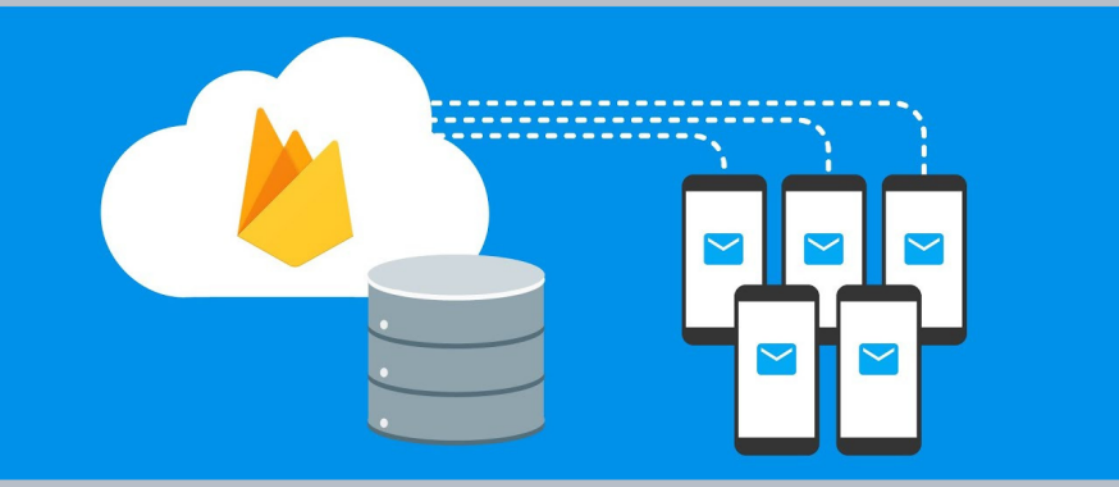
Manajemen Jaringan Menggunakan Firebase Cloud Massaging Berbasis Android Teori dan Praktek



Disusun Oleh :
Dessyanto Boedi Prasetyo
Rizki Inka Miftah
Rifki Indra Perwira

ISBN 978-602-5534-68-3

Penerbit LPPM UPN "Veteran" Yogyakarta

Manajemen Jaringan Menggunakan Firebase Cloud Massaging Berbasis Android

Teori dan Praktek

Disusun Oleh :
Dessyanto Boedi Prasetyo
Rizki Inka Miftah
Rifki Indra Perwira

Penerbit LPPM UPN "Veteran" Yogyakarta

KATA PENGANTAR

Puji syukur kehadirat Allah SWT yang senantiasa memberikan rahmat, hidayah serta ridhonya sehingga penulis dapat menyelesaikan buku yang berjudul "**Manajemen Jaringan Menggunakan Firebase Cloud Massaging Berbasis Android,Teori dan Praktek**".

Salawat serta salam diberikan kepada jujungan Nabi Muhammad SAW yang telah membawa risalah yang benar sebagai pedoman menuju jalan yang lurus yang di ridhoi Allah SWT. Buku ini merupakan salah satu instrument output penelitian. Buku ini berisi manajemen jaringan dan monitoring bagi admin menggunakan mikrotik.

Terlepas dari semua itu, penulis menyadari sepenuhnya bahwa masih ada kekurangan baik dari segi susunan kalimat maupun tata bahasanya. Oleh karena itu dengan tangan terbuka penulis menerima segala saran dan kritik dari pembaca agar penulis dapat memperbaiki makalah ilmiah ini. Akhir kata penulis berharap semoga buku ini dapat memberikan manfaat maupun inpirasi terhadap pembaca. Semoga Allah SWT selalu meridhoi semua umatnya, Amiin Ya Robbal Alamin.

Yogyakarta, September 2019

Penulis

DAFTAR ISI

BAB I
PENDAHULUAN

1.1 Latar Belakang Masalah

Jaringan internet merupakan komponen penting bagi semua pihak. Namun masalah koneksi data pada jaringan internet seringkali terjadi. Masalah seperti *timeout*, *data traffic* padat, hingga safety sering menyebabkan ketidaknyamanan para pengguna. Maka dari itu, jaringan internet memerlukan alat pengatur jalur lalu-lintas data agar tepat pada sasaran bernama *router*. Perangkat *router* yang didukung dengan fasilitas-fasilitas yang dimiliki dapat mengatasi permasalahan yang terjadi pada jaringan. Dalam hal ini *router* yang akan dibahas adalah *router Mikrotik* yang tersedia pada Universitas Pembangunan Nasional "Veteran" Yogyakarta.

Karena suatu jaringan harus selalu dipantau agar jaringan tetap terkendali dengan baik, maka seorang teknisi/*administrator* tentu tidak bisa selalu berada di ruang kantor pengelolaan untuk memantau kondisi *router mikrotik* secara *real-time*. Untuk mengatasi hal tersebut, pada penelitian ini akan mengembangkan sistem pengiriman notifikasi berupa info aktifitas-aktifitas yang terjadi pada *router mikrotik* berisikan parameter seperti *limit bandwidth*, perubahan *IP*, *reboot device*, dan pesan *error*. Pesan *error* ini akan berguna untuk mengatasi masalah keamanan, seperti kegiatan ilegal saat seseorang melakukan proses *login* ke *router mikrotik* gagal.

Pada kasus ini menggunakan teknologi *Firebase Cloud Messaging* (*FCM*) yang berfungsi sebagai *web service* penyedia layanan *push notification* atau pesan notifikasi yang berisikan aktifitas-aktifitas yang terjadi pada *router mikrotik*, dan juga berperan untuk menjembatani antara *server* dengan perangkat *Android* agar dapat terjadi aktifitas pengiriman pesan (*push notification*).

Dari penjabaran permasalahan di atas, aplikasi *Network Notification System* berbasis *Android* dengan menggunakan teknologi *Firebase Cloud Messaging* (*FCM*) dapat menjadi alternatif untuk membantu kesigapan kinerja teknisi/*administrator* jaringan untuk memberikan pelayanan terbaik saat terjadi permasalahan jaringan internet di lingkungan civitas akademika Universitas Pembangunan Nasional "Veteran" Yogyakarta.

BAB II
TINJAUAN LITERATUR

2.1 Monitoring Jaringan

Monitoring jaringan komputer (Pradikta, 2013) adalah proses pengumpulan dan melakukan analisis terhadap data-data pada lalu lintas jaringan dengan tujuan memaksimalkan seluruh sumber daya yang dimiliki jaringan komputer. *Monitoring* jaringan ini merupakan bagian dari manajemen jaringan. *Monitoring* Jaringan Komputer dapat dibagi menjadi 2 bagian yaitu :

a) Connection Monitoring

Connection monitoring adalah teknik *monitoring* jaringan yang dapat dilakukan dengan melakukan tes *ping* antara *monitoring station* dan *device target*, sehingga dapat diketahui bila koneksi terputus.

b) Traffic Monitoring

Traffic monitoring adalah teknik monitoring jaringan dengan melihat paket aktual dari *traffic* pada jaringan dan menghasilkan laporan berdasarkan *traffic* jaringan.

2.1.1 Tujuan Monitoring Jaringan

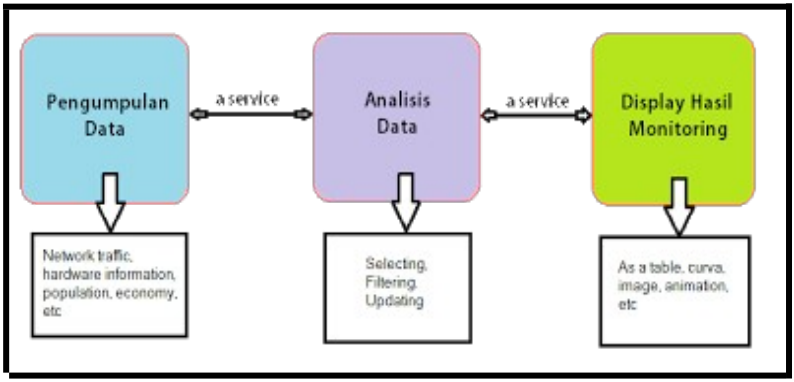
Tujuan *monitoring* jaringan adalah untuk mengumpulkan informasi yang berguna dari berbagai bagian jaringan sehingga jaringan dapat diatur dan dikontrol dengan menggunakan informasi yang telah terkumpul. Berikut ini beberapa alasan utama dilakukan *monitoring* jaringan :

a) Untuk menjaga stabilitas jaringan.
b) Sulit untuk mengawasi apa yang sedang terjadi di dalam jaringan yang memiliki sejumlah besar mesin (*host*) tanpa alat pengawas yang baik
c) Untuk mendeteksi kesalahan pada jaringan, *gateway*, *server*, maupun *user*.

d) CalibriUntuk memberitahu *trouble* kepada *administrator* jaringan secepatnya. Mempermudah analisis *troubleshooting* pada jaringan.
e) Mendokumentasikan jaringan.

2.1.2 Tahapan Monitoring Jaringan

Secara garis besar tahapan dalam sebuah sistem *monitoring* terbagi ke dalam tiga proses besar, yaitu:
a) Proses di dalam pengumpulan data *monitoring*.
b) Proses di dalam analisis data *monitoring*.
c) Proses di dalam menampilkan data hasil *monitoring*.



Gambar 2.1 Tahapan *Monitoring* Jaringan

Analogi proses dapat dilihat pada gambar 2.1 Sumber data dapat berupa *network traffic*, informasi mengenai *hardware*, atau sumber-sumber lain yang ingin diperoleh informasi mengenai dirinya. Proses dalam analisis data dapat berupa pemilihan data dari sejumlah data telah terkumpul atau bisa juga berupa manipulasi data sehingga diperoleh informasi yang diharapkan. Sedangkan tahap menampilkan data hasil *monitoring* menjadi informasi yang berguna di dalam pengambilan keputusan atau kebijakan terhadap sistem yang

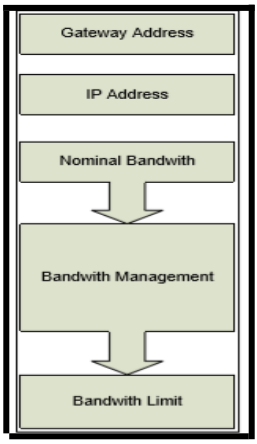
sedang berjalan dapat berupa sebuah tabel, gambar, gambar kurva, atau bahkan animasi.

Aksi yang terjadi di dalam sebuah sistem *monitoring* berbentuk *service*, yaitu suatu proses yang terus-menerus berjalan pada interval waktu tertentu. Proses yang dijalankan dapat berupa pengumpulan data dari objek yang di-*monitor*, atau melakukan analisis data yang telah diperoleh dan menampilkannya.

2.2 Manajemen Jaringan

Pada penelitian mengenai pembuatan aplikasi yang mampu memfasilitasi *admin* jaringan pada LABKOM dalam membatasi penggunaan *bandwith* dengan memanfaatkan *HTB-tools* sebagai aplikasi pendukung dalam memanajemen *bandwith* dan pembatasan akses *gateway* untuk *port* tertentu dan blokir *website* dengan fasilitas dari *IP Tables* dari sistem operasi *Ubuntu* (Wun et al., 2014).

Sistem ini dilengkapi dengan fasilitas manajemen jaringan diantaranya blok *website* dan *open/close gateway* dan juga sistem untuk melakukan limitasi *bandwidth*. Dari modul blok *website* dan *open/close gateway* menggunakan sabuah aplikasi *pre-instaled* pada *Ubuntu* yaitu "*iptables*". Untuk modul *limit bandwidth* menggunakan aplikasi tambahan yaitu *HTBTools* untuk memudahkan pengguna dalam memanajemen *bandwidth* dan banyaknya *support* dan tutorial penggunaan *HTB-Tools* untuk membantu *maintenance* jika ada permasalahan pada limitasi *bandwidth*. Penggunaan *HTB-Tools* memudahkan *developer* aplikasi mengintegrasikan sistem yang dibangun dengan *HTB-Tools*. Konfigurasi untuk manajemen *bandwidth* pada *HTB-Tools* dapat diakkses di "*/etc/htb/eth0-qos.cfg*" dan yang utama dalam memanejemen *bandwidth* adalah *total banwith*, *minimum limit*, *gateway*, *IP address*, dan *subnet* dari jaringan LABKOM.



Gambar 2.2 Blok Diagram *Management Bandwidth*

2.2.1 Bandwidth Internet

IPCop sebagai alat manajemen trafik dan *bandwidth* internet memudahkan *administrator* dalam melakukan pengaturan *bandwidth* dan *monitoring* trafik dan penggunaan *bandwidth* internet melalui media *web*. Dalam pengujian dilakukan terhadap dua jaringan besar di Politeknik Telkom, Bandung yaitu jaringan publik dan jaringan staf. Dari hasil pengujian, terbukti bahwa *IPCop* berhasil mengoptimalkan dan membatasi *bandwidth* yang sampai pada *client*. Terbukti dengan sesuainya *throughput* yang didapat *client* dengan *throughput* yang diatur pada *IPCop* (Riza, 2011).

Istilah *bandwidth* dapat diartikan sebagai kapasitas atau daya tampung suatu *channel* komunikasi (medium komunikasi) untuk dapat dilewati sejumlah *traffic* informasi atau data dalam satuan waktu tertentu. Umumnya *bandwidth* dihitung dalam satuan *bit*, *kbit* atau *bps* (*byte per second*). Pengalokasian

bandwidth yang tepat bisa menjadi salah satu cara dalam menjamin kualitas suatu layanan jaringan (*QoS = Quality Of Services*). Sedangkan istilah *traffic* dapat didefinisikan sebagai banyaknya informasi yang melewati suatu *channel* komunikasi (medium komunikasi).

TCP dan UDP

Penerapan penggunaan pustaka *SNMP++* yang berorientasi objek dalam membuat sebuah aplikasi perangkat lunak, berfungsi menginformasikan statistik layanan-layanan *TCP* dan *UDP* dalam sebuah jaringan lokal *TCP/IP* (Sagita et al., 2011).

TCP bersifat andal (*reliable*) dan berorientasi koneksi (*connection oriented*). *TCP* memberikan layanan yang andal, karena ia melakukan proses pengiriman *acknowledgment*, yaitu bahwa *TCP* penerima segmen akan memberitahukan kepada pengirim bahwa segmen telah diterimanya. Bila sampai batas waktu yang telah ditetapkan *acknowledgment* penerima belum sampai kepada pengirim, maka pengirim melakukan *re-send* segmen ke penerima.

Sifat layanan *TCP* disebut berorientasi koneksi karena *TCP* pengirim akan lebih dulu memberitahukan bahwa ia akan membangun komunikasi kepada penerima. Bila penerima setuju, komunikasi baru dapat berjalan. Selain itu, layanan *TCP* juga membuat pihak penerima data untuk melakukan pengurutan segmen yang diterimanya, jika segmen datang tidak berurutan. Proses ini disebut proses pengurutan data (*data sequencing*). Layanan *TCP* dibangun dengan membuat hubungan antara pihak pengirim dan penerima data dan dianggap telah terjalin jika keduanya telah membuat suatu *endpoint*, yang disebut *socket*. *Socket* terdiri atas pasangan nomor alamat *IP 32-bit* dan *port 16-bit*.

UDP tidak berorientasi koneksi (*connectionless oriented*) dan juga tidak andal. Hal ini disebabkan ia tidak mengenal

mekanisme *acknowledgment* dan pengiriman ulang. Dalam melakukan pengiriman paket data, layanan *UDP* tidak mengenal nomor urut paket sehingga tidak melakukan proses pengurutan paket.

Topologi Jaringan

Cacti merupakan aplikasi *opensource* yang digunakan secara gratis yang berbasis *RRDTool* (*Round-Robin Database Tool*) untuk membuat penyimpanan data grafik. Lalu-lintas trafik pada jaringan dapat di pantau penggunaanya. *Mail notification* ditambahkan untuk mempercepat penyampaian informasi mengenai status *node*, *link* atau *device* jika terjadi gangguan pada jaringan *VPN-MPLS WAN*. Sistem pemantauan *traffic* jaringan diharapkan memberikan informasi keadaan jaringan *WAN* dan pemakaian *traffic*, *CPU* dan *memory* di tiap *node* serta dapat membuat sistem pelaporan dan notifikasi *email* bagi pengelola jaringan. Memudahkan pengelolaan jaringan dan mempercepat informasi gangguan jaringan pada *PT PLN* (*Persero*) Distribusi Jawa Tengah dan Daerah Istimewa Yogyakarta (Kamto and Putra, 2015).

Jaringan computer dapat dibedakan menjadi 3 kelompok berdasarkan jarak dan wilayah kerjanya:

a) *Local Area Network* (*LAN*)

Local Area Network (*LAN*) digunakan untuk menghubungkan beberapa komputer pribadi dan *workstation* di suatu perusahaan yang menggunakan peralatan secara bersamaan dan saling bertukar informasi.

b) *Metropolitan Area Network* (*MAN*)

Metropolitan Area Network (*MAN*) merupakan versi *LAN* dengan kapasitas lebih besar. *MAN* merupakan alternatif pembuatan jaringan komputer antar perusahaan dalam suatu wilayah yang sama.

c) Wide Area Network (*WAN*)

Wide Area Network (*WAN*) adalah jaringan yang memiliki jarak yang sangat luas, karena radiusnya dapat mencakup sebuah negara atau bahkan benua.

2.3 Cloud Computing

National Institute of Standards and Technology (NIST), *Information Technology Laboratory* memberikan dua buah catatan mengenai pengertian komputasi awan. Pertama, komputasi awan masih merupakan paradigma yang berkembang. Definisi, kasus penggunaan, teknologi yang mendasari, masalah, risiko, dan manfaat akan terus disempurnakan melalui perdebatan baik oleh sektor publik maupun swasta. Definisi, atribut, dan karakteristik akan berkembang dan berubah dari waktu ke waktu. Kedua, industri komputasi awan merupakan ekosistem besar dengan banyak model, vendor, dan pangsa pasar. Definisi ini mencoba untuk mencakup semua pendekatan berbagai awan (Peter Mell, 2011).

Jadi, komputasi awan adalah model untuk memungkinkan kenyaman, *on-demand* akses jaringan untuk memanfaatkan bersama suatu sumber daya komputasi yang terkonfigurasi (misalnya, jaringan, *server*, penyimpanan, aplikasi, dan layanan) yang dapat secara cepat diberikan dan dirilis dengan upaya manajemen yang minimal atau interaksi penyedia layanan. Model komputasi awan mendorong ketersediaan dan terdiri dari lima karakteristik, tiga model layanan, dan empat model penyebaran.

2.3.1 Karakteristik Cloud Computing

NIST mengidentifikasi lima karakteristik penting dari komputasi awan (Peter Mell, 2011) sebagai berikut:

a) *On-demand self-service*. Pengguna dapat memesan dan mengelola layanan tanpa interaksi manusia dengan penyedia layanan, misalnya melalui sebuah portal *web* dan manajemen

antarmuka. Pengadaan dan perlengkapan layanan serta sumberdaya yang terkait terjadi secara otomatis pada penyedia.

b) *Broad network access*. Kemampuan yang tersedia melalui jaringan dan diakses melalui mekanisme standar, yang mengenalkan penggunaan berbagai platform (misalnya, telepon selular, laptop, dan PDA).

c) *Resource pooling*. Penyatuan sumberdaya komputasi yang dimiliki penyedia untuk melayani beberapa konsumen menggunakan model multi-penyewa, dengan sumberdaya fisik dan virtual yang berbeda, ditetapkan secara dinamis dan ditugaskan sesuai dengan permintaan konsumen. Ada rasa kemandirian lokasi bahwa pelanggan umumnya tidak memiliki kontrol atau pengetahuan atas keberadaan lokasi sumberdaya yang disediakan, tetapi ada kemungkinan dapat menentukan lokasi di tingkat yang lebih tinggi (misalnya, negara, negara bagian, atau *data center*).

d) *Rapid elasticity*. Kemampuan dapat dengan cepat dan elastis ditetapkan.

e) *Measured Service*.

Sistem komputasi awan secara otomatis mengawasi dan mengoptimalkan penggunaan sumberdaya dengan memanfaatkan kemampuan pengukuran (*metering*) pada beberapa tingkat yang sesuai dengan jenis layanan (misalnya, penyimpanan, pemrosesan, *bandwidth*, dan akun pengguna aktif).

BAB III
PENGEMBANGAN SISTEM

Pada bab III ini akan dijabarkan penjelasan mengenai pengembangan sistem, alat dan bahan penelitian, jalannya penelitian, serta perancangan sistem. Penelitian dilakukan dengan cara membuat sebuah *web service* dan melakukan konfigurasi pada *router mikrotik* agar dapat mengirimkan parameter-parameter yang terjadi pada jaringan internet secara *real-time*, dan kemudian dikirimkan pesan notifikasi ke dalam aplikasi *Android*.

3.1 Metodologi Pengembangan Sistem

Metodologi yang digunakan pada proses pengembangan sistem ini yaitu *Guide Lines for Rappid Application Engineering* (*GRAPPLE*).

Metodologi *Guide Lines for Rappid Application Engineering* (*GRAPPLE*) memiliki lima tahapan yaitu :

a) Pengumpulan kebutuhan (*Requirement gathering*)
b) Analisis (*Analysis*)
c) Perancangan (*Design*)
d) Pengembangan (*Development*)
e) Penyebaran (*Deployment*)

Berikut penjelasannya;

3.1.1 Pengumpulan Kebutuhan (Requirement Gathering)

Pada tahap ini dilakukan proses identifikasi dan merencanakan teknologi yang digunakan untuk pembuatan aplikasi dan perancangan *server*. Proses identifikasi dilakukan dengan melakukan wawancara kepada pihak *administrator* jaringan Universitas Pembangunan Nasional "Veteran" Yogyakarta. Tahap perencanaan dibutuhkan untuk proses jalannya pengembangan aplikasi *Network Notification System* menggunakan teknologi *Firebase Cloud Messaging (FCM)* berbasis *Android* supaya tidak berjalan secara acak dan

ditentukan empat parameter yang dibutuhkan yaitu *limit bandwidth*, perubahan *IP*, informasi *error*, dan *device reboot*.

3.1.2 Hasil Analisis yang Dilakukan

Pada tahap ini dilakukan analisis dengan cara memetakan hasil observasi dan wawancara dengan pihak *administrator* jaringan Universitas Pembangunan Nasional "Veteran" Yogyakarta tentang

informasi bahwa selama ini banyak laporan masuk dari satuan kerja mengenai lambatnya proses penanganan jaringan dikarenakan tingkat mobilitas yang tinggi dan keterbatasan jumlah *administrator* menyebabkan kegiatan pemantauan jaringan tidak berjalan secara *real time*, maka dari itu dibutuhkan sebuah terobosan aplikasi yang dapat memberikan pesan notifikasi mengenai kegiatan (*event*) yang berhubungan dengan *router mikrotik* dan berisikan empat parameter pesan yaitu *bandwidth*, *error*, perubahan *IP*, dan *reboot device*.

3.1.3 Perancangan (Design)

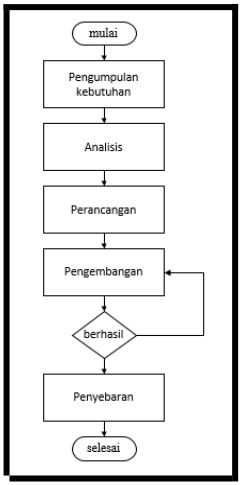
Pada tahap desain perancangan merupakan hasil dari proses analisis yang digambarkan melalui perancangan *prototype user interface* berupa mockup desain *web admin* beserta desain aplikasi *Android* yang akan dibuat.

3.1.4 Pengembangan (Development)

Pada tahap ini proses pengembangan sistem (*system development*) dilakukan dengan proses konfigurasi pendukung aplikasi yaitu *server*, *mikrotik*, dan *FCM*. Selanjutnya, mulai dengan proses *coding* untuk membuat aplikasi *Android* dengan aplikasi *Android studio*, kemudian dibuat *web admin* dengan bahasa *PHP* menggunakan aplikasi *sublime text*, kemudian dilakukan proses pengujian untuk memastikan sistem yang dibuat dapat berjalan sesuai perancangan awal.

3.1.5 Penyebaran (Deployment)

Pada tahap penyebaran ini akan dilakukan pendistribusian aplikasi kepada pengguna dalam hal ini administrator jaringan UPN "Veteran" Yogyakarta.



Gambar 3.1 *Flowchart* Proses Pengembangan Sistem

3.2 Bahan Penelitian

Berikut ini merupakan bahan penelitian yang dibutuhkan dalam proses jalannya penelitian pengembangan aplikasi aplikasi *Network Notification System* berbasis *Android* dengan menggunakan teknologi *Firebase Cloud Messaging* (*FCM*). Alat-alat penelitian yang dibutuhkan antara lain:

a) Perangkat Keras (*Hardware*)
Antara lain:

1. Laptop dengan spesifikasi *Processor Intel Core i3* 2,10 GHz, *RAM* 6 GB, *Hardisk* 500 GB dengan Sistem Operasi *Linux Ubuntu* 17.04 *Zesty Zapus* yang dijadikan sebagai *web server, database,* dan juga untuk pembuatan aplikasi (*coding*).
2. *Mikrotik RB941 yang akan dikonfigurasi*.

3. Perangkat *Android* dengan spesifikasi *Processor Quad-core* 1.40 GHz, *RAM* 2 GB, *Storage* 32 GB sebagai perangkat uji coba aplikasi.

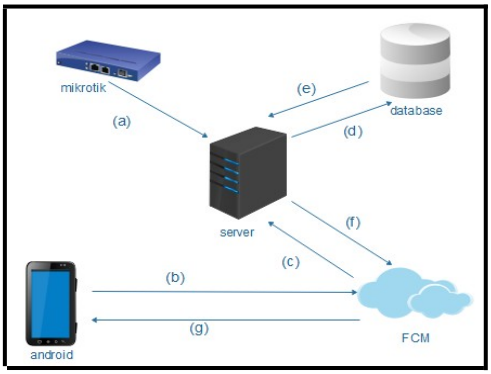
b) Perangkat Lunak (*Software*)

Antara lain :

1. *Android Studio*
   Merupakan aplikasi pembuat aplikasi *Android* (*coding*).
2. *Star UML*
   Merupakan aplikasi pembuat rancang aplikasi (*flowchart* dan *UML* (*Unified Modelling Language*)).
3. *Microsoft Word*
   Merupakan aplikasi pembuat naskah laporan hasil dokumentasi dari penelitian ini.
4. *Sublime Text*
   Aplikasi untuk pembuatan halaman *admin* dengan menggunakan bahasa *PHP*.
5. *LAMP (Linux Apache, MySQL, PHP)*
   Sebuah *web server* pada OS *Linux* yang akan digunakan dalam penelitian. *Linux* sebagai sistem operasi, *apache* berfungsi sebagai *web service*, *MySQL* sebagai *database*, dan *PHP* sebagai bahasa pemrograman.
6. *Winbox*
   Aplikasi yang digunakan untuk konfigurasi *router mikrotik*.
7. *Balsamiq Mockups*
   Aplikasi pembuat desain *user interface* pada aplikasi *Android* dan juga halaman *web admin.*

3.3 Arsitektur Sistem

Pada bagian ini akan dijelaskan mengenai arsitektur dari proses jalannya sistem seperti gambar 3.2 berikut.

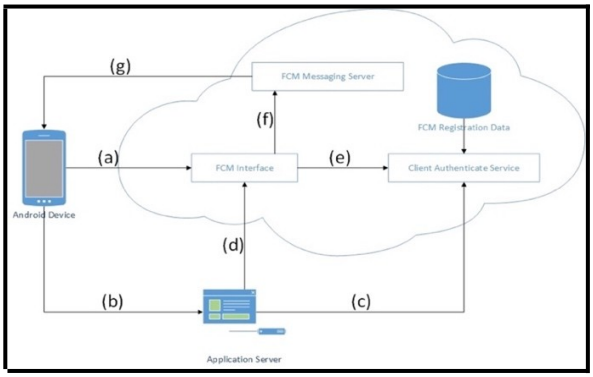


Gambar 3.2 Arsitektur Sistem

a) *Mikrotik* akan terus menerus mengirimkan info *log* ke *server* untuk selanjutnya di lakukan proses *parsing* sesuai parameter yang dibutuhkan.
b) Ketika aplikasi *user* pada perangkat *Android* dijalankan dan melakukan registrasi, *device* akan mendapatkan identitas berupa *token* dari *FCM*.
c) *Firebase Cloud Messaging* (*FCM*) mengirim data *user* yang telah melakukan registrasi beserta *token device*nya.
d) *Server* menjalankan fungsi *insert data* untuk memasukkan data ke dalam *database*. Pada tahap ini juga *server* akan melakukan proses *parsing* data *log mikrotik* dan mengirimnya ke dalam *database*.
e) *Server* dapat memperoleh data pesan maupun *users* untuk dikelola maupun pesan untuk dikirimkan menuju *user*.
f) *Server* mengirimkan pesan kepada user melalui *FCM*.
g) *Firebase Cloud Messaging* (*FCM*) mengirimkan pesan *push notification* ke perangkat *Android*.

3.4 Arsitektur Cloud

Berikut penjelasan mengenai arsitektur daripada *Firebase Cloud Messaging* (*FCM*) pada khususnya seperti pada gambar 3.3 berikut ini.

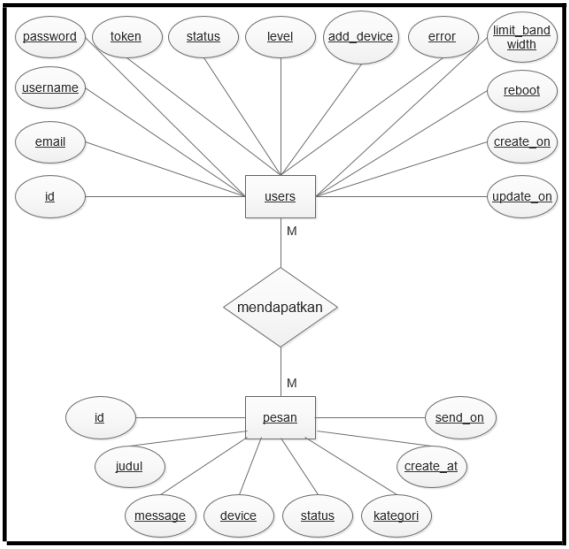


Gambar 3.3 Arsitektur *Firebase Cloud Messaging* (*FCM*)

a) *Android device* akan teregistrasi ke layanan *Firebase Cloud Messaging* (*FCM*) dan menerima ID register.
b) ID register dikirimkan menuju *application server*.
c) *Application server* meminta autentikasi ke *Client Authenticate Service* dari *Firebase Cloud Messaging* (*FCM*) untuk mendapatkan *token*.
d) Pesan dikirimkan ke perangkat *android* berdasarkan *token* yang didapatkan sebelumnya.
e) *Token* dan ID register diverifikasi kecocokannya.
f) Pesan akan diarahkan untuk segera dikirim ke perangkat *android*.
Pesan berhasil terkirim sebagai notifikasi.

3.5 Entity Relationship Diagram (ERD)

Entity Relationship Diagram (*ERD*) yang akan menggambarkan proses perancangan *database*. ERD pada sistem yang dibuat ini memiliki 2 buah entitas yang terdiri *users* dan *pesan*. Kedua entitas tersebut memiliki hubungan *many-to-many*, karena setiap *user* dapat mendapatkan banyak pesan.



Gambar 3.4 *Entity Relationship Diagram* (*ERD*)

3.6.1 Struktur Tabel

Pada bagian ini akan dijabarkan struktur tabel-tabel yang tersedia pada *database* sistem yang dibuat, struktur tabel terdiri dari tabel *users* dan tabel *pesan*. Berikut ini merupakan struktur tabel *users*.

Tabel 3.1 Struktur Tabel *Users*

No	Nama Field	Type	Size	Keterangan
1	*Id*	*Int*	11	*ID user*
2	*Email*	*Varchar*	100	*Email user*
3	*Username*	*Varchar*	100	Nama *user*
4	*Password*	*Text*		*Password user*
5	*Token*	*Varchar*	255	*ID device* terdaftar
6	*Status*	*Int*	2	Status *user* dikirim pesan

7	*Level*	*Varchar*	10	*User* sebagai *admin*/*client*
8	*Add_device*	*Int*	2	Status *user* menerima pesan perubahan *device*
9	*Error*	*Int*	2	Status *user* menerima pesan *error*
10	*Limit_bandwidth*	*Int*	2	Status *user* menerima pesan *limit bandwidth*
11	*Reboot*	*Int*	2	Status *user* menerima pesan saat *device reboot*
12	*Create_at*	*Datetime*		Waktu data users didaftar
13	*Update_on*	*Datetime*		Waktu data users diperbaharui

Tabel *users* di atas terdiri dari 13 buah *field* sesuai dengan tipe, ukuran, dan keterangannya. *ID users* berguna sebagai penanda *user*, *email*, *username*, dan *password* sebagai data dari *user, token* berarti *ID Device* yang terdaftar dalam *firebase*, untuk *status* berarti status *user* untuk dikirimkan pesan notifikasi atau tidak, *add_device*, *error*, *limit_bandwidth*, dan *reboot* merupakan jenis pesan yang akan dikirimkan sesuai kategori yang dipilih *user*. *Create_at* sebagai penunjuk waktu data *user* didaftarkan, untuk *update_on* sebagai penunjuk waktu saat terjadi perubahan data *user*. Sementara berikut ini merupakan struktur tabel *pesan*.

Tabel 3.2 Struktur Tabel Pesan

No	Nama Field	Type	Size	Keterangan
1	*Id*	*Int*	11	ID pesan
2	*Judul*	*Varchar*	200	Judul pesan
3	*Message*	*Text*		Isi Pesan
4	*Device*	*Varchar*	20	*Device* pengirim pesan
5	*Status*	*Int*	2	Status Pesan Terkirim
6	*Kategori*	*Varchar*	200	Jenis Pesan
7	*Create_at*	*Datetime*		Waktu pesan dibuat
8	*Send_on*	*Datetime*		Waktu pesan dikirim

Tabel *pesan* di atas terdiri dari 8 buah *field* sesuai dengan tipe, ukuran, dan keterangannya. *ID pesan* berguna sebagai penanda pesan, untuk data judul pesan, isi pesan, dan *device* pengirim nantinya akan ditampilkan pada aplikasi *android* berupa pesan notifikasi. Pada *field status* berarti pesan yang terdaftar dalam *database* telah terkirim atau belum, kategori berarti jenis pesan yang terdaftar. *Create_at* sebagai penunjuk waktu kejadian *log mikrotik*, untuk *update_on* sebagai penunjuk waktu saat pesan telah diterima oleh *user* pada perangkat *android*.

3.6 Perancangan *User Interface*

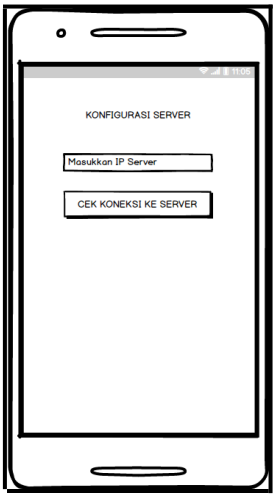
Perancangan antar muka (*Interface*) dilakukan sebelum diimplementasikan ke dalam bentuk aplikasi sebenarnya, terdiri dari *Mockup* aplikasi *Android* dan halaman *web admin*.

3.7.1 Mockup Halaman Awal

Dalam perancangan *mockup* halaman awal terdapat teks selamat datang, logo aplikasi, beserta tombol untuk melakukan registrasi *device*.

Gambar 3.5 *Mockup* Halaman Awal

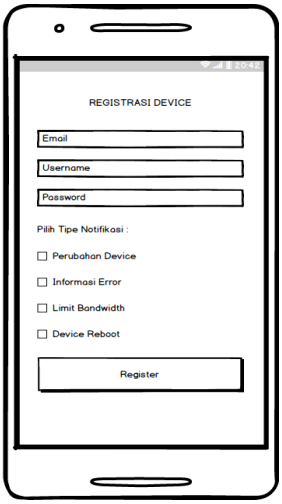
3.7.2 Mockup Halaman Koneksi ke Server



Gambar 3.6 *Mockup* Halaman Koneksi ke *Server*

3.7.3 Mockup Halaman Registrasi

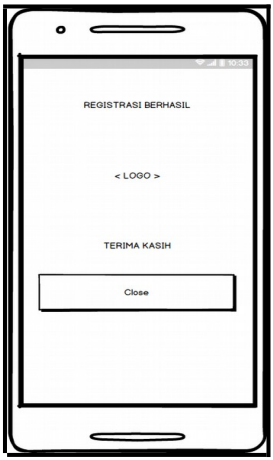
Dalam perancangan *mockup* halaman registrasi *user* akan menginputkan data *email*, *username*, dan *password* yang aktif, data yang dikirimkan akan tersimpan ke dalam *database*. Selanjutnya terdapat kotak cek (*check box*) untuk memilih info apa saja yang dibutuhkan *user* untuk dikirimkan pesan notifikasi.



Gambar 3.7 *Mockup* Halaman Registrasi

3.7.4 Mockup Halaman Registrasi Berhasil

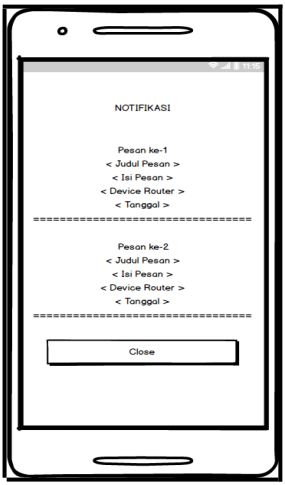
Dalam perancangan *mockup* halaman registrasi berhasil terdapat teks terima kasih, *device user* telah terdaftar, beserta tombol *close*.



Gambar 3.8 *Mockup* Halaman Registrasi Berhasil

3.7.5 Mockup Halaman Isi Pesan

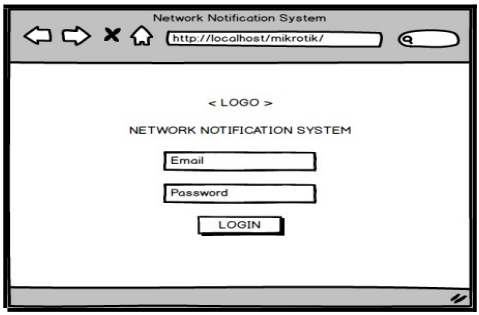
Dalam perancangan *mockup* halaman isi pesan terdapat teks isi dari pesan notifikasi yang diterima dengan data judul pesan, isi pesan, device pengirim, dan tanggal kejadian (*event*) yang ada.



Gambar 3.9 *Mockup* Halaman Isi Pesan

3.7.6 Mockup Halaman Login Admin

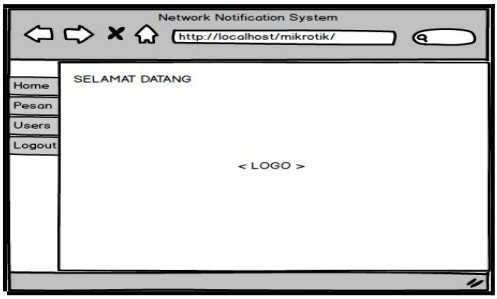
Dalam perancangan *mockup* halaman *login admin*, terdapat logo *Network Notification System*, kolom isi *email*, kolom isi *password*, dan tombol *login* untuk masuk ke halaman utama *web admin*.



Gambar 3.10 *Mockup* Halaman *Login Admin*

3.7.7 Mockup Halaman Awal Admin

Dalam perancangan *mockup* halaman awal di *web admin*, terdapat logo *Network Notification System*, *sidebar* yang berisikan menu *Home,* menu Pesan, menu *Users*, dan menu untuk *Logout*.

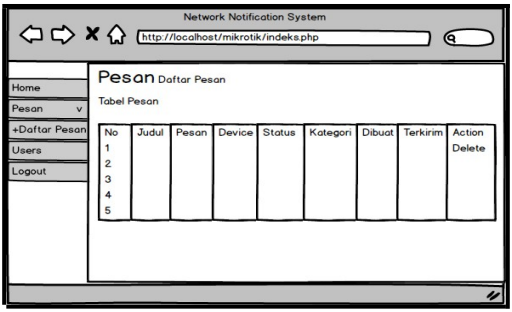


Gambar 3.11 *Mockup* Halaman Awal *Web Admin*

3.7.8 Mockup Halaman Menu Daftar Pesan

Dalam perancangan *mockup* halaman menu daftar pesan, terdapat tabel pesan yang berisi data nomor, judul pesan, isi pesan, *device* pengirim, status terkirim, kategori pesan, tanggal

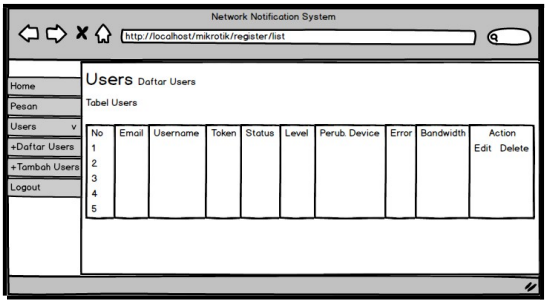
pesan dibuat, tanggal pesan terkirim, dan menu *action* untuk *edit* pesan maupun menghapus pesan yang tersedia.



Gambar 3.12 *Mockup* Halaman Menu Pesan

3.7.9 Mockup Halaman Menu Daftar Users

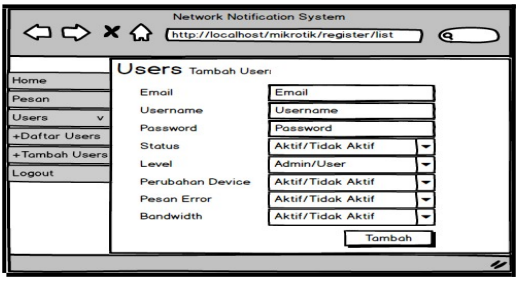
Dalam perancangan *mockup* halaman menu daftar *users*, terdapat tabel *users* yang berisi data nomor, *email user*, *username*, *token* yang terdaftar, status aktif *user*, level user, status pilihan notifikasi untuk perubahan *device*, pesan *error*, atau pesan mengenai *bandwidth*, kemudian terdapat menu *action* untuk *edit users* maupun menghapus *user*.



Gambar 3.13 *Mockup* Halaman Menu Daftar *Users*

3.7.10 Mockup Halaman Menu Tambah Users

Dalam perancangan *mockup* halaman menu tambah users, terdapat *form* yang berfungsi untuk menambah data *users* yang berisi sesuai dengan gambar di bawah. *Mockup* ini juga hampir sama dengan menu *edit user* yang perbedaannya terletak pada data yang masih kosong dengan tombol "*Edit User"*.



Gambar 3.14 *Mockup* Halaman Menu Tambah *Users*

3.7 Pengujian Sistem

Pengujian aplikasi *Network Monitoring System* ini meliputi tingkat keberhasilan sistem dalam proses pengiriman maupun penerimaan informasi kegiatan (*event*) yang terjadi pada *device router mikrotik* ke dalam aplikasi berbasis *Android* melalui *web service* yang telah dikonfigurasi sebelumnya dengan menggunakan pengujian fungsionalitas dengan cara melakukan simulasi mengoperasikan seluruh rangkaian berjalannya sistem mulai dari proses registrasi data pada aplikasi *android* dari *administrator,* proses pengiriman empat parameter *log mikrotik,* perbandingan beberapa pernagkat *router mikrotik*, percobaan fungsi-fungsi pada web admin untuk melihat data pesan maupun *users*, dan proses pengiriman data hingga muncul pesan notifikasi pada perangkat *Android*.

BAB IV
IMPLEMENTASI DAN PEMBAHASAN

Pada bab IV ini akan dijabarkan penjelasan mengenai proses implementasi sistem mulai dari proses konfigurasi *server*, *mikrotik*, dan *firebase*, serta beberapa *source code* yang dibutuhkan untuk membangun sistem seperti bahasan sebagai berikut.

4.1 Konfigurasi Server

4.1.1 Instalasi LAMP (Linux, Apache, MySQL, PHP)

Pada tahap ini akan dilakukan proses instalasi *LAMP* yang merupakan sebuah paket perangkat lunak untuk menjalankan tugasnya masing-masing. *Linux* berfungsi sebagai sistem operasi, *Apache* sebagai *web service*, *MySQL* sebagai *database*nya, *PHP* sebagai bahasa pemrogramannya. Perintah yang dijalankan di *terminal ubuntu* sebagai berikut.

```
sudo apt-get install apache2 mysql-client mysql-
server php7.0 libapache2-mod-php7.0
```

Modul Program 4.1 Perintah Instalasi *LAMP*

4.1.2 Konfigurasi RSYSLOG

Pada tahap ini dilakukan proses instalasi *RSYSLOG* beserta konfigurasinya. *RSYSLOG* merupakan fitur pada *linux* yang dapat digunakan untuk melakukan aktivitas *remote logging* seperti perintah yang dijalankan sebagai berikut.

```
sudo apt-get install rsyslog
sudo apt-get install rsyslog-mysql
```

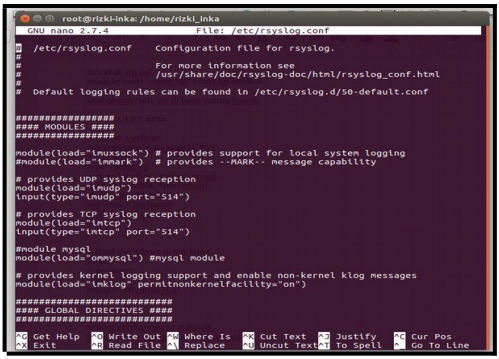
Modul Program 4.2 Perintah Instalasi *RSYSLOG*

Setelah dilakukan proses instalasi, langkah selanjutnya yaitu mengkonfigurasi file *RSYSLOG* dengan perintah berikut.

```
nano /etc/rsyslog.conf
```

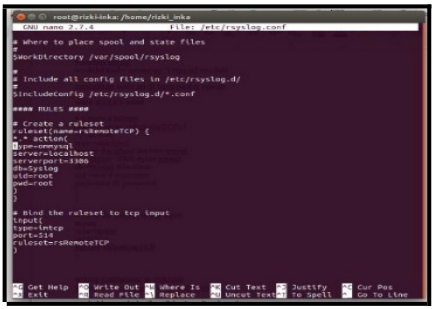
Modul Program 4.3 Perintah Memasuki File Konfigurasi *RSYSLOG*

Setelah memasuki file konfigurasi *RSYSLOG* langkah pertama yaitu dengan menghilangkan tanda pagar (#) pada baris *module* dan *input* pada *UDP syslog* dan *TCP syslog.*



Gambar 4.1 Konfigurasi *RSYSLOG* Pertama

Langkah selanjutnya dengan menambahkan kode *rules* dan alamat *IP* yang akan dijadikan sumber dari *remote logging* dari *RSYSLOG* pada baris bagian bawah *file* konfigurasi seperti pada gambar 4.2.

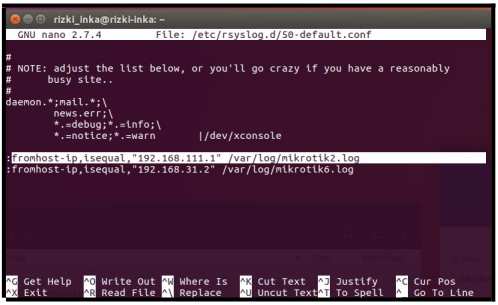


Gambar 4.2 Konfigurasi *RSYSLOG* Kedua

Langkah selanjutnya dengan mengedit daftar *IP mikrotik* yang akan dikenali *PC server* untuk menerima *log file* dan menyimpannya pada berkas *mikrotik.log* dengan perintah sebagai berikut.

```
nano /etc/rsyslog.d/50-default.conf
```

Modul Program 4.4 Perintah Menambahkan *List IP Mikrotik* Pengirim *Log File*

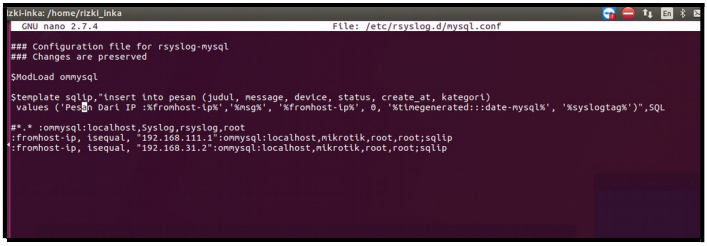


Gambar 4.3 Menambahkan *List IP Mikrotik* Pengirim *Log File*

Setelah *log file* masuk ke dalam *PC Server*, selanjutnya dikirimkan menuju *database MySQL* untuk selanjutnya dikirimkan menuju aplikasi *android* dengan perintah sebagai berikut.

```
nano /etc/rsyslog.d/mysql.conf
```

Modul Program 4.5 Perintah Mengirimkan *Log File* ke *database MySQL*

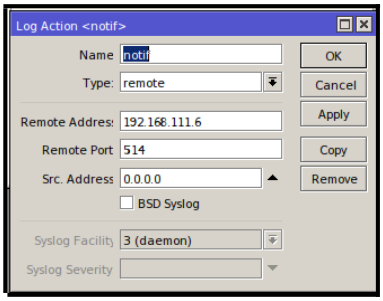


Gambar 4.4 Mengirimkan *Log File* ke *database MySQL*

4.2 Konfigurasi Router Mikrotik

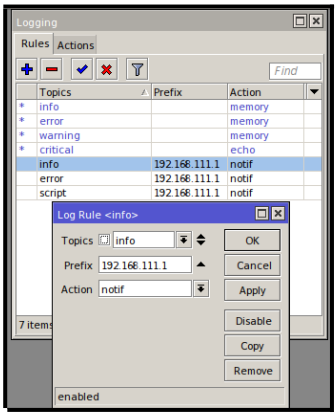
4.2.1 Parameter Limit Bandwidth

Mengkonfigurasi *router mikrotik* pada parameter *Limit Bandwidth* dilakukan dengan menambahkan *actions* dan *rules* pad menu *system* → *logging* untuk menghubungkan *file log* dari *router mikrotik* menuju ke server yang telah dibuat pada tahap sebelumnya. Menu *actions* ditambahi fitur dengan nama "notif", *type* diisi dengan *remote*, dan *remote address* diisi dengan alamat *IP* dari *server*.



Gambar 4.5 Menambahkan Menu *Actions*

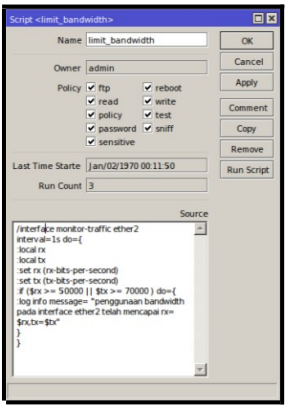
Selanjutnya memasuki *tab rules*, untuk menambahkan *log action* yang telah dibuat sebelumnya ke menu *rules* ini.



Gambar 4.6 Menambahkan Menu *Rules*

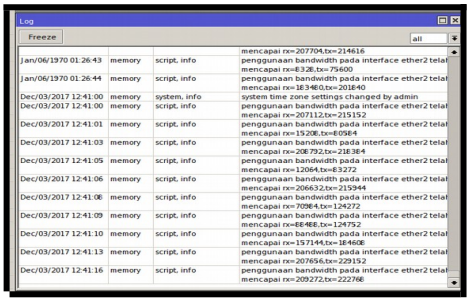
Selanjutnya yaitu menambahkan sebuah *script* pada menu *system* → *scripts* yang berisi *script limit bandwidth*, langkah ini dilakukan karena pada *system logging mikrotik* belum terdapat *log* berisi peringatan batasan (*limit*) dari *bandwidth* yang telah digunakan untuk kemudian dilakukan perintah *run script*.

Perintah yang ditambahkan sebagai berikut.



Gambar 4.7 Menambahkan *Script Limit Bandwidth*

Kemudian pada prosesnya jika terdapat keadaan kapasitas *bandwidth* mencapai batas yang telah ditentukan, maka akan tampil pada *log file* yang nantinya akan dikirimkan ke *device android*.

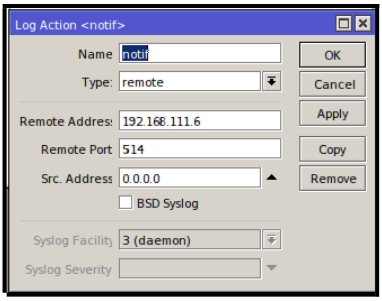


Gambar 4.8 *Log File* Parameter *Limit Bandwidth*

4.2.2 Parameter Perubahan IP

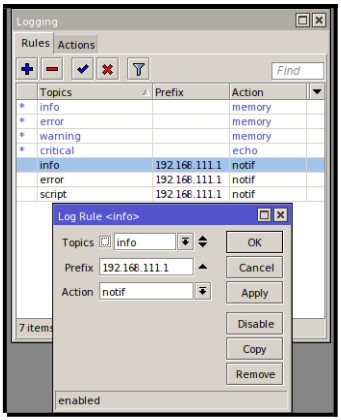
Langkah yang dilakukan untuk mengkonfigurasi *router mikrotik* untuk parameter Perubahan *IP* yaitu dengan menambahkan *actions* dan *rules* pada menu *system* → *logging* untuk menghubungkan *file log* dari *router mikrotik* menuju ke *server* yang telah dibuat pada tahap sebelumnya. Pada menu

actions ditambahkan fitur dengan nama "notif", *type* diisi dengan *remote*, dan *remote address* diisi dengan alamat *IP* dari *server*.



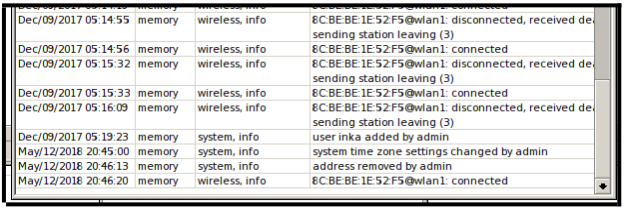
Gambar 4.9 Menambahkan Menu *Actions*

Selanjutnya memasuki *tab rules*, untuk menambahkan *log action* yang telah dibuat sebelumnya ke menu *rules* ini.



Gambar 4.10 Menambahkan Menu *Rules*

Langkah selanjutnya yaitu mengecek adanya perubahan *IP* yang terjadi dapat dicek pada menu *IP* → *IP Address*. Jika terjadi adanya perubahan *IP* oleh *admin* ataupun *user* lain dapat terlihat pada *log file* yang tertera pada gambar 4.11 yang nantinya akan dikirimkan menuju *device android*.

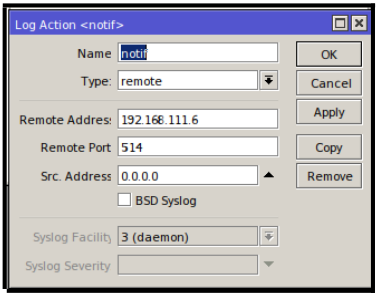


Dec/09/2017 05:14:55	memory	wireless, info	8C:BE:BE:1E:52:F5@wlan1: disconnected, received de... sending station leaving (3)
Dec/09/2017 05:14:56	memory	wireless, info	8C:BE:BE:1E:52:F5@wlan1: connected
Dec/09/2017 05:15:32	memory	wireless, info	8C:BE:BE:1E:52:F5@wlan1: disconnected, received de... sending station leaving (3)
Dec/09/2017 05:15:33	memory	wireless, info	8C:BE:BE:1E:52:F5@wlan1: connected
Dec/09/2017 05:16:09	memory	wireless, info	8C:BE:BE:1E:52:F5@wlan1: disconnected, received de... sending station leaving (3)
Dec/09/2017 05:19:23	memory	system, info	user inka added by admin
May/12/2018 20:45:00	memory	system, info	system time zone settings changed by admin
May/12/2018 20:46:13	memory	system, info	address removed by admin
May/12/2018 20:46:20	memory	wireless, info	8C:BE:BE:1E:52:F5@wlan1: connected

Gambar 4.11 *Log File* Parameter Perubahan IP

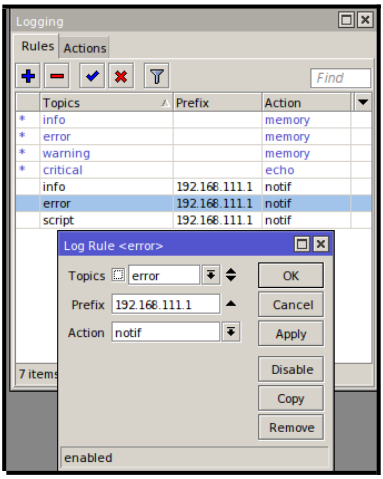
4.2.3 Parameter Informasi Error

Langkah yang dilakukan untuk mengkonfigurasi *router mikrotik* untuk parameter informasi *error* yaitu dengan menambahkan *actions* dan *rules* pada menu *system* → *logging* untuk menghubungkan *file log* dari *router mikrotik* menuju ke server yang telah dibuat pada tahap sebelumnya. Pada menu *actions* ditambahkan fitur dengan nama "notif", *type* diisi dengan *remote*, dan *remote address* diisi dengan alamat *IP* dari *server*.



Gambar 4.12 Menambahkan Menu *Actions*

Selanjutnya menambahkan *log action* yang telah dibuat sebelumnya ke menu *rules* ini.



Gambar 4.13 Menambahkan Menu *Rules*

Jika terjadi adanya informasi *error* dapat terlihat pada *log file* yang tertera pada gambar 4.14 yang nantinya akan dikirimkan menuju *device android*.

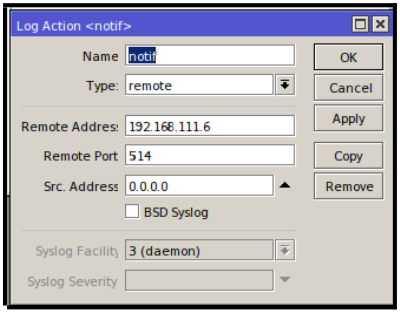
May/13/2018 00:12:19	memory	interface, info	ether3 link up (speed 100M, full duplex)
May/13/2018 00:12:57	memory	interface, info	ether3 link down
May/13/2018 00:12:59	memory	interface, info	ether4 link up (speed 100M, full duplex)
May/13/2018 00:14:33	memory	interface, info	ether4 link down
May/13/2018 00:14:34	memory	interface, info	ether4 link up (speed 100M, full duplex)
May/13/2018 00:15:48	memory	system, info, account	user admin logged in from 192.168.111.6 via winbox
May/13/2018 00:16:03	memory	system, info, account	user admin logged out from 192.168.111.6 via winbox
May/13/2018 00:16:12	memory	system, error, critical	login failure for user admin from 192.168.111.6 via winbox
May/13/2018 00:16:19	memory	system, info, account	user admin logged in from 192.168.111.6 via winbox

Gambar 4.14 *Log File* Parameter Informasi *Error*

4.2.4 Parameter Device Reboot

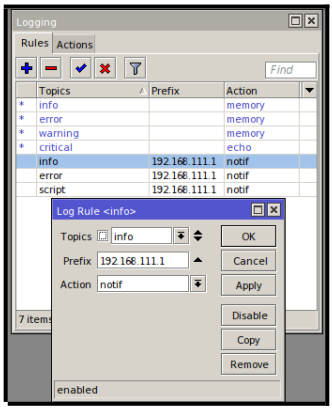
Langkah yang dilakukan untuk mengkonfigurasi *router mikrotik* untuk parameter *device reboot* yaitu dengan menambahkan *actions* dan *rules* pada menu *system* → *logging* untuk menghubungkan *file log* dari *router mikrotik* menuju ke server yang telah dibuat pada tahap sebelumnya. Pada menu *actions* ditambahkan fitur dengan nama "notif", *type* diisi

dengan *remote*, dan *remote address* diisi dengan alamat *IP* dari *server*.



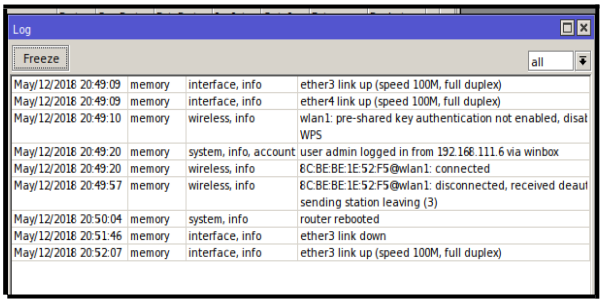
Gambar 4.15 Menambahkan Menu *Actions*

Selanjutnya memasuki *tab rules*, untuk menambahkan *log action* yang telah dibuat sebelumnya ke menu *rules* ini.



Gambar 4.16 Menambahkan Menu *Rules*

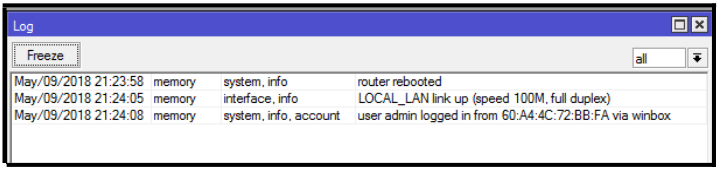
Jika terjadi adanya informasi *device mikrotik* melakukan *reboot* dapat terlihat pada *log file* yang tertera pada gambar 4.17 yang nantinya akan dikirimkan menuju *device android.*



Gambar 4.17 *Log File* Parameter *Device Reboot*

Namun terdapat kekurangan pada parameter *device reboot* ini yaitu pada *device router mikrotik* A dapat memunculkan *log* berupa "*router rebooted*"dan mengirimkannya ke dalam *database MySQL*, namun pada *device router mikrotik* B *log* "*router rebooted*" tidak dapat dikirimkan menuju *database MySQL*.

Hal ini terjadi karena perbedaan karakteristik beberapa *router mikrotik* dalam hal pencatatan *log activity*-nya, khususnya dalam kasus *reboot device* mayoritas *mikrotik* mengirimkan *log* "*router rebooted*" sebelum *device* menyala (*link up*) sedangkan proses pengiriman *log file* dari perangkat *router mikrotik* membutuhkan daya (setelah *link up*) sebagai contoh pada *mikrotik* B yang *log file*-nya dapat dilihat pada gambar 4.18.

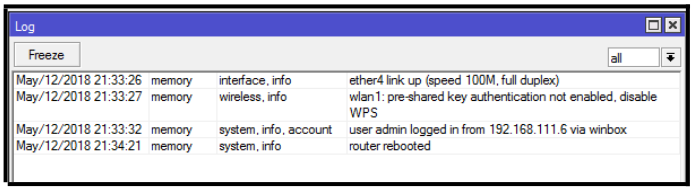


Gambar 4.18 *Log Reboot Device Mikrotik* B

Pada skema ini *log* mengenai *reboot* tidak dapat terkirimkan karena sistem yang dirancang dengan menggunakan

RSYSLOG ini hanya memungkinkan mengirimkan file *log mikrotik* setelah *device* menyala (*link up*).

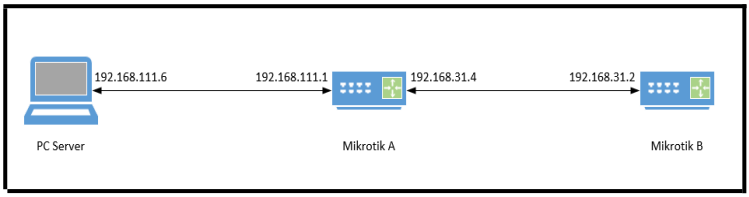
Jadi pada penelitian ini untuk kasus parameter *reboot* hanya dapat mengirimkan *log* dari *mikrotik* A saja karena pada *device* tersebut karakteristik pengiriman *log reboot*-nya terjadi setelah *device* menyala (*link up*) seperti pada gambar 4.19.



Gambar 4.19 *Log Reboot Device Mikrotik* A

4.2.5 Menghubungkan Dua Device Router Mikrotik

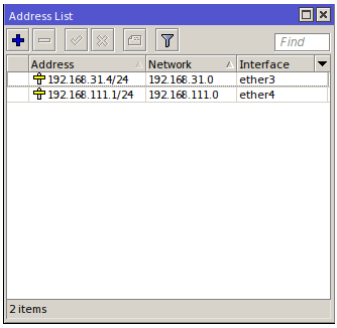
Untuk menghubungkan dua buah *device router mikrotik* dibutuhkan langkah *routing* yang disertai proses *port forwarding* agar kedua *device router mikrotik* tersebut dapat mengirimkan data *log* menuju ke komputer *server* dengan gambaran sebagai berikut.



Gambar 4.20 Skema Topologi Jaringan

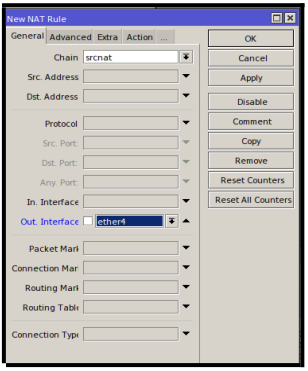
4.2.5.1 Konfigurasi Router Mikrotik A

Langkah-langkah konfigurasi pada *device router mikrotik A* dimulai dengan menambahkan *IP address* 192.168.111.1/24 pada *interface ethernet* 4 yang terhubung ke *PC Server* dan *IP* 192.168.31.4/24 pada *interface ethernet* 3 sehingga menjadi seperti gambar berikut.

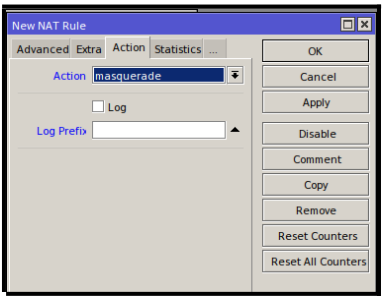


Gambar 4.21 Menambahkan *IP Address*

Selanjutnya yaitu menambahkan fungsi *masqurade* pada *NAT firewall* dengan masuk ke menu *IP* → *firewall*, pada tab *general* isikan *chain srcnat* dan *output interface* pada *ethernet 3* dan *ethernet 4*. Kemudian memasuki tab *action* pilih *masquerade*



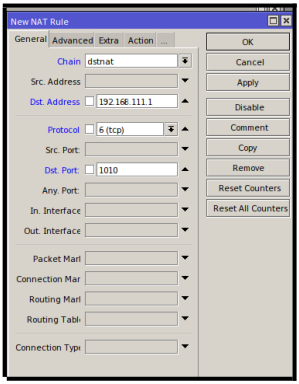
Gambar 4.22 Konfigurasi *Masquerade* Bagian Pertama



Gambar 4.23 Konfigurasi *Masquerade* Bagian Kedua

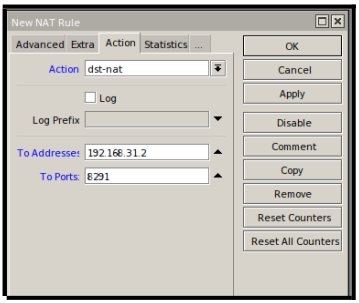
Langkah selanjutnya yaitu menambahkan *port forwarding* pada *mikrotik A* agar dapat menhubungkan *mikrotik B* menuju ke *PC Server*.

Tambahkan *chain dstnat*, *dst address* isikan *IP mikrotik A* sebagai penghubungnya dengan *port* 1010.



Gambar 4.24 Konfigurasi *Port Forwarding* Bagian Pertama

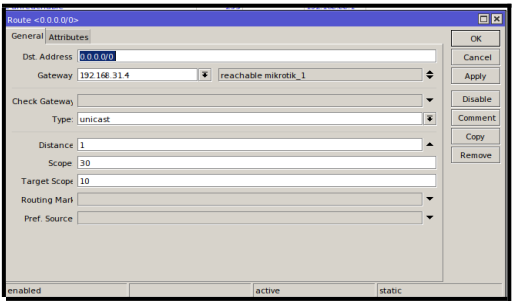
Tambahkan *action dst-nat* menuju *IP address mikrotik B* 192.168.31.2 dengan *port default winbox* yaitu 8291.



Gambar 4.25 Konfigurasi *Port Forwarding* Bagian Kedua

4.2.5.2 Konfigurasi Router Mikrotik B

Pada bagian ini akan dijelaskan mengenai langkah-langkah konfigurasi pada *device router mikrotik B* dimulai dengan menambahkan *route list* pada menu *IP→routes* dengan mengisikan *Dst. Address* 0.0.0.0/0 dengan *gateway* 192.168.31.4 sebagai *IP mikrotik A*.

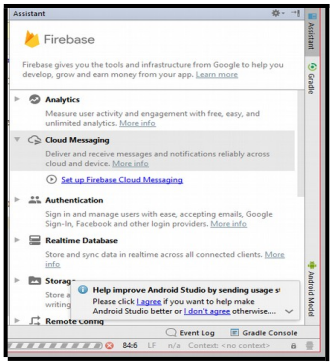


Gambar 4.26 *Menambah* Route List

Konfigurasi Firebase

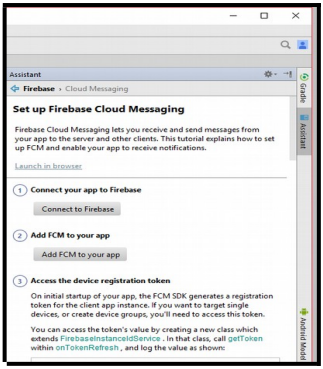
Pada tahap ini akan dilakukan proses konfigurasi *firebase cloud messaging* pada *project android* yang akan dibuat. Firebase cloud messaging ini merupakan perantara antara *server* dengan *device* yang akan didaftarkan untuk menerima *push notification*. Langkah pertama daftarkan project yang akan

dibuat pada android stuido, masuk ke menu tools → firebase kemudian akan muncul jendela seperti di gambar.



Gambar 4.27 *Tools Firebase* pada *Android Studio*

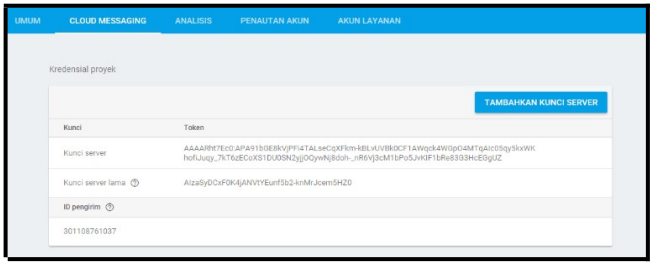
Selanjutnya pilih fitur set-up firebase cloud messaging*, dan akan muncul jendela baru pada gambar 4.28. Pilih* connect to firebase *dan, kemudian pilih* add FCM to your app. *Hingga langkah ini selesai* project android *yang akan dibuat telah terhubung ke* firebase.



Gambar 4.28 *Menu* Firebase Cloud Messaging *(*FCM*)*

Selanjutnya yaitu masuk ke *website firebase* untuk mengambil *firebase token* yang dibutuhkan pada *server*.

```
foreach ($listPesan as $value) {

        $syslogtag = $value->kategori;
        $message = $value->message;

        if (strpos($syslogtag, 'error') !== false) {
          $id = $value->id;
          $judul = $value->judul;
          $message = $value->message;
          $device = $value->device;
          $date = $value->create_at;
          $notif = notifikasi($listerror, $judul , $message, $device, $date);

          if($notif->success > 0){
            $updatepesan = $this->mPesan->updatePesan($id);
          }

        }elseif (strpos($syslogtag, 'script') !== false) {
          if (strpos($message, 'penggunaan bandwidth') !== false) {
            $id = $value->id;
            $judul = $value->judul;
            $message = $value->message;
            $device = $value->device;
            $date = $value->create_at;
            $notif = notifikasi($listbandwidth, $judul , $message, $device, $date);

            if($notif->success > 0){
              $updatepesan = $this->mPesan->updatePesan($id);
            }
          }

           }elseif ( strpos($message, 'address added') !== false || strpos($message,
'address removed') !== false) {
            $id = $value->id;
            $judul = $value->judul;
            $message = $value->message;
            $device = $value->device;
            $date = $value->create_at;
            $notif = notifikasi($listadddevice, $judul , $message, $device, $date);

            if($notif->success > 0){
              $updatepesan = $this->mPesan->updatePesan($id);
            }

           }elseif ( strpos($message, 'router reboot') !== false || strpos($message,
'router was reboot') !== false) {
            $id = $value->id;
            $judul = $value->judul;
            $message = $value->message;
            $device = $value->device;
```



Gambar 4.29 *Firebase Token* pada Menu *Cloud Messaging*

4.3 Parsing Log Mikrotik

Log mikrotik tentunya akan mengirimkan semua isi dari *log* menuju ke server melalui *remote syslog*. Namun hanya data yang diperlukan akan masuk ke dalam *database SQL* untuk seterusnya dikirim ke *device user*. Untuk memilah data-data yang dibutuhkan dibutuhkan proses *parsing* yang terdapat pada fungsi *PHP* seperti pada modul berikut ini.

Modul Program 4.6 Fungsi *PHP Parsing Data Log Mikrotik*

```
public class CheckServer extends AppCompatActivity {    EditText etIp;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_check_server);
        etIp = (EditText)findViewById(R.id.ip);
    }
    public void konekServer(View arg){
        final String ip = etIp.getText().toString().trim();
        String ipPattern = "[0-9]+\\.+[0-9]+\\.+[0-9]+\\.+[0-9]+";
        if (ip.matches(ipPattern)){
            URL url = null;
            HttpURLConnection connection = null;
            BuatKonfigurasi bk = new BuatKonfigurasi(this);
            int chekConfigIp = bk.checkValue("ip");
            Log.d("Test config ip", Integer.toString(chekConfigIp));
            if (chekConfigIp == 0) {
                bk.addConfig("ip", ip);
            }else{
                bk.updateConfig("ip", ip);
            }

            String ipNew = bk.getValue("ip");
```

Modul Program 4.7 Lanjutan Fungsi *PHP Parsing Data Log Mikrotik*

```
public class CheckServer extends AppCompatActivity {

EditText etIp;


   @Override
   protected void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.activity_check_server);
      etIp = (EditText)findViewById(R.id.ip);
   }
   public void konekServer(View arg){
      final String ip = etIp.getText().toString().trim();
      String ipPattern = "[0-9]+\\.+[0-9]+\\.+[0-9]+\\.+[0-9]+";
      if (ip.matches(ipPattern)){
         URL url = null;
         HttpURLConnection connection = null;
         BuatKonfigurasi bk = new BuatKonfigurasi(this);
         int chekConfigIp = bk.checkValue("ip");
         Log.d("Test config ip", Integer.toString(chekConfigIp));
         if (chekConfigIp == 0) {
            bk.addConfig("ip", ip);
         }else{
            bk.updateConfig("ip", ip);
         }

         String ipNew = bk.getValue("ip");
         try{
            url = new URL("http://"+ipNew+"/mikrotik/index.php/register");
```

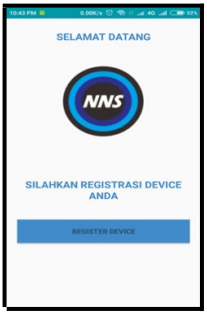
Penjelasan dari modul program 4.6 dan 4.7 sebagai cara *parsing* pesan yang berasal dari *log mikrotik* yaitu sistem akan membaca daftar dari tabel pesan, kemudian dideklarasikan *$syslogtag* sebagai data "kategori" dan *$message* sebagai data "*message*" pada tabel pesan. Jika sistem membaca kata "*error*" pada data "kategori" dan mengambil data yang dibutuhkan yaitu *id*, *judul*, *message*, *device*, dan *create_at* untuk kemudian

dikirimkan menuju *firebase* berupa data *json*. Begitu pula dengan kategori pesan notifikasi lainnya yaitu penggunaan *bandwidth*, perubahan *device*, dan *reboot device* yang termasuk ke dalam kategori *$syslogtag* berupa kata "*script*" kemudian mencari data pesan berupa "penggunaan *bandwidth*" untuk notifikasi *limit bandwidth*, "*address added*" atau "*address removed*" untuk notifikasi perubahan *device*, dan "*router reboot*" untuk kategori notifikasi *reboot device*.

4.4 Simulasi Sistem

Pada bagian ini akan dijelaskan mengenai langkah-langkah dari proses berjalannya sistem mulai dari *parsing* data *log mikrotik*, berikut penjelasannya:

a) *Router mikrotik* akan terus mengirimkan file log jika terdapat aktivitas menuju ke *server* dengan menggunakan *remote syslog* yang telah diterangkan sebelumnya.
b) Selanjutnya *user* melakukan registrasi pada *device android* dengan urutan sebagai berikut.



Gambar 4.30 Halaman Awal Aplikasi

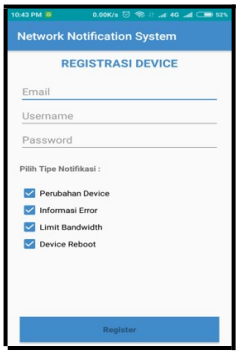
Pada saat masuk ke aplikasi u*ser* akan disuguhi tampilan awal berupa logo aplikasi dengan tombol "register device", klik tombol tersebut untuk masuk ke halaman berikutnya.

Gambar 4.31 Halaman Koneksi *Server*

Pada bagian ini *user* harus memasukkan alamat *IP* dari *server* agar dapat terkoneksi untuk menuju langkah selanjutnya. *Source code* dari halaman koneksi *server* dapat dilihat sebagai berikut.

Modul Program 4.8 Fungsi Koneksi *Server*

Penjelasan dari modul program 4.8 yaitu cara mengkoneksikan aplikasi *android* menuju *server* dengan cara mencocokkan alamat *IP* yang diketikkan pada aplikasi dengan *server*. Jika alamat *IP* cocok maka aplikasi berlanjut ke halaman registrasi, namun jika gagal akan muncul peringatan berupa format *IP* salah atau gagal konek ke *server*. Berikut ini merupakan tampilan halaman registrasi setelah proses koneksi *server* berhasil.



Gambar 4.32 Halaman Registrasi

Pada bagian ini *user* melakukan registrasi dengan meng*input* data dan kebutuhan tipe notifikasi dengan mencentang pilihan yang tersedia. Jika registrasi berhasil, data akan tersimpan ke dalam *database*. Namun jika gagal, muncul pesan *error* sesuai letak kesalahannya, berikut ini *source code* dari halaman registrasi untuk memasukkan data ke dalam *database*.

Modul Program 4.9 Fungsi Proses Registrasi

Penjelasan dari modul program 4.9 pada bagian atas merupakan fungsi data yang diterima setelah proses registrasi

```
    public function insertUsers(){
                        $email = $this->input->post("email");
                        $password = password_hash($this->input->post('password'),
PASSWORD_BCRYPT);
                        $username = $this->input->post("username");
                        $reboot = $this->input->post("reboot");
                        $adddevice = $this->input->post("addDevice");
                        $error = $this->input->post("error");
    $bandwidth = $this->input->post("bandwidth");
    $token = $this->input->post("token");
                        $create_at = date('Y-m-d H:i:s');
        $update_on = date('Y-m-d H:i:s');
                $this->db->set("email", $email);


                        $this->db->set("add_device", $adddevice);
                        $this->db->set("error", $error);
                        return ($this->db->affected_rows() != 1) ? false : true;
    }
```

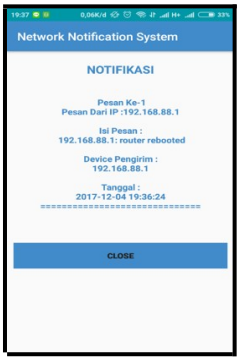
berhasil dilakukan, sementara untuk bagian bawah merupakan fungsi untuk mengisi data ke dalam *database*.

c) Setelah berhasil registrasi, *Firebase Cloud Messaging* (*FCM*) akan melakukan *generate token* sesuai *device* yang didaftarkan dan memasukkan data beserta *token* ke dalam *database*.

d) Ketika ada pesan yang tersedia dalam *database* tabel pesan, *server* akan mengambil data tersebut dan mengirimnya ke *device android* melalui *FCM*.

e) Pesan notifikasi berhasil diterima oleh *device user* seperti contoh gambar 4.33.

f)



Gambar 4.33 Halaman Tampil Pesan

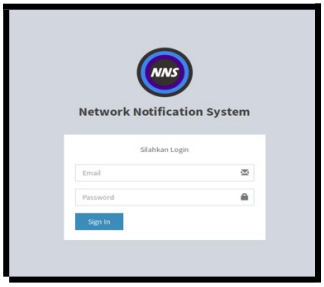
g) Data pesan yang berhasil diterima oleh *user*, akan dapat dikelola atau di*review* kembali melalui halaman *web admin*.

4.5 Halaman Kelola Web Admin

Berikut penjelasan tahapan-tahapan *admin* dalam melihat data pesan yang masuk mengelola data pesan maupun data *users*.

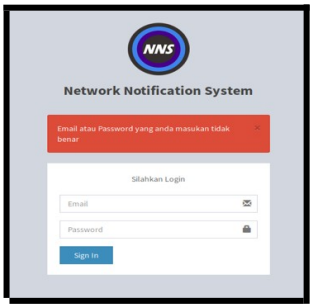
a) Langkah pertama yang dilakukan yaitu melakukan proses *login* dengan mengisi data *email*, dan *password* yang terdaftar.

b)



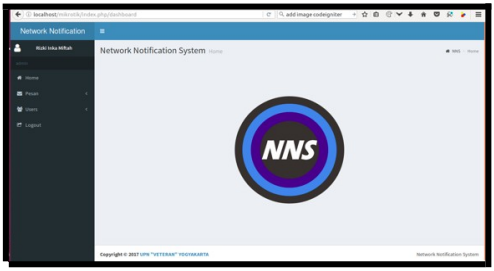
Gambar 4.34 Halaman *Login*

c) Namun jika proses *login* gagal dikarenakan data yang salah, maka akan muncul seperti ini.



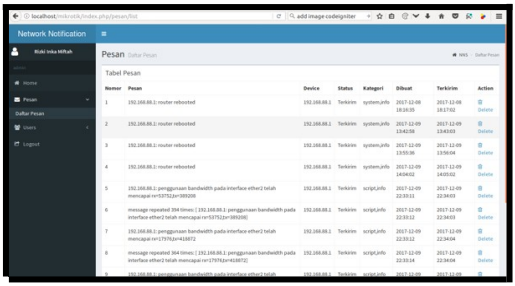
Gambar 4.35 Proses *Login* Gagal

d) Jika berhasil maka akan masuk ke halaman awal dari *web admin* seperti gambar berikut ini.



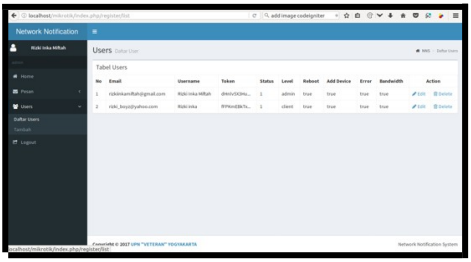
Gambar 4.36 Halaman Awal

e) Untuk melihat daftar pesan yang telah dikirimkan ke *device user*, pilih menu pesan.



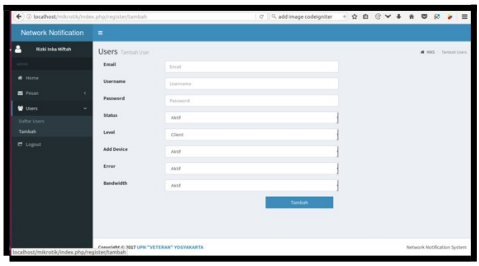
Gambar 4.37 Halaman Daftar Pesan

f) **Data users dapat d**ilihat dan diubah datanya dengan memlih menu users, lalu daftar users untuk menampilkan data users tersedia beserta tombol edit untuk mengubah data dan tombol delete untuk menghapus pesan.



Gambar 4.38 Halaman Daftar *Users*

g) Selanjutnya terdapat fitur tambah *users* untuk menambahkan *user* baru untuk menerima pesan notifikasi.

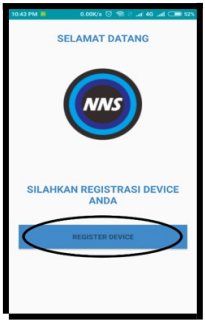


Gambar 4.39 Halaman Tambah *Users*

4.6 Pengujian Fungsionalitas Aplikasi Network Notification System

Uji coba fungsionalitas ini berfungsi untuk menguji aplikasi apakah berjalan dengan baik atau tidak. Kemudian dilakukan evaluasi terhadap kesalahan atau kekurangan pada aplikasi. Untuk menguji fungsionalitas dari aplikasi ini dilakukan sesuai langkah-langkah berikut.

1. Jalankan aplikasi *Network Notification System* di *android*, klik menu *register device* untuk selanjutnya masuk ke halaman koneksi *server*.

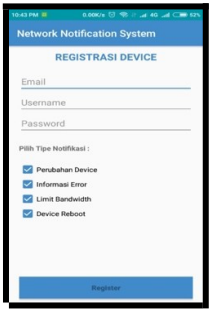


Gambar 4.40 Halaman Awal Aplikasi

2. Setelah masuk ke halaman koneksi *server*, masukkan *IP address* dari *server*.

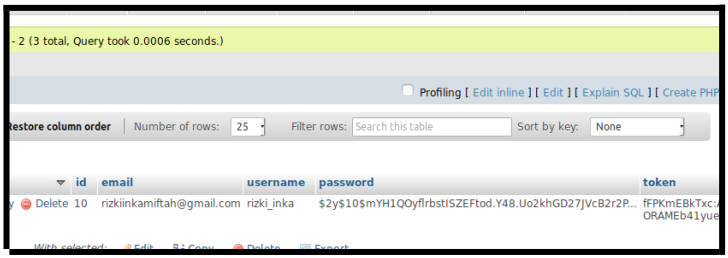
Gambar 4.41 Halaman Koneksi *Server*

3. Jika berhasil *connect*, selanjutnya berganti menuju halaman registrasi, isi data dan kebutuhan notifkasi yang diinginkan berupa empat buah parameter.



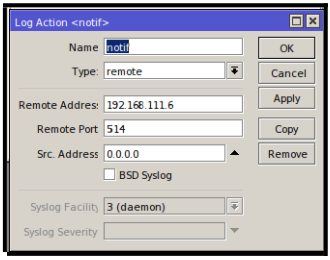
Gambar 4.42 Halaman Registrasi

4. Jika berhasil maka muncul halaman sukses registrasi, namun jika belum berhasil akan muncul peringatan.
5. Data hasil registrasi dapat dicek pada *database MySQL* maupun halaman *web admin*.



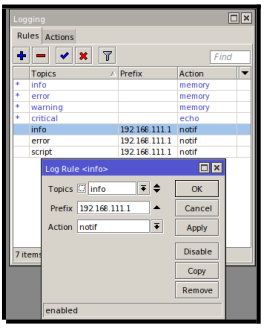
Gambar 4.43 Data Tabel *Users*

6. Pengujian proses mengirim *log file* dari *router mikrotik* menuju ke *database MySQL* yang pertama yaitu untuk menguji pengiriman *log file* parameter *limit bandwidth*, langkah yang dilakukan untuk mengkonfigurasi *router mikrotik* pada parameter *Limit Bandwidth* dengan menambahkan *actions* dan *rules* pada menu *system* → *logging* untuk menghubungkan *file log* dari *router mikrotik* menuju ke server yang telah dibuat pada tahap sebelumnya. Pada menu *actions* ditambahkan fitur dengan nama "notif", *type* diisi dengan *remote*, dan *remote address* diisi dengan alamat *IP* dari *server*.



Gambar 4.44 Menambahkan Menu *Actions*

Selanjutnya memasuki *tab rules*, untuk menambahkan *log action* yang telah dibuat sebelumnya ke menu *rules* ini.



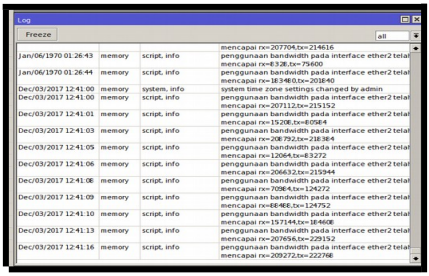
Gambar 4.45 Menambahkan Menu *Rules*

Selanjutnya yaitu menambahkan sebuah *script* pada menu *system* → *scripts* yang berisi *script limit bandwidth*, langkah ini dilakukan dikarenakan pada *system logging mikrotik* belum terdapat *log* berisi peringatan batasan (*limit*) dari *bandwidth* yang telah digunakan untuk kemudian dilakukan perintah *run script*.

Perintah yang ditambahkan sebagai berikut.

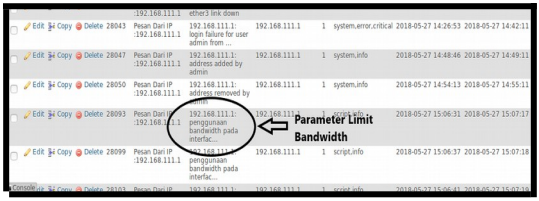
Gambar 4.46 Menambahkan *Script Limit Bandwidth*

Kemudian pada prosesnya jika terdapat keadaan kapasitas *bandwidth* mencapai batas yang telah ditentukan, maka akan tampil pada *log file* yang nantinya akan dikirimkan ke *device android*.



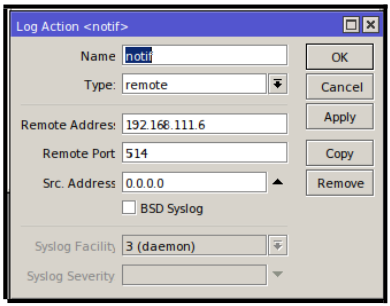
Gambar 4.47 *Log File* Parameter *Limit Bandwidth*

Selanjutnya dicek pada *database MySQL* maka akan muncul pesan berupa parameter *limit bandwidth.*



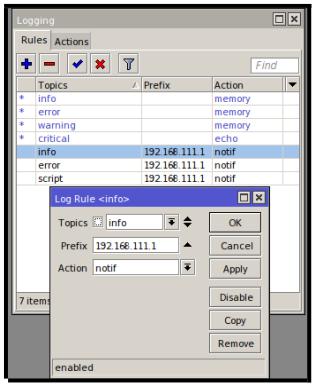
Gambar 4.48 Data Tabel Pesan Parameter *Limit Bandwidth*

7. Untuk parameter kedua yaitu perubahan *IP* langkah yang dilakukan untuk mengkonfigurasi *router mikrotik* untuk parameter Perubahan *IP* yaitu dengan menambahkan *actions* dan *rules* pada menu *system* → *logging* untuk menghubungkan *file log* dari *router mikrotik* menuju ke server yang telah dibuat pada tahap sebelumnya. Pada menu *actions* ditambahkan fitur dengan nama "notif", *type* diisi dengan *remote*, dan *remote address* diisi dengan alamat *IP* dari *server*.



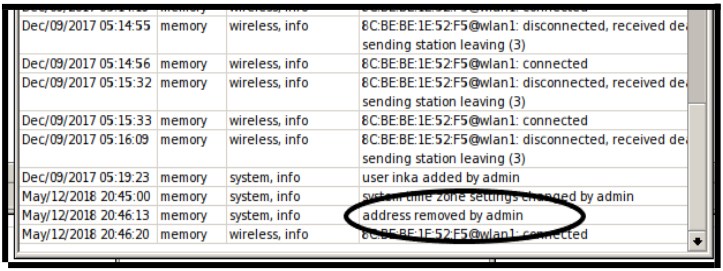
Gambar 4.49 Menambahkan Menu *Actions*

Pada *tab rules*, tambahkan *log action* yang telah dibuat sebelumnya ke menu *rules* ini.



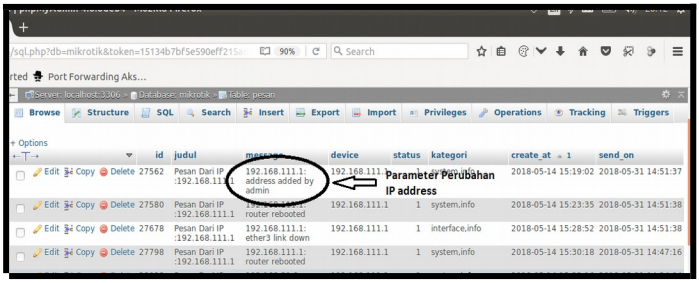
Gambar 4.50 Menambahkan Menu *Rules*

Selanjutnya yaitu mengecek jika adanya perubahan *IP* yang terjadi dapat dicek pada menu *IP* → *IP Address*. Jika terjadi adanya perubahan *IP* oleh *admin* ataupun *user* lain dapat terlihat pada *log file* yang nantinya akan dikirimkan menuju *device android*.



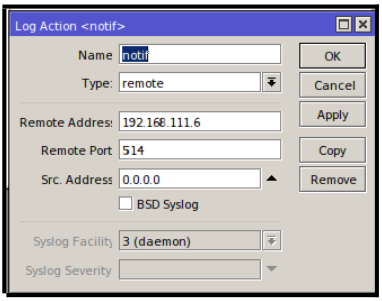
Gambar 4.51 *Log File* Parameter Perubahan *IP*

Selanjutnya dicek pada *database MySQL* maka akan muncul pesan berupa parameter Perubahan *IP*.



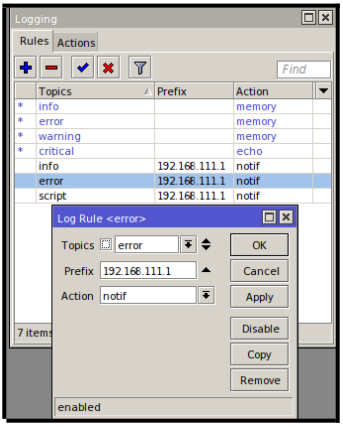
Gambar 4.52 Data Tabel Pesan Parameter Perubahan *IP*

8. Parameter ketiga yaitu informasi *error* langkah yang dilakukan untuk mengkonfigurasi *router mikrotik* untuk parameter informasi *error* yaitu dengan menambahkan *actions* dan *rules* pada menu *system* → *logging* untuk menghubungkan *file log* dari *router mikrotik* menuju ke server yang telah dibuat pada tahap sebelumnya. Pada menu *actions* ditambahkan fitur dengan nama "notif", *type* diisi dengan *remote*, dan *remote address* diisi dengan alamat *IP* dari *server*.



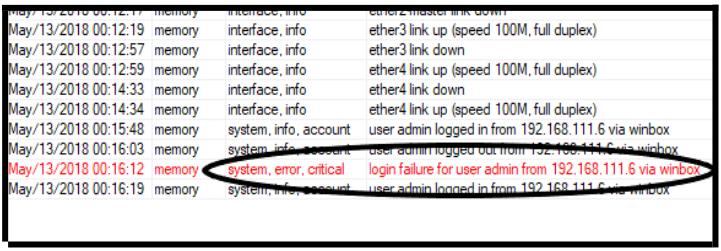
Gambar 4.53 Menambahkan Menu *Actions*

Selanjutnya memasuki *tab rules*, untuk menambahkan *log action* yang telah dibuat sebelumnya ke menu *rules* ini.



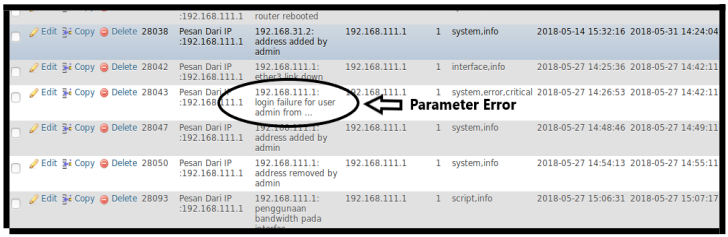
Gambar 4.54 Menambahkan Menu *Rules*

Jika terjadi adanya informasi *error* dapat terlihat pada *log file* yang nantinya akan dikirimkan menuju *device android*.



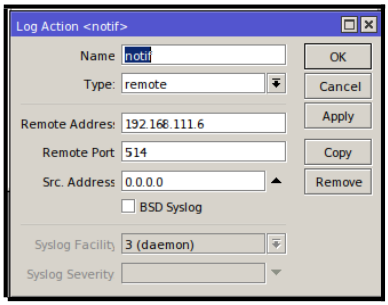
Gambar 4.55 *Log File* Parameter Informasi *Error*

Selanjutnya dicek pada *database MySQL* maka akan muncul pesan berupa parameter informasi *error*.



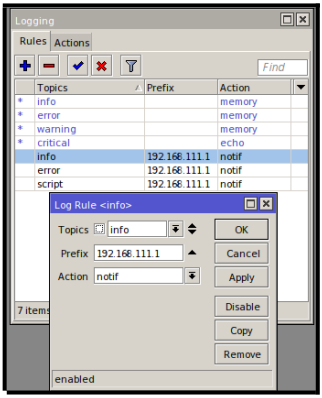
Gambar 4.56 Data Tabel Pesan Parameter *Error*

9. Parameter keempat yaitu *reboot device* langkah yang dilakukan untuk mengkonfigurasi *router mikrotik* untuk parameter *device reboot* yaitu dengan menambahkan *actions* dan *rules* pada menu *system* → *logging* untuk menghubungkan *file log* dari *router mikrotik* menuju ke server yang telah dibuat pada tahap sebelumnya. Pada menu *actions* ditambahkan fitur dengan nama "notif", *type* diisi dengan *remote*, dan *remote address* diisi dengan alamat *IP* dari *server*.



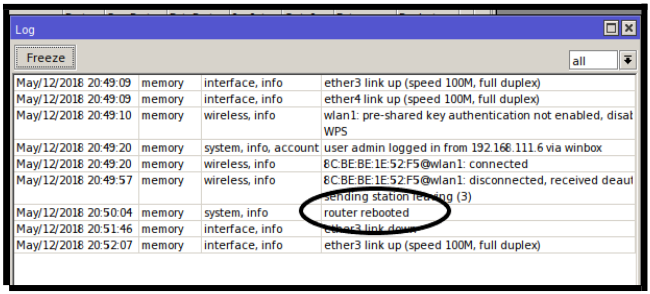
Gambar 4.57 Menambahkan Menu *Actions*

Selanjutnya memasuki *tab rules*, untuk menambahkan *log action* yang telah dibuat sebelumnya ke menu *rules* ini.



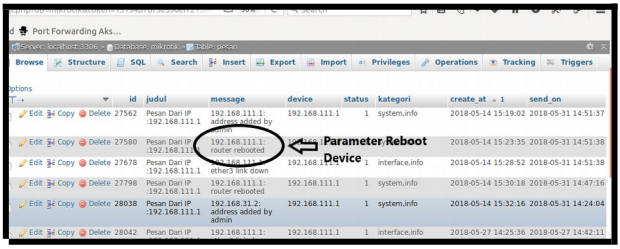
Gambar 4.58 Menambahkan Menu *Rules*

Jika terjadi *device mikrotik* melakukan *reboot* dapat terlihat pada *log file* yang tertera pada gambar 4.17 yang nantinya akan dikirimkan menuju *device android.*



Gambar 4.59 *Log File* Parameter *Reboot Device*

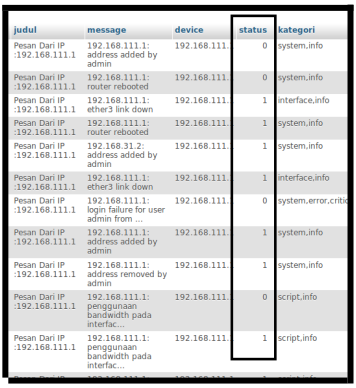
Selanjutnya dicek pada *database MySQL* maka akan muncul pesan berupa parameter informasi *error*.



Gambar 4.60 Data Tabel Pesan Parameter *Reboot Device*

Begitu pula dilakukan percobaan pengiriman pada perangkat *router mikrotik* kedua dengan parameter yang sama.

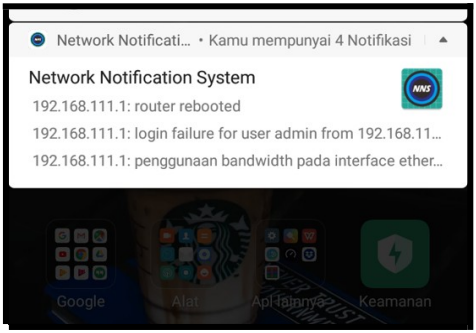
10. Selanjutnya dilakukan proses pengiriman data dari *database MySQL* menuju ke perangkat *android*. Jadi, pada proses ini *server* akan terus menjalankan sebuah URL (*http://localhost/mikrotik/pesan/sendpesan*) setiap menitnya dengan bantuan *crontab* untuk mengirim pesan dalam bentuk *array*. Dalam fungsi *sendpesan*, status "0" berarti pesan akan dikirimkan dan jika statusnya menjadi "1", berarti tidak akan dikirimkan kembali.



judul	message	device	status	kategori
Pesan Dari IP :192.168.111.1	192.168.111.1: address added by admin	192.168.111.	0	system,info
Pesan Dari IP :192.168.111.1	192.168.111.1: router rebooted	192.168.111.	0	system,info
Pesan Dari IP :192.168.111.1	192.168.111.1: ether3 link down	192.168.111.	1	interface,info
Pesan Dari IP :192.168.111.1	192.168.111.1: router rebooted	192.168.111.	1	system,info
Pesan Dari IP :192.168.111.1	192.168.31.2: address added by admin	192.168.111.	1	system,info
Pesan Dari IP :192.168.111.1	192.168.111.1: ether3 link down	192.168.111.	1	interface,info
Pesan Dari IP :192.168.111.1	192.168.111.1: login failure for user admin from ...	192.168.111.	0	system,error,criti
Pesan Dari IP :192.168.111.1	192.168.111.1: address added by admin	192.168.111.	1	system,info
Pesan Dari IP :192.168.111.1	192.168.111.1: address removed by admin	192.168.111.	1	system,info
Pesan Dari IP :192.168.111.1	192.168.111.1: penggunaan bandwidth pada interfac...	192.168.111.	0	script,info
Pesan Dari IP :192.168.111.1	192.168.111.1: penggunaan bandwidth pada interfac...	192.168.111.	1	script,info

Gambar 4.61 Status Tabel Pesan Terkirim Data

11. Pesan notifikasi akan muncul pada layar *popup* *android* berupa jumlah pesan dan *preview* pesan yang akan ditampilkan.



Gambar 4.62 *Popup* Notifikasi Pesan *Android*

12. Klik pesan notifikasi, maka akan muncul halaman *display* pesan. Berikut tampilan pesan dari parameter *limit bandwidth* dan parameter informasi *error*.

Gambar 4.63 Halaman Tampil Pesan 1

Sedangkan gambar berikut merupakan tampilan pesan dari parameter perubahan *IP* dan *reboot device* pada perangkat *android*.

Gambar 4.64 Halaman Tampil Pesan 2

Dari rangkaian percobaan pengujian fungsionalitas yang telah dilakukan, fungsi-fungsi yang terdapat pada aplikasi *Network Notification System* berjalan dengan baik, mulai dari proses registrasi, pengiriman *log file* dari empat buah parameter

pada perangkat *router mikrotik* menuju ke dalam *database* hingga sukses mengirimkan pesan notifikasi menuju perangkat *android*.

Namun terdapat kendala pada parameter reboot device di router mikrotik kedua yang tidak dapat terkirimkan. Hal ini dikarenakan terdapat perbedaan karakteristik beberapa *router mikrotik* dalam hal pencatatan *log activity*-nya, khususnya dalam kasus *reboot device* mayoritas *mikrotik* mengirimkan *log* "*router rebooted*" sebelum *device* menyala (*link up*) sedangkan proses pengiriman *log file* dari perangkat *router mikrotik* membutuhkan daya (setelah *link up*). Pada umumnya aplikasi *Network Notification System* telah diuji berjalan dengan baik untuk memantau kondisi *router mikrotik* secara optimal.

DAFTAR PUSTAKA

Darmawan, E., Santoso, S., 2017. Perancangan dan Pembuatan Sistem Pengumuman Akademis Berbasis Tag Menggunakan REST Web Service. ULTIMA INFOSYS 8, 48–53.

Diaz, R.A.N., Novianti, K.D.P., Parasu, I.B.E., 2017. Mobile Application Untuk Traffic Monitoring Wilayah Provinsi Bali. E-Proc. KNSI STIKOM Bali 67–72.

Fatria, F.B., 2011. Pengembangan Fitur Nagios Untuk Pemantauan Jaringan Berbasis Sms (Short Message Service). Universitas Islam Negeri Sultan Syarif Kasim Riau.

Fiade, A., Maula, A.A., Suseno, H.B., 2013. Aplikasi Monitoring Jaringan Berbasis Mobile Web dengan Sistem Notifikasi Berbasis SMS Gateway. J. Tek. Inform. 6.

Himmi, M.H.S.A.M., Abidin, S., 2015. Rancang Bangun Aplikasi Monitoring Network Berbasis Web Menggunakan HTML5 Pada Dinas Pendidikan Kabupaten Blitar. J. Mhs. Fak. Sains Dan Teknol.

Kadir, A., 2014. From Zero to A Pro: Pemrograman Aplikasi Android. Penerbit ANDI.

Kamto, K., Putra, T.W.A., 2015. Pemantauan Traffic Jaringan VPN-MPLS WAN (Studi Kasus: PT PLN (Persero) Distribusi Jawa Tengah Dan Daerah Istimewa Yogyakarta). J. Teknol. Inf. Dan Komun. 6, 15–24.

KUSUMA, F.I., 2015a. Perancangan Sistem Monitoring Perangkat Jaringan Berbasis SNMP. Universitas Muhammadiyah Surakarta.

Moroney, L., 2017. Firebase Cloud Messaging, in: The Definitive Guide to Firebase. Springer, pp. 163–188.

Nurzam, F., Fajri, I.N., Prabowo, D., 2017. Rancang Bangun Aplikasi Media Laporan Aspirasi dengan Firebase Cloud Messaging Berbasis Mobile. SEMNASTEKNOMEDIA online 5, 4–5.

Ohara, G.J., TELKOM, S.T.T., 2005. Aplikasi Sistem Monitoring Berbasis Web Untuk Open Cluster. Bdg. Sekol. Tinggi Teknologi TELKOM.

Peter Mell, T.G., 2011. The NIST Definition of Cloud Computing.

Pradikta, R., Affandi, A., Setijadi, E., 2013. Rancang Bangun Aplikasi Monitoring Jaringan dengan Menggunakan Simple Network Management Protocol. J. Tek. ITS 2, A154–A159.

Prayoga, F., 2016. Perancangan Prototype Aplikasi Pengumuman Kelas Menggunakan Teknologi Firebase Cloud Message pada Android. Program Studi Teknik Informatika FTI-UKSW.

Riza, T.A., 2011. Implementasi Manajemen Traffic Dan Bandwidth Internet Dengan IPCOP. J. INKOM 4, 22–28.

Sagita, A., Prasetijo, A.B., Isnanto, R.R., 2011. Aplikasi Pemantau Statistik Layanan-Layanan TCP dan UDP Berbasiskan SNMP++ Pada Sebuah Jaringan Area Lokal. Jurusan Teknik Elektro Fakultas Teknik UNDIP.

Schmuller, J., 1999. Understanding the Foundations of the UML. Sams publishing.

Sumardi, R., 2017. Aplikasi Mobile Notification Informasi Perkuliahan Berbasis Android. STMIK AKAKOM Yogyakarta.

Terplan, K., 1995. Benchmarking for effective network management. McGraw-Hill, Inc.

Wun, Y.A.O., Sukmaaji, A., Jatmika, K., 2014. Monitoring Trafik Jaringan Dan Pengaturan Pc Router Berbasis Web (Studi Kasus: Labkom Stikom Surabaya). J. JSIKA 3, 212–221.

Manajemen Jaringan Menggunakan Firebase Cloud Massaging Berbasis Android Teori dan Praktek

Disusun Oleh : Dessyanto Boedi Prasetyo, Rizki Inka Miftah, Rifki Indra Perwira

Penerbit LPPM UPN "Veteran" Yogyakarta

ISBN 978-602-5534-68-3

REPUBLIK INDONESIA
KEMENTERIAN HUKUM DAN HAK ASASI MANUSIA

SURAT PENCATATAN
CIPTAAN

Dalam rangka pelindungan ciptaan di bidang ilmu pengetahuan, seni dan sastra berdasarkan Undang-Undang Nomor 28 Tahun 2014 tentang Hak Cipta, dengan ini menerangkan:

Nomor dan tanggal permohonan	:	EC00201975107, 9 Oktober 2019
Pencipta		
Nama	:	**DESSYANTO BOEDI PRASETYO, RIZKI INKA MIFTAH, , dkk**
Alamat	:	Perum Griya Taman Asri Blok G-301 Rt.001/Rw.035 Kel./Desa Donoharjo, Kec. Ngaglik, Kab. Sleman, D.I. Yogyakarta, Sleman, Di Yogyakarta, 55581
Kewarganegaraan	:	Indonesia
Pemegang Hak Cipta		
Nama	:	**DESSYANTO BOEDI PRASETYO, RIZKI INKA MIFTAH, , dkk**
Alamat	:	Perum Griya Taman Asri Blok G-301 Rt.001/Rw.035 Kel./Desa Donoharjo, Kec. Ngaglik, Kab. Sleman, D.I. Yogyakarta, Sleman, 22, 55581
Kewarganegaraan	:	Indonesia
Jenis Ciptaan	:	**Buku**
Judul Ciptaan	:	**Manajemen Jaringan Menggunakan Firebase Cloud Messaging Berbasis Android Teori Dan Praktek**
Tanggal dan tempat diumumkan untuk pertama kali di wilayah Indonesia atau di luar wilayah Indonesia	:	11 September 2019, di Yogyakarta
Jangka waktu pelindungan	:	Berlaku selama hidup Pencipta dan terus berlangsung selama 70 (tujuh puluh) tahun setelah Pencipta meninggal dunia, terhitung mulai tanggal 1 Januari tahun berikutnya.
Nomor pencatatan	:	000158013

adalah benar berdasarkan keterangan yang diberikan oleh Pemohon.
Surat Pencatatan Hak Cipta atau produk Hak terkait ini sesuai dengan Pasal 72 Undang-Undang Nomor 28 Tahun 2014 tentang Hak Cipta.

a.n. MENTERI HUKUM DAN HAK ASASI MANUSIA
DIREKTUR JENDERAL KEKAYAAN INTELEKTUAL

Dr. Freddy Harris, S.H., LL.M., ACCS.
NIP. 196611181994031001

LAMPIRAN PENCIPTA

No	Nama	Alamat
1	DESSYANTO BOEDI PRASETYO	Perum Griya Taman Asri Blok G-301 Rt.001/Rw.035 Kel./Desa Donoharjo, Kec. Ngaglik, Kab. Sleman, D.I. Yogyakarta
2	RIZKI INKA MIFTAH	Perum Griya Satria Blok G-4 Rt.001/Rw.010 Kel./Desa Bantarsoka, Kec. Purwokerto Barat, Kab. Banyumas, Prov. Jawa Tengah
3	RIFKI INDRA PERWIRA	Jatimulyo Baru Blok E/11 Rt.028/Rw.006 Kel. Kricak, Kec. Tegalrejo, Kota Yogyakarta, D.I. Yogyakarta

LAMPIRAN PEMEGANG

No	Nama	Alamat
1	DESSYANTO BOEDI PRASETYO	Perum Griya Taman Asri Blok G-301 Rt.001/Rw.035 Kel./Desa Donoharjo, Kec. Ngaglik, Kab. Sleman, D.I. Yogyakarta
2	RIZKI INKA MIFTAH	Perum Griya Satria Blok G-4 Rt.001/Rw.010 Kel./Desa Bantarsoka, Kec. Purwokerto Barat, Kab. Banyumas, Prov. Jawa Tengah
3	RIFKI INDRA PERWIRA	Jatimulyo Baru Blok E/11 Rt.028/Rw.006 Kel. Kricak, Kec. Tegalrejo, Kota Yogyakarta, D.I. Yogyakarta