Journal of Applied Informatics and Computing (JAIC)
Vol.x, No.x, Juli 201x, pp. xx~xx
e-ISSN: 2548-6861

Pengembangan Aplikasi Otomatisasi Konfigurasi Perangkat Jaringan Berbasis *Website* Menggunakan Python

Willy Rivaldo Manurung[1]*, Andy Triwinarko[2]*
* Teknologi Rekayasa Multimedia, Politeknik Negeri Batam
4311911008.mj@students.polibatam.ac.id [1], andy@polibatam.ac.id [2]

Article Info

Article history:

Received …
Revised …
Accepted …

Keyword:

Configuration Management,
Django,
GNS3,
Network Automation,
Python

ABSTRACT

In today's digital era, the need for networks and the Internet is growing, resulting in computer networks that are increasingly dynamic and complex. Constantly changing network conditions and increasing connected devices require automation solutions to reduce manual configuration errors and time. This research aims to develop a web-based network device automation system that can eliminate repetitive work, consolidate device configuration and information into one platform, and reduce errors by using programming logic for efficient network service management. The Rapid Application Development (RAD) methodology was used during the development process to ensure speed and flexibility. Black-box testing demonstrated the functional success of the application. Based on the analysis, the application can save about 97% of the time compared to manual configuration. With an intuitive interface, the application makes it possible to manage devices and configurations more efficiently.

I. PENDAHULUAN

Di era digital saat ini, kebutuhan akan jaringan dan internet semakin meningkat pesat, sehingga menghasilkan jaringan komputer yang semakin dinamis dan kompleks. Istilah "dinamis dan kompleks" mengimplikasikan bahwa kondisi jaringan akan terus berubah dan semakin banyak perangkat yang akan terhubung ke jaringan komputer. Secara umum, jaringan komputer terdiri dari beberapa perangkat berkabel atau nirkabel yang saling terhubung, seperti *access point*, *router*, *switch*, dan sejumlah komputer. Jelaslah bahwa agar jaringan dapat berfungsi secara optimal, perangkat jaringan harus dikonfigurasi terlebih dahulu oleh administrator jaringan. Tantangan dalam mengelola jaringan semakin bertambah ketika ada banyak perangkat jaringan dari *vendor* yang berbeda, dengan konfigurasi manual secara satu-per-satu pada semua perangkat jaringan tentu akan memerlukan banyak waktu menyelesaikannya. Solusi otomatisasi jaringan dibutuhkan dengan tujuan untuk meningkatkan efisiensi dan akurasi dalam pengelolaan jaringan karena konfigurasi manual rentan terhadap kesalahan dan membutuhkan lebih banyak waktu. [1], [2].

Otomatisasi jaringan adalah sarana atau metode untuk menyelesaikan tugas lebih cepat dengan menggunakan logika pemrograman untuk mengelola layanan jaringan [3]. Melalui otomatisasi jaringan, administrator jaringan dapat menghilangkan tugas-tugas yang bersifat berulang seperti pencadangan *(backup)*, melihat informasi perangkat, melakukan konfigurasi dan verifikasi hanya melalui satu *platform* [4]. Kegiatan konfigurasi manual tidak efisien dan rentan terhadap kesalahan. Menurut beberapa laporan industri, kesalahan manusia menyumbang setidaknya 40% dari kegagalan jaringan (beberapa perkiraan mencapai 80%) [5]. Dalam Laporan Cisco 2020 *Business Resilience Network Survey Report* [6], 50% responden memprioritaskan otomatisasi jaringan untuk mengatasi gangguan yang terjadi saat ini. Pemanfaatan skrip Python memastikan penerapan perintah konfigurasi yang konsisten, sehingga dapat menghilangkan potensi kesalahan konfigurasi yang dilakukan secara manual oleh administrator jaringan.

Python menjadi salah satu bahasa pemrograman paling populer untuk pekerjaan otomatisasi jaringan. Python memiliki dukungan komunitas yang luas dan menawarkan banyak *library* yang mendukung otomatisasi jaringan, seperti Paramiko [7]. Paramiko menyediakan fungsi untuk membuat koneksi *Secure Shell v2* (SSHv2) ke berbagai perangkat jaringan. Hal ini memungkinkan eksekusi skrip yang berisi konfigurasi dan pengiriman skrip tersebut ke berbagai

e-ISSN: 2548-6861

perangkat yang akan dikonfigurasi melalui protokol *Secure Shell* (SSH) [8].

Terdapat beberapa buku dan penelitian yang membahas atau berhubungan dengan otomatisasi jaringan. Salah satu buku yang relevan adalah [9]. Buku ini menjelaskan tentang implementasi otomatisasi jaringan menggunakan bahasa pemrograman Python dengan memanfaatkan beberapa *library* yang mendukung otomatisasi jaringan, seperti Paramiko, Netmiko, NAPALM, PyNTC, dan Ansible. Implementasi dilakukan dengan menggunakan sistem operasi Linux Ubuntu versi 16.04. Pengujian konfigurasi dilakukan pada *router* Cisco.

Penelitian lain oleh [10] membahas implementasi otomatisasi jaringan berbasis *website* pada perangkat jaringan Cisco dan Mikrotik. Aplikasi yang dibuat dapat menampilkan jumlah perangkat *router*, menambahkan konfigurasi, memverifikasi konfigurasi, dan melihat riwayat aktivitas. Pengujian dilakukan dengan menggunakan metode *black-box* dan *white*-box, hasil penelitian menunjukkan bahwa aplikasi dapat melakukan konfigurasi penambahan *loopback interface* pada tiga buah *router* Cisco dan tiga buah *router* Mikrotik sekaligus.

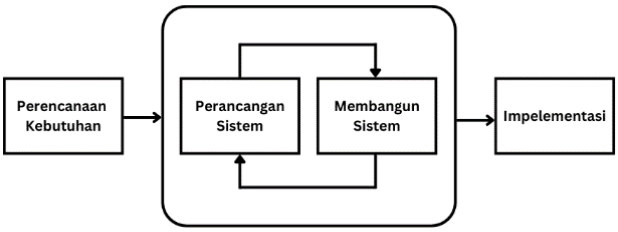
Penelitian lain oleh [11] membahas tentang pengembangan aplikasi otomatisasi jaringan berbasis *website* untuk melakukan konfigurasi *IP gateway* secara otomatis. Penelitian ini menggunakan metode pengujian *black-box* untuk menguji fungsionalitas aplikasi dan pengujian koneksi untuk memvalidasi konfigurasi penambahan *IP gateway*. Hasil pengujian menunjukkan berhasilnya penambahan *IP gateway* pada perangkat *router* Cisco dan Mikrotik yang diverifikasi dengan berhasil melakukan *ping* ke alamat 8.8.8.8 atau ke google.com.

Penelitian lain oleh [12] membahas implementasi penggunaan skrip Python dalam otomatisasi jaringan untuk mengelola *router* Mikrotik yang dimiliki PT Link Net Tbk dalam melakukan konfigurasi awal atau *provisioning* untuk pengguna baru. Metode yang digunakan adalah NDLC dalam membangun jaringan. Implementasi dilakukan menggunakan sistem operasi Linux Ubuntu versi 20.04.1 LTS, Python, dan Paramiko. Pengujian dilakukan menggunakan dua buah skrip Python yaitu "provisioning.py" untuk melakukan konfigurasi pada pelanggan baru dan "mass-update-config1.py" untuk melakukan pembaharuan konfigurasi massal *radius* dan SNMP *(Simple Network Management Protocol)* pada CPE *(Customer-Premise Equipment)* yang sudah ada. Hasil pengujian menunjukkan konfigurasi secara massal berhasil pada 10 perangkat *router* hanya dalam waktu sekitar 25 detik.

Tujuan dari penelitian ini adalah mengembangkan aplikasi berbasis *web*, untuk mengotomatisasi konfigurasi perangkat jaringan sehingga dapat mengeliminasi tugas yang bersifat berulang seperti pencadangan (*backup)*, mengkonsolidasikan informasi perangkat, konfigurasi dan verifikasi dalam satu *platform*. Pengembangan aplikasi dibuat menggunakan bahasa pemrograman Python, *library* Paramiko dan diintegrasikan dengan *website* yang dibangun menggunakan *framework* Django. Hasilnya aplikasi dapat melakukan konfigurasi *dynamic routing* OSPF *(Open Shortest Path First)*, penambahan VLAN *(Virtual Local Area Network)*, dan konfigurasi berbasis *command line* secara simultan pada beberapa perangkat jaringan. Selain itu juga dapat melakukan verifikasi konfigurasi, memiliki fungsi CRUD *(Create, Read, Update, Delete)*, menjalankan pencadangan *(backup)* secara manual dan terjadwal, serta dapat melihat riwayat aktivitas. Metode pengujian yang digunakan adalah *black-box* untuk mengetahui keberhasilan fungsional aplikasi dengan mengamati hasil *input* dan *output* dari aplikasi [13]. Selain itu dilakukan analisis efisiensi waktu dengan membandingkan konsumsi waktu konfigurasi otomatis dengan manual. Konfigurasi yang diujikan adalah *dynamic routing* OSPF pada 3 *router* Cisco dan 3 *router* Mikrotik. Pengujian dilakukan secara maya *(virtual)* menggunakan aplikasi simulasi jaringan GNS3 sebelum diimplementasikan ke lingkungan nyata. Batasan pengembangan aplikasi berfokus pada otomatisasi pencadangan *(backup)* terjadwal dan konfigurasi *dynamic routing* OSPF pada perangkat *router* Cisco dan Mikrotik.

II. METODE

Penelitian dimulai dengan melakukan tinjauan literatur pada beberapa artikel dan buku yang berkaitan dengan implementasi otomatisasi jaringan. Metode yang digunakan adalah RAD *(Rapid Application Development)* sebagai metode pendekatan berorientasi objek untuk pengembangan sistem yang bertujuan untuk mempersingkat waktu antara desain dan implementasi [2], [14], [15]. Terdapat empat langkah dalam pengembangan sistem, yaitu perencanaan kebutuhan, perancangan sistem, membangun sistem, dan implementasi. Tahapan pada metode RAD dapat dilihat pada Gambar 1.



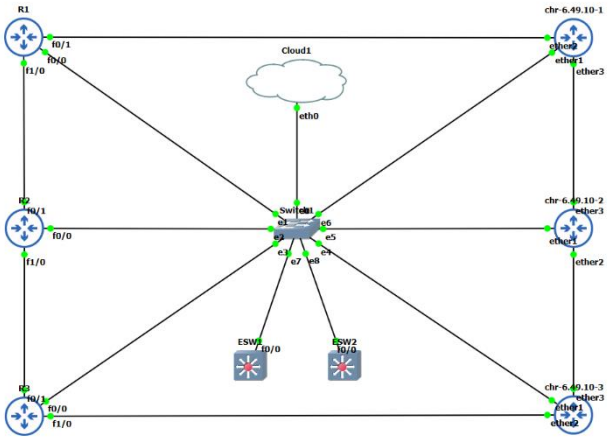
Gambar 1. Metode RAD yang memiliki 4 tahapan, yaitu perencanaan kebutuhan, perancangan sistem, membangun sistem, dan implementasi.

Tahapan pertama dalam pengembangan aplikasi adalah perencanaan kebutuhan. Pada tahap ini, identifikasi kebutuhan perangkat keras dan perangkat lunak dilakukan untuk memastikan semua elemen yang diperlukan tersedia dan kompatibel. Pada penelitian ini, perangkat keras yang digunakan adalah satu unit laptop dengan spesifikasi Intel Core i5, 16GB RAM, dan 512GB SSD untuk menjalankan semua perangkat lunak dalam pengembangan aplikasi.

Perangkat lunak terdiri dari sistem operasi Windows 11 sebagai *server/controller*. Bahasa pemrograman Python 3.12.1 digunakan untuk membuat logika konfigurasi, sementara *library* Paramiko 3.4.0 digunakan untuk melakukan koneksi *secure shell* (SSH) dan mengeksekusi perintah secara aman ke perangkat jaringan. Visual Studio Code sebagai kode editor dan Django dipilih sebagai *web framework* karena menyediakan fitur *templating, otentikasi pengguna*, dan *Object-Relational Mapping* (ORM) untuk mendefinisikan model dan basis data sebagai tempat penyimpanan data perangkat jaringan. GNS3 2.2.43 sebagai alat simulasi jaringan, dengan VMware Workstation 17 Pro untuk menjalankan GNS3 VM. Selain itu *image* perangkat Cisco dan Mikrotik dipasang ke dalam aplikasi GNS3 untuk simulasi.

Selain kebutuhan perangkat keras dan perangkat lunak, aplikasi ini juga memiliki kebutuhan fungsional dan non-fungsional. Kebutuhan fungsional aplikasi memungkinkan pengguna untuk mendaftar, masuk, dan keluar dari aplikasi; menambah konfigurasi *dynamic routing Open Shortest Path First* (OSPF); menambahkan konfigurasi *Virtual Local Area Network* (VLAN); melakukan konfigurasi berbasis *command line*; melakukan verifikasi konfigurasi; menjalankan pencadangan *(backup)* secara manual dan terjadwal; melakukan operasi CRUD *(Create, Read, Update, Delete)* pada perangkat jaringan, dan melihat riwayat aktivitas. Kebutuhan non-fungsional aplikasi mencakup kompatibilitas terhadap berbagai browser.

Tahapan kedua dalam pengembangan aplikasi adalah perancangan sistem. Pada tahapan ini dirancang sebuah topologi dan desain sistem. Topologi jaringan dibuat menggunakan aplikasi GNS3 dimana terdapat satu *cloud* yang merupakan perangkat fisik laptop yang dijadikan sebagai *server/controller*, 3 *router* Cisco c3725, 3 *router* Mikrotik CHR-6.49.10 dan 2 *switch* Cisco. Desain topologi dapat dilihat pada Gambar 2.



Gambar 2. Desain topologi jaringan yang terdiri dari satu *cloud*, tiga *router* Cisco, tiga *router* Mikrotik, dan dua *switch* Cisco (EtherSwitch).

Setiap perangkat telah diberikan alokasi alamat *IP* yang harus dikonfigurasi terlebih dahulu pada masing-masing perangkat. Terdapat 9 buah perangkat yang harus dikonfigurasi alamat *IP* sesuai pada gambar topologi jaringan yang sudah dirancang sebelumnya. Alokasi alamat *IP* dapat dilihat pada Tabel 1.

TABEL I
ALOKASI ALAMAT IP

No	Perangkat	Antarmuka	Alamat IP
1	Cloud/Controller	GNS3 eth0	192.168.227.128/24
2	R1 Cisco	Fa0/0	192.168.227.11/24
		Fa0/1	10.10.11.1/30
		Fa1/0	10.10.12.1/30
3	R2 Cisco	Fa0/0	192.168.227.12/24
		Fa0/1	10.10.12.2/30
		Fa1/0	10.10.13.1/30
4	R3 Cisco	Fa0/0	192.168.227.13/24
		Fa0/1	10.10.13.2/30
		Fa1/0	10.10.14.1/30
5	CHR-6.49.10-1 Mikrotik	Ether1	192.168.227.14/24
		Ether2	10.10.11.2/30
		Ether3	10.10.16.2/30
6	CHR-6.49.10-2 Mikrotik	Ether1	192.168.227.15/24
		Ether2	10.10.15.2/30
		Ether3	10.10.16.1/30
7	CHR-6.49.10-3 Mikrotik	Ether1	192.168.227.16/24
		Ether2	10.10.14.2/30
		Ether3	10.10.15.1/30
8	ESW1 Cisco	Fa0/0	192.168.227.17/24
9	ESW2 Cisco	Fa0/0	192.168.227.18/24

Selain itu, setiap perangkat juga harus dikonfigurasi untuk mengaktifkan SSHv2 agar dapat diakses dari jarak jauh oleh *server/controller*. Konfigurasi SSHv2 dapat dilihat pada Gambar 3.

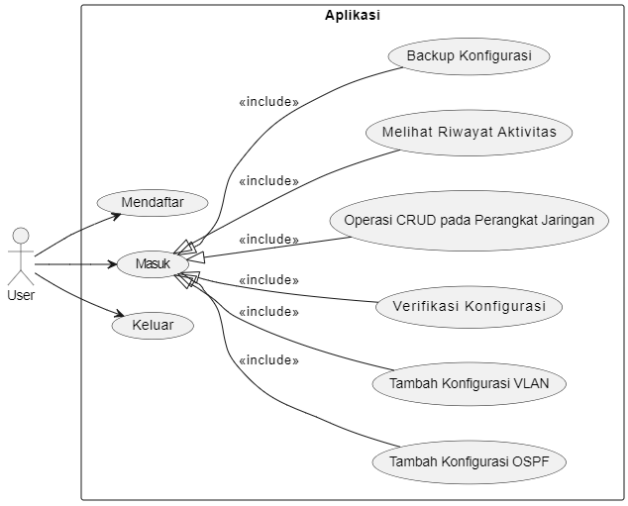
```
R1(config)#username cisco password cisco
R1(config)#username cisco privilege 15
R1(config)#ip domain-name anco.local
R1(config)#crypto key generate rsa modulus 1024
The name for the keys will be: R1.anco.local

% The key modulus size is 1024 bits
% Generating 1024 bit RSA keys, keys will be non-exportable...[OK]

R1(config)#line vty 0 15
R1(config-line)#login local
R1(config-line)#end
R1#wr
Building configuration...
[OK]
R1#
*Mar  1 01:12:09.555: %SSH-5-ENABLED: SSH 1.99 has been enabled
*Mar  1 01:12:09.619: %SYS-5-CONFIG_I: Configured from console by console
R1#
```

Gambar 3. Konfigurasi SSHv2 pada terminal router Cisco

Perancangan desain sistem dibuat menggunakan diagram *Unified Modelling Language* (UML). Penelitian ini menggunakan *use case diagram* untuk mengilustrasikan interaksi fungsi utama sistem dan pengguna [16]. Pada *use case diagram*, seorang Administrator jaringan digambarkan sebagai aktor yang berinteraksi dengan sistem. Aplikasi yang akan dikembangkan diberi nama ANCO, akronim dari

e-ISSN: 2548-6861

Automated Network Configuration. Aktor dapat mendaftar dan masuk ke dalam aplikasi. Aktor dapat mengatur data perangkat jaringan melalui operasi CRUD, melakukan konfigurasi *dynamic routing* OSPF, menambah VLAN, dan konfigurasi lewat *command line* secara simultan, melihat riwayat aktivitas, melakukan *backup* dan keluar dari aplikasi. Desain *use case diagram* dapat dilihat pada Gambar 4.



Gambar 4. *Use Case Diagram* dimana aktor digambarkan sebagai seorang administrator jaringan yang berinteraksi dengan aplikasi atau sistem.

Tahapan ketiga adalah membangun sistem. Pada tahapan ini aplikasi dibangun menggunakan menggunakan Python sebagai bahasa pemrograman dan menggunakan *library* Paramiko sebagai modul melakukan koneksi *secure shell* (SSH) ke perangkat jaringan yang terhubung dan *framework* Django untuk membangun aplikasi berbasis *website*.

Tahapan keempat adalah implementasi. Implementasi dilakukan secara simulasi dengan menggunakan aplikasi GNS3 sebelum diimplementasikan ke lingkungan nyata. Pengujian dilakukan menggunakan metode *black-box* untuk mengetahui keberhasilan fungsional aplikasi yang sudah dirancang sebelumnya.

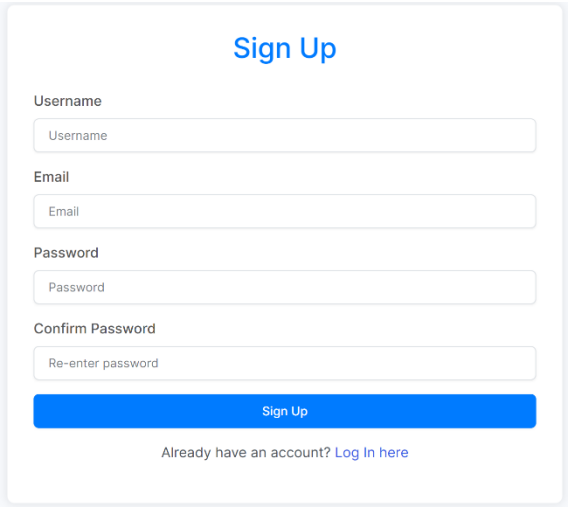
III. HASIL DAN PEMBAHASAN

Penelitian ini menghasilkan sebuah aplikasi otomatisasi konfigurasi perangkat jaringan berbasis *website* yang diberi nama ANCO yang merupakan akronim dari *Automated Network Configuration*. Aplikasi ini hanya dijalankan oleh seorang administrator jaringan dalam melakukan konfigurasi perangkat jaringan.

A. Tampilan Antarmuka Aplikasi

Aplikasi ini memiliki tampilan antarmuka yang terdiri dari beberapa halaman berdasarkan fungsinya masing-masing, yaitu *Sign Up*, *Login*, *Dashboard*, *Network Device*, *Dropdown Config (*OSPF *Config*, VLAN *Config*, *Command Line*, *Check Configuration)*, *Backup*, dan *History*. Pengembangan antarmuka aplikasi menggunakan *frontend toolkit* Bootstrap v5.3 dengan tujuan untuk membuat aplikasi responsif dan mempercepat proses pembuatannya karena sudah tersedia kumpulan komponen *User Interface* (UI) yang siap pakai.

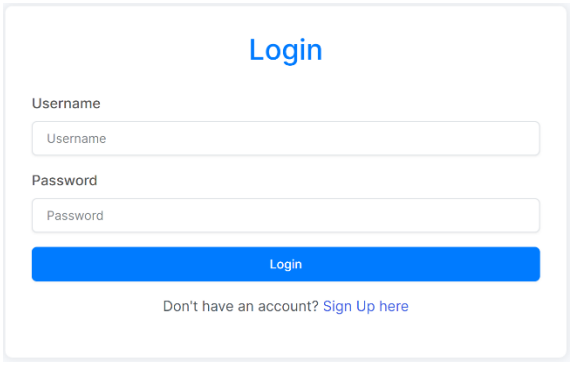
1) *Halaman Sign Up:* Halaman ini digunakan untuk mendaftarkan akun baru. Pada halaman ini, pengguna baru diminta untuk mengisi informasi berikut: *username*, *email*, *password*, dan *confirm password*. Setelah mengisi informasi tersebut, pengguna dapat klik tombol "*Sign Up*" untuk memproses pendaftaran. Jika data yang dimasukkan benar, pengguna baru akan dialihkan ke halaman *Login*. Namun, jika terdapat kesalahan dalam pengisian, data tidak akan diproses dan akan ditampilkan pesan kesalahannya dan diharuskan mengisi ulang informasi yang diminta. Pengguna juga memiliki opsi untuk langsung masuk ke aplikasi jika sudah memiliki akun sebelumnya dengan cara klik pada tautan "*Already have an account? Log In here*". Tampilan halaman *Sign Up* dapat dilihat pada Gambar 5.



Gambar 5. Halaman *Sign Up* untuk mendaftarkan akun pengguna baru.

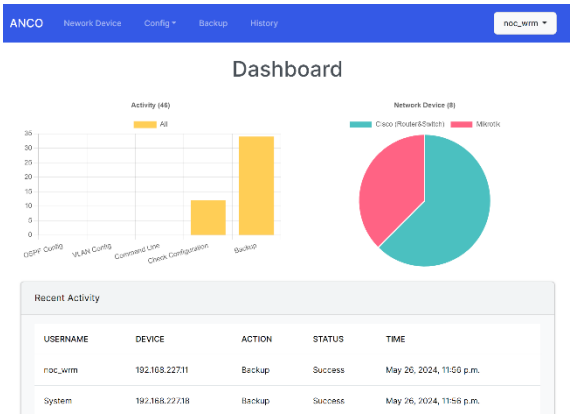
2) *Halaman Login:* Halaman ini digunakan untuk masuk ke dalam aplikasi. Seorang administrator terlebih dahulu harus melakukan proses *login* sebelum dapat menggunakan aplikasi. Hal ini dilakukan untuk mencegah penggunaan aplikasi oleh pihak yang tidak berwenang. pengguna harus memasukkan *username* dan *password* pada formulir *login* yang disediakan dengan benar agar dapat masuk ke dalam aplikasi. Jika data yang dimasukkan tidak benar, pengguna akan

kembali diarahkan ke halaman *login* untuk memasukkan ulang *username* dan *password*. Pengguna juga memiliki opsi untuk mendaftar akun baru jika belum memiliki akun sebelumnya dengan cara klik pada tautan "*Don't have an account? Sign Up here*". Tampilan halaman *Login* dapat dilihat pada Gambar 6.



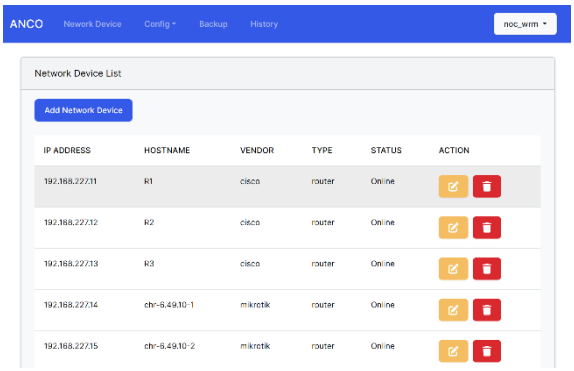
Gambar 6. Halaman *Login* untuk masuk ke dalam aplikasi oleh pengguna terdaftar.

3) *Halaman Dashboard:* Halaman *dashboard* menjadi halaman pertama yang diakses pengguna setelah berhasil masuk *(login)* ke aplikasi. Halaman ini menampilkan informasi total perangkat yang terhubung ke aplikasi, dan aktivitas terakhir dalam bentuk grafik dan tabel. Pada bagian *navbar* terdapat 4 menu utama, yaitu *Network Device*, *Dropdown Config*, *Backup*, dan *History*. Pada bagian ujung *navbar* ditampilkan juga nama pengguna beserta tombol *"Logout"* dalam bentuk *dropdown*. Tampilan halaman *dashboard* dapat dilihat pada Gambar 7.



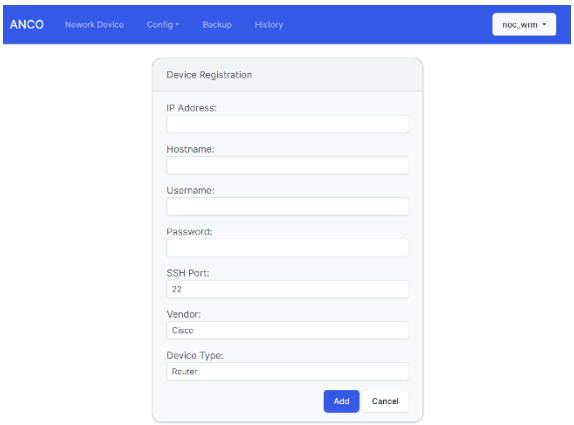
Gambar 7. Halaman *dashboard* menampilkan informasi total jumlah perangkat dari kedua *vendor* Cisco dan Mikrotik, beserta aktivitas terakhir dalam bentuk grafik dan tabel.

4) *Halaman Network Device:* Halaman ini menampilkan daftar perangkat jaringan yang terhubung ke aplikasi. Informasi yang ditampilkan berupa *IP address*, *hostname*, *vendo*r, *type*, *status*, dan *action* yang memiliki dua opsi yaitu *edit* dan *delete*. Pada halaman ini juga disediakan sebuah tombol "*Add Network Device*" untuk mendaftarkan perangkat jaringan ke dalam basis data aplikasi. Tampilan halaman *Network Device* dapat dilihat pada Gambar 8.



Gambar 8. Halaman *Network Device* digunakan untuk menambahkan perangkat, menampilkan informasi perangkat yang tersimpan, dan tindakan yang memiliki 2 opsi, yaitu *edit* dan *delete*.

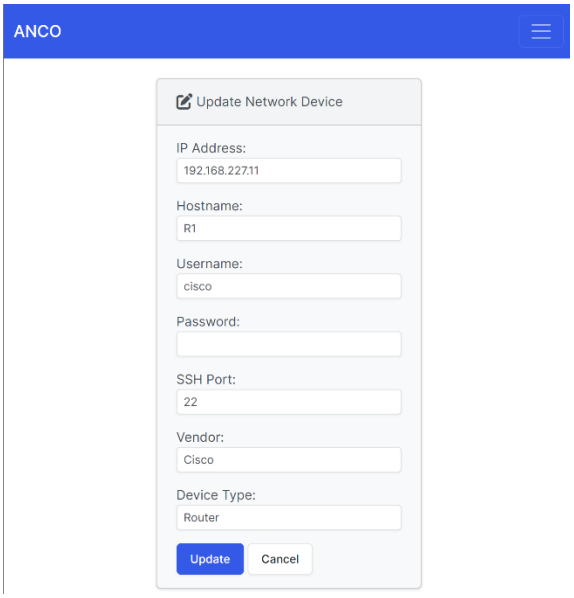
Pengguna dapat menambahkan perangkat jaringan ke dalam basis data aplikasi dengan klik tombol *"Add Network Device"* yang akan dialihkan ke halaman yang berisi formulir registrasi. Beberapa informasi yang perlu diisi untuk menambahkan perangkat, yaitu *IP address*, *hostname*, *username*, *password*, SSH *port*, *vendor*, dan *device type*. Tampilan formulir registrasi perangkat dapat dilihat pada Gambar 9.



Gambar 9. Tampilan formulir registrasi penambahan perangkat jaringan ke basis data sistem.

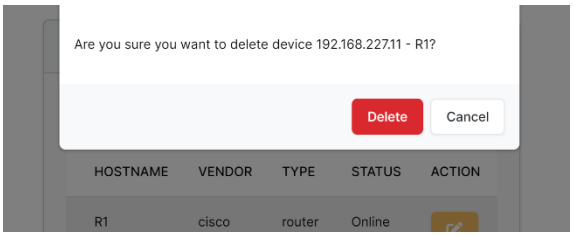
e-ISSN: 2548-6861

Selain fungsi menambahkan perangkat, pengguna juga dapat mengubah data perangkat dengan klik tombol "*Edit"* pada kolom *action*. Pengguna akan dialihkan ke halaman yang sama seperti halaman penambahan perangkat, pengguna dapat mengubah data perangkat pada formulir yang disediakan. Setelah selesai mengubah data, pengguna dapat menekan tombol "*Update*" untuk menyimpan perubahan ke basis data. Tampilan formulir sunting perangkat dapat dilihat pada Gambar 10.



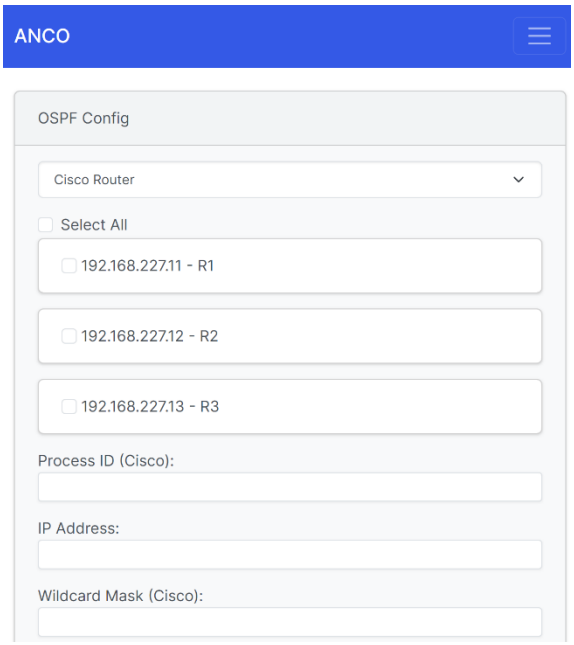
Gambar 10. Tampilan formulir sunting data perangkat jaringan untuk memperbaharui informasi perangkat.

Selain fungsi perubahan data, pengguna juga dapat menghapus data perangkat jaringan dari basis data apabila sudah tidak digunakan dengan klik tombol "*Delete"* pada kolom *action*. Tampilan fungsi *delete* dapat dilihat pada Gambar 11.



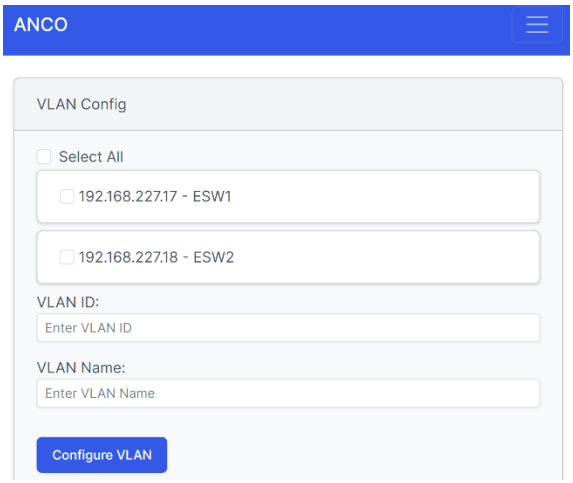
Gambar 11. Fungsi *delete* untuk menghapus data perangkat jaringan dari basis data.

5) *Halaman Config:* Halaman ini digunakan untuk melakukan konfigurasi pada perangkat jaringan yang terhubung ke sistem aplikasi. Halaman ini memiliki *submenu*, yaitu OSPF *Config*, VLAN *Config*, *Command Line*, dan *Check Configuration*. Sebelum melakukan konfigurasi pengguna harus memilih perangkat yang akan dikonfigurasi pada daftar perangkat yang tersedia. Pada konfigurasi *dynamic routing* OSPF pengguna hanya perlu memasukkan beberapa parameter konfigurasi pada formulir yang sudah disediakan, yaitu *process ID* (Cisco), *IP address*, *wildcard mask* (Cisco), *subnet mask*/*prefiks*, dan *area ID*. Setelah selesai pengguna dapat klik tombol "*Submit*" untuk mengirimkan skrip konfigurasi ke perangkat jaringan yang telah dipilih secara simultan. Tampilan halaman konfigurasi OSPF dapat dilihat pada gambar 12.



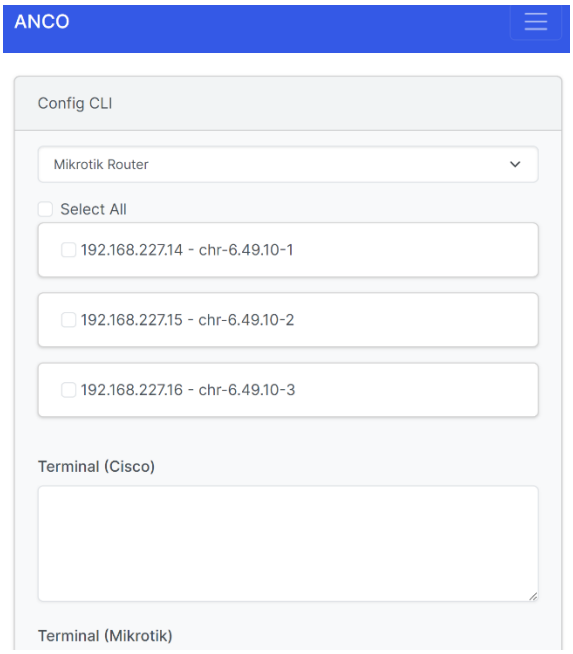
Gambar 12. Tampilan formulir konfigurasi *dynamic routing* OSPF dengan memilih daftar perangkat *Cisco*

Pada konfigurasi penambahan VLAN daftar perangkat yang disediakan hanya perangkat *switch* Cisco. pengguna hanya perlu memasukkan nomor VLAN, dan nama VLAN. Setelah selesai pengguna dapat klik tombol "*Submit*" untuk mengirimkan skrip konfigurasi ke perangkat jaringan yang telah dipilih secara simultan. Tampilan halaman konfigurasi VLAN dapat dilihat pada gambar 13.



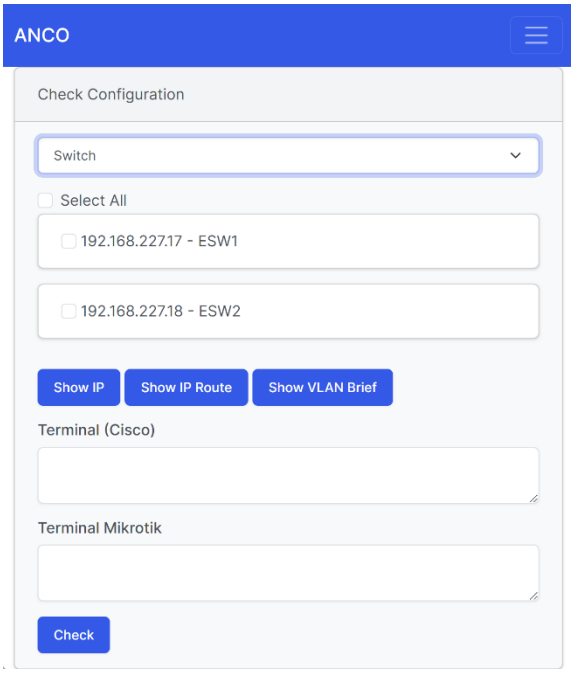
Gambar 13. Tampilan formulir konfigurasi VLAN pada *switch* Cisco

Selain konfigurasi *dynamic routing* OSPF dan VLAN dengan menggunakan formulir, pengguna juga dapat melakukan konfigurasi lain berbasis *command line*. Fitur ini memungkinkan pengguna tidak perlu lagi masuk secara satu-per-satu ke setiap perangkat jaringan karena dengan menggunakan terminal konfigurasi yang disediakan, pengguna dapat melakukan konfigurasi secara simultan pada semua perangkat jaringan. Tampilan halaman konfigurasi berbasis *command line* dapat dilihat pada Gambar 14.



Gambar 14. Tampilan Halaman *Config* untuk melakukan konfigurasi pada perangkat jaringan berbasis *command line*.

Halaman Check Configuration: Halaman ini digunakan untuk memeriksa konfigurasi pada perangkat jaringan. Terdapat beberapa tombol yang dapat digunakan untuk memeriksa konfigurasi umum seperti *"Show IP"*, *"Show IP Route"*, dan *"Show VLAN Brief"*. Selain itu, disediakan juga fitur *command line* untuk memeriksa konfigurasi lainnya. Tampilan halaman dapat dilihat pada Gambar 15.



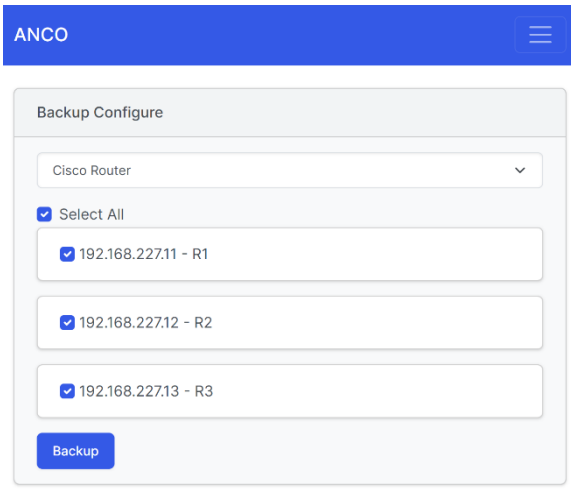
Gambar 15. Tampilan halaman *check configuration* untuk memeriksa hasil konfigurasi perangkat *switch* Cisco.

Sebelum melakukan pemeriksaan konfigurasi, pengguna harus terlebih dahulu memilih perangkat apa saja yang akan diperiksa pada daftar perangkat yang tersedia. Setelah memilih perangkat, pengguna dapat menekan tombol yang menyediakan fungsi cek konfigurasi umum atau menuliskan skrip pada *textarea* sesuai dengan *vendor* yang dipilih. Setelah selesai, pengguna dapat melakukan klik tombol "*Submit*" untuk mendapatkan hasil pengecekan. skrip akan dijalankan secara simultan pada semua perangkat yang dipilih. Tampilan halaman hasil pengecekan dapat dilihat pada Gambar 16.

e-ISSN: 2548-6861

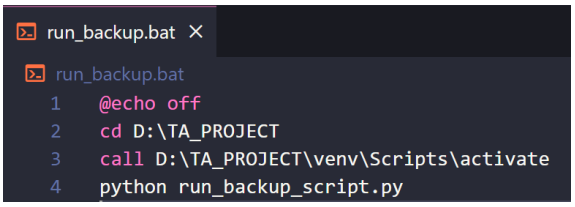
Gambar 16. Tampilan hasil pengecekan konfigurasi VLAN pada perangkat *switch* Cisco.

6) *Halaman Backup:* Halaman ini digunakan untuk melakukan pencadangan konfigurasi perangkat. Pengguna dapat memilih perangkat yang akan dicadangkan pada daftar perangkat yang tersedia. Hasil pencadangan dalam bentuk file.txt yang disimpan pada direktori proyek dengan nama folder "*backups*". Folder akan dibuat secara otomatis melalui skrip konfigurasi jika belum tersedia. Tampilan halaman *backup* dapat dilihat pada Gambar 17.

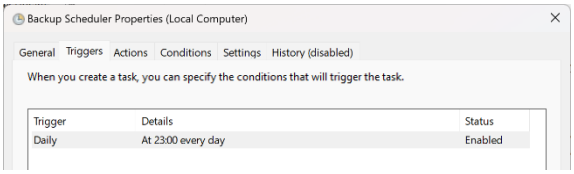


Gambar 17. Tampilan halaman *backup* untuk melakukan pencadangan pada perangkat *router* Cisco

Selain pencadangan secara manual oleh pengguna, aplikasi dapat melakukan pencadangan otomatis dengan skrip yang akan dijalankan secara terjadwal oleh sistem. Skrip ini dipanggil dengan bantuan program Windows Task Scheduler yang diatur untuk menjalankan program pada pukul 23:00 setiap hari. Tampilan skrip *backup* dan pengaturan program Windows Task Scheduler dapat dilihat pada Gambar 18 dan Gambar 19.

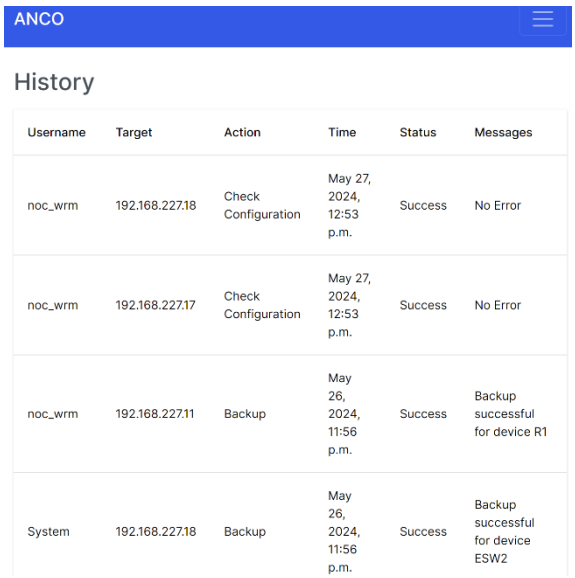


Gambar 18. Skrip pencadangan yang dijalankan oleh *task scheduler*.



Gambar 19 tugas pencadangan diatur pada pukul 23:00 setiap hari menggunakan *task scheduler*.

7) *Halaman History:* Halaman ini menampilkan riwayat aktivitas pengguna atau administrator jaringan. Menu ini menampilkan informasi aktivitas konfigurasi, *backup*, dan pemeriksaan konfigurasi yang dilakukan oleh pengguna. Selain itu, juga ditampilkan informasi *username*, alamat *IP*, aksi, waktu eksekusi, status dan pesan. Tampilan Halaman *history* dapat dilihat pada Gambar 20.



Gambar 20. Halaman *history* menampilkan aktivitas yang dilakukan oleh pengguna.

B. Black-box Testing

Pengujian *black-box* dilakukan untuk mengetahui keberhasilan fungsionalitas aplikasi yang sudah dirancang sebelumnya dengan mengamati hasil *input* dan *output* dari aplikasi. Hasil pengujian dapat dilihat pada Tabel 2.

TABEL II
HASIL PENGUJIAN BLACK BOX

No	Fitur	Skenario	Hasil
1	*Sign Up*	Mendaftarkan pengguna baru dan memverifikasi pengguna berhasil terdaftar di basis data	berhasil
2	*Login*	Masuk ke dalam aplikasi menggunakan akun yang sudah didaftarkan	berhasil
3	CRUD	Menambahkan perangkat ke basis data	berhasil
		Melihat daftar dan status perangkat yang terdaftar	berhasil
		Melakukan pembaharuan data perangkat	berhasil
		Menghapus perangkat yang terdaftar	berhasil
4	Konfigurasi	Menambahkan konfigurasi *dynamic routing* OSPF pada *router*	berhasil
		Menambahkan konfigurasi VLAN pada *switch*	berhasil
		Menambahkan konfigurasi menggunakan *command line*	berhasil
		Menjalankan dan melihat hasil verifikasi konfigurasi	berhasil
6	*Backup*	Menjalankan *backup* oleh pengguna	berhasil
		Menjalankan *backup* oleh Sistem (otomatis)	berhasil
7	*Logout*	Keluar dari aplikasi	berhasil

C. Analisa Efisiensi Konfigurasi Manual vs Otomatis

Pengujian efisiensi konfigurasi manual dengan konfigurasi otomatis melibatkan pengujian konfigurasi *dynamic routing* OSPF *single-area*. Hal ini sesuai dengan fitur yang telah dibuat pada aplikasi. Konfigurasi diimplementasikan pada 3 *router* Cisco dan 3 *router* Mikrotik. Proses konfigurasi menggunakan aplikasi Putty dengan melakukan *remote* pada perangkat melalui protokol SSH. Perhitungan waktu konfigurasi menggunakan aplikasi Wireshark dengan merekam lalu lintas paket SSH. Perhitungan waktu dimulai sejak paket SSH pertama kali ditangkap atau proses dimulai hingga selesai melakukan konfigurasi. Dilakukan pengujian sebanyak 5 kali untuk memperoleh lebih banyak data dan membantu dalam memvalidasi konsistensi hasil. Hasil pengujian dapat dilihat pada Tabel 3 dan Tabel 4.

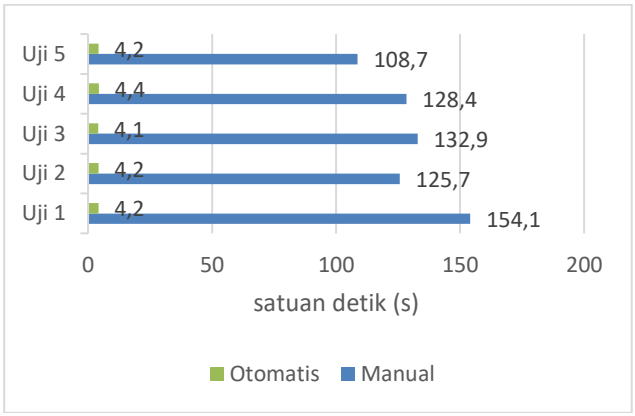
TABEL III
HASIL PENGUJIAN KONFIGURASI MANUAL ROUTING OSPF

Uji	Cisco			Mikrotik			Total	*Mean*
	R1	R2	R3	R1	R2	R3		
1	25.0	30.4	41.0	20.9	16.8	20.0	154.1*s*	25.7*s*
2	25.5	24.0	25.1	17.7	14.1	19.3	125.7	21.0*s*
3	36.5	21.2	23.1	21.4	15.4	15.3	132.9*s*	22.2*s*
4	28.7	26.0	21,9	13.8	18.8	19,2	128.4*s*	21.4*s*
5	25.9	19,0	18,6	18.4	11.4	15.4	108.7	18.1*s*
Rata-rata waktu konfigurasi seluruh perangkat: 130*s*								

TABEL IV
HASIL KONFIGURASI OTOMATIS ROUTING OSPF

Uji	Jumlah perangkat	Waktu	Mean
1	6 (Cisco & Mikrotik)	4.2*s*	4.2*s*
2		4.2*s*	
3		4.1*s*	
4		4.4*s*	
5		4.2*s*	

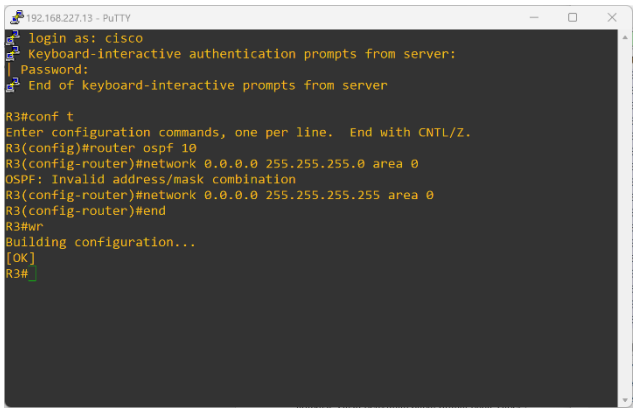
Konfigurasi otomatis hanya membutuhkan rata-rata waktu sekitar 4,2 detik untuk melakukan konfigurasi pada seluruh perangkat, sedangkan konfigurasi manual membutuhkan rata-rata waktu sekitar 130 detik. Hal ini menunjukkan konfigurasi otomatis dapat menghemat waktu 125,8 detik atau sekitar 97% dibandingkan dengan manual.



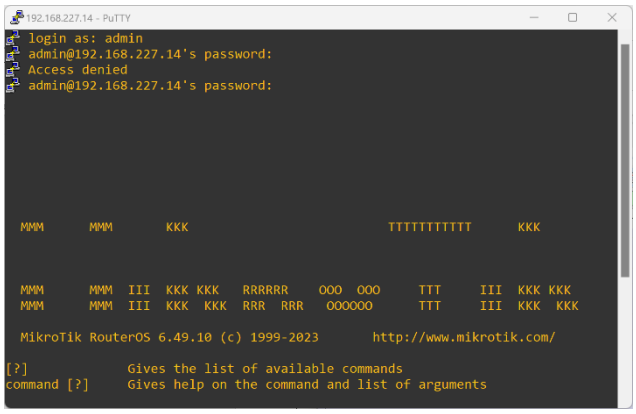
Gambar 21. Waktu rata-rata konfigurasi seluruh perangkat

Konsistensi waktu konfigurasi otomatis juga lebih tinggi dibandingkan dengan manual. Dapat dilihat pada Gambar 21 waktu konfigurasi manual bervariasi dari 108,7 detik hingga 154,1 detik, sedangkan waktu konfigurasi otomatis cukup konsisten, berkisar dari 4,1 detik hingga 4,4 detik. Selain itu ditemukan 3 kesalahan dalam penulisan konfigurasi manual terdiri dari 2 kesalahan dalam penentuan *wildcard mask* pada *router* Cisco dan 1 kesalahan penulisan *password* pada *router* Mikrotik seperti terlihat pada Gambar 22 dan 23.

e-ISSN: 2548-6861



Gambar 22. Temuan kesalahan penulisan *wildcard mask* konfigurasi *dynamic routing* OSPF pada *router* Cisco



Gambar 23. Temuan kesalahan login password pada *router* Mikrotik

Berdasarkan hasil pengujian dan analisis yang sudah dilakukan dapat disimpulkan, penggunaan aplikasi otomatisasi konfigurasi perangkat jaringan berbasis web memberikan peningkatan efisiensi yang signifikan dalam hal waktu dan konsistensi konfigurasi dibandingkan dengan metode konfigurasi manual.

IV. KESIMPULAN

Penelitian ini menghasilkan sebuah aplikasi otomatisasi konfigurasi perangkat jaringan berbasis *website* yang dikembangkan menggunakan pemrograman Python, *library* Paramiko, dan *framework* Django. Aplikasi ini memiliki kemampuan untuk menjalankan operasi CRUD pada perangkat jaringan, melakukan konfigurasi OSPF, VLAN, dan konfigurasi lewat *command line* secara simultan. Aplikasi juga dapat digunakan untuk memeriksa dan menampilkan hasil konfigurasi, melakukan *backup* secara manual dan terjadwal, dan melihat riwayat aktivitas pengguna. Pengujian pada aplikasi dilakukan menggunakan metode *black-box* menunjukkan keberhasilan fungsional aplikasi yang sudah dirancang sebelumnya. Selain itu dilakukan juga analisa efisiensi antara konfigurasi otomatis dan konfigurasi manual. Berdasarkan analisa yang sudah dilakukan konfigurasi otomatis hanya membutuhkan rata-rata waktu sekitar 4,2 detik untuk melakukan konfigurasi pada seluruh perangkat, sedangkan konfigurasi manual membutuhkan rata-rata waktu sekitar 130 detik. Hal ini menunjukkan konfigurasi otomatis dapat menghemat waktu 125,8 detik atau sekitar 97% dibandingkan dengan manual. Selain itu, aplikasi ini juga mengurangi kesalahan konfigurasi yang umum terjadi pada metode manual.

Penulis menyadari masih banyak keterbatasan fungsi pada aplikasi yang dikembangkan saat ini. Saran untuk pengembangan selanjutnya adalah menyediakan fitur formulir *routing static* dan *dynamic* yang lebih lengkap, penambahan fitur lain seperti pengaturan *firewall* dan *Access Control List* (ACL), serta mendukung perangkat dari *vendor* lain.

UCAPAN TERIMA KASIH

Penulis mengucapkan terima kasih kepada Politeknik Negeri Batam dan semua individu yang telah berkontribusi sehingga penelitian ini dapat diselesaikan.

DAFTAR PUSTAKA

[1] V. V. S. S. S. Balaram, C. Mukundha, dan S. Bhutada, "Enhancement of network administration through software defined networks," *IOSR Journal of Computer Engineering (IOSR-JCE)*, vol. 18, no. 1, hlm. 30–36, 2016, doi: 10.9790/0661-18113036.

[2] R. A. Wiryawan dan N. R. Rosyid, "Pengembangan aplikasi otomatisasi administrasi jaringan berbasis website menggunakan bahasa pemrograman python," *Jurnal SIMETRIS*, vol. 10, no. 2, hlm. 741–752, Nov 2019, doi: 10.24176/simet.v10i2.3589.

[3] G. S. Santyadiputra, I. M. E. Listartha, dan G. A. J. Saskara, "The effectiveness of Automatic Network Administration (ANA) in network automation simulation at Universitas Pendidikan Ganesha," dalam *Journal of Physics: Conference Series*, IOP Publishing Ltd, Mar 2021. doi: 10.1088/1742-6596/1810/1/012028.

[4] Red Hat, *Network Automation for Everyone: Modernize Your Network With Red Hat Ansible Automation*. Red Hat, Inc., 2021. Diakses: 24 Februari 2023. [Daring]. Tersedia pada: https://www.redhat.com/en/engage/network-automation-everyone-201901070306

[5] K. Fultz, M. Krishnarao, dan S. Wu, *Network Automation VMware ® Special Edition*. Hoboken, New Jersey: John Wiley & Sons, Inc., 2020. Diakses: 24 Februari 2023. [Daring]. Tersedia pada: https://www.vmware.com/content/microsites/learn/en/544472_REG.html

[6] Cisco, "2021 Global Networking Trends Report," 2020. Diakses: 24 Februari 2023. [Daring]. Tersedia pada: https://www.cisco.com/c/dam/en/us/solutions/enterprise-networks/2021-networking-report.pdf

[7] A. Ratan, *Practical Network Automation*, 2 ed. Birmingham, UK: Packt Publishing, 2018. Diakses: 24 Februari 2023. [Daring]. Tersedia pada: https://learning.oreilly.com/library/view/practical-network-automation/9781789955651/

[8] K. Nugroho, A. D. Abrariansyah, dan S. Ikhwan, "Perbandingan kinerja library paramiko dan netmiko dalam proses otomasi jaringan," *InfoTekJar : Jurnal Nasional Informatika dan Teknologi Jaringan*, vol. 5, no. 1, hlm. 1–8, Sep 2020, doi: 10.30743/infotekjar.v5i1.2758.

[9] A. R. Komarudin, *Otomatisasi Administrasi Jaringan Dengan Script Python*. Jasakom, 2018.

[10] S. Nugroho dan B. Pujiarto, "Network automation pada beberapa perangkat router menggunakan pemrograman python," *Jurnal Teknologi Informasi dan Ilmu Komputer (JTIIK)*, vol. 9, no. 1, hlm. 79–86, 2022, doi: 10.25126/jtiik.2022913947.

[11] E. S. Ginting, Suroso, dan I. Hadi, "Pengujian Konfigurasi Otomatis Penambahan Gateway Pada Virtual Router Menggunakan Aplikasi Otomatisasi Jaringan Berbasis Web," *Jurnal Media Informatika Budidarma*, vol. 4, hlm. 1126–1131, 2020, doi: 10.30865/mib.v4i4.2485.

[12] M. Fahmi, M. Maisyaroh, I. Komarudin, S. Faizah, dan I. Fadhillah, "Otomatisasi Jaringan Menggunakan Script Python Untuk Penyediaan Konfigurasi Internet Dan Manajemen Mikrotik," *Bina Insani ICT Journal*, vol. 8, no. 1, hlm. 53–62, Jun 2021, doi: 10.51211/biict.v8i1.1517.

[13] Y. C. E. Paksi dan I. R. Widiasari, "Perancangan dan implementasi website network automation pada RT RW NET Dusun Senden Magelang dengan framework django," *Jurnal Teknik Informatika (JUTIF)*, vol. 3, no. 5, hlm. 1313–1322, Okt 2022, doi: 10.20884/1.jutif.2022.3.5.350.

[14] K. E. Kendall dan J. E. Kendall, *Systems Analysis And Design*, 8 ed. Upper Saddle River, New Jersey: Pearson Education, Inc., 2010.

[15] M. S. J. Pradana, A. Hendrawan, B. V. Christioko, dan A. P. R. Pinem, "Implementasi sistem administrasi di Unit Pelaksana Teknis Pusat Pengembangan Publikasi Ilmiah Dosen Universitas Semarang berbasis website," *Jurnal Teknologi Informasi dan Ilmu Komputer (JTIIK)*, vol. 9, no. 3, hlm. 501–506, 2022, doi: 10.25126/jtiik.2021864089.

[16] A. Dennis, B. H. Wixom, dan D. Tegarden, *Systems Analysis & Design An Object-Oriented Approach With UML*, 5 ed. Hoboken, New Jersey: John Wiley & Sons, Inc, 2015.