Implementasi Pengiriman Pesan Broadcast dengan Redis Pub/Sub dan Bahasa Pemrograman Nim

Tania Rachel[1], Yerymia Alfa Susetyo[2]
Universitas Kristen Satya Wacana, Jl. Dr. O. Notohamidjojo, Salatiga
Email: 672018370@student.uksw.edu[1], yeremia.alfa@uksw.edu[2]

Abstrack – Message broadcasting in the company is very important to keep the information in it up-to-date, in a large company that have many outlets that scattered in various location, the amount of the message that must be sent become a problem and the time allocated to send the message become longer. So In this study, the researchers implemented message delivery using Redis/Pubsub and Nim programming language. By using Redis Pub/Sub the sender only needs to send messages to one channel and the message will received by many subscriber channel. The result of the implementation is server application that used to publish a message and client application that used to subscribe a channel. Based on the test, the result is 86% that means the users "strongly agree" that this reserach can incresae the effectiveness of message delivery, and the result program in this study are very small in size so they are easier to distribute.

Keywords – Redis Pub/Sub, Nim *Language*, *Broadcast message delivery*.

Intisari - Penyebaran pesan pada perusahaan sangat penting untuk menjaga informasi didalamnya tetap *up-to-date*, pada perusahaan berskala besar yang memiliki banyak gerai yang tersebar, banyaknya pesan yang harus dikirim menjadi permasalahan tersendiri serta waktu yang digunakan untuk mengirimkan pesan akan menjadi lebih lama. Sehingga pada penelitian ini sistem pengiriman pesan *broadcast* diimplementasikan menggunakan Redis Pub/Sub dan bahasa pemrograman Nim, pada Redis Pub/Sub pengirim hanya perlu men-*publish* pesan ke dalam suatu *channel* dan pesan tersebut akan sampai pada banyak *subscriber channel*. Implementasi menghasilkan aplikasi *server* yang digunakan untuk *publish* pesan dan aplikasi *client* yang digunakan untuk men-*subscribe channel*. Berdasarkan pengujian yang dilakukan, didapatkan hasil 86% yang menunjukkan *user* "sangat setuju" jika penelitian ini dapat meningkatkan efektifitas pengiriman pesan. Hasil program yang dihasilkan pada penelitian ini juga berukuran sangat kecil sehingga mudah untuk di distribusikan.

Kata Kunci – Redis Pub/Sub, Bahasa Pemrograman Nim, Pengiriman pesan *broadcast*

I. PENDAHULUAN

Pada era modern saat ini, bisnis retail modern di Indonesia semakin berkembang pesat. Semakin besarnya sebuah perusahaan retail, maka semakin besar juga proses bisnis yang ada didalamnya. Salah satu hal yang penting bagi proses bisnis sebuah perusahaan adalah informasi [1]. Tanpa adanya persebaran informasi, proses bisnis tidak akan dapat berjalan dengan baik. Salah satu bentuk penyebaran informasi dalam lingkup perusahaan adalah pengiriman pesan melalui jaringan internal perusahaan dari kantor pusat ke komputer-komputer gerai yang berada di berbagai tempat [2].

Hampir seluruh aspek kehidupan manusia saat ini tidak dapat dilepaskan dari penggunaan komputer yang semakin meluas [14], bagi perusahaan retail berskala besar, kecepatan penyebaran pesan sangatlah penting untuk menjaga informasi tetap *up-to-date* [3]. Pesan yang dikirim juga perlu tersampaikan secara merata ke komputer di seluruh gerai yang ada. Akan

tetapi dengan bertambahnya jumlah gerai yang tersedia, mengakibatkan jumlah pesan yang perlu dikirimkan menjadi lebih banyak. Sehingga waktu yang digunakan dalam proses pengiriman pesan menjadi lebih lama.

Pada umumnya pengiriman pesan dapat dilakukan menggunakan *email*, namun pesan yang dikirimkan terkadang tidak terbaca oleh karyawan pada gerai sehingga informasi tidak dapat tersampaikan dengan baik [4].

Oleh karena itu perusahaan membutuhkan sistem pengiriman pesan yang efektif untuk mengirimkan pesan sekaligus ke banyak penerima secara *realtime,* sehingga proses bisnis dapat berjalan dengan baik tanpa adanya kekeliruan yang disebabkan oleh informasi yang terlambat atau tidak tersampaikan.

Redis merupakan penyimpanan struktur data dalam memori yang dapat berfungsi sebagai *message broker*. *Message broker* adalah *middleware* yang digunakan sebagai perantara yang menghubungkan pengirim dan penerima pesan [5].

Message broker pada Redis diimplementasikan dengan menggunakan fitur Redis Pub/Sub, Pada Redis Pub/Sub, penerima berperan sebagai *subscriber* dan pengirim sebagai *publisher* yang berkomunikasi melalui sebuah *channel* didalam *server* Redis [6]. Redis Pub/Sub dapat meningkatkan efektifitas dalam pengiriman pesan karena *publisher* hanya memerlukan sekali kirim pesan ke suatu channel*,* lalu komputer-komputer gerai yang telah melakukan *subscribe* akan menerima pesan tersebut. Redis Pub/Sub memiliki kecepatan yang sangat tinggi sehingga pesan dapat disampaikan secara *realtime* [7].

Bahasa pemrograman Nim merupakan salah satu bahasa pemrograman yang didukung oleh Redis. Bahasa pemrograman Nim dapat menghasilkan *executable file* berukuran kecil dan tanpa dependensi tambahan, sehingga aplikasi yang dibangun menjadi lebih mudah didistribusikan ke komputer-komputer gerai [8].

Berdasarkan uraian yang telah disampaikan, Redis pub/sub dan bahasa pemrograman Nim dapat digunakan dalam pembuatan sistem pengiriman pesan sehingga permasalahan yang dihadapi perusahaan dapat teratasi dengan baik.

II. SIGNIFIKANSI STUDI

A. Studi Literatur

Pada penelitian yang berjudul "Implementasi Sistem Pesan Popup Antar Perguruan Tinggi dengan Memanfaatkan Pemrograman Socket Dinamis". Pengiriman pesan di dalam suatu perguruan tinggi yang dilakukan dengan metode konvensional seperti Pos, surat elektronik, dan media sosial ke banyak penerima memiliki beberapa kerugian diantaranya adalah waktu, akun yang tidak valid, dan pesan yang tidak terbaca karena terlewatkan oleh penerimanya. Oleh karena itu dibutuhkan sistem pengiriman pesan dari 1 pengirim ke banyak penerima yang dapat mengatasi permasalahan-permasalahan tersebut. Sistem yang dibangun pada penelitian ini terbukti dapat meningkatkan efektifitas dan mengatasi permasalahan pengiriman pesan yang dihadapi dari 1 perguruan tinggi ke perguruan tinggi lainnya [4].

Pada penelitian berjudul "Monitoring Penggunaan Daya Listrik menggunakan Protokol MQTT berbasis Web" peneliti melakukan monitoring secara *realtime* menggunakan protokol MQTT. Protokol MQTT merupakan sistem pengiriman pesan yang berbasis *publish-subscribe*. *Publish-subscribe* dalam MQTT memungkinkan untuk mengirim dan menerima pesan berdasarkan topik yang diinginkan. Data yang dikirimkan oleh *publisher* adalah *Voltage*, *Current*, dan *Power* yang akan diteruskan ke *broker*, lalu data akan diteruskan oleh *broker* ke *subscriber* topik, yang akan ditampilkan pada *interface web*. Peneliti menyimpulkan sistem *publish-subscribe* yang dibangun dapat menyampaikan pesan dengan nilai *delay* yang relatif kecil. Hal ini dibuktikan dari 10 kali pengujian yang menghasilkan nilai delay rata-rata sebesar 0,011. Selain itu peneliti juga melakukan pengujian *delay* pada *bandwidth* yang terbatas dan

didapatkan hasil rata-rata 0,015 sehingga dibuktikan bahwa sistem *publish-subscribe* dapat berjalan dengan baik dalam jaringan yang terbatas [9].

Pada penelitian yang berjudul "*Low Latency Message Brokers*" membandingkan beberapa *message broker* yaitu Redis, RabbitMQ dan Apache Kafka. Penelitian ini membahas perbandingan fitur-fitur dari ketiga *message broker*, fitur yang dibandingkan yaitu *Message Delivery Guarantee, Message Persistence, Message Ordering, Throughput, Latency, Availability dan Scalability*. Berdasarkan hasil perbandingan, baik Redis, RabbitMQ maupun Kafka memiliki keunggulannya masing-masing. Keunggulan dari Redis adalah *Throughput* yang sangat tinggi, *Throughput* merupakan ukuran dari seberapa cepat data dapat ditransfer dari produsen ke konsumen. Redis juga mempunyai rentang waktu pengiriman dan penerimaan pesan (*latency*) yang paling rendah daripada *message broker* yang lain. Selain itu Redis mensupport 49 bahasa dan beragam struktur data yaitu *Strings, hashes, lists sets*, dan *bitmaps*. Dari perbandingan yang dilakukan peneliti, dapat disimpulkan Redis cocok digunakan untuk membuat aplikasi untuk proses data kecil, tetapi membutuhkan kecepatan yang tinggi. Redis juga cocok digunakan untuk proses *upload* data yang dapat dilihat secara *realtime* [10].

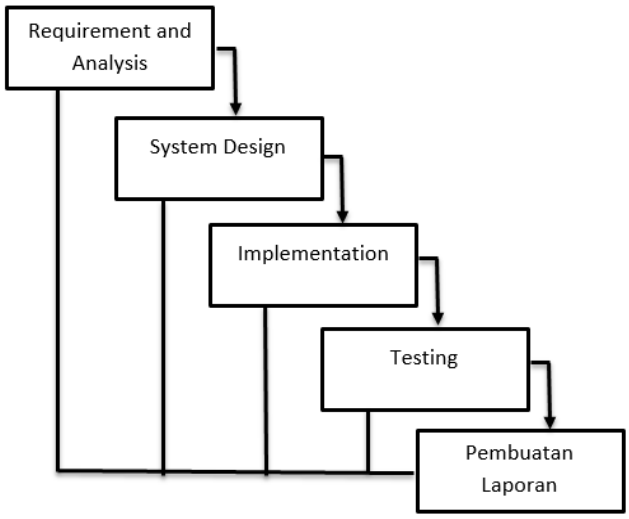
Redis (*Remote Dictionary Server*) merupakan penyimpanan struktur data di dalam memori yang dapat berfungsi sebagai *database, cache,* serta *message broker*. Karena bekerja di dalam memori Redis dapat mengakses data dengan sangat cepat, salah satu fitur yang diberikan oleh Redis adalah *publish/subscribe* [5].

Redis Pub/Sub mempunyai konsep dimana *publisher*(pengirim) berkomunikasi dengan *subscriber*(penerima) secara *asynchronous, publisher* akan mengirim pesan melalui *channel* di dalam server Redis dimana *channel* tersebut juga yang akan di-*subscribe* oleh *subscriber* untuk mendapatkan pesan [7].

Nim adalah bahasa pemrograman yang muncul pada tahun 2008. Nim mengkombinasikan konsep bahasa pemrograman yang telah ada seperti Python, Ada, dan Modula, tetapi secara keseluruhan *syntax* yang ada pada Nim menyerupai Python yang menggunakan indentasi. Nim didesain untuk menjadi bahasa pemrograman yang efisien, ekspresif dan elegan. Nim merupakan *compiled language* yang mengubah *source code* programnya kedalam bahasa pemrograman C, yang akan dikirimkan ke *compiler* C. *Compiler* C akan menghasilkan file *executable* yang independen, kecil, cepat, dan mudah didistribusikan [8].

B. Metode Penelitian

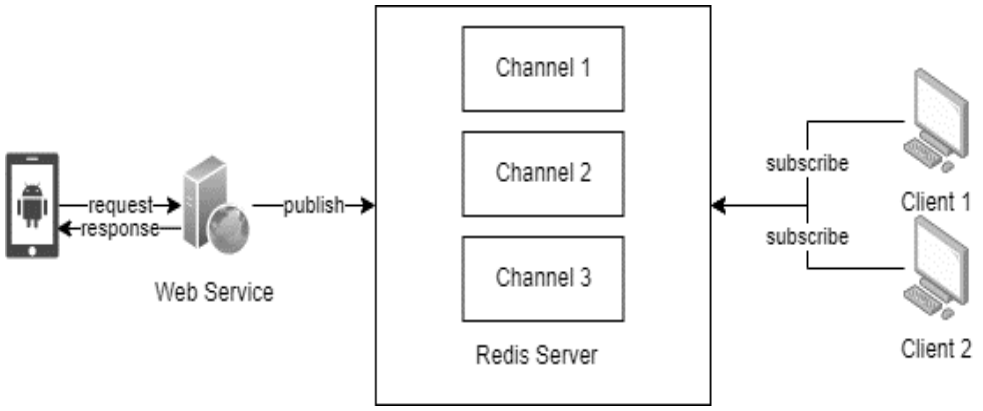
Metode yang digunakan dalam Implementasi Pengiriman Pesan dengan Redis Pub/Sub dan Bahasa Pemrograman Nim adalah waterfall yang dapat dilihat pada gambar 1.



Gambar 1. Metode *waterfall*

Tahapan pertama dari metode *waterfall* adalah mempersiapkan dan menganalisa kebutuhan dari *software* yang akan dikerjakan. Kebutuhan dari penelitian ini adalah bagaimana mengirim pesan dari pusat kepada gerai-gerai di berbagai daerah secara *realtime*. Pada tahap ini pula dilakukan analisis dengan studi literatur yang dapat digunakan sebagai referensi untuk membantu penelitian yang dilakukan.

Setelah melakukan analisis maka dibuatlah arsitektur dari aplikasi yang ditunjukkan pada gambar 2.

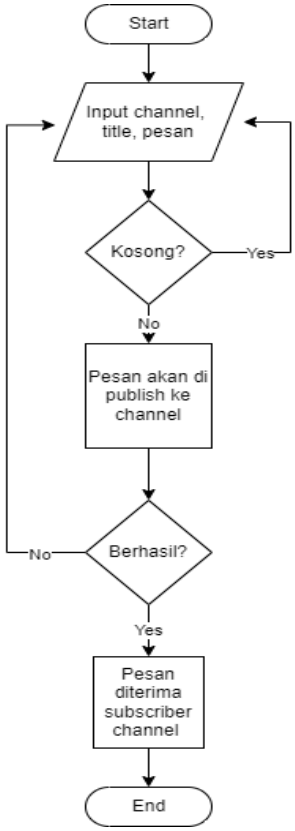


Gambar 2. Arsitektur aplikasi

Proses pengiriman pesan pada penelitian ini berjalan pada jaringan intranet. Pada penelitian ini terdapat dua aplikasi yaitu dari sisi *server* dan *client*. Aplikasi *server* memiliki basis android yang akan melakukan *request* dengan method HTTP POST dan menerima *response* dari *web service* [11]. *Web Service* akan mengirim pesan (*publish*) ke Redis Server, lalu aplikasi client yang terdapat pada komputer, diprogram untuk dapat menerima pesan dengan men-*subscribe* channel yang telah ditentukan. *Web service* dan aplikasi *client* dibangun menggunakan bahasa pemrograman Nim.

Gambar 3 merupakan *flowchart* dari aplikasi pengiriman pesan yang dibuat. Pada tahapan awal, admin sebagai *user* pada sisi *server* akan memasukkan beberapa inputan berupa *string* untuk mengirim pesan yaitu *channel*, judul dan isi pesan. Pada proses ini juga terjadi pengecekan apakah inputan masih kosong atau sudah diisi. Jika inputan telah sesuai maka pesan akan di *publish* ke *channel* didalam server Redis dan jika berhasil maka pesan akan diterima oleh *subscriber channel* yaitu komputer-komputer *client.*

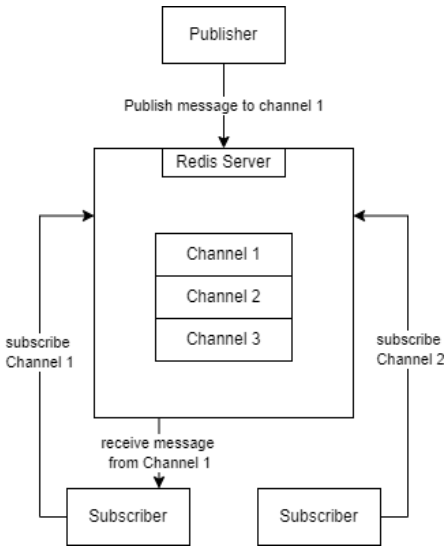
Dari desain yang telah dibuat, desain diimplementasikan menjadi sebuah program dengan bahasa pemrograman dan *tools* yang telah ditentukan yaitu dengan bahasa Nim dan Redis Pub/Sub, lalu program yang dibuat akan diuji fungsionalitas dan kesesuaiannya terhadap desain yang telah dibuat. Lalu tahap akhir dari penelitian adalah penyusunan laporan dari penelitian yang telah dilakukan.



Gambar 3. *Flowchart* aplikasi

III. HASIL DAN PEMBAHASAN

Implementasi pembuatan aplikasi didasarkan pada arsitektur Redis Pub/Sub yang terdapat pada gambar 4.



Gambar 4. Arsitektur Redis Pub/Sub

Pada dasarnya Redis Pub/Sub merupakan salah satu fitur *message broker* dari Redis, *message broker* menyediakan sarana standar untuk menangani aliran data antar aplikasi seperti memvalidasi, menyimpan, memetakan dan mengirimkan pesan ke tujuan yang sesuai, sekalipun aplikasi tersebut dibangun dengan bahasa pemrograman dan platform yang berbeda, sehingga fokus pekerjaan dapat terpusat pada logika inti dalam membangun aplikasi. Redis Pub/Sub dirancang untuk menyiarkan pesan *broadcast* langsung di dalam suatu sistem yang memiliki kondisi dimana *low latency* dan *huge throughput* sangat dibutuhkan.

Redis Pub/Sub memiliki beberapa *command* yang dapat digunakan. Pada penelitian ini *command* yang digunakan adalah *PUBLISH* yang berfungsi untuk mengirim pesan ke *channel* dan *SUBSCRIBE* yang berfungsi untuk dapat menerima pesan dari sebuah *channel*. *Channel* terdapat pada *server* Redis yang berperan seperti ruangan yang menghubungkan pesan dari *publisher* agar dapat sampai kepada *subscriber*.

Dari arsitektur aplikasi yang terdapat pada desain, Aplikasi android berfungsi untuk menerima inputan berupa string *channel* yang terdapat pada *combo box*. String *channel* yang tersedia adalah Channel 1, Channel 2 dan Channel 3. Inputan selanjutnya adalah string *title* yaitu judul dari pesan, dan string *message* yaitu isi dari pesan yang akan dikirim. Dari aplikasi android, *string* yang diinputkan akan diubah ke bentuk JSON untuk dilakukan request post ke *web service* yang telah dibuat.

Pada *web service* yang ditunjukkan pada gambar 5, kiriman berupa JSON akan diubah kembali ke bentuk *string* untuk dimasukkan ke dalam parameter pada procedure redisService yang terdapat pada baris program ke 8.

```
1  post "/sendMessage":
2     var count = 0
3     var countAll = 0
4     var data = parseJson(request.body)
5     var channel = data["channel"].getStr
6     var title = data["title"].getStr
7     var message = data["message"].getStr
8     var catchReturn = redisService(channel,title,message)
```

Gambar 5. Kode Program pada *web service*

Pada *procedure* redisService dilakukan proses *publish* pesan ke dalam sebuah *channel* sesuai parameter yang ada, proses ini terdapat pada gambar 6. Seperti arsitektur Redis Pub/Sub pada gambar 4, *channel* terdapat pada sebuah *server* Redis sehingga dalam proses ini *server* Redis harus dihidupkan dan dikonfigurasi dengan IP dan Port yang sesuai dari sebuah jaringan intranet. konfigurasi *server* Redis dapat dilakukan dengan mengganti *bind* IP pada file **redis.windows.conf**.

Proses *publish* pesan dapat dilakukan dengan import redis pada program, lalu seperti yang terlihat pada baris program ke 7 *host* dan *port* harus ditetapkan sesuai dimana *server* Redis berjalan. Lalu publish dapat dilakukan sesuai dengan baris program ke 14 *procedure publish* memiliki 3 parameter yaitu pub, destination dan bodyMessage. Pub merupakan dimana IP dan *port server* Redis berjalan, destination merupakan nama *channel* dimana *publish* akan dilakukan, dan bodyMessage yaitu 1 baris string yang merupakan gabungan dari *title* dan *message* yg akan di-*publish* ke channel pada *server* Redis yang telah diatur. Redis *publish* memiliki kembalian berupa *integer*, *integer* 0 menandakan pesan tidak terkirim ke *channel* tersebut.

```
1   import redis
2   proc redisService(channel:string,title:string,message:string) :seq[string]
3   =
4       var flag = true
5       var returnResponse : seq[string] = @[]
6       while flag:
7           let pub = open(host="192.168.1.3",port = 6379.Port)
8           var redtitle = title
9           var redMessage = message
10          var bodyMessage = redTitle&"|"&redMessage
11          let listChannel = @["Channel 1","Channel 2","Channel 3"]
12          if channel == "All Channel":
13              for destination in listChannel:
14                  let pubs =  publish(pub,destination,bodyMessage)
15                  if pubs==0:
16                      returnResponse.add("unsuccess")
17                  else:
                        returnResponse.add("success")
```

Gambar 6. Kode Program Redis *Publish* pada *Server*

Dari sisi *client* sebagai *subscriber*, program dibuat untuk dapat menerima pesan dengan men-*subscribe* channel, kode program untuk proses ini terdapat pada gambar 7. Jika dilihat dari arsitektur Redis Pub/Sub yang terdapat pada gambar 4 untuk dapat men-*subscribe channel*, IP harus disesuaikan dengan IP dimana *server* Redis berjalan sesuai pada baris kode ke 4. *Procedure* untuk *subscribe channel* dijalankan secara *asynchronous*, proses *asynchronous* dibutuhkan agar saat admin men-*publish* banyak pesan, program dapat menerima pesan lain yang dikirimkan saat proses menerima pesan sebelumnya belum selesai, sehingga pesan dapat diterima secara *realtime*.

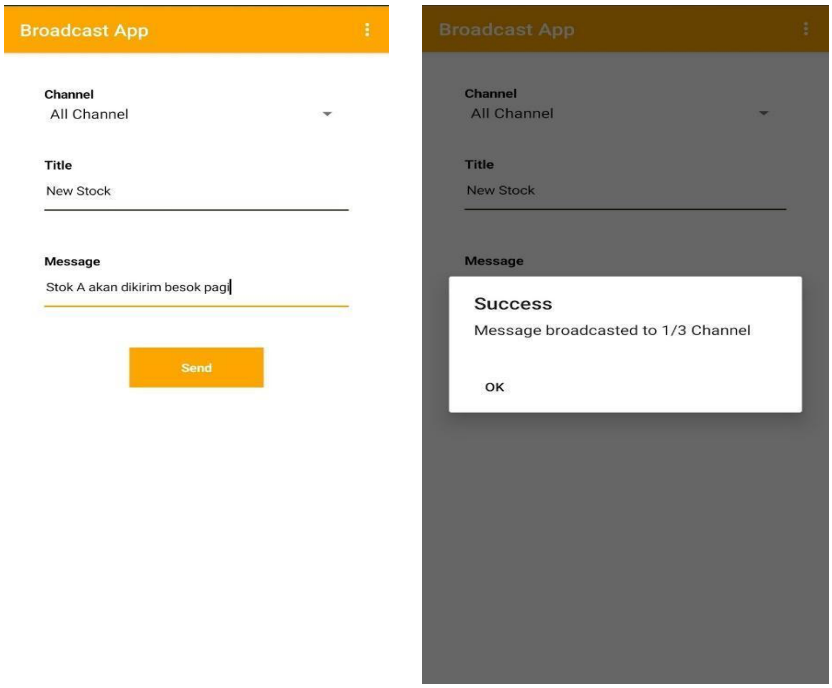
Lalu untuk men-*subscribe* channel, dilakukan dengan memanggil *procedure* **sub.subscribe** pada baris kode ke 5, variabel sub merupakan dimana IP *server* Redis berjalan dan parameter didalamnya merupakan nama *channel* yang akan di *subscribe*. Lalu *procedure* **sub.nextMessage** digunakan untuk dapat mengambil pesan yang di-*publish* dari channel yang telah di-*subscribe*. Pesan diterima dalam bentuk 1 baris string yang akan ditampilkan dalam bentuk *message box* pada komputer *client*.

1 komputer *client* dapat men-*subscribe* beberapa *channel* untuk mendapatkan pesan. Saat pesan di-*publish* pada channel yang tidak di-*subscribe* maka komputer *client* tidak akan mendapatkan pesan tersebut.

```
1   proc main() {.async.} =
2      var loop=true
3      while loop:
4      let sub = await openAsync("192.168.1.3")
5      await sub.subscribe("Channel 1")
6      let message = (await sub.nextMessage()).message
```

Gambar 7. Kode Program *Redis Subscribe* pada *Client*

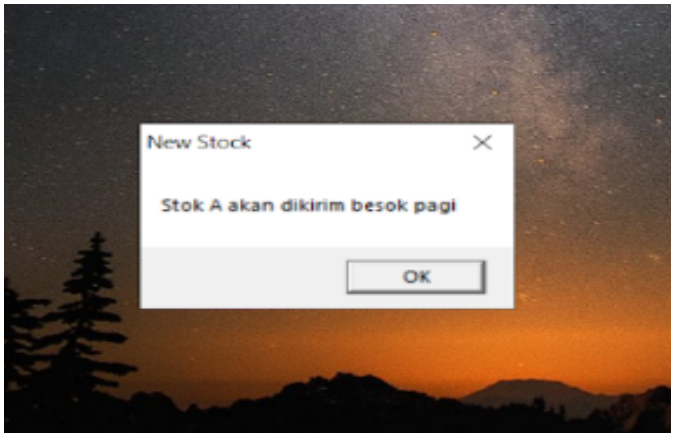
Gambar 8 a merupakan tampilan dari aplikasi pengiriman pesan. Saat admin menyentuh tombol *send* maka akan muncul notifikasi status pengiriman pesan seperti yang terlihat di gambar 8 b.



(a) (b)

Gambar 8. (a) UI aplikasi (b) Notifikasi sukses

Saat berhasil mengirim ke *channel* maka aplikasi *client* akan menerima message *box* seperti pada gambar 9. Hasil aplikasi *client* yang dibuat menggunakan bahasa pemrograman Nim dalam penelitian ini memiliki ukuran *executable file* sangat kecil yaitu sebesar 576KB.



Gambar 9. *Message box* pada komputer *client*

TABEL I
BLACKBOX TESTING PADA APLIKASI PENGIRIMAN PESAN

Pengujian	Hasil yang diharapkan	Pengamatan	Kesimpulkan
Input data pada All Channel, Judul, Pesan	Data berhasil di publish dan pesan diterima oleh client subscriber Channel 1, Channel 2, dan Channel 3 dalam bentuk message box	Data berhasil di publish dan pesan diterima oleh client subscriber Channel 1, Channel 2, dan Channel 3 dalam bentuk message box	Diterima
Input data pada Channel 1, Judul, Pesan	Data berhasil di publish dan pesan diterima oleh client subscriber Channel 1 dalam bentuk message box	Data berhasil di publish dan pesan diterima oleh client subscriber Channel 1 dalam bentuk message box	Diterima
Input data pada Channel 2, Judul, Pesan	Data berhasil di publish dan pesan diterima oleh client subscriber Channel 2 dalam bentuk message box	Data berhasil di publish dan pesan diterima oleh client subscriber Channel 2 dalam bentuk message box	Diterima
Input data pada Channel 3, Judul, Pesan	Data berhasil di publish dan pesan diterima oleh client subscriber Channel 3 dalam bentuk message box	Data berhasil di publish dan pesan diterima oleh client subscriber Channel 3 dalam bentuk message box	Diterima
Tidak mengisi Inputan Judul dan Pesan	Muncul notifikasi field belum terisi	Muncul notifikasi field belum terisi	Diterima
Tidak mengisi Inputan Judul	Muncul notifikasi field Judul belum terisi	Muncul notifikasi field Judul belum terisi	Diterima
Tidak mengisi Inputan Pesan	Muncul notifikasi field Pesan belum terisi	Muncul notifikasi field Pesan belum terisi	Diterima

Hasil pengujian *blackbox testing* pada aplikasi ditunjukkan pada tabel I. Pada pengujian ini menunjukkan bahwa aplikasi dapat mengirim pesan kepada *client-client* sesuai *channel* yang dituju.

Lalu dilakukan penyebaran kuesioner yang melibatkan 15 orang internal IT perusahaan untuk mengetahui apakah sistem aplikasi pengiriman pesan ini efektif dalam menyampaikan pesan *broadcast* di dalam perusahaan. Hasil kuesioner yang disebarkan ditunjukkan pada tabel II.

TABEL III
HASIL KUESIONER

Kode Pertanyaan	Pertanyaan	Jawaban					Jumlah
		Ax1	Ax2	Ax3	Ax4	Ax5	
1	Apakah menurut anda, adanya aplikasi pengiriman pesan ini dapat mempermudah admin untuk mengirimkan pesan ke banyak penerima sekaligus?	-	1	-	5	9	67

2	Apakah menurut anda, sistem pengiriman pesan ini lebih baik untuk diterapkan pada perusahaan jika dibandingkan dengan metode pengiriman konvensional lainnya?	-	-	-	10	5	65
3	Apakah menurut anda, sistem pengiriman pesan ini dapat meningkatkan atensi dari client sebagai penerima pesan?	-	-	2	9	4	62

Dengan menggunakan skala likert, kolom jumlah yang terdapat pada tabel 2 didapatkan pada rumus perhitungan (1)[12]

$$\sum T \times Pn \quad (1)$$

Keterangan:
T = Total responden
Pn = Skor likert

Dimana skor likert didapatkan dari tabel III.

TABEL IIIII
SKOR LIKERT

Kode Jawaban	Deskripsi Jawaban	Skor
Ax5	Sangat Setuju	5
Ax4	Setuju	4
Ax3	Cukup	3
Ax2	Tidak Setuju	2
Ax1	Sangat Tidak Setuju	1

Lalu dari skor yang didapatkan, dihitung hasil interpretasi Y dimana skor tertinggi likert dikalikan dengan jumlah responden dan didapatkan hasil 5x15 = 75. Selanjutnya dilakukan perhitungan hasil seperti pada tabel IV.

TABEL IVV
HASIL PERHITUNGAN KUESIONER

Kode Pertanyaan	(Jumlah/Y) x 100%
1	(67/75) x100% = 89%
2	(65/75) x100% = 86%
3	(62/75) x100% = 82%

TABEL V
INDEX RANGE SKALA LIKERT

Index Range	Hasil
0% - 19,99%	Sangat Tidak Setuju
20% - 39,99%	Tidak Setuju
40% - 59,99%	Kurang Setuju
60% - 79,99%	Setuju
80 – 100%	Sangat Setuju

Dari hasil rata-rata yang dijumlahkan dari tabel IV didapatkan hasil 86%, dan jika dilihat pada index range skala likert yang ditunjukkan pada tabel V [13]. Hasil tersebut termasuk di dalam hasil "Sangat Setuju" sehingga dapat disimpulkan sistem aplikasi pengiriman pesan ini efektif dalam menyampaikan pesan *broadcast* di dalam perusahaan.

IV. KESIMPULAN

Pada penelitian Implementasi Pengiriman Pesan dengan Redis Pub/Sub dan Bahasa Pemrograman Nim yang dilakukan, dapat diambil kesimpulan bahwa proses pengiriman pesan dengan memanfaatkan Redis Pub/Sub dapat menyampaikan pesan secara efektif yang dibuktikan dari hasil kuesioner yang menghasilkan nilai 86% atau "sangat setuju" dari para user. Untuk dapat mengirim pesan admin hanya perlu mengirim pesan ke *channel* tujuan maka komputer-komputer *client* yang men-*subscribe channel* akan menerima pesan tersebut. Penggunaan bahasa pemrograman Nim dalam penelitian ini menghasilkan *executable file* aplikasi yang berukuran kecil, sehingga aplikasi lebih ringan dan mudah didistribusikan.

Penelitian ini hanya berfokus dalam proses pengiriman dan penerimaan pesan, diharapkan pada penelitian selanjutnya fitur pendukung lainnya seperti penyimpanan pesan dan keamanan pesan dapat dikembangkan lebih lanjut.

REFERENSI

[1] S. C. Cahyodi and R. W. Arifin, "Sistem Informasi Point of Sales Berbasis Web Pada Colony Amaranta Bekasi," vol. 1, no. 2, pp. 189–204, 2017.

[2] D. Yadav, D. P. Sharma, and B. Keshwani, "A Study of Intranet over Cloud," Int. J. New Innov. Eng. Technol., vol. 7, no. 2, pp. 1–6, 2017.

[3] Y. Chen, T. Mao, and B. Yu, "A reliable messaging middleware for financial institutions," ACM Int. Conf. Proceeding Ser., pp. 108–112, 2017, doi: 10.1145/3162957.3163050.

[4] S. A. N. Afiah, "Implementasi sistem pesan popup antar perguruan tinggi dengan memanfaatkan pemrograman soket dinamis," J. Inf. Sains dan Teknol., 2019.

[5] F. Febriyani, E. S. Pramukantoro, and F. A. Bachtiar, "Perbandingan Kinerja Redis, Mosquitto, dan MongoDB sebagai Message Broker pada IoT Middleware | Jurnal Pengembangan Teknologi Informasi dan Ilmu Komputer," J-Ptiik.ub.ac.id, vol. 03, no. 07, pp. 6816–6823, 2019.

[6] D. Dedousis, N. Zacheilas, and V. Kalogeraki, "On the fly load balancing to address hot topics in topic-based pub/sub systems," *Proc. - Int. Conf. Distrib. Comput. Syst.*, vol. 2018-July, no. c, pp. 76–86, 2018, doi: 10.1109/ICDCS.2018.00018.

[7] R. K. Singh and H. K. Verma, "Redis-Based Messaging Queue and Cache-Enabled Parallel Processing Social Media Analytics Framework," Comput. J., vol. 00, no. 00, 2020, doi: 10.1093/comjnl/bxaa114.

[8] D. Picheta, in Nim in action, Shelter Island: Manning, 2017, pp. 4–6.

[9] R. Z. Pratama and H. Nurwarsito, "Monitoring Penggunaan Daya Listrik menggunakan Protokol MQTT berbasis Web," vol. 3, no. 11, pp. 10820–10826, 2019.

[10] R. G. Hegde and G. S. Nagaraja, "Low Latency Message Brokers," no. May, pp. 2731–2738, 2020.

[11] A. Tedyyana, M. Fauzi, F. Ratnawati, *Revamp Keamanan Web Service Milik PT XYZ Menggunakan REST API, Digit. Zo. J. Teknol. Inf. dan Komun.*2021;vol. 12(1):1–10, doi: 10.31849/digitalzone.v12i1.6378.

[12] V. H. Pranatawijaya, W. Widiatry, R. Priskila, and P. B. A. A. Putra, "Penerapan Skala Likert dan Skala Dikotomi Pada Kuesioner Online," *J. Sains dan Inform.*, vol. 5, no. 2, pp. 128–137, 2019, doi: 10.34128/jsi.v5i2.185.

[13] R. Hardianto, Z. Zamzami, and W. Wirdahchoiriah, "Efektifitas Penerapan Blended Learning Terhadap Hasil Belajar Mahasiswa Di Unilak," *INOVTEK Polbeng - Seri Inform.*, vol. 5, no. 1, p. 106, 2020, doi: 10.35314/isi.v5i1.1002.
[14] A Tedyyana, Supria, "*Perancangan Sistem Pendeteksi Dan Pencegahan Penyebaran Malware Melalui SMS Gateway*," INOVTEK Polbeng - Seri Inform., vol. 3, no. 1, 2018.