Sistem Automasi *Zero Touch Provisioning Routerboard* Berbasis GraphQL API

Leonardus Daniel Krisnayuda P.*[1], Eugenius Kau Suni[2]
Program Studi Teknik Informatika, Fakultas Ilmu Komputer, Universitas Mercu Buana
e-mail: [1]41518120074@student.mercubuana.ac.id , [2]eugenius@mercubuana.ac.id
*Penulis Korespondensi

Diterima: 27 Juni 2023; Direvisi: 5 Oktober 2023; Disetujui: 9 Oktober 2023

Abstrak

Sistem perangkat jaringan modern identik dengan adanya pemisahan antara control plane dengan data plane. Dalam sistem perangkat jaringan modern, idealnya sebuah perangkat jaringan memiliki manajemen sistem yang terpusat. Sistem manajemen terpusat yang baik memiliki kemampuan tidak sekedar monitoring, melainkan juga provisioning dan kegiatan administratif lain. Permasalahan yang dihadapi yakni perangkat jaringan routerboard tidak mampu melakukan provisioning secara mandiri dan otomatis. Untuk mengatasi masalah tersebut, perlu adanya sebuah sistem manajemen modern yang mampu melakukan provisioning secara otomatis, monitoring, dan juga administrasi konfigurasi secara massal. Dengan melakukan observasi dan studi literatur diketahui penggunaan routerboard dapat dimaksimalkan dengan fitur command line berupa HTTP request dan pemanfaatan arsitektur API. Penelitian ini mampu membuat sistem automasi zero touch provisioning routerboard, dengan menggunakan framework PHP Laravel dan GraphQL API. Hasil penelitian membuktikan bahwa sistem automasi mampu melakukan zero touch provisioning dengan memanfaatkan command line HTTP request untuk routerboard berkomunikasi ke server controller. Sedangkan untuk administrasi konfigurasi antara server controller ke routerboard, dilakukan melalui proses pemanggilan API. Pengujian yang dilakukan menunjukkan bahwa rata-rata durasi yang dibutuhkan dalam proses zero touch provisioning routerboard yakni 8.851ms.

Kata kunci: *network automation, routerboard, graphQL, automasi, python*

Abstract

Modern network device systems are identical with the separation between control plane and data plane. In a modern network device system, ideally a network device has a centralized management system. A good centralized management system has the capability not only for monitoring, but also for provisioning and other administrative activities. The problem faced is that routerboard network devices are unable to provision independently and automatically. To overcome this problem, it is necessary to have a modern management system capable of automatic provisioning, monitoring, and administration of configurations in bulk. By observing and studying the literature, it is known that the use of the routerboard can be maximized with the command line features in the form of HTTP requests and the use of the API architecture. This research was able to create a zero-touch provisioning routerboard automation system, using the Laravel PHP framework and the GraphQL API. The results of the study prove that the automation system is capable of zero touch provisioning by utilizing the HTTP request command line for the routerboard to communicate with the server controller. As for configuration administration between the server controller and the routerboard, it is carried out through the API calling process. The tests carried out show that the average duration required in the routerboard zero touch provisioning process is 8.851ms.

***Keywords**: network automation, routerboard, graphQL, automation, python*

1. PENDAHULUAN

Tren perkembangan teknologi informasi pada khususnya teknologi jaringan informasi sangatlah masif dalam satu dekade ke belakang. Hal tersebut seiring pula peningkatan variasi perangkat jaringan yang aktif beroperasi. Banyak pengembang perangkat keras menciptakan perangkat *router* jaringan yang kaya akan fitur. Fitur – fitur yang masif dikembangkan dalam satu dekade ke belakang diantaranya *zero touch provisioning, Software Defined Networking (SDN), Software Defined Wide Area Network (SD-WAN),* dan *Programmability Network*. Jika diteliti, arah pengembangan jaringan semakin hari semakin mendekatkan pada konsep jaringan yang dapat diprogram dan berjalan di atas sebuah sistem perangkat lunak terotomasi.

Dengan banyaknya perangkat yang aktif beroperasi, tentunya beban kerja seorang administrator jaringan semakin kompleks. Guna membantu kerja administrator jaringan, perlu adanya sistem monitoring yang tepat dan mampu meringankan pekerjaan administrator sehari – hari. Dari pengamatan praktek secara umum, sistem monitoring mengandalkan protokol *Simple Network Management Protocol* (SNMP). Mekanisme SNMP berupa *polling* berkala dari SNMP *Manager* ke seluruh SNMP *Agent*. Data hasil *polling* kemudian ditampilkan dalam bentuk grafik pada aplikasi web.

Salah satu *router* jaringan yang beredar di pasaran adalah *Routerboard* Mikrotik. Routerboard Mikrotik merupakan salah satu vendor perangkat keras dan perangkat lunak yang salah satu fungsinya sebagai *router* [1]. Menurut tren pengembangan perangkat jaringan di Indonesia, dalam kurun waktu 5 tahun terakhir riset pengembangan terkait Mikrotik mengalami kenaikan yang signifikan, seiring dengan jumlah komunitasnya. Hal ini membuktikan bahwa Routerboard Mikrotik memiliki peminat besar di Indonesia. Selain itu, berdasarkan survei yang dilakukan oleh tim Mikrotik melalui acara Mikrotik User Meeting (MuM) Indonesia di 2018, Mikrotik User Meeting Indonesia memecahkan rekor sebagai Mikrotik User Meeting dengan partisipan terbanyak disepanjang acara Mikrotik User Meeting yang pernah diadakan oleh tim Mikrotik pada beberapa tempat di dunia sejak 2006 [2].

Dari informasi dan survei yang telah dilakukan, disimpulkan bahwa pengguna Routerboard Mikrotik di Indonesia sangat banyak, mulai dari kalangan penyedia layanan Internet, instansi pemerintah, perusahaan swasta, usaha mikro, dan juga instansi pendidikan. Sebagian telah mengimplementasikan Routerboard dengan sistem *monitoring*. Sebagian sisanya perangkat dibiarkan operasional tanpa dimonitor. Perlu adanya perubahan mekanisme dalam sistem *monitoring*, dimana tidak sekedar *monitoring*, melainkan juga mampu mengautomasikan konfigurasi. Tugas berupa konfigurasi perangkat dalam skala puluhan, ratusan, hingga ribuan perangkat, tentunya memerlukan waktu yang panjang, memicu adanya *human-error*, dan berdampak pada performa dan *output* yang dihasilkan [3].

Berdasarkan uraian permasalahan di atas, penelitian ini ditujukan untuk mengembangkan sebuah sistem automasi jaringan berbasis web untuk memfasilitasi fitur *provisioning* secara otomatis. Beberapa penelitian sebelumnya mengenai pemanfaatan teknologi informasi untuk membantu manajemen jaringan seperti dalam penelitian [4] yang menggunakan sistem berbasis web untuk mengautomasikan perangkat Cisco CSR1000V. Aplikasi As-RaD mampu berkomunikasi dengan perangkat jaringan komputer arsitektur REST-API. Hasil dari pengujian automasi jaringan menggunakan As-RaD menunjukkan hasil performa lebih cepat 75% dibanding dengan metode Paramiko, dan 92% lebih cepat dari metode NAPALM dalam proses manajemen jaringan. Penelitian [5] diketahui bahwa aplikasi *mobile* dapat digunakan untuk mengakses *router mikrotik* dengan menggunakan informasi *login* yang tersimpan pada mikrotik. Selain itu, aplikasi *mobile* berhasil untuk mengatur *bandwidth* untuk *user* yang terhubung ke dalam jaringan. Aplikasi *mobile* juga dapat digunakan untuk membuat tabel *routing*, *monitoring traffic*, menampilkan *resource* dari mikrotik serta me-monitoring jaringan menggunakan *ping*.

Penelitian [6] menunjukkan bahwa metode Scrum dapat digunakan untuk membuat sistem manajemen yang dapat diakses dalam skala besar. Sistem aplikasi ini juga dapat digunakan oleh perusahaan ataupun pemilik toko atau pebisnis untuk membantu mereka dalam mencari jasa seorang *content creator* guna membantu mereka dalam membuat konten yang menarik. Penelitian [7] disimpulkan terdapat perbedaan penggunaan GraphQL dengan RESTful API dalam hal penciptaannya. GraphQL terdiri dari tiga komponen yakni Mutation, Query, dan Type sebelum mengakses data menggunakan Data Manipulation Language (DML), sedangkan RESTful API berkomunikasi langsung menggunakan DML. Berdasar hasil penelitian, RESTful terbukti lebih responsif dalam hal penarikan data, sedangkan GraphQL lebih dinamis karena *response* tergantung pada *request*.

Penelitian [8] diketahui rata-rata respons data pada REST-API berbasis GraphQL dalam *framework* Express disimpulkan berperforma lebih baik daripada *native* GraphQL dalam hal menangani *request*. Kemudian GraphQL unggul dalam proses menampilkan data ke sistem aplikasi, dimana sistem tersebut mampu menentukan atribut *custom* sehingga tidak ada data yang tidak terpakai. Hal ini mengoptimalkan besarnya *bandwidth* yang dipakai.

Dari kelima penelitian di atas, disimpulkan bahwa penelitian mengenai sistem automasi jaringan sangat bergantung dengan protokol komunikasi yang digunakan. Ada yang menggunakan protokol REST-API di sisi *controller*, ada juga yang memanfaatkan fitur API yang tertanam pada perangkat jaringan. Selain itu, penelitian mengenai protokol GraphQL dalam konteks proses automasi masih jarang digunakan. Penelitian ini didukung juga oleh bukti konkret bahwa perangkat Routerboard MikroTik tidak memiliki mekanisme *built-in* berupa automasi *provisioning*. Semua konfigurasi perangkat Routerboard masih perlu peran *user* / administrator. Maka perlu aplikasi yang mampu melakukan *provisioning* secara otomatis, tanpa peran *user* / administrator untuk konfigurasi, dan juga langsung terintegrasi dengan sistem monitoring. Inilah yang menjadi gagasan utama peneliti memilih untuk membuat sistem *Zero Touch Provisioning* perangkat routerboard MikroTik.

Penelitian ini berupa rancangan dan implementasi sistem automasi jaringan yang menggunakan protokol GraphQL. Dalam sistem automasi yang telah dirancang, terdapat tiga fitur, meliputi (1) mampu melakukan eksekusi *provisioning* secara otomatis, (2) memberikan informasi mengenai perangkat yang sinkron dengan sistem automasi, (3) memberikan kemampuan *push config* secara massal dan terdistribusi. Dengan fitur yang telah dirancang, proses administrasi dan *monitoring* jaringan dapat lebih komprehensif dan administrator jaringan merasa percaya menggunakannya.

2. METODE PENELITIAN

Proses penelitian menggunakan metode *waterfall* untuk merancang dan mengimplementasikan sistem automasi. Di dalam metode ini, setiap proses yang dilakukan saling berkaitan satu sama lain. Sebuah tahapan harus tercapai sebelum berlanjut ke tahapan berikutnya. Tahapan penelitian secara detail dijelaskan sebagai berikut :

1. Observasi dan Studi Literatur
   Proses observasi pada sebuah penelitian pada prinsipnya dibangun dengan mengumpulkan data informasi dari sumber internal berupa data operasional, dan juga sumber informasi eksternal seperti studi literatur [9]. Observasi jaringan dilakukan dengan mengamati kondisi jaringan Routerboard yang sudah operasional. Studi literatur ditujukan untuk memahami teori yang mendukung sebuah sistem sebelum mengambil keputusan yang dicapai [10], dalam hal ini berupa automasi jaringan.
2. Perancangan Sistem
   Proses perancangan sistem mengacu pada hasil observasi yang dilakukan. Untuk membuat sebuah sistem, perlu pemahaman dalam hal konfigurasi routerboard beserta tekniknya dalam meningkatkan efektivitas dan efisiensi operasional [11]. Pada tahapan ini proses yang dilakukan berupa desain rancangan *backend* automasi jaringan beserta

script pendukungnya.

3. Implementasi
   Proses implementasi layanan pendukung sistem automasi jaringan yang dirancang mulai dilakukan. Konsep *network automation* memiliki persamaan dengan konsep *Software Defined Network* (SDN). Kedua konsep tersebut mampu terintegrasi dengan arsitektur jaringan secara multi-layer, multi-vendor, dan bermacam koneksi jaringan yang memuat aplikasi *user* secara heterogen dan terotomasi [12]. Kunci dari keberhasilan automasi jaringan yakni ekosistem sentral guna mengeliminasi operasional perangkat yang bersifat *hardware-centric* [13]. Ekosistem pada tahapan implementasi perlu adanya perantara untuk proses komunikasi menggunakan API. GraphQL merupakan pemrograman *query* pada arsitektur API untuk memproses data berdasarkan tipe yang dikonfigurasikan pada server. GraphQL API bertindak sebagai perantara data dengan aplikasi [14]. Proses *query* data pada GraphQL dibentuk dari variabel dan field yang diatur dalam Schema pada sisi server [7]. Penelitian terhadap API di tahun 2019, menguji sampel sebanyak 104 penelitian yang dikelompokkan berdasar metodologinya. Sebanyak 63 jurnal penelitian menggunakan metodologi eksperimen, 21 jurnal penelitian tanpa metodologi, 15 penelitian secara kualitatif, 3 penelitian kuantitatif, dan 2 penelitian menggunakan metodologi mix [15]. Proses yang dilakukan sistem automasi yakni melakukan proses API-call ke perangkat routerboard dengan maksud melakukan *zero-touch provisioning*. Setelah sampel perangkat berhasil *provisioning* langkah berikutnya perangkat mengirim data yang mengindikasikan selesai *provisioning*, disalurkan melalui protokol GraphQL API.
4. Pengujian
   Proses pengujian sistem dilakukan secara *black-box* untuk menguji keberhasilan fungsi *zero touch provisioning*. Proses pengujian dilakukan dalam skala puluhan perangkat untuk mencari atribut terbaik dari setiap hasil pengujian.
5. Analisa
   Pada tahapan analisa, hal yang dilakukan yakni adalah inspeksi terhadap hasil keluaran setiap *proses zero touch provisioning*.
6. Kesimpulan
   Proses penarikan kesimpulan dilakukan setelah semua tahapan berhasil ditempuh, mulai dari observasi hingga analisa pengujian.

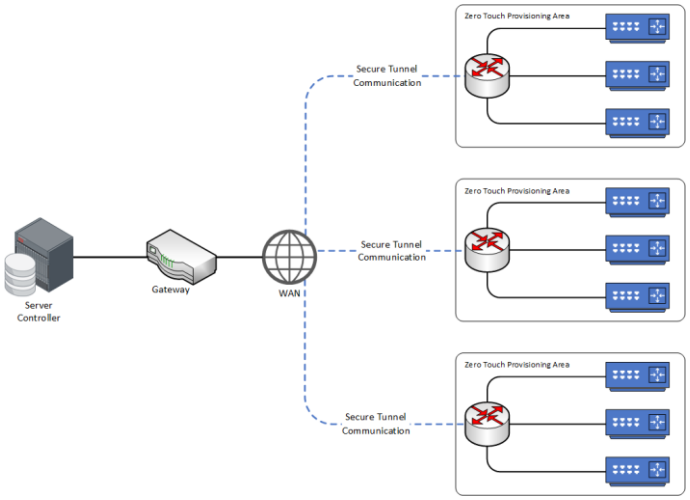
3. HASIL DAN PEMBAHASAN

Dari tahapan penelitian yang dilakukan, diketahui bahwa sistem monitoring dan manajemen Routerboard masih manual secara *Graphical User Interface*. Per tahun 2023, belum ada mekanisme automasi yang *built-in* di dalam perangkat Routerboard. Mengingat jaringan yang sudah ada terpisah jauh secara geografis, maka rumusan topologi sistem automasi dapat dilihat pada Gambar 1.

Pada Gambar 1 terlihat bahwa setiap master *router* didesain memiliki zona *provisioning* masing – masing sesuai dengan konfigurasi. Selain itu, proses *zero touch provisioning* mewajibkan master *router* terhubung secara *tunnel* ke *controller* melalui WAN Internet. Kebutuhan utama dalam pembuatan sistem automasi adalah proses untuk menjembatani komunikasi antara *router* yang belum terprovisioning dengan server, melalui protokol GraphQL API. Untuk mengakses server *controller*, jalur yang ditempuh master *router* mengharuskan via Gateway. Gateway yang dirancang menggunakan Routerboard MikroTik, sedangkan server *controller* menggunakan sistem operasi Linux.

Mengacu pada topologi yang dirancang, maka spesifikasi *hardware* yang digunakan dalam proses penelitian ini dapat dilihat dalam Tabel 1. Sedangkan secara *software*, sistem automasi ini dibangun menggunakan *framework* bahasa pemrograman PHP Laravel yang di dalamnya terdapat *dependencies* seperti GraphQL, *asynchrounous processor*, dan paket

pelengkap lain yang terkait.



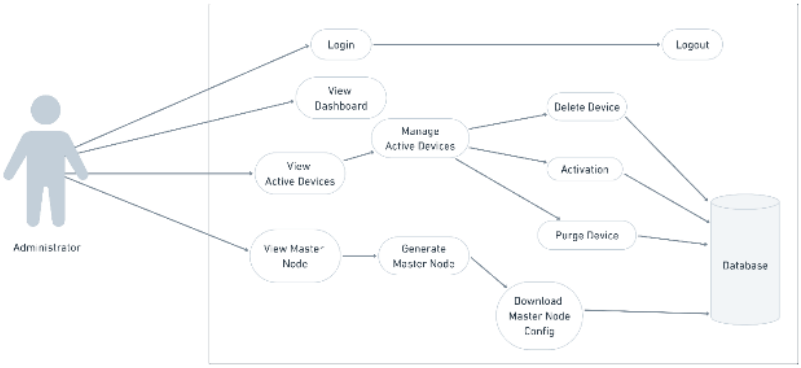
Gambar 1. Topologi Sistem

Tabel 1. Spesifikasi Sistem

Spesifikasi	Server Controller	Gateway
RAM	4 GB	256 Mb
CPU	4 Core	1 Core
HDD	40 GB	1 GB
Port	1 Gigabit Ethernet	5 Gigabit Ethernet

Tabel 1 menunjukkan kebutuhan dan spesifikasi sistem yang diperlukan berdasarkan topologi jaringan yang telah dirancang. Kebutuhan sistem untuk *hardware* meliputi RAM, CPU, HDD dan Port.

Rancangan sistem yang ditampilkan dalam penelitian ini adalah *use case* dan *activity diagram* yang terkait dengan fitur *zero touch provisioning*. *Use case* terdapat dalam gambar 2, sedangkan *activity diagram* dapat dilihat dalam gambar 3 dan gambar 4.



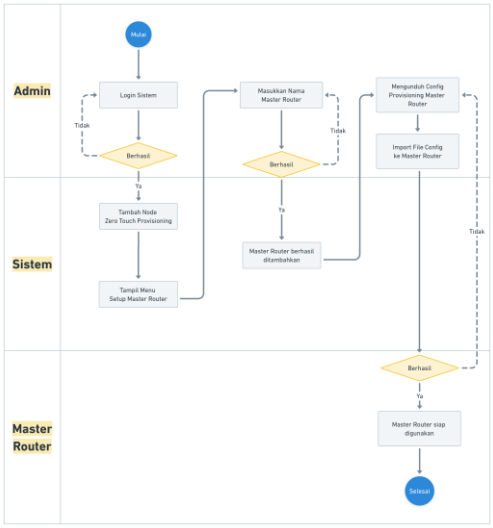
Gambar 2. Use Case Diagram

Use case diagram dalam gambar 2 menjelaskan rancangan fungsi yang ada pada sistem automasi. Administrator mampu melakukan pengelolaan seperti login, melihat dashboard, melakukan list perangkat aktif, list master router, download konfigurasi master router dan proses *zero touch provisioning* yang terkait.

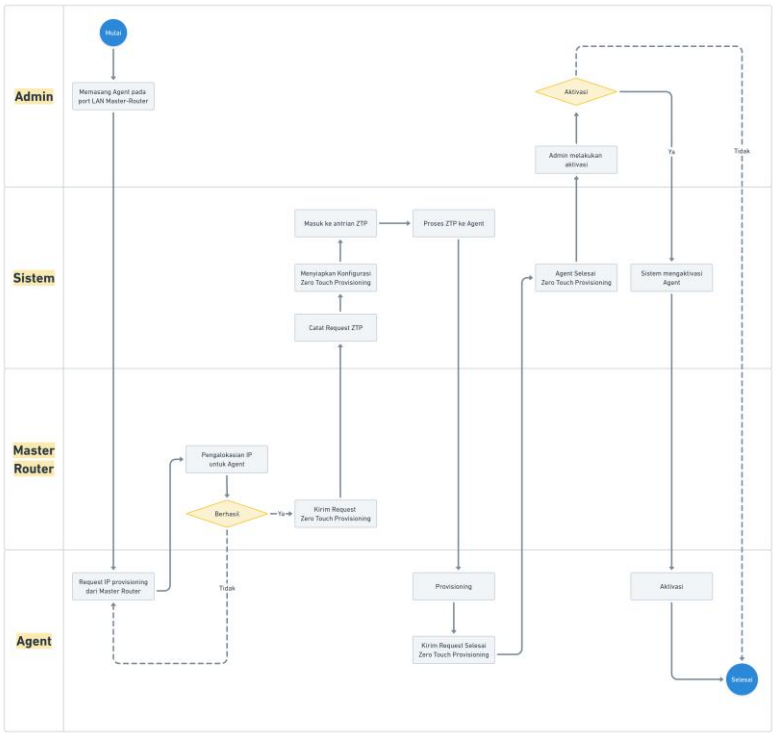
Activity diagram pada gambar 3 menunjukkan proses *provisioning* master router. Proses *provisioning* master *router* dimulai dari administrator *login* ke sistem dan melakukan *generate* konfigurasi sesuai kebutuhan. Setelah *script* berhasil *generate*, administrator perlu mengunduh

file tersebut dan melakukan *import* pada router yang masih kosong untuk dijadikan master router. Master router yang melayani proses ZTP untuk calon perangkat yang akan dilakukan *provisioning*.

Activity diagram pada gambar 4 menjelaskan proses *zero touch provisioning*. Proses *zero touch provisioning* dilakukan pada *router* yang masih kosong untuk dijadikan router *agent*. Proses dimulai dalam kondisi master *router* yang sinkron dengan sistem automasi dan sudah siap melayani *router* yang akan dijadikan *agent*. Administrator memasang *agent* ke master *router* secara fisikal. Kemudian, master *router* akan menjalankan serangkaian automasi yang di dalamnya berisi *template* konfigurasi untuk diimplementasi pada *agent*. Setelah proses *zero touch provisioning* selesai, sistem akan mengirim notifikasi. Selain itu muncul *agent* pada *dashboard list* perangkat baru, yang siap dilakukan aktivasi.



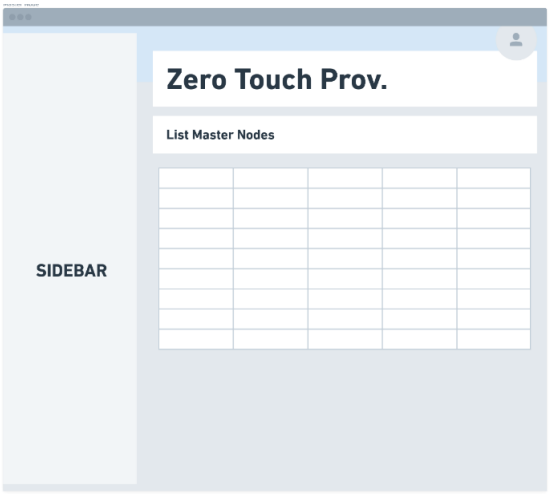
Gambar 3. Activity Diagram Provisioning Master Router



Gambar 4. Activity Diagram Zero Touch Provisioning

Gambaran *prototype* tampilan tampilan daftar perangkat yang aktif dapat dilihat dalam gambar 5. Dalam gambar 5, pada bagian tabel daftar *active devices*, berisi informasi mengenai *agent* yang berhasil sinkron ke sistem. Informasi agent yang ditampilkan meliputi *hostname*, *ip address*, *serial number*, dan juga status *last seen.*

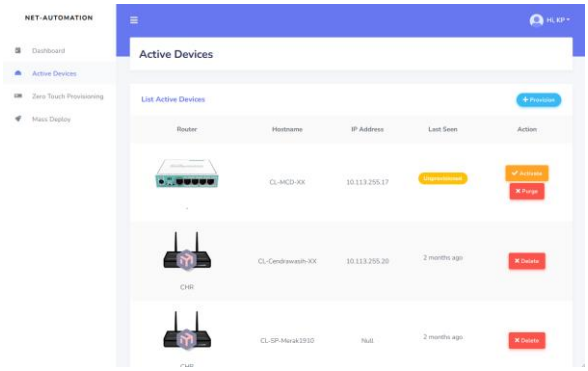
Gambar 5. *Prototype* Daftar Perangkat Aktif



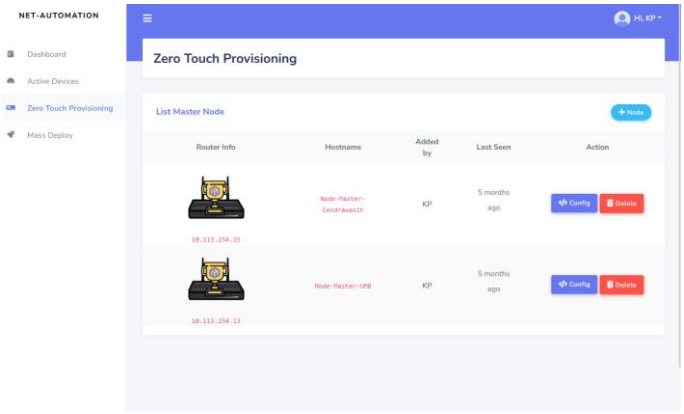
Gambar 6. *Prototype* Daftar Master Router ZTP

Gambaran *prototype* tampilan daftar master router untuk keperluan *Zero Touch Provisioning* (ZTP) dapat dilihat dalam gambar 6. Dalam gambar 6, pada bagian tabel daftar *master nodes*, berisi informasi mengenai master *router* yang berhasil sinkron ke sistem. Informasi yang ditampilkan meliputi *hostname*, *ip address*, administrator pembuat master router, panel *action*, dan juga status *last seen*. Berikut adalah output implementasi dari *prototype* yang telah dirancang.

Hasil realisasi prototype Daftar Perangkat Aktif dan Daftar Master Router ZTP dapat dilihat dalam Gambar 7 dan Gambar 8. Gambar 7 menunjukkan beberapa perangkat router aktif. Gambar 8 menunjukkan beberapa perangkat router yang berhasil melakukan sinkronisasi ke sistem.



Gambar 7. Implementasi Daftar Perangkat Aktif



Gambar 8. Implementasi Daftar Master Router ZTP

Proses implementasi GraphQL untuk *router* dijelaskan pada gambar 9. Berdasarkan gambar 9, GraphQL yang diimplementasikan memiliki Query dan Mutation yang di dalamnya terdapat *function*. *Function* telah dilakukan *mapping* antara parameter dan tipe data yang sesuai.

Implementasi function GraphQL API untuk master *router* terdapat dalam gambar 10. Gambar 10 menunjukkan fungsi *zero touch provisioning* untuk master *router* dengan menggunakan *function* GraphQL API.

```
type Query {
  routers: [User!]! @field(resolver: "RouterQuery@all")
  router(id: ID!): User @field(resolver: "RouterQuery@find")
  checkUpdate(serial_number: ID): CheckUpdate  @field(resolver: "RouterQuery@updateTask")
}

type Mutation {
  init(
    serial_number: String!,
    hostname: String!,
    mgmt_ip: String,
  ): String @field(resolver: "RouterMutation@initial")

  resource(
    serial_number: String!,
    board_name: String!,
    cpu_usage: String!,
    hostname: String!,
    memory_usage: String!,
    latency: String!,
    mgmt_ip: String,
  ): String @field(resolver: "RouterMutation@updateResource")

  ztpProvisioning(
    master_address: String!,
    client_api_port: String!,
  ): String @field(resolver: "RouterMutation@ztpProvisioning")

}

type CheckUpdate {
  serial_number: String!
  file: String!
}
```

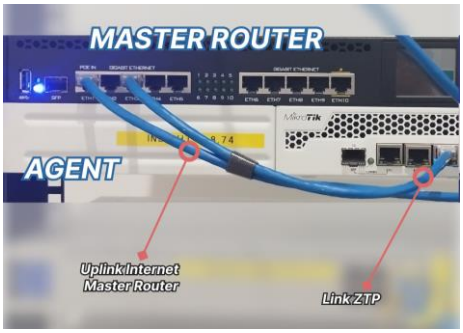
Gambar 9. GraphQL Router

```
type Mutation {
  initMaster(
    serial_number: String!,
    hostname: String!
  ): String @field(resolver: "RouterMutation@initMaster")

  masterResource(
    serial_number: String!,
    board_name: String!,
    cpu_usage: String!,
    memory_usage: String!,
    latency_controller: String!,
    mgmt_ip: String
  ): String @field(resolver: "RouterMutation@updateResourceMaster")
}
```

Gambar 10. GraphQL Master Router

Proses pengujian fungsi *zero touch provisioning* terlihat pada gambar 11. Gambar 11 menunjukkan router yang belum terprovisioning dihubungkan ke port LAN master router.



Gambar 11. Pengujian ZTP

Pengujian fungsionalitas dari sistem automasi yang telah dirancang dilakukan dengan menerapkan metode pengujian blackbox. Hasil pengujian blackbox terdapat dalam tabel 2. Kesimpulan pengujian blackbox dalam tabel 2 menunjukkan hasil hasil yang sesuai dengan skenario yang telah dibuat untuk menguji fungsionalitas sistem.

Tabel 2. Hasil Pengujian Blackbox

No	Nama Pengujian	Output Harapan	Output Sebenarnya	Kesimpulan
1	Login sistem menggunakan kredensial yang tidak terdaftar	Sistem menolak kredensial yang dikirimkan dan muncul tampilan '*Invalid Username / Password*'	Tidak berhasil login dan muncul tampilan '*Invalid Username/Password*'	Sesuai
2	Login sistem menggunakan kredensial yang terdaftar	Sistem mencocokkan data kredensial dengan informasi yang tersimpan di database, dan akan *redirect* ke dashboard	Berhasil login dan *redirect* ke dashboard	Sesuai
3	Provisioning agent dengan kondisi master router tidak terkoneksi ke server controller	Agent selesai provisioning. Namun data agent router pada server controller tidak ditemukan	Agent router selesai provisioning konfigurasi namun belum selesai aktivasi oleh user	Sesuai
4	Setup master router dengan kondisi server controller offline	Data master router tidak tersimpan di sistem dan fungsi zero touch provisioning tidak berhasil	Proses zero touch provisioning tidak berhasil dilakukan terhadap router agent, sehingga seluruh data master router maupun router agent tidak tersimpan dan tidak sinkron dengan server controller	Sesuai
5	*Generate* konfigurasi master router menggunakan hostname master router yang telah disimpan	Sistem melakukan cek data terkait duplikasi master router. Jika diketahui duplikat data, maka proses *generate* konfigurasi master router ditolak dan muncul error "*Duplicate data*"	Data master router tidak ditambahkan dan muncul error "*Duplicate data*"	Sesuai
6	Router agent yang akan dilakukan *zero touch provisioning* dihubungkan pada port WAN / Uplink disisi Master Router	Trigger provisioning tidak terdeteksi oleh master router, sehingga *trigger zero touch provisioning* tidak berhasil dieksekusi	Router agent tidak dilakukan *zero touch provisioning*	Sesuai

No	Nama Pengujian	Output Harapan	Output Sebenarnya	Kesimpulan
7	Router agent yang akan dilakukan *zero touch provisioning* dihubungkan pada port LAN disisi Master Router	Trigger provisioning terdeteksi oleh master router, hal ini memacu server controller untuk *push command zero touch provisioning*	Proses *zero touch provisioning* berhasil dieksekusi, dan router agent siap untuk diaktivasi sebelum digunakan.	Sesuai
8	Trigger provisioning berupa data payload GraphQL dikirimkan oleh master router secara tidak utuh	Sistem melakukan validasi input payload GraphQL yang dikirimkan sebelum melakukan eksekusi *zero touch provisioning*	Proses provisioning berhenti dan tidak berhasil diselesaikan	Sesuai
9	Provisioning dalam kondisi master router tidak aktif	*Pooling* berkala yang dilakukan sistem mendeteksi master router nonaktif, maka sistem menolak update data provisioning dari master router tersebut.	Agent yang terkoneksi secara benar ke master router tidak dapat dilakukan proses *zero touch provisioning*	Sesuai

Hasil durasi provisioning dalam proses ZTP tercantum dalam tabel 3. Tabel 3 menunjukkan durasi waktu mulai dan waktu selesai dalam 10 kali pengujian.

Tabel 3. Durasi Provisioning

Pengujian	Menit Mulai	Menit Selesai	Durasi (ms)
1	00:28:01.116	00:28:10.116	9.001
2	00:29:13.347	00:29:22.331	8.984
3	00:29:27.001	00:29:36.001	9.000
4	00:29:37.823	00:29:46.721	8.898
5	00:45:20.317	00:45:29.259	8.942
6	00:45:43.556	00:45:52.509	8.953
7	00:46:26.791	00:46:35.726	8.935
8	00:46:39.964	00:46:48.831	8.867
9	00:47:01.378	00:47:10.246	8.868
10	00:47:12.173	00:47:20.232	8.059

4. KESIMPULAN

Dalam penelitian "Sistem Automasi Zero Touch Provisioning Routerboard Berbasis GraphQL API" dapat disimpulkan beberapa poin sebagai berikut: (a) Sistem automasi yang dirancang mampu digunakan untuk *provisioning* dan eksekusi konfigurasi secara massal. (b) Proses komunikasi dalam sistem automasi jaringan routerboard dapat disalurkan menggunakan protokol GraphQL. (c) Eksekusi konfigurasi secara massal dapat memudahkan pekerjaan repetitif dalam operasional industri Internet Service Provider. (d) Proses zero touch provisioning mampu dilakukan dan membutuhkan durasi selama 8.851ms untuk satu kali *provisioning*. (e) Mass Deploy yang dirancang pada sistem automasi ini relatif lebih cepat dan stabil dalam melakukan eksekusi konfigurasi secara massal. Diharapkan pada penelitian berikutnya, zero touch provisioning dapat menerapkan validasi dan adaptif terhadap berbagai jenis perangkat jaringan.

DAFTAR PUSTAKA

[1] B. K. Simpony, "Simple Queue Untuk Manajemen User dan Bandwidth di Jaringan Hotspot Menggunakan Mikrotik," *Jurnal Informatika*, vol. 8, no. 1, 2021, [Online]. Available: http://ejournal.bsi.ac.id/ejurnal/index.php/ji

[2] A. R. Komarudin, "Unlocking Indonesia's Digital Business Opportunity using Mikrotik," Bali, 2019.

[3] H. Shakirat Oluwatosin, "Client-Server Model," Ver. IX, 2014. [Online]. Available: www.iosrjournals.orgwww.iosrjournals.org

[4] A. F. Rochim, A. Rafi, A. Fauzi, and K. T. Martono, "As-RaD System as a Design Model of the Network Automation Configuration System Based on the REST-API and Django Framework," *Kinetik: Game Technology, Information System, Computer Network, Computing, Electronics, and Control*, pp. 291–298, Nov. 2020, doi: 10.22219/kinetik.v5i4.1093.

[5] Y. H. Tasanah Assakur, M. S. Fahrudin, and F. Ferdiansyah, "Implementasi API Mikrotik untuk Management Router Berbasis Android (Studi Kasus: PT Sigma Adi Perkasa)," *Jurnal Sains dan Informatika*, vol. 6, no. 1, pp. 92–101, Jun. 2020, doi: 10.34128/jsi.v6i1.217.

[6] B. F. Wicaksono, R. Ariansyah, and I. Handriani, "Sistem Aplikasi 'Pinda' Untuk Mencari Content Creator Menggunakan Metode Scrum," Journal of Computer Engineering, System and Science, vol. 4, no. 2, 2019.

[7] D. A. Hartina, A. Lawi, and B. L. E. Panggabean, "Performance Analysis of GraphQL and RESTful in SIM LP2M of the Hasanuddin University," in *2018 2nd East Indonesia Conference on Computer and Information Technology (EIConCIT)*, IEEE, Nov. 2018, pp. 237–240. doi: 10.1109/EIConCIT.2018.8878524.

[8] G. S. Mas Diyasa, G. S. Budiwitjaksono, H. Amarul Ma'rufi, and I. A. W. Sampurno, "Comparative Analysis of Rest and GraphQL Technology on Nodejs-Based Api Development," Galaxy Science, Apr. 2021. doi: 10.11594/nstp.2021.0908.

[9] E. K. Suni, "Analisis dan Perancangan Data Warehouse Untuk Mendukung Keputusan Redaksi Televisi Menggunakan Metode Nine-Step Kimball," *Jurnal Teknik Informatika*, vol. 11, no. 2, pp. 197–206, Nov. 2018, doi: 10.15408/jti.v11i2.8560.

[10] M. R. Fajar and E. K. Suni, "Sistem Pendukung Keputusan Karyawan Teladan Menggunakan Algoritma SAW Pada PT Semesta Citra Media," *Jurnal Informatika*, vol. 8, no. 2, 2021, [Online]. Available: http://ejournal.bsi.ac.id/ejurnal/index.php/ji

[11] E. K. Suni and H. I. Maulana, "Penerapan Digital Signature Untuk Mengesahan Proposal Hibah Dikti Menggunakan Secure Hash Algorithm," Journal of Information Technology and Computer Science, vol. 5, no. 2, 2020.

[12] D. Rafique and L. Velasco, "Machine learning for network automation: Overview, architecture, and applications [Invited Tutorial]," *Journal of Optical Communications and Networking*, vol. 10, no. 10, pp. D126–D143, Oct. 2018, doi: 10.1364/JOCN.10.00D126.

[13] R. Casellas, R. Martínez, R. Vilalta, and R. Muñoz, "Control, Management, and Orchestration of Optical Networks: Evolution, Trends, and Challenges," *Journal of Lightwave Technology*, vol. 36, no. 7, pp. 1390–1402, Apr. 2018, doi: 10.1109/JLT.2018.2793464.

[14] The GraphQL Foundation, "Introduction to GraphQL," 2020. https://graphql.org/learn/

[15] J. Ofoeda, R. Boateng, and J. Effah, "Application programming interface (API) research: A review of the past to inform the future," *International Journal of Enterprise Information Systems*, vol. 15, no. 3. IGI Global, pp. 76–95, Jul. 01, 2019. doi: 10.4018/IJEIS.2019070105.