024; Accepted; 02 April 2024; Published: 30 April 2024

   a. Bagian <div *class*="modal *fade*" ...> adalah modal yang akan ditampilkan saat tombol "*ON*" atau *"OFF"* di klik.
   b. Modal ini berfungsi untuk memberikan konfirmasi sebelum mengubah status pada perangkat *Mikrotik*. Jika pengguna mengklik "*Confirm*", maka fungsi *Javascript* akan melakukan pemrosesan melalui AJAX ke file proses.*php*.
5. Data *Table*:
   a. Di dalam sebuah <div *class*="*container* mt-4">, terdapat sebuah tabel dengan *class* "*table table-striped*". Tabel ini akan menampilkan data dari perangkat *Mikrotik*.
   b. *PHP* digunakan untuk mengambil data dri perangkat *Mikrotik* dan menampilkan data tersebut sebagai baris-baris pada tabel.
   c. Setiap baris tabel menampilkan informasi seperti nomor, akses internet, status, kecepatan, dan tombol *"ON"* atau *"OFF"* (sesuai dengan status).
6. *Javascript* (jQuery):
   a. *Script* ini akan dieksekusi ketika halaman sudah siap (dalam fungsi $(document).ready(function() { ... })). Ketika tombol *"ON"* atau *"OFF"* pada baris tabel diklik (*class* .button-toggle), maka fungsi *Javascript* akan menampilkan modal konfirmasi dan menyiapkan AJAX untuk mengirim data ke proses.*php*.
   b. Ketika tombol "*Confirm*" pada modal diklik, AJAX akan mengirim data (id dan status) ke proses.*php*, yang kemudian akan memproses permintaan tersebut.
   c. Setelah selesai, halaman akan dire*load* agar tampilan tabel terbaru ditampilkan.
7. Penutup *PHP*:
   Di akhir file, terdapat penutup untuk koneksi ke perangkat *Mikrotik* menggunakan *Library Routeros_API.class.php*.

Kemudian, penulis membuat *script* baru dan letakan didalam satu direktori, yang berfungsi untuk mengendalikan perangkat *Mikrotik* dengan menggunakan *Routeros API*.

1. Memasukan *Library API Mikrotik*:
   Pada baris pertama, kode require('*Routeros_API.class.php*'); digunakan untuk menyertakan file *Routeros_API.class.php*, yang merupakan *Library* untuk menghubungkan dan berinteraksi dengan perangkat *Mikrotik* melalui *API*.
2. Koneksi ke Perangkat *Mikrotik*:
   Baris selanjutnya mendefinisikan variabel-variabel yang digunakan untuk koneksi ke perangkat *Mikrotik*.
   $*Router_address*, $*Router_user*, dan $*Router_pass* adalah variabel yang berisi alamat IP perangkat *Mikrotik*, nama pengguna, dan kata sandi untuk masuk ke perangkat.
3. Inisialisasi Objek *Class* `*Routeros_API*`:
   Pada baris selanjutnya, `$*API* = new *Routeros_API*()`; digunakan untuk membuat objek dari kelas *Routeros_API*. Objek ini nantinya akan digunakan untuk berinteraksi dengan perangkat *Mikrotik*.
4. Koneksi ke *Mikrotik*:
   `$*API->connect*($*Router_address, $Router_user, $Router_pass)`; digunakan untuk melakukan koneksi ke perangkat *Mikrotik* dengan menggunakan informasi koneksi yang telah ditentukan sebelumnya. Jika koneksi berhasil, *script* akan dapat berkomunikasi dengan perangkat *Mikrotik*.
5. Menerima Data dari Halaman *WEB*:
   Selanjutnya, kode ini akan menerima data dari halaman *WEB* melalui permintaan AJAX dengan menggunakan metode *POST*. Data yang diterima adalah "ID" (ID perangkat) dan "status" (status yang akan diubah, '*true*' untuk hidup dan '*false*' untuk mati).
6. Mengubah Status Perangkat *Mikrotik*:
   Setelah menerima data dari halaman *WEB*, *script* ini akan menggunakan *Routeros API* untuk mengubah status perangkat *Mikrotik*.
   Baris berikutnya berisi perintah *API* untuk mengubah status perangkat menggunakan metode *write*(). Perangkat dengan ID yang sesuai akan ditemukan dan statusnya akan diubah sesuai dengan data yang diterima dari halaman *WEB*.
7. Redirect Halaman Setelah Perubahan:
   Setelah status perangkat *Mikrotik* diubah, *script* akan melakukan redirect ke halaman *WEB* bernama *dashboard.php*. Redirect dilakukan menggunakan *Javascript* dengan menggunakan perintah *window.location.href='dashboard.php';*.
   Ini akan membawa pengguna kembali ke halaman *dashboard.php*, yang kemungkinan berisi tabel atau tampilan terbaru dari status perangkat *Mikrotik*.

https://doi.org/10.33998/processor.2024.19.1.1641
Submitted: 02 Maret 2024; Reviewed: 04 Maret 2024; Accepted; 02 April 2024; Published: 30 April 2024

4.1.3.Rancangan Pengujian

Rancangan Pengujian dilakukan untuk mengevaluasi kinerja dan fungsionalitas dari implementasi *Mikrotik API* berbasis *WEB* menggunakan *PHP*. Pengujian ini mencakup serangkaian skenario dan kasus uji untuk memverifikasi apakah sistem dapat beroperasi sebagaimana yang diharapkan.

1. Pengujian Kebutuhan:
    a. Pengujian analisis kebutuhan untuk memastikan persyaratan sistem telah dipahami secara mendalam.
    b. Verifikasi bahwa fitur-fitur yang diinginkan, terutama terkait pengelolaan *Filter rule* pada perangkat *Mikrotik*, telah terklarifikasi dengan jelas melalui komunikasi intensif dengan pihak UPT Lab Universitas AMIKOM Yogyakarta.
2. Pengujian Perancangan Antarmuka *WEB*:
    a. Uji fungsionalitas antarmuka *WEB* untuk memastikan kemudahan dalam manajemen *Filter rule* dan pengaturan aturan-aturan pada jaringan.
    b. Pastikan antarmuka *WEB* telah dirancang dengan baik sesuai dengan kebutuhan yang telah terdefinisi.
3. Pengujian Implementasi *MIKROTIK-API*:
    a. Uji fungsionalitas *MIKROTIK-API* untuk memastikan penghubung antara aplikasi *WEB* dan perangkat *Mikrotik* berfungsi dengan baik.
    b. Verifikasi bahwa data dan konfigurasi *Filter rule* dapat diterapkan dengan benar pada perangkat *Mikrotik* melalui fitur ini.
    c. Pastikan informasi dari perangkat *Mikrotik* dapat diambil dan ditampilkan dengan akurat melalui aplikasi *WEB*.
4. Pengujian Integrasi Sistem:
    a. Uji integrasi antara antarmuka *WEB*, *MIKROTIK-API*, dan perangkat *Mikrotik* untuk memastikan seluruh sistem berinteraksi dengan benar.
    b. Pastikan data yang ditampilkan di antarmuka *WEB* sesuai dengan data yang ada di perangkat *Mikrotik* setelah diterapkan melalui *MIKROTIK-API*.
5. Pengujian Keamanan:
    a. Lakukan pengujian keamanan untuk mengidentifikasi potensi celah keamanan pada aplikasi *WEB* dan *API*.
    b. Pastikan hanya pengguna yang memiliki akses yang tepat yang dapat melakukan manajemen *Filter rule* dan konfigurasi perangkat *Mikrotik*.
6. Pengujian Performa:
    a. Uji performa aplikasi *WEB* dengan menggunakan *Visual Studio Code* dan *platform server* XAMPP untuk memastikan kinerjanya optimal.
    b. Pastikan aplikasi tetap responsif dan berkinerja baik saat diuji secara lokal.
7. Pengujian Pengguna (*User Acceptance Testing*):
    a. Melibatkan pengguna dari pihak UPT Lab Universitas AMIKOM Yogyakarta untuk menguji aplikasi dan memberikan umpan balik.
    b. Pastikan aplikasi memenuhi harapan dan kebutuhan pengguna serta mudah digunakan.
8. Pengujian Keseluruhan Sistem:
    a. Uji seluruh sistem di lingkungan produksi untuk memastikan seluruh komponen berfungsi dengan baik bersamaan.
    b. Pastikan sistem telah terintegrasi dengan perangkat *Mikrotik* dengan baik melalui *API* dan antarmuka *WEB*.

4.2. Implementasi

4.2.1.Implementasi Rancangan

Mengimplementasikan *Mikrotik Filter rule* menggunakan *Mikrotik API* adalah proses mengintegrasikan *API* (*Application Programming Interface*) yang disediakan oleh *Mikrotik* dengan aplikasi atau sistem yang ingin dibuat. Dalam hal mengatur *Rule Filter* di perangkat *Mikrotik* menggunakan kode program atau skrip *PHP*.
Untuk mengimplementasikan *Mikrotik Filter rule* menggunakan *Mikrotik API*, ada beberapa langkah sebagai berikut:

1. Pastikan perangkat *Mikrotik* sudah terhubung ke jaringan.
2. Unduh `*Routeros_API.class.php*` melalui *Website* resmi *Mikrotik*. Setelah di unduh, file tersebut diletakkan di direktori project.

https://doi.org/10.33998/processor.2024.19.1.1641
Submitted: 02 Maret 2024; Reviewed: 04 Maret 2024; Accepted; 02 April 2024; Published: 30 April 2024

3. Pastikan *Port API* yang ada pada *script* `*Routeros_API.class.php*` sudah sesuai dengan *port* yang ada pada *Service* di*Mikrotik*.
4. Buat tampilan frontend dari halaman *WEB*, termasuk pemuatan *jQuery* dan *Bootstrap*.

Dasboard.php:

```
<!DOCTYPE html>
<html>

<head>
<script
src="https://ajax.googleapis.com/ajax/libs/jquery/3.6.0/jquery.min.js"></scr
ipt>
<link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css"
>
<script
src="https://maxcdn.bootstrapcdn.com/bootstrap/4.5.2/js/bootstrap.min.js"></
script>
<link rel="stylesheet"href="https://cdnjs.cloudflare.com/ajax/libs/twitter-
bootstrap/4.5.2/css/bootstrap.css">
<link rel="stylesheet"
href="https://cdn.datatables.net/1.13.5/css/dataTables.bootstrap4.min.css">
<script
src="https://cdn.datatables.net/1.13.5/js/jquery.dataTables.min.js"></script
>
<script
src="https://cdn.datatables.net/1.13.5/js/dataTables.bootstrap4.min.js"></sc
ript>
</head>
```

Gambar 2. *Head Dashboard.php*

Berdasarkan gambar 2, Dengan *jQuery* dan *Bootstrap* ditambahkan dapat menggunakan fitur-fitur dan komponen yang disediakan oleh keduanya untuk mengembangkan tampilan *frontend* yang menarik dan responsif.

5. Membuat navigasi (*navbar*) di bagian atas halaman *WEB* dengan logo "UPT LAB AMIKOM YOGYAKARTA" dan tombol "*Logout*" di kanan atas. Dan Style untuk membuat navigasi tetap berada diatas ketika *scrolling* halaman.

Dasboard.php:

```
<body>
<nav class="navbar navbar-expand-lg navbar-dark bg-primary">
<a class="navbar-brand" href="#">
<img src="logo-amikom.png" alt="Amikom Yogyakarta" width="30"
height="30" class="d-inline-block align-top">
UPT LAB AMIKOM YOGYAKARTA
</a>
<button class="navbar-toggler" type="button" data-
toggle="collapse" data-target="#navbarNav" aria-
controls="navbarNav" aria-expanded="false" aria-label="Toggle
navigation">
<span class="navbar-toggler-icon"></span>
</button>
<div class="collapse navbar-collapse" id="navbarNav">
<ul class="navbar-nav ml-auto">
<li class="nav-item">
<a class="nav-link" href="logout.php">Logout</a>
</li>
</ul>
</div>
</nav>
<style>
fixed-top {
position: fixed;
top: 0;
width: 100%;
}
</style>
```

Gambar 3. *Body Dashboard.php*

Berdasarkan gambar 3, Mengatur posisi *navbar* sebagai "*fixed*" menggunakan CSS untuk menjadikannya tetap berada di bagian atas ketika halaman di-*scroll*. Dengan demikian, pengguna akan memiliki akses cepat ke informasi penting ketika menjelajahi halaman, dan gaya "*fixed*" akan memastikan navigasi tetap terlihat selama interaksi dengan konten halaman.

6. Membuat modal konfirmasi (*confirmation* Modal) yang akan muncul saat pengguna mengklik tombol aksi pada tabel. Modal ini berisi pesan konfirmasi *"Are you sure you want to perform this action?"* dan dua tombol yaitu "*Cancel*" dan "*Confirm*".

https://doi.org/10.33998/processor.2024.19.1.1641
Submitted: 02 Maret 2024; Reviewed: 04 Maret 2024; Accepted; 02 April 2024; Published: 30 April 2024

Dasboard.php:

```
<div class="modal fade" id="confirmationModal" tabindex="-1"
role="dialog" aria-labelledby="confirmationModalLabel"
aria-hidden="true">
<div class="modal-dialog" role="document">
<div class="modal-content">
<div class="modal-header">
<h5 class="modal-title"
id="confirmationModalLabel">Confirmation</h5>
<button type="button" class="close" data-dismiss="modal" aria-
label="Close">
<span aria-hidden="true">&times;</span>
</button>
</div>
<div class="modal-body">
Are you sure you want to perform this action?
</div>
<div class="modal-footer">
<button type="button" class="btn btn-secondary" data-
dismiss="modal">Cancel</button>
<button type="button" class="btn btn-primary"
id="confirmAction">Confirm</button>
</div>
```

Gambar 4. Modal Konfirmasi *Dashboard.php*

Berdasarkan gambar 4, Membuat modal konfirmasi (*confirmation* modal) melibatkan penggunaan elemen *HTML* seperti * * yang diatur dengan gaya CSS agar muncul sebagai jendela pop-up saat pengguna mengklik tombol aksi pada tabel. Modal ini mengandung pesan klarifikasi seperti "*Are you sure you want to perform this action*?" yang bertujuan untuk memastikan niat pengguna sebelum melanjutkan. Modal biasanya dilengkapi dengan dua tombol, "*Cancel*" untuk membatalkan tindakan dan "*Confirm*" untuk mengkonfirmasi tindakan tersebut. Fungsi *JavaScript* dapat ditambahkan untuk mengelola perilaku tombol-tombol ini, seperti menutup modal saat "*Cancel*" ditekan atau melanjutkan tindakan setelah "*Confirm*" diklik. Ini memberikan pengalaman interaktif yang lebih baik bagi pengguna sebelum mereka melakukan tindakan yang mungkin bersifat permanen atau penting.

7. Membuat tabel dengan menggunakan *Bootstrap* untuk menampilkan data perangkat yang terhubung ke *Mikrotik*. Tabel ini memiliki beberapa kolom yaitu "No", "Akses Internet", "Status", "Kecepatan", dan "*Action*".

Dasboard.php:

```
<div class="container mt-4">
<table id='dashboard' class="table table-striped">
<thead class="thead-dark">
<tr>
<th scope="col">No</th>
<th scope="col">Akses Internet</th>
<th scope="col">Status</th>
</tr>
</thead>
```

Gambar 5. Tabel *Dashboard.php*

Berdasarkan gambar 5, Membuat tabel dengan menggunakan *Bootstrap* untuk menampilkan data perangkat yang terhubung ke *Mikrotik* melibatkan penggunaan kelas-kelas *Bootstrap* yang sesuai untuk mengatur tampilan responsif dan menarik. Tabel ini memiliki lima kolom: "*No*" untuk nomor urut, "Akses Internet" untuk menampilkan jenis akses yang diberikan, "Status" untuk menampilkan status terkini dari perangkat, "Kecepatan" untuk menunjukkan kecepatan koneksi, dan "*Action*" untuk menyediakan tombol aksi seperti "*Edit*" atau "*Delete*". Dengan menggabungkan elemen *HTML* tabel dengan kelas-kelas *Bootstrap*, Penguna dapat menciptakan tampilan tabel yang rapi dan berespons dengan baik terhadap perangkat yang berbeda.

8. Memuat *Library Routeros_API.class.php* yang dibutuhkan untuk berinteraksi dengan perangkat *Mikrotik*.

Dasboard.php:

```
<?php
// Library API Mikrotik
require('routeros_api.class.php');
```

Gambar 6. *Library API Dashboard.php*

Berdasarkan gambar 6, Dengan memuat Routeros_API.class.php ke dalam proyek, *developer* dapat mengakses semua fungsi yang diperlukan untuk berinteraksi dengan perangkat Mikrotik secara efisien dan terstruktur, membantu dalam otomatisasi tugas-tugas administratif dan pengelolaan perangkat jaringan.

https://doi.org/10.33998/processor.2024.19.1.1641
Submitted: 02 Maret 2024; Reviewed: 04 Maret 2024; Accepted; 02 April 2024; Published: 30 April 2024

9. Mendefinisikan variabel $*Router_address*, $*Router_user*, dan $*Router_pass* yang digunakan untuk koneksi ke perangkat *Mikrotik*.

Dasboard.php:

```
// Koneksi User Mikrotik
$router_address = $ip;
$router_user    = $user;
$router_pass    = $password;
// Object dari class routeros_api
$API            = new routeros_api();
// Debug status/Kesalahan koneksi
$API->debug = false;
// Koneksi sementara ke Mikrotik
$API->connect($router_address, $router_user, $router_pass);
```

Gambar 7. Koneksi *Router Dashboard.php*

Berdasarkan gambar 7, Mendefinisikan variabel $*Router_address*, $*Router_user*, dan $*Router_pass* merupakan langkah awal dalam mengatur parameter koneksi ke perangkat Mikrotik melalui bahasa pemrograman seperti PHP. Variabel $*Router_address* berisi alamat IP atau *hostname* perangkat Mikrotik yang dituju, $*Router_user* memuat nama pengguna yang valid untuk otentikasi, dan $*Router_pass* berisi kata sandi yang sesuai. Dengan nilai-nilai yang tepat untuk ketiga variabel ini, Penguna mempersiapkan informasi kredensial yang diperlukan untuk membuka koneksi ke perangkat Mikrotik dari dalam kode program, memungkinkan akses dan interaksi dengan perangkat secara aman dan sesuai dengan otorisasi yang diberikan.

10. Mengirim perintah ke perangkat *Mikrotik* untuk mendapatkan data *Filter rule* menggunakan perintah */ip/Firewall/Filter/print*. Hasilnya akan disimpan dalam variabel $*Rules*.

Dasboard.php:

```
// Query melihat filter rule mikrotik
$API->write('/ip/firewall/filter/print');
// Menjalankan Query Write
$rules = $API->read();
```

Gambar 8. *Write Query Filter rule Dashboard.php*

Berdsarkan gambar 8, Dengan cara ini, Pengguna mendapatkan akses ke data konfigurasi firewall yang relevan dan dapat memprosesnya dalam kode program untuk keperluan selanjutnya, seperti tampilan informasi.

11. Melakukan *Loop foreach* untuk mengolah dan menampilkan data *Filter rule* pada tabel. Setiap *Rule* akan ditampilkan dalam baris tabel yang berisi nomor, nama perangkat, status (Hidup atau Mati), kecepatan akses internet (*Down* dan *Up*), serta tombol aksi (*ON* atau *OFF*).

Dasboard.php:

```
foreach ($rules as $index => $res) {
$status = $res['disabled'] == 'true' ? 'Hidup' : 'Mati';?
<tr>
<th scope="row"><?php echo $index + 1; ?></th>
<td><?php echo $res['comment']; ?></td>
<td>
<form>
<input type="hidden" name="status" value="<?php echo
$res['disabled'] == 'false' ? 'true' : 'false'; ?>">
<input type="hidden" name="id" value="<?php echo $res['.id']; ?>">
<button type="button" class="btn button-toggle <?php echo
$res['disabled'] == 'true' ? 'btn-success' : 'btn-danger'; ?>"
data-status="<?php echo $res['disabled'] == 'false' ? 'true' :
'false'; ?>">
<?php echo $res['disabled'] == 'false' ? 'OFF' : 'ON'; ?>
</button>
</form>
</td>
</tr>
<?php } ?>
</tbody>
</table>
</div>
```

Gambar 9. *Loop foreach Dashboard.php*

Berdasarkan gambar 9, Melakukan *loop foreach* untuk mengolah dan menampilkan data *Filter rule* pada tabel melibatkan iterasi melalui setiap aturan filter yang diperoleh sebelumnya. Pada setiap iterasi, informasi seperti nomor, nama perangkat, status (hidup atau mati), kecepatan akses internet (*down* dan *up*),

https://doi.org/10.33998/processor.2024.19.1.1641
Submitted: 02 Maret 2024; Reviewed: 04 Maret 2024; Accepted; 02 April 2024; Published: 30 April 2024

dan tombol aksi (*on* atau *off*) akan diambil dari data tersebut. Selanjutnya, informasi ini akan dimasukkan ke dalam baris tabel yang dibangun dalam kode *HTML* atau framework seperti *Bootstrap*.

12. *Script Javascript* menggunakan *JQuery* untuk mengatur perilaku tombol aksi pada tabel. Ketika pengguna mengklik tombol aksi (*ON* atau *OFF*), akan muncul modal konfirmasi untuk memastikan pengguna ingin melakukan perubahan. Jika pengguna mengklik tombol "*Confirm*" pada modal, akan dilakukan permintaan AJAX ke proses.*php* untuk memproses perubahan status akses internet pada perangkat,

Dasboard.php:

```
<script>
$(document).ready(function() {
$('#dashboard').DataTable( {"lengthMenu": [[100, "All", 50, 25,
10], [100, "All", 50, 25, 10]]} );} );
// DataTable('#dashboard');
$(document).ready(function() {
$(".button-toggle").click(function() {
var button = $(this);
var status = button.data('status');
var id = button.siblings('input[name="id"]').val();
$("#confirmationModal").modal('show');
$("#confirmAction").off('click').on('click', function() {
$.ajax({
url: "proses.php",
type: "POST",
data: {
id: id,
status: status
},
success: function(response) {
location.reload();
},
error: function(xhr, status, error) {
console.log(xhr.responseText);
}
});
$("#confirmationModal").modal('hide');
});
});
});
window.addEventListener('scroll', function () {
const navbar = document.querySelector('.navbar');
const offset = 100;
if (window.pageYOffset > offset) {
navbar.classList.add('fixed-top');
} else {
navbar.classList.remove('fixed-top');
</script>
```

Gambar 10. Java *Script Dashboard.php*

Berdasarkan gambar 10, Jika pengguna memilih tombol "*Confirm*" pada modal, *script* akan menginisiasi permintaan AJAX ke file proses.*php* dengan tujuan memproses perubahan status akses internet pada perangkat. Proses AJAX ini memungkinkan pengiriman data dan permintaan ke *server* secara asinkron tanpa perlu memuat ulang halaman, sehingga perubahan dapat diterapkan secara responsif. Dengan mengintegrasikan *JavaScript*, *jQuery*, modal konfirmasi, dan permintaan AJAX, pengguna dapat dengan aman dan nyaman mengubah status akses internet pada perangkat dengan dukungan visual yang kuat.

13. *Script PHP* untuk memutuskan koneksi ke perangkat *Mikrotik* dengan menggunakan $*API*->dis*connect*();. Perintah ini akan dieksekusi setelah data selesai ditampilkan pada tabel.

Dasboard.php:

```
<?php
// Memutus koneksi ke mikrotik
$API->disconnect();
?>
```

Gambar 11. *API* Dis*connect Dashboard.php*

Berdasarkan gambar 11, *Script PHP* untuk memutuskan koneksi ke perangkat Mikrotik dengan menggunakan *$API->disconnect();* adalah langkah penting dalam pengelolaan koneksi yang baik. Setelah data selesai ditampilkan pada tabel dan interaksi antara *PHP* dan *Mikrotik* melalui *API* telah selesai dilakukan, perintah *$API->disconnect();* akan dieksekusi untuk mengakhiri koneksi dengan perangkat *Mikrotik*. Tindakan ini bertujuan untuk memastikan bahwa sumber daya jaringan yang digunakan oleh koneksi tidak terus berlanjut secara tidak perlu, sehingga membebaskan sumber daya dan memungkinkan koneksi lain ke perangkat *Mikrotik*. Dengan mengakhiri koneksi dengan benar setelah penggunaan selesai,

https://doi.org/10.33998/processor.2024.19.1.1641
Submitted: 02 Maret 2024; Reviewed: 04 Maret 2024; Accepted; 02 April 2024; Published: 30 April 2024

skrip *PHP* menjaga efisiensi dan ketersediaan sumber daya jaringan yang lebih baik untuk aplikasi dan interaksi selanjutnya.

14. Buat satu *script* baru yang berfungsi untuk mengendalikan perangkat *Mikrotik* dengan menggunakan *Routeros API*. Dan disebut oleh penulis yaitu `Proses.*php*`
15. Memuat `*Library Routeros_API.class.php*` yang dibutuhkan untuk berinteraksi dengan perangkat *Mikrotik*.
16. Mendefinisikan variabel $*Router_address*, $*Router_user*, dan $*Router_pass* yang digunakan untuk koneksi ke perangkat *Mikrotik*.
17. Membuat objek $*API* dari *class Routeros_API* yang akan digunakan untuk berinteraksi dengan perangkat *Mikrotik*.
18. Membuat koneksi sementara ke perangkat *Mikrotik* dengan menggunakan $*API*->*connect*($*Router_address*, $*Router_user*, $*Router_pass*).
19. Mengambil nilai ID dan status dari permintaan AJAX yang dikirimkan dari halaman sebelumnya (frontend).

Proses.php
```
$id = $_POST['id'];
$status = $_POST['status'];
```

Gambar 12. *POST* Proses.*php*

Berdasarkan gambar 12, Mengambil nilai ID dan status dari permintaan AJAX yang dikirimkan dari halaman sebelumnya (*frontend*) melibatkan proses ekstraksi informasi yang dikirimkan melalui permintaan AJAX. Ketika pengguna berinteraksi dengan tombol aksi pada *frontend*, data seperti ID dan status diambil dari permintaan tersebut.

20. Menuliskan perintah untuk mengubah status *Filter rule* pada perangkat *Mikrotik*. Perintah ini menggunakan /ip/*Firewall*/*Filter*/set untuk mengatur properti *Filter rule*, kemudian diikuti dengan data id dan disabled (status baru) yang diperoleh dari permintaan AJAX.

Proses.php
```
$API->write("/ip/firewall/filter/set", false);
$API->Write("=.id=$id", false);
$API->write("=disabled=$status");
```

Gambar 13. *Write Filter rule* Proses.*php*

Berdasarkan gambar 13, Perintah ini akan diikuti dengan data seperti ID aturan dan nilai baru untuk properti "*disabled*" (status baru) yang diperoleh dari permintaan AJAX. Dengan menggabungkan elemen-elemen ini, *developer* menyusun perintah yang menginstruksikan perangkat *Mikrotik* untuk memodifikasi status aturan *filter* sesuai permintaan pengguna. Proses ini memungkinkan pengguna mengontrol dan mengelola perangkat *Mikrotik* secara dinamis melalui kode *PHP* yang terhubung dengan *API*-nya.

21. Menjalankan perintah dengan menggunakan $*API*->read(), sehingga perubahan status *Filter rule* pada perangkat *Mikrotik* akan diimplementasikan.

Proses.php
```
$result = $API->read();
if($status == 'true'){
    echo "
    <script>
    window.location.href='dashboard.php';
    </script>
    ";
} else {
    echo "
    <script>
    window.location.href='dashboard.php';
    </script>
    ";
}
?>
```

Gambar 14. Read Result *Filter rule* Proses.*php*

Berdasarkan gambar 14, Dengan menggunakan $*API->read()*, perintah dieksekusi untuk membaca atau mengekstrak informasi yang mengandung perubahan status *Filter rule* dari perangkat *Mikrotik*. Setelah pengguna telah menyusun dan mengirimkan perintah untuk mengubah status *filte rule*, perangkat *Mikrotik* akan merespons dengan informasi yang mencerminkan perubahan yang telah diterapkan.

22. Setelah status berhasil diubah, *script* ini akan memberikan respon berupa *script Javascript* untuk melakukan redirect ke halaman *dashboard.php* pada frontend. Jika status adalah '*true*' (Hidup), maka pengguna akan diarahkan kembali ke *dashboard.php* untuk menampilkan tabel dengan status terbaru. Jika

https://doi.org/10.33998/processor.2024.19.1.1641
Submitted: 02 Maret 2024; Reviewed: 04 Maret 2024; Accepted; 02 April 2024; Published: 30 April 2024

status adalah '*false*' (Mati), maka pengguna akan diarahkan kembali ke *dashboard.php* untuk menampilkan tabel dengan status terbaru.

23. Untuk meningkatkan keamanan dalam pengimplementasian penelitian ini, diperlukan sistem *login* untuk mengontrol akses pengguna ke *Mikrotik API*. Dengan cara ini, hanya pengguna yang memiliki otorisasi yang dapat mengakses dan mengelola perangkat *Mikrotik*. Buat *Script* yang berisi halaman *login* yang tersambung ke *Mikrotik* untuk menjadi keamanan sebelum masuk kemenu *dashboard*.
24. Buat Bagian *Head* dari halaman *Login*, yang memuat beberapa *script* dan stylesheet untuk jQuery, *Bootstrap*, dan CSS.
25. Buat Bagian *Body* dari halaman *Login*, yang berisi elemen-elemen untuk menampilkan form *login*. Terdapat sebuah form dengan tiga input fields yaitu "IP *Router*", "*Username*", dan "*Password*". Pengguna diminta untuk mengisi informasi ini sebelum melakukan *login*.
26. Membuat Kode *PHP* yang dipanggil saat form *login* dikirimkan (saat tombol "*Login*" ditekan). Implementasi ini menggunakan $_*POST* untuk mengakses data yang dikirimkan melalui form. Setelah mendapatkan nilai IP *Router*, *Username*, dan *Password* yang diisi pengguna

Login.php

```
<div class="card-body">
<?php
if (isset($_POST['login'])){
require('routeros_api.class.php');
$router_address = $_POST['ip'];
$router_user = $_POST['username'];
$router_pass = $_POST['password'];
// Object dari class routeros_api
$API            = new routeros_api();
// Debug status/Kesalahan koneksi
$API->debug = false;
// Koneksi sementara ke Mikrotik
if ($API->connect($router_address, $router_user, $router_pass)){
session_start();
$_SESSION['ip'] = $_POST['ip'];
$_SESSION['username'] = $_POST['username'];
$_SESSION['password'] = $_POST['password'];
header("Location: dashboard.php");
} else {
echo "<script type='text/javascript'>alert('Login Gagal. Usernasme
/ Password Salah!');</script>";
// echo 'Login Gagal, Username / Password Salah';
}
}
?>
<form method="post" action="">
<div class="form-group">
<label for="username">IP Router:</label>
<input type="text" class="form-control" id="ip" name="ip"
required>
</div>
<div class="form-group">
<label for="username">Username:</label>
<input type="text" class="form-control" id="username"
name="username" required>
</div>
<div class="form-group">
<label for="password">Password:</label>
<input type="password" class="form-control" id="password"
name="password" required>
</div>
<div class="text-center">
<button type="submit" name="login" class="btn btn-
primary">Login</button>
</div>
```

Gambar 15. *Script Login.php*

Berdasarkan gambar 15, Membuat kode *PHP* yang dipanggil saat *form login* dikirimkan melibatkan penanganan data yang diterima dari *form*. Saat pengguna menekan tombol "*Login*", data yang diisikan dalam *form*, seperti alamat IP *Router*, *username*, dan *password*, dikirimkan melalui metode *POST*. Dalam kode *PHP* ini, *developer* akan menggunakan variabel *$_POST* untuk mengakses dan mengolah data yang diterima. Dengan mengambil nilai dari *$_POST*, sistem bisa mendapatkan informasi yang diisi oleh pengguna dalam *form login*.

27. Setelah Membuat Sistem *login* yang akan membantu mengontrol akses pengguna ke *Mikrotik API*, Sistem *login* akan memastikan bahwa hanya pengguna yang memiliki otoritas yang sesuai yang dapat mengakses *Mikrotik API*,
28. Buat satu *script* baru yang berfungsi untuk membuat sistem *logout* yang akan membantu melindungi akun dan informasi pengguna dengan memastikan bahwa sesi pengguna dihapus dengan aman setelah mereka selesai menggunakan layanan.

https://doi.org/10.33998/processor.2024.19.1.1641
Submitted: 02 Maret 2024; Reviewed: 04 Maret 2024; Accepted; 02 April 2024; Published: 30 April 2024

Logout.php

```php
<?php
session_start();
session_destroy();
header("Location: login.php");
?>
```

Gambar 16. *Logout.php*

Berdasarkan gambar 16, Membuat satu *script* baru yang berfungsi untuk membuat sistem *logout* adalah langkah penting dalam menjaga keamanan dan privasi pengguna. *Script* ini bertujuan untuk melindungi akun dan informasi pengguna dengan memastikan bahwa sesi pengguna diakhiri dengan aman setelah mereka selesai menggunakan layanan. Ketika pengguna memilih tombol "*Logout*" atau tindakan serupa, *script* akan dieksekusi untuk menghapus data sesi yang terkait dengan pengguna tersebut. Ini melibatkan menghancurkan data penyimpanan sementara yang terkait dengan sesi, seperti token atau informasi lain yang mengidentifikasi sesi pengguna. Dengan melakukan ini, *script logout* memastikan bahwa akses tanpa izin atau potensi risiko keamanan pada akun pengguna dapat dihindari setelah pengguna meninggalkan layanan, memberikan lapisan perlindungan tambahan bagi pengguna dan data mereka.

29. Terakhir buat *script* untuk membuat Tabel selalu *realtime* saat ada perubahan konfigurasi / *scheduler* yang terpasang pada *Mikrotik* tabel *Filter rule* pada *WEB* akan mengikuti.
30. Buat session_start();: Memulai atau melanjutkan sesi yang telah dimulai sebelumnya. Sesi digunakan untuk menyimpan data pengguna yang telah *login*, yaitu *username* dan *password* yang disimpan di sesi saat *login* berhasil. $*user* = $_SESSION['*username*']; dan $*password* = $_SESSION['*password*'];: Mengambil data *username* dan *password* dari sesi yang telah disimpan sebelumnya saat *login* berhasil.
31. Mendefinisikan variabel $*Router_address*, $*Router_user*, dan $*Router_pass* yang digunakan untuk koneksi ke perangkat *Mikrotik*. Membuat objek $*API* dari *class Routeros_API* yang akan digunakan untuk berinteraksi dengan perangkat *Mikrotik*. Mengatur variabel $*API->debug* menjadi *false*, yang berarti tidak akan menampilkan pesan *debug*. Membuat koneksi sementara ke perangkat *Mikrotik* dengan menggunakan $*API->connect* ($*Router_address*, $*Router_user*, $*Router_pass*).
32. Selanjutnya, dilakukan perulangan *foreach* untuk setiap aturan *Filter* yang diperoleh dari perangkat *Mikrotik*

realtime.php

```php
$output = '';
foreach ($rules as $index => $res) {
if (isset($res['comment']) && strpos($res['comment'], '(API)') !==
false) {
$status = $res['disabled'] == 'true' ? 'Hidup' : 'Mati';
$output .= '<tr>';
$output .= '<td>' . (isset($res['comment']) ? $res['comment'] :
'Comment Filter Rule Tidak ada') . '</td>';
$output .= '<td>';
$output .= '<form>';
$output .= '<input type="hidden" name="status" value="' .
($res['disabled'] == 'false' ? 'true' : 'false') . '">';
$output .= '<input type="hidden" name="id" value="' . $res['.id'] .
'">';
$output .= '<button type="button" class="btn button-toggle ' .
($res['disabled'] == 'true' ? 'btn-success' : 'btn-danger') . '"
data-status="' . ($res['disabled'] == 'false' ? 'true' : 'false')
. '">';
$output .= $res['disabled'] == 'false' ? 'OFF' : 'ON';
$output .= '</button>';
$output .= '</form>';
$output .= '</td>';
$output .= '</tr>';
}
}
echo $output;
$API->disconnect();
?>
```

Gambar 17. *foreach realtime.php*

Berdasarkan gambar 17, Dilakukan perulangan *foreach* untuk setiap aturan *Filter* yang diperoleh dari perangkat *Mikrotik*. Dengan menggunakan struktur perulangan *foreach*, setiap aturan *Filter* akan diambil satu per satu dari kumpulan data yang diterima sebelumnya dari perangkat *Mikrotik*.

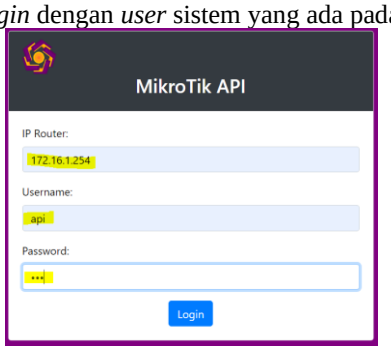
33. Dengan adanya *realtime.php* maka tabel *dashboard.php* akan menjadi *realtime* sesuai konfigurasi yang ada di *Mikrotik*. Dan *realtime.php* akan dipanggil melalui sebuat *Javascript* Setinterval pada *dashboard php* untuk dapat merefresh tabel dengan otomatis.

https://doi.org/10.33998/processor.2024.19.1.1641
Submitted: 02 Maret 2024; Reviewed: 04 Maret 2024; Accepted; 02 April 2024; Published: 30 April 2024

4.3. Pengujian

4.3.1.Pengujian Hasil Implementasi

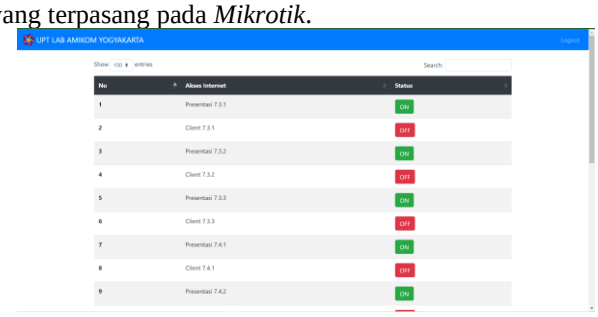
1. Koneksi ke *Router Mikrotik*
    a. Tujuan: Memastikan bahwa aplikasi *WEB* dapat terhubung dan berkomunikasi dengan *Router Mikrotik* melalui *API*.
    b. Langkah Pengujian :
        i. Buka Aplikasi *WEB*.
        ii. Lakukan *login* dengan *user* sistem yang ada pada *Mikrotik*.



Gambar 18. Pengujian *Login* Admin

Berdasarkan gambar 18, Melakukan *login* dengan *user* sistem yang ada pada *Mikrotik* adalah tahap di mana pengguna menggunakan kredensial yang valid, seperti *username* dan *password*, untuk mengakses sistem *Mikrotik*. Setelah informasi login diberikan, permintaan otentikasi dikirimkan ke perangkat Mikrotik untuk memeriksa kecocokan kredensial dengan data yang ada dalam sistem. Jika informasi yang diberikan sesuai, pengguna akan diberikan akses terbatas sesuai dengan hak akses yang terkait dengan akun tersebut. Dengan melewati tahap ini, pengguna dapat menjalankan operasi tertentu sesuai dengan izin yang mereka miliki dalam sistem *Mikrotik*, dan memungkinkan interaksi yang sah dan aman dengan perangkat tersebut.

        iii. Verifikasi koneksi ke *Router Mikrotik* dengan mengakses informasi berupa *Filter rule* yang terpasang pada *Mikrotik*.

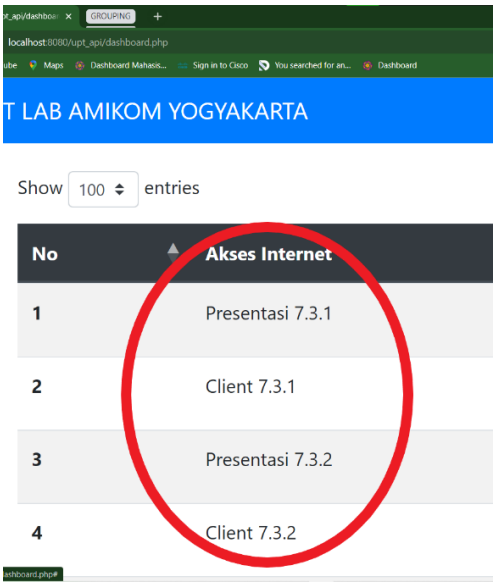


Gambar 19. Akses Informasi *Filter rule*

Berdasarkan gambar 19, Verifikasi koneksi ke *Router Mikrotik* dengan mengakses informasi berupa *Filter rule* yang terpasang pada *Mikrotik* melibatkan pengambilan data yang ada pada perangkat tersebut untuk memastikan bahwa koneksi telah berhasil diatur dan informasi dapat diakses. Dalam konteks ini, pengambilan data *Filter rule* yang terpasang menjadi indikator bahwa komunikasi antara *API* dan perangkat *Mikrotik* telah berfungsi dengan baik. Dengan mengakses dan mengambil informasi ini, Anda memvalidasi koneksi dan memverifikasi bahwa sistem dapat berinteraksi dengan perangkat *Mikrotik* sesuai yang diharapkan. Hal ini menjadi langkah penting dalam memastikan bahwa aplikasi atau layanan yang menggunakan data dari perangkat Mikrotik akan berjalan dengan lancar dan akurat.

2. Konfigurasi *Router Mikrotik*
    a. Tujuan: Menguji kemampuan aplikasi *WEB* untuk mengkonfigurasi *Router Mikrotik* melalui *API*.
    b. Langkah Pengujian :
        i. Pilih Salah satu Akses Internet yang ingin di ubah.

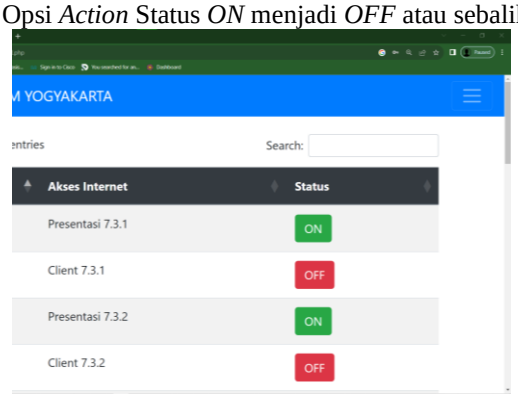
https://doi.org/10.33998/processor.2024.19.1.1641
Submitted: 02 Maret 2024; Reviewed: 04 Maret 2024; Accepted; 02 April 2024; Published: 30 April 2024



Gambar 20. Pengujian Konfigurasi

Berdasarkan gambar 20, pengguna akan melihat daftar akses internet yang tersedia dan memilih salah satu dari opsi tersebut sesuai kebutuhan.

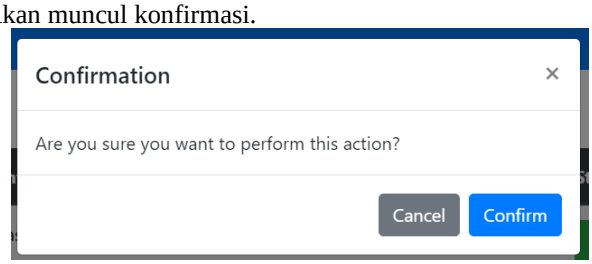
ii. Pilih Opsi *Action* Status *ON* menjadi *OFF* atau sebaliknya.



Gambar 21. Pengujian *Action* Status

Berdasarkan gambar 21, Pengguna dapat memilih tindakan yang ingin diambil terhadap suatu status yang sedang aktif (ON), dengan opsi untuk mengubahnya menjadi tidak aktif (OFF), atau sebaliknya.

iii. Akan muncul konfirmasi.



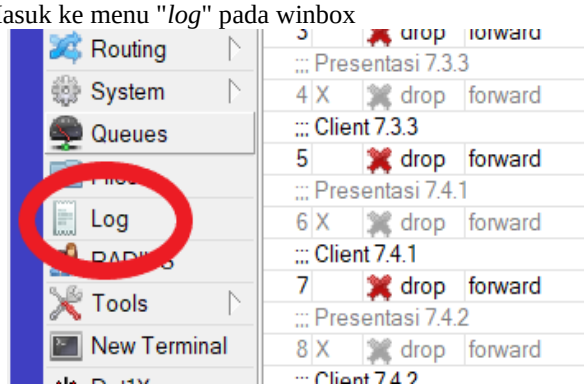
Gambar 22. Pengujian Konfirmasi

Berdasarkan gambar 22, Setelah tindakan sebelumnya, pengguna akan dihadapkan pada pilihan konfirmasi yang meminta mereka untuk memutuskan apakah mereka ingin melanjutkan tindakan tersebut atau membatalkannya. Dalam hal ini, dua opsi akan ditampilkan: "*Confirm*" untuk melanjutkan tindakan yang diambil, atau "*Cancel*" untuk membatalkan tindakan tersebut.

https://doi.org/10.33998/processor.2024.19.1.1641
Submitted: 02 Maret 2024; Reviewed: 04 Maret 2024; Accepted; 02 April 2024; Published: 30 April 2024

iv. Tekan "*Confirm*" Untuk mengubah konfigurasi.

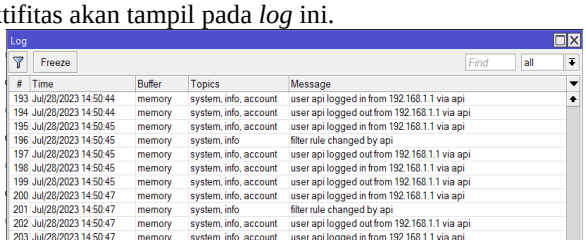
3. *Monitoring Mikrotik API*
    a. Tujuan: Memverifikasi kemampuan aplikasi *WEB* untuk jaringan menggunakan data yang diperoleh dari *Router Mikrotik*.
    b. Langkah Pengujian :
        i. *Login Mikrotik* melalui Winbox.
        ii. Masuk ke menu "*log*" pada winbox



Gambar 23. Pengujian *Monitoring*

Berdasarkan gambar 23, langkah di mana pengujian membuka bagian *log* pada antarmuka manajemen perangkat *MikroTik* menggunakan aplikasi Winbox.

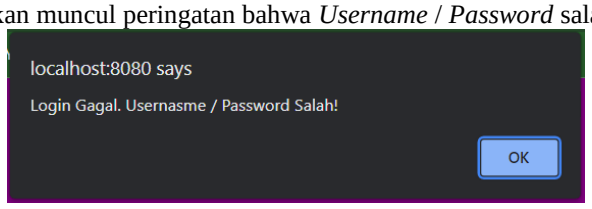
iii. Lihat *Log* yang menampilkan semua konfigurasi dan amati pada menu *Log*. Semua aktifitas akan tampil pada *log* ini.



Gambar 24. Pengujian *Log* Winbox

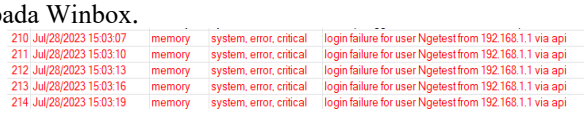
Berdasarkan gambar 22, Dalam konteks ini, menu "*log*" berisi catatan tentang peristiwa dan aktivitas yang terjadi pada perangkat *MikroTik*, termasuk informasi seperti status koneksi, perubahan konfigurasi, dan tindakan jaringan lainnya.

4. Keamanan
    a. Tujuan: Mengidentifikasi potensi masalah keamanan pada aplikasi *WEB*.
    b. Langkah Pengujian :
        i. Masuk ke aplikasi *WEB* dengan memasukan *Username* dan *password* salah. Maka akan muncul peringatan bahwa *Username* / *Password* salah.



Gambar 25. Pengujian *Password* Salah

Berdasarkan gambar 25, Dengan memasukkan *username* dan *password* yang salah menghasilkan peringatan yang memberitahu pengguna bahwa upaya masuk telah gagal karena kredensial yang diberikan tidak cocok. Dan dapat dilihat melalui "*Log*" pada Winbox.



Gambar 26. Pengujian *Login* Gagal Pada *Log* Winbox

https://doi.org/10.33998/processor.2024.19.1.1641
Submitted: 02 Maret 2024; Reviewed: 04 Maret 2024; Accepted; 02 April 2024; Published: 30 April 2024

Berdasarkan gambar 26, Setiap percobaan *login* ke perangkat *MikroTik* melalui *API* dengan menggunakan kredensial yang salah akan dicatat dan terekam dalam catatan *log*.

ii. *Logout* Dan Kembali ke menu *Dashboard* dengan mengunakan histori *browser*. Maka tampilan akan kembali ke menu *Login*, karena *User* Season telah Habis.

4.3.2.Hasil Pengujian

Hasil dari pengujian sistem menunjukkan bahwa implementasi *Mikrotik API* berbasis *WEB* menggunakan *PHP* untuk konfigurasi *Filter rule* berjalan sesuai dengan harapan. Semua skenario pengujian berhasil dilaksanakan dengan baik, dan sistem dapat beroperasi dengan stabil dan responsif. Tidak ditemukan masalah keamanan yang signifikan selama pengujian di UPT Universitas Amikom Yogyakarta.

5. KESIMPULAN DAN SARAN

5.1. Kesimpulan

Berdasarkan hasil dari pengujian yang telah dibahas, maka penulis mengambil kesimpulan menghubungkan *Mikrotik API* dan *Mikrotik Routeros* menggunakan *PHP*, yaitu dengan memberi akses dan pengelolaan fitur pada *Routeros* melalui *API Service* yang telah diaktifkan di perangkat *Mikrotik* dan akses untuk alamat IP yang diizinkan. Dengan menggunakan *Library* atau *class API* yang telah tersedia di *Website Mikrotik* lalu dihubungkan pada *script PHP*, maka *Filter rule* dapat diedit secara efisien, dan dapat membatasi akses pengguna melalui *WEB browser*. Penggunaan *Mikrotik API* dalam pengelolaan *Filter rule* memerlukan keamanan. Penerapan autentikasi *login* pada aplikasi *WEB* harus dilakukan untuk mencegah akses yang tidak sah dan melindungi jaringan dari potensi ancaman keamanan.

5.2. Saran

Dari penelitian yang telah dilakukan, penulis memberikan beberapa saran sebagai berikut.

1. Pastikan tampilan row-group tabel *Filter rule* lebih mudah dibaca dan dipahami oleh operator atau staff UPT.
2. Implementasikan *port knocking* untuk keamanan 2 l*api*s sebelum dapat akses untuk terhubung ke *Mikrotik*.
3. Berikan pilihan atau fitur *Filter* untuk memungkinkan operator atau staff UPT untuk menampilkan hanya *Filter rule* yang relevan dengan UPT Lab Gedung 2 atau Gedung 7.

DAFTAR PUSTAKA

[1] A. Zakaria, A. Prihantara, and A. A. Hartono, “Integrasi Application Programing Interface, PHP, dan MySQL untuk Otomatisasi Verifikasi dan Aktifasi Pengguna Layanan Hotspot Mikrotik,” JUITA J. Inform., vol. 7, no. 2, p. 63, 2019, doi: 10.30595/juita.v7i2.4361.

[2] I Kadek Juni Arta and Nyoman Bagus Suweta Nugraha, “Implementasi Aplikasi User Management Hotspot Mikrotik Berbasis Php Dengan Application Programing Interface (Api) Dan Framework Bootstrap,” J. Resist. (Rekayasa Sist. Komputer), vol. 3, no. 1, pp. 66–71, 2020, doi: 10.31598/jurnalresistor.v3i1.466.

[3] L. Y. Said, A. H. Jatmika, and I. W. A. Arimbawa, “Sistem Pendaftaran Hotspot Online Berbasis WEB Menggunakan Mikrotik API, PHP, MySql Pada SMK Plus Nurul Hakim Kediri,” J. Teknol. Informasi, Komputer, dan Apl. (JTIKA), vol. 1, no. 2, pp. 141–148, 2019, doi: 10.29303/jtika.v1i2.28.

[4] H. Kurniawan, J. Dedy Irawan, and F. . Ariwibisono, “Implementasi Squid proxy Pada Mikrotik Dan Monitoring Traffic Jaringan Berbasis Website,” JATI (Jurnal Mhs. Tek. Inform., vol. 4, no. 2, pp. 136–143, 2020, doi: 10.36040/jati.v4i2.2691.

[5] R. Z. Alhamri, T. A. Cinderatama, K. Eliyen, and A. Heriadi, “Pengembangan Aplikasi Monitoring Jaringan Berbasis Android Studi Kasus Puskom PSDKU Polinema di Kota Kediri,” INOVTEK Polbeng - Seri Inform., vol. 6, no. 2, p. 269, 2021, doi: 10.35314/isi.v6i2.2136.

[6] R. A. Dalimunthe and H. Saputra, “IMPLEMENTATION AND ANALYSIS OF MIKROTIK API MONITORING OF NETWORK USAGE,” JURTEKSI (Jurnal Teknol., 2022, [Online]. Available: https://jurnal.stmikroyal.ac.id/index.php/jurteksi/article/view/1920.

https://doi.org/10.33998/processor.2024.19.1.1641
Submitted: 02 Maret 2024; Reviewed: 04 Maret 2024; Accepted; 02 April 2024; Published: 30 April 2024