Perancangan Backend pada Sistem Otomasi Konfigurasi dan Monitoring Perangkat Jaringan Multivendor pada Arsitektur Microservice

1st Jesikapna Kristina Br Tarigan
Fakultas Teknik Elektro
Universitas Bandung
Bandung, Indonesia
Jesikristina@student.telkomuniversity.ac.id

2nd Favian Dewanta
Fakultas Teknik Elektro
Universitas Bandung
Bandung, Indonesia
favian@telkomuniversity.ac.id

3rd Bagus Aditya
Fakultas Teknik Elektro
Universitas Bandung
Bandung, Indonesia
goesaditya@telkomuniversity.ac.id

***Abstrak* — Dalam dunia jaringan, kompleksitas integrasi perangkat dari vendor yang berbeda menjadi tantangan tersendiri bagi perusahaan atau penyedia layanan internet (ISP). Keberagaman vendor perangkat jaringan ini disebabkan oleh keunggulan yang dimiliki oleh masing-masing vendor dalam sektor tertentu. Namun, integrasi perangkat dari vendor yang berbeda dapat menimbulkan masalah dalam menerapkan otomasi jaringan, terutama dalam hal integrasi SNMP (Simple Network Management Protocol) ke dalam program otomasi jaringan multi vendor. Dalam penelitian sebelumnya, Python telah berhasil diimplementasikan sebagai bahasa pemrograman yang efektif untuk otomasi jaringan, dengan dukungan dari beberapa library seperti Paramiko dan Netmiko untuk konfigurasi perangkat jaringan multivendor melalui protokol SSH (Secure Shell). Protokol SSH terbukti efektif dalam menyederhanakan proses konfigurasi dan konfigurasi perangkat jaringan secara massal. Oleh karena itu, dalam penelitian ini, penulis menggunakan protokol SSH sebagai dasar dalam pengembangan sistem otomasi konfigurasi dan monitoring Perangkat Jaringan Multivendor. Selain itu, protokol SNMP juga digunakan sebagai alat bantu untuk menciptakan tool otomasi yang lebih canggih. Dari hasil penelitian ini didapatkan bahwa untuk melakukan konfigurasi perangkat dari vendor yang berbeda-beda dalam satu waktu hanya membutuhkan waktu yang singkat. Pada fitur configure didapatkan rata-rata waktu 7,085 detik dan pada fitur verify config didapatkan rata-rata waktu 8,68 detik.**

Kata Kunci : Network Automation, Multivendor Networking, konfigurasi Automation, Monitoring Automation

I. PENDAHULUAN

Vendor merupakan pihak yang menjadi pemasok peralatan yang dibutuhkan perusahaan. Dalam hal ini vendor yang dimaksud berfokus di infrastruktur jaringan suatu perusahaan, seiring berkembangnya suatu perusahaan atau ISP maka perusahaan tersebut menggunakan vendor perangkat jaringan yang bervariatif[1]. Hal ini bisa terjadi karena beberapa vendor memiliki keunggulan pada sektor tertentu, yang membuat perusahaan memiliki beberapa perangkat dari vendor yang berbeda, hal ini dapat menimbulkan masalah ketika ingin mengintegrasikan otomasi pada perangkat dengan vendor yang berbeda, dikarenakan setiap vendor memiliki tool property yang beda pula.

Kompleksitas masalah Network automation multi vendor ini terdapat pada integrasi *Simple Network Management Protocol (SNMP)* ke logika program otomasi multi vendor. Hal tersebut bertujuan agar setiap sensor yang terbaca pada SNMP dapat digunakan sebagai trigger untuk menjalankan script automation[2]. Pada penelitian sebelumnya python diimplementasikan pada dunia jaringan karena terdapat beberapa library yang mendukung untuk otomasi jaringan seperti paramiko dan netmiko[3] untuk melakukan konfigurasi terhadap perangkat jaringan multivendor dengan menggunakan protokol SSH dan terbukti dapat berjalan dengan baik serta dapat membantu untuk konfigurasi banyak perangkat sekaligus dengan lebih mudah[4].

Saat ini, terdapat beberapa *automation tool* yang banyak digunakan, salah satunya yaitu *ansible tower*. *Ansible tower* tidak memiliki pemantauan untuk menjalankan otomasi sehingga otomasi yang berjalan hanya berdasarkan waktu. Permasalahan tersebut akan teratasi dengan website yang dibuat, dimana otomasi dapat dijalankan berdasarkan kondisi yang terjadi pada perangkat jaringan secara *real-time.*

II. KAJIAN TEORI

A. *Ansible Tower* dengan SNMP

Secara umum pengoprasian Ansible Tower dan Ansible Playbook memiliki tahapan yang sama, Ansible Tower adalah versi GUI *(Graphical user interface)* dari *Ansible Playbook* sehingga ketika ingin menghubungkan Ansible Tower dengan SNMP maka digunakan Ansible Playbook untuk melakukan koneksi ke SNMP, *Ansible Playbook* juga menggunakan SSH sebagai koneksinya ke SNMP hal ini membuat *Ansible Playbook* dapat digunakan pada perangkat Multivendor.

B. *API (Application Programming Interface)* dengan SNMP

API dapat diintegrasikan dengan SNMP melalui pembuatan *backend* untuk interkoneksi dengan perangkat Jaringan, pembuatan backend API dengan setiap vendor dapat dicapai melalui pembuatan backend eksklusif untuk setiap vendor, hal ini dikarenakan setiap vendor memiliki aturan yang berbeda untuk dapat terkoneksi dengan API perangkatnya.

C. SSH dengan SNMP

SSH (*Secure Shell*) adalah protokol yang digunakan untuk mengamankan komunikasi dalam jaringan. Sedangkan SNMP adalah untuk memantau dan mengelola perangkat jaringan seperti router, switch, dan server. Paramiko adalah sebuah library Python yang menyediakan implementasi dari protokol SSH untuk berkomunikasi dengan perangkat jarak jauh secara aman. Modul ini memungkinkan pengembang Python membuat koneksi SSH, mengirim perintah ke perangkat jarak jauh, dan mentransfer file melalui protokol SSH. Fitur utama yaitu

membuat template automasi pada website untuk seluruh atau sebagian perangkat jika terdapat kondisi tertentu, fitur ini dapat menjawab masalah kompatibilitas antar vendor dalam menciptakan sebuah perintah automasi.

III. METODE

Dalam perancangan backend pada sistem otomasi konfigurasi dan *monitoring* perangkat jaringan multivendor pada arsitektur *microservic*e ini, menggunakan beberapa metode antara lain:

A. Kajian Literatur

Pada metode kajian literatur, penulis mempelajari jurnal-jurnal yang relevan dengan topik yang penulis ambil yaitu sistem otomasi konfigurasi dan monitoring perangkat jaringan multivendor pada arsitektur microservice. Jurnal-jurnal tersebut dapat menjadi referensi dalam perancangan *backend* pada sistem ini. Jurnal tersebut juga membantu jika ada ketidakpahaman dalam pengerjaan *backend* pada sistem ini.

B. Perancangan Sistem

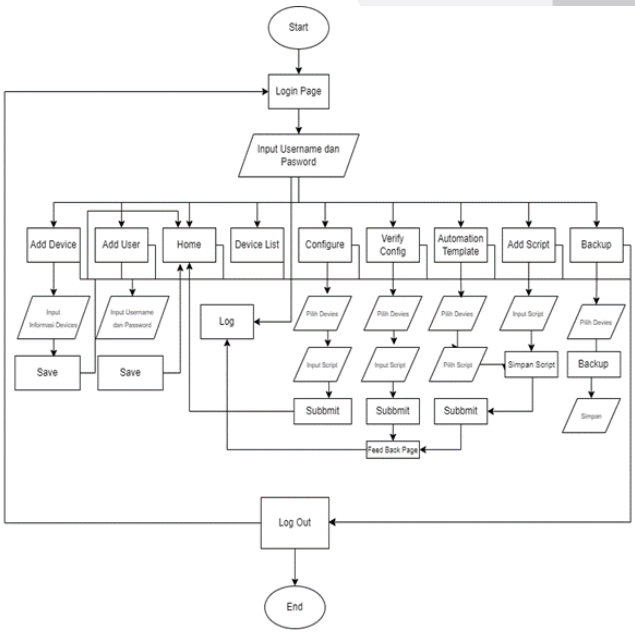
Pada penelitian ini, desain sistem merupakan metode yang penting untuk dilakukan. Hal tersebut dikarenakan desain sistem akan menjadi acuan dalam mengembangkan solusi yang efektif selama pengerjaan. Proses dalam desain sistem meliputi:

1. Identifikasi Kebutuhan

Pada tahap ini, kebutuhan yang digunakan dalam melakukan penelitian yaitu *Visual Studio Code* yang berfungsi sebagai media dalam perancangan *code*.

2. Perancangan Konsep

Dalam tahap ini, hal yang dilakukan yaitu membuat *flowchart* yang bertujuan untuk menggambarkan keseluruhan sistem yang akan dirancang. *Flowchart* ini akan mempermudah dalam memahami alur kerja dari sistem. Berikut merupakan *flowchart* dari sistem otomasi konfigurasi dan monitoring perangkat jaringan multivendor pada arsitektur *microservice.*



GAMBAR 1
Flowchart Sistem

Gambar 1 menjelaskan alur kerja sebuah aplikasi manajemen perangkat dan otomatisasi konfigurasi. Proses dimulai dari halaman *login*, di mana pengguna harus memasukkan nama pengguna dan kata sandi untuk mengakses aplikasi. Setelah berhasil login, pengguna dapat mengakses berbagai fitur utama aplikasi seperti menambahkan perangkat baru melalui fitur *Add Device*, menambahkan pengguna baru dengan *Add User*, kembali ke halaman utama menggunakan Home, melihat daftar perangkat melalui *Device List*, serta mengkonfigurasi perangkat tertentu menggunakan Configure. Selain itu, pengguna dapat memverifikasi konfigurasi perangkat melalui *Verify Config*, mengelola *template* otomatisasi dengan *Automation Template*, menambahkan *script* baru menggunakan *Add Script*, dan melakukan *backup* data perangkat melalui fitur *Backup*.

C. Implementasi Sistem

Pada implementasi sistem, terdapat satu bahasa pemograman serta satu *framework* yang digunakan yaitu:

1. Bahasa Pemrograman *Python*

Python merupakan bahasa pemrograman yang mudah untuk dipahami. *Python* memiliki beberapa *library* yang mendukung untuk otomasi jaringan seperti *netmiko* dan *paramiko* yang dapat dimanfaatkan dalam perencanaan sistem otomasi konfigurasi dan monitoring perangkat jaringan multivendor pada arsitektur *microservice*. Perangkat jaringan multivendor dapat dilakukan konfigurasi dengan satu kali konfigurasi saja menggunakan bahasa pemrograman python [5]. Dokumentasi penggunaan bahasa pemrograman *python* dapat diakses dengan mudah, sehingga dalam proses pembuatan *backend*, *python* menjadi salah satu pilihan yang biasanya digunakan oleh *network engineer* dalam melakukan monitoring secara *real-time* berbasis jaringan. Dalam penggunaan python, semua perangkat jaringan dapat dikonfigurasikan menggunakan *SSH protocol*[6].

2. *Framework Django*

Django merupakan *framework* yang tersedia pada bahasa pemrograman *python*[7]. *Django* digunakan untuk membantu perancangan web secara cepat dan efisien[8]. *Framework django* memiliki kemiripan dengan arsitektur *microservice* dan juga dapat digunakan dengan mudah pada arsitektur *microservice*[9]. Pada perancangan sistem otomasi konfigurasi dan *monitoring* perangkat jaringan *multivendor* pada arsitektur *microservice*, *framework django* akan menjadi tempat dimana *database* akan disimpan.

IV. HASIL DAN PEMBAHASAN

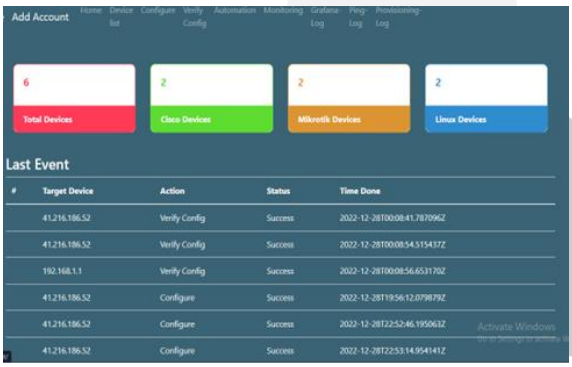
Setelah melalui tahap perancangan, berikut merupakan hasil dari implementasi perancangan backend pada sistem otomasi konfigurasi dan monitoring perangkat jaringan multivendor pada arsitektur *microservice.* Adapun *website* yang dihasilkan memiliki fitur yang dapat melakukan otomasi konfigurasi serta memonitoring perangkat multivendor.

A. *Login* dan *Logout* View

GAMBAR 2
Login dan logout page

Pada Gambar 2 fitur *login* dan *logout*, mengimplementasikan metode get(), yang akan dipanggil ketika ada permintaan HTTP GET ke endpoint yang terkait dengan *view* ini. Dalam metode get(), pengguna akan keluar dari website menggunakan fungsi logout(request) untuk menghapus sesi pengguna yang terkait dengan permintaan, dan kemudian pengguna akan diarahkan kembali ke halaman *login* menggunakan fungsi redirect('login'). Pada fitur *login*, *user* diminta untuk memasukan *usermane* dan *passsword*. Lalu setelah berhasil login maka *user* akan dibawa ke halaman berikutnya. Jika user sudah selesai menggunakan *website*, *user* dapat memilih *logout* dan *user* akan dibawa ke halaman awal.

B. *Home View*

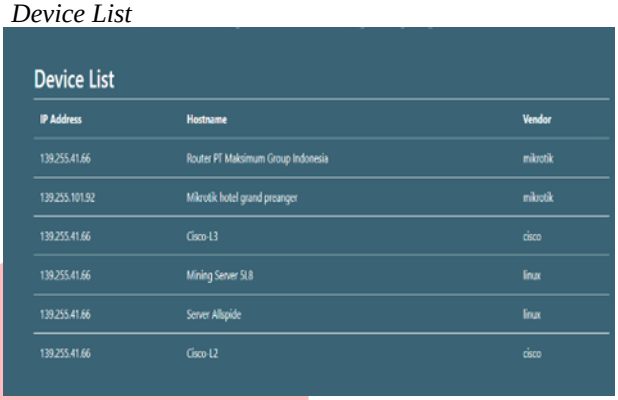


GAMBAR 3
Home page

Pada Gambar 3, kelas HomeDataView mewarisi fungsi dari APIView dan mengimplementasikan metode get(), yang akan dipanggil saat ada permintaan HTTP GET ke endpoint yang terkait dengan *view* ini. Dalam metode get(), objek HomePageDataProviderImpl dibuat untuk mengambil data dari sistem. Data tersebut kemudian diambil menggunakan beberapa metode dari objek data_provider, seperti get_all_device_count(), get_cisco_device_count(), dan sebagainya. Data ini dikelompokkan dalam sebuah *dictionary* yang disebut context. Akhirnya, *context* dikirim sebagai *respons* dalam format JSON menggunakan objek *Response*. JSON merupakan format yang populer untuk bertukar data adalam proses pembuatan web[10]

Pada menu *home* akan terdapat tampilan *website*, dimana *user* dapat melihat log dari aktivitas terakhir yang user lakukan di *website*. Log tersebut berisikan target *device*, *actions*, *status*, dan *time done*

C. *Device List*



GAMBAR 4
Device list page

Pada Gambar 4, kelas DeviceListView mewarisi fungsi dari APIView dan mengimplementasikan metode get(),yang dipanggil ketika ada permintaan HTTP GET ke *endpoint* yang terkait dengan *view* ini. Dalam metode get(), semua objek *Device* diambil dari *database* menggunakan Device.objects.all(). Objek Device kemudian disterilsasikan menggunakan serializer DeviceDerializer dengan parameter many=True untuk menangani banyak objek. Hasil tersebut kemudian dikirim sebagai *respons* dalam format JSON menggunakan objek *Response* Menu *device list* dikatakan berhasil jika *user* sudah dapat memasukan lebih dari satu perangkat jaringan dengan vendor yang berbeda. Adapun menu ini menampilkan *ip address*, *hostname*, serta nama vendor yang sudah dimasukkan.

D. *Configure View*

GAMBAR 5
Configure page

Pada Gambar 5, kelas ConfigureView mewarisi fungsi dari APIView dan mengimplementasikan dua metode: get() dan post(). Metode get() akan dipanggil saat ada permintaan HTTP GET ke endpoint yang terkait dengan view ini, dan akan merender halaman HTML config.html dengan data yang dikumpulkan, seperti daftar perangkat yang telah diserialisasikan menggunakan serializer DeviceSerializer dan mode 'Configure'. Metode post() akan dipanggil saat ada permintaan HTTP POST ke *endpoint* yang sama, dan akan melakukan konfigurasi perangkat yang dipilih berdasarkan *input* dari pengguna, seperti perintah MikroTik, Cisco, dan Linux yang diambil dari form HTML. Setelah konfigurasi dilakukan, pengguna akan diarahkan kembali ke halaman beranda.

Pengujian pada menu *configure*, dilakukan dengan mengkonfigurasi sepuluh perangkat jaringan dari vendor yang berdeda. Hasil pengujian menunjukkan rata-rata waktu yang dibutuhkan untuk melakukan *configure* yaitu 7,085 detik.

E. *Verify Config View*

GAMBAR 6
Config View

Pada gambar 6, kelas VerifyConfigView mewarisi fungsi dari APIView dan mengimplementasikan dua metode: get() dan post(). Metode get() akan dipanggil saat ada permintaan HTTP GET ke endpoint yang terkait dengan view ini, dan akan merender halaman HTML config.html dengan data yang dikumpulkan, seperti daftar perangkat yang telah diserialisasikan menggunakan *serializer Device Serializer* dan mode 'Verify Config'. Metode post() akan dipanggil saat ada permintaan HTTP POST ke endpoint yang sama, dan akan melakukan verifikasi konfigurasi perangkat yang dipilih berdasarkan input dari pengguna, seperti perintah MikroTik, Cisco, dan Linux yang diambil dari form HTML. Setelah verifikasi dilakukan, hasilnya akan di render dalam halaman HTML verify_result.html. Jika terjadi kesalahan, log akan dicatat dalam database dengan status error.

Pengujian pada menu *verify config*, dilakukan dengan melakukan verifikasi mengkonfigurasi sepuluh perangkat jaringan dari vendor yang berbeda-beda. Hasil pengujian menunjukkan rata-rata waktu yang dibutuhkan yaitu 8.68 detik.

V. KESIMPULAN

Pada penelitian ini, telah berhasil membuat kode perancangan backend pada sistem otomasi konfigurasi dan monitoring perangkat jaringan multivendor pada arsitektur microservice. Adapun fungsi yang dapat dilakukan menggunakan website ini yaitu otomasi konfigurasi serta memonitoring perangkat multivendor. Terdapat lima kelas yang berhasil dibuat yaitu *Login* dan *Logout, home, Device List, Configure*, dan *Verify Config.*

Adapun pengujian yang dilakukan yaitu melakukan login dan logout, melakukan configure dan verify config pada perangkat jaringan dengan vendor yang berbeda-beda pula. Hal tersebut mengidentifikasikan bahwa perancangan backend pada sistem otomasi konfigurasi dan monitoring perangkat jaringan multivendor pada arsitektur microservice berjalan sesuai rancangan yang sudah dibuat.

REFERENSI

[1] T. Muhammad and M. Tahir Munir , "Network Automation," *European Journal of Technology* , vol. 7, no. 3, 2023.

[2] E. d. Costa and S. Mesquita , "Computer Network Management and Monitoring System With SNMP and QoS," *Timor-Leste Journal of Engineering and Science* , vol. 3, no. 1, 2022.

[3] B. Choi, Introduction to Python Network Automation, Sydney: Apress, 2021.

[4] S. Nugroho and B. Pujiarto, "Network Automation pada Beberapa Perangkat Router," *Jurnal Teknologi Informasi dan Ilmu Komputer (JTIIK)*, vol. 9, no. 1, 2022.

[5] M. Mashudi, "Network Autonation Menggunakan Bahasa Pemograman Python," *Jurnal Teknik industri, Sistem informasi dan Teknik informatika*, vol. 1, no. 2, 2022.

[6] G. Milios, "Network Automation," *School of Science & Technology, Master of Science (MSc) in Cybersecurity*, vol. SID: 3307190015, 2021.

[7] N. Bakshi, P. Kumar, A. Rastogi and D. Surya, "Spring Framework vs Django Framework: A Comparative Study," *International Research Journal of Engineering and Technology (IRJET)* , vol. 7, no. 6, 2020.

[8] Y. C. E. Paksi and I. R. Widiasari, "Website Network Automation Design and Implementation in RT RW NET Senden Dusun Magelang with Django Framework," *Jurnal Teknik Informatika (JUTIF)*, vol. 3, no. 5, 2022.

[9] M. Oksa, "Web API Development and Intregation for Microservice Functioality in Web," *UNIVERSITY OF JYVÄSKYLÄ*, vol. 7, 2016.

[10] P. Bourhis, J. L. Reutter and D. Vrgoč, "JSON : Data model and query languages," *Information Systems*, 2019.