



**UNIVERSITAS INDONESIA**

**PENGEMBANGAN APLIKASI MATLAB CLIENT DAN SERVER  
BERBASIS MESSAGING DENGAN RABBITMQ**

**SKRIPSI**

**FAHRI NURUL HIDAYAT  
0706271701**

**FAKULTAS ILMU KOMPUTER  
PROGRAM STUDI ILMU KOMPUTER  
DEPOK  
JUNI 2011**



**UNIVERSITAS INDONESIA**

**PENGEMBANGAN APLIKASI MATLAB CLIENT DAN SERVER  
BERBASIS MESSAGING DENGAN RABBITMQ**

**SKRIPSI**

**Diajukan sebagai salah satu syarat untuk memperoleh gelar  
Sarjana Ilmu Komputer**

**FAHRI NURUL HIDAYAT**

**0706271701**

**FAKULTAS ILMU KOMPUTER  
PROGRAM STUDI ILMU KOMPUTER**

**DEPOK**

**JUNI 2011**

## **HALAMAN PERNYATAAN ORISINALITAS**

**Skripsi ini adalah hasil karya saya sendiri,  
dan semua sumber baik yang dikutip maupun dirujuk  
telah saya nyatakan dengan benar.**

**Nama : Fahri Nurul Hidayat**  
**NPM : 0706271701**  
**Tanda Tangan :**

**Tanggal : 30 Juni 2011**

## **HALAMAN PENGESAHAN**

Skripsi ini diajukan oleh :

Nama : Fahri Nurul Hidayat

NPM : 0706271701

Program Studi : Ilmu Komputer

Judul Skripsi : Pengembangan Aplikasi MATLAB Client dan Server  
berbasis Messaging dengan RabbitMQ

**Telah berhasil dipertahankan di hadapan Dewan Penguji dan diterima sebagai bagian persyaratan yang diperlukan untuk memperoleh gelar Sarjana Ilmu Komputer pada Program Studi Ilmu Komputer, Fakultas Ilmu Komputer, Universitas Indonesia.**

## **DEWAN PENGUJI**

Pembimbing : Agus Kurniawan S.T., M.Si ( )

Penguji : Lim Yohanes Stefanus, Drs, M.Math, Ph.D. ( )

Penguji : Satrio Baskoro Yudhoatmojo, S.Kom, MTI ( )

Ditetapkan di : Depok

Tanggal : 8 Juli 2011

## KATA PENGANTAR

Alhamdulillahirobbil a'amin. Atas berkat dan rahmat Allah SWT, penulis dapat menyelesaikan tugas akhir dan penyusunan laporan skripsi dengan judul: Pengembangan Aplikasi MATLAB Client dan Server berbasis Messaging dengan RabbitMQ.

Pada kesempatan ini, penulis ingin menyampaikan terima kasih yang kepada semua pihak yang telah ikut serta memberikan dukungan serta bantuan mencakup dorongan semangat dan moral, sehingga akhirnya, tugas akhir ini dapat berjalan lancar seperti sebagaimana mestinya. Penulis mengucapkan terima kasih kepada:

1. Kedua orang tua penulis yang selalu menyertai langkah penulis dalam setiap do'a mereka.
2. Kedua saudara kandung penulis, Ifa Nurul Utami dan Fadhila Nurul Hastuti, yang selalu menemani dan memberi semangat kepada penulis.
3. Bapak Agus Kurniawan, selaku pembimbing tugas akhir yang telah mengarahkan penulis dalam melaksanakan tugas akhir ini hingga proses penyusunan laporan.
4. Bapak Widijanto S. Nugroho, selaku pembimbing akademik yang telah memberikan arahan kepada penulis selama masa studi di Fakultas Ilmu Komputer, Universitas Indonesia.
5. Dosen, staf, mahasiswa, dan seluruh keluarga besar Fasilkom UI yang namanya tidak bisa penulis sebutkan satu per satu di sini, terima kasih atas segala bentuk bantuan dan dukungannya.

Penulis menyadari masih terdapat kekurangan pada penyusunan laporan ini. Oleh karena itu, penulis dengan tangan terbuka bersedia menerima kritik dan saran yang berguna. Semoga karya ini bermanfaat bagi pembaca semua.

Depok, 30 Juni 2011

Fahri Nurul Hidayat

## HALAMAN PERNYATAAN PERSETUJUAN PUBLIKASI TUGAS AKHIR UNTUK KEPENTINGAN AKADEMIS

Sebagai sivitas akademik Universitas Indonesia, saya yang bertanda tangan di bawah ini:

**Nama** : Fahri Nurul Hidayat  
**NPM** : 0706271701  
**Program Studi** : Ilmu Komputer  
**Fakultas** : Ilmu Komputer  
**Jenis Karya** : Skripsi

demikian pengembangan ilmu pengetahuan, menyetujui untuk memberikan kepada Universitas Indonesia **Hak Bebas Royalti Noneksklusif (Non-exclusive Royalty Free Right)** atas karya ilmiah saya yang berjudul:

Pengembangan Aplikasi MATLAB Client dan Server berbasis Messaging dengan RabbitMQ

berserta perangkat yang ada (jika diperlukan). Dengan Hak Bebas Royalti Noneksklusif ini Universitas Indonesia berhak menyimpan, mengalihmedia/formatkan, mengelola dalam bentuk pangkalan data (database), merawat, dan memublikasikan tugas akhir saya selama tetap mencantumkan nama saya sebagai penulis/pencipta dan sebagai pemilik Hak Cipta.

Demikian pernyataan ini saya buat dengan sebenarnya.

Dibuat di : Depok  
Pada tanggal : 30 Juni 2011  
Yang menyatakan

(Fahri Nurul Hidayat)

## ABSTRAK

Nama : Fahri Nurul Hidayat  
Program Studi : Ilmu Komputer  
Judul : Pengembangan Aplikasi MATLAB Client dan Server  
berbasis Messaging dengan RabbitMQ

MATLAB adalah perangkat lunak yang umum digunakan sebagai lingkungan analisis data, visualisasi data, dan komputasi numerik. Namun, sumber daya yang dibutuhkan MATLAB untuk beroperasi cukup besar. Oleh karena itu, ada kebutuhan arsitektur sistem interaksi yang dapat membagi beban pengoperasian MATLAB secara terpisah. Penelitian ini mencoba untuk mengimplementasikan sistem interaksi MATLAB dengan paradigma *client - server* berbasis *messaging*. Pada sistem ini, RabbitMQ digunakan sebagai *message broker* yang menggunakan teknik komunikasi *remote procedure call*. Hasil implementasi menunjukkan MATLAB *server* dapat mengolah perintah - perintah dasar MATLAB dan MATLAB *client* dapat berperan sebagai media interaksi yang menampilkan hasil komputasi MATLAB berupa teks dan grafik.

Kata Kunci:

MATLAB, *client-server architecture*, *messaging pattern*, *remote procedure call*, RabbitMQ

## ABSTRACT

Name : Fahri Nurul Hidayat  
Study Program : Computer Science  
Title : MATLAB Client and Server Application Development based on  
Messaging using RabbitMQ

MATLAB is an interactive environment for algorithm development, data visualization, data analysis, and numeric computation. However, the resources needed to operate MATLAB are large enough. Therefore, there is a need for interaction system architecture which can divide the load of MATLAB operation separately. This study attempts to implement the architecture of MATLAB interaction with the paradigm of client - server based on messaging. In this architecture, RabbitMQ is used as a message broker that uses remote procedure call as communication techniques. System implementation results show that the MATLAB server can process the basic commands of MATLAB and MATLAB client can act as a medium of interaction that displays the MATLAB computational results as text and graphics.

Keywords:

MATLAB, *messaging pattern*, *remote procedure call*, RabbitMQ



## DAFTAR ISI

<b>HALAMAN JUDUL</b>	<b>i</b>
<b>LEMBAR PERNYATAAN ORISINALITAS</b>	<b>ii</b>
<b>LEMBAR PENGESAHAN</b>	<b>iii</b>
<b>KATA PENGANTAR</b>	<b>iv</b>
<b>LEMBAR PERSETUJUAN PUBLIKASI ILMIAH</b>	<b>v</b>
<b>ABSTRAK</b>	<b>vi</b>
<b>DAFTAR ISI</b>	<b>viii</b>
<b>DAFTAR GAMBAR</b>	<b>xi</b>
<b>DAFTAR TABEL</b>	<b>xii</b>
<b>DAFTAR ALGORITMA</b>	<b>xiii</b>
<b>DAFTAR LAMPIRAN</b>	<b>xiv</b>
<b>1 PENDAHULUAN</b>	<b>1</b>
1.1 Latar Belakang	1
1.2 Permasalahan	2
1.2.1 Definisi Permasalahan	2
1.2.2 Batasan Permasalahan	2
1.3 Tujuan	3
1.4 Metodologi Penelitian	3
1.5 Sistematika Penulisan	4
<b>2 LANDASAN TEORI</b>	<b>5</b>
2.1 <i>Message-Oriented Middleware</i>	5
2.2 RabbitMQ	6
2.3 MATLAB	9
2.3.1 JMatLink	9
2.4 <i>Remote Procedure Call (RPC)</i>	9
2.5 Teknologi Java	10
2.6 <i>Software Development Life Cycle</i>	11

<b>3</b>	<b>PERANCANGAN DAN IMPLEMENTASI</b>	<b>12</b>
3.1	Arsitektur Sistem	12
3.1.1	<i>Request Format Data</i>	12
3.1.2	<i>Response Format Data</i>	13
3.2	<i>Use Case Diagram</i>	14
3.3	MATLAB Client	14
3.3.1	<i>Functional Requirement</i>	14
3.3.1.1	<i>Connect to Message Broker</i>	14
3.3.1.2	<i>Execute Query</i>	14
3.3.1.3	<i>View Image</i>	14
3.3.1.4	<i>Save Image</i>	15
3.3.1.5	<i>View Variable History</i>	15
3.3.1.6	<i>Input Variable From History</i>	15
3.3.1.7	<i>View Command History</i>	15
3.3.1.8	<i>Input Command From History</i>	15
3.3.2	<i>Nonfunctional Requirement</i>	15
3.3.2.1	<i>User Interface Requirement</i>	15
3.3.2.2	<i>Error Handling Requirement</i>	15
3.3.3	<i>Activity Diagram</i>	16
3.3.4	<i>User Interface</i>	21
3.3.4.1	<i>Message Broker Host Input Dialog</i>	21
3.3.4.2	<i>Console Panel</i>	23
3.3.4.3	<i>Image Panel</i>	24
3.3.4.4	<i>Status Panel</i>	25
3.3.4.5	<i>Variable History Panel</i>	26
3.3.4.6	<i>Command History Panel</i>	27
3.3.5	Teknik dan Algoritma	27
3.3.5.1	Akses RabbitMQ Message Broker	27
3.3.5.2	Pengiriman Perintah dan Penerimaan Respon	29
3.3.5.3	Variable History	30
3.3.5.4	Command History	31
3.3.5.5	Menampilkan Gambar	31
3.3.5.6	Menyimpan Gambar	32
3.4	MATLAB Server	33
3.4.1	<i>Functional Requirement</i>	33
3.4.1.1	<i>Connect to Message Broker</i>	33
3.4.1.2	<i>Client Session Management</i>	33
3.4.1.3	<i>Execute Query</i>	33
3.4.1.4	<i>Retrieve Image</i>	33
3.4.1.5	<i>Close Matlab Engine</i>	34
3.4.2	Teknik dan Algoritma	34
3.4.2.1	Akses RabbitMQ message broker	34
3.4.2.2	Session Management	35
3.4.2.3	Evaluasi Query MATLAB dan Pengambilan Hasil	36
3.4.2.4	Pengambilan Gambar	37
3.4.2.5	Tutup Koneksi	38

<b>4</b>	<b>PENGUJIAN</b>	<b>39</b>
4.1	Pengujian dengan Hasil <i>String</i>	40
4.2	Pengujian dengan Hasil Gambar	41
4.3	Pengujian <i>Multi-User</i> dengan <i>Session</i> yang Berbeda	42
<b>5</b>	<b>KESIMPULAN DAN SARAN</b>	<b>44</b>
5.1	Kesimpulan	44
5.2	Saran	44
	<b>DAFTAR REFERENSI</b>	<b>45</b>

## DAFTAR GAMBAR

Gambar 2.1	Komunikasi <i>message-oriented middleware</i> [3] . . . . .	5
Gambar 2.2	Notasi <i>producer</i> [9] . . . . .	6
Gambar 2.3	Notasi <i>consumer</i> [9] . . . . .	6
Gambar 2.4	Notasi <i>queue</i> [9] . . . . .	6
Gambar 2.5	Simple Queues [9] . . . . .	7
Gambar 2.6	Work Queues [10] . . . . .	7
Gambar 2.7	Publish/Subscribe [11] . . . . .	7
Gambar 2.8	Routing [12] . . . . .	8
Gambar 2.9	Topics [13] . . . . .	8
Gambar 2.10	Remote Procedure Call [14] . . . . .	8
Gambar 2.11	MATLAB [1] . . . . .	9
Gambar 2.12	Diagram alur eksekusi pada RPC . . . . .	10
Gambar 2.13	Tiga tahapan umum SDLC [18] . . . . .	11
Gambar 3.1	<i>Arsitektur sistem</i> . . . . .	12
Gambar 3.2	<i>Use Case Diagram</i> . . . . .	14
Gambar 3.3	<i>Activity Diagram</i> untuk <i>Connect to Message Broker</i> . . . . .	16
Gambar 3.4	<i>Activity Diagram</i> untuk <i>Execute Query</i> . . . . .	17
Gambar 3.5	<i>Activity Diagram</i> untuk <i>Save Image</i> . . . . .	18
Gambar 3.6	<i>Activity Diagram</i> untuk <i>Input Variable from History</i> . . . . .	19
Gambar 3.7	<i>Activity Diagram</i> untuk <i>Input Command from History</i> . . . . .	20
Gambar 3.8	Tampilan utama MATLAB Client . . . . .	21
Gambar 3.9	<i>Message Broker Host Input Dialog</i> . . . . .	21
Gambar 3.10	<i>Message Broker Host Error Input Dialog</i> . . . . .	22
Gambar 3.11	<i>Console Panel</i> . . . . .	23
Gambar 3.12	<i>Image Panel</i> . . . . .	24
Gambar 3.13	<i>Save Image Dialog</i> . . . . .	25
Gambar 3.14	<i>Status Panel</i> . . . . .	25
Gambar 3.15	<i>Variable History Panel</i> . . . . .	26
Gambar 3.16	<i>Command History Panel</i> . . . . .	27

## DAFTAR TABEL

Tabel 3.1	Deskripsi paramater pada pesan <i>request</i> . . . . .	13
Tabel 3.2	Contoh pesan <i>request</i> dan <i>response</i> . . . . .	13
Tabel 4.1	<i>Platform</i> pengujian MATLAB <i>Server</i> . . . . .	39
Tabel 4.2	<i>Platform</i> pengujian MATLAB <i>Client</i> . . . . .	39
Tabel 4.3	<i>Platform</i> pengujian RabbitMQ <i>message broker</i> . . . . .	39
Tabel 4.4	Pengujian dengan Hasil <i>String</i> . . . . .	40
Tabel 4.5	Pengujian dengan Hasil Gambar . . . . .	41
Tabel 4.6	Pengujian <i>Multi-User</i> dengan <i>Session</i> yang Berbeda . . . . .	43

## DAFTAR ALGORITMA

Algoritma 3.1	Inisialisasi koneksi pada MatlabRPC . . . . .	28
Algoritma 3.2	Pemakaian MatlabRPC oleh MATLAB <i>Client</i> . . . . .	29
Algoritma 3.3	Pengiriman <i>request</i> dan penerimaan <i>response</i> pada MATLAB <i>Client</i> . . . . .	30
Algoritma 3.4	<i>Variable History</i> pada MATLAB <i>Client</i> . . . . .	31
Algoritma 3.5	<i>Command History</i> pada MATLAB <i>Client</i> . . . . .	31
Algoritma 3.6	Menampilkan gambar pada MATLAB <i>Client</i> . . . . .	32
Algoritma 3.7	Menyimpan gambar pada MATLAB <i>Client</i> . . . . .	33
Algoritma 3.8	Akses RabbitMQ <i>message broker</i> pada MATLAB <i>Server</i> . . .	35
Algoritma 3.9	<i>Session Management</i> pada MATLAB <i>Server</i> . . . . .	36
Algoritma 3.10	Evaluasi <i>query</i> dan pengambilan hasil pada MATLAB <i>Server</i>	37
Algoritma 3.11	Pengambilan gambar pada MATLAB <i>Server</i> . . . . .	38
Algoritma 3.12	Tutup koneksi <i>engine</i> MATLAB pada MATLAB <i>Server</i> . . .	38

## DAFTAR LAMPIRAN

Lampiran 1	JMatLink <i>Method Summary</i> . . . . .	47
Lampiran 2	Instalasi MATLAB <i>Server</i> dan MATLAB <i>Client</i> (untuk <i>End User</i> ) . . . . .	50
Lampiran 3	Tahapan Kompilasi Kode Sumber MATLAB <i>Server</i> dan MATLAB <i>Client</i> (untuk <i>Developer</i> ) . . . . .	51

# BAB 1

## PENDAHULUAN

Bab ini menjelaskan tentang latar belakang dilakukannya penelitian, rumusan masalah, tujuan yang ingin dicapai, ruang lingkup yang membatasi penelitian, tahapan dan metodologi yang penulis gunakan dalam melaksanakan penelitian, serta sistematika penulisan laporan.

### 1.1 Latar Belakang

MATLAB adalah perangkat lunak yang digunakan sebagai lingkungan pengembangan algoritma, analisis data, visualisasi data, dan komputasi numerik [1]. Dengan kemampuan tersebut, MATLAB sering digunakan oleh kalangan akademisi, industri, dan *engineer* [2]. Namun, untuk beroperasi, MATLAB membutuhkan sumber daya yang besar. Oleh karena itu, diperlukan rancangan arsitektur sistem sehingga pembebanan sumber daya dalam menjalankan MATLAB dapat terbagi. Pada tugas akhir ini, penulis mencoba mengimplementasikan arsitektur berupa *client* dan *server* pada MATLAB. MATLAB *Client* berperan sebagai media interaksi bagi pengguna dalam meng-*input* perintah MATLAB dan menampilkan respon dari hasil pengolahan MATLAB *Server*.

Di sisi lain, rancangan sistem diharapkan dapat menangani *client* dalam jumlah banyak dan bersifat *scalable* dan *decouple*. Untuk permasalahan ini, *messaging pattern* diimplementasikan dalam sistem ini. *Messaging pattern* adalah arsitektur sistem berorientasi jaringan dengan basis pengiriman pesan dalam berkomunikasi antar sistem [3]. Dengan *messaging pattern*, sistem memiliki karakteristik *decouple* karena pengirim dan penerima pesan tidak mengetahui secara detil bagaimana pesan ditransmisikan [4]. Pada tugas akhir ini, sistem akan menggunakan RabbitMQ sebagai *message broker* yang menjembatani antar *server* dan *client*. Untuk komunikasi yang bersifat *synchronous*, sistem ini akan mengadopsi teknik *remote procedure call* dengan bantuan RabbitMQ.

RabbitMQ adalah *message broker* yang mengimplementasikan Advanced Message Queuing Protocol (AMQP). RabbitMQ dipilih karena memiliki karakter yang *reliable* dan cepat. Selain itu, RabbitMQ sudah diadopsi oleh banyak perusahaan - perusahaan besar, seperti Soocial.com, Ocean Observatories Initiative (OOI), Digg.com, NASA Nebula, Delicious.com, dan BBC [5] [6].



## 1.2 Permasalahan

Pada bagian ini akan dijelaskan mengenai definisi permasalahan yang Penulis hadapi dan ingin diselesaikan serta asumsi dan batasan yang digunakan dalam menyelesaikannya.

### 1.2.1 Definisi Permasalahan

Berikut adalah pertanyaan yang ingin dijawab melalui penelitian ini:

1. Bagaimana membangun arsitektur *client - server* dengan pendekatan *messaging* yang menggunakan RabbitMQ sebagai *message-oriented middle-ware*?
2. Bagaimana mengimplementasi metoda komunikasi *Remote Procedure Call* pada RabbitMQ?
3. Apa saja fitur yang diimplementasikan pada MATLAB *client* berbasis Java Swing (*desktop*)?
4. Bagaimana mengimplementasi MATLAB *server* yang terhubung pada RabbitMQ *message broker*?

### 1.2.2 Batasan Permasalahan

Beberapa batasan yang diberikan dalam melakukan penelitian ini:

1. Rancangan arsitektur terdiri dari aplikasi MATLAB *client*, MATLAB *server*, dan RabbitMQ *message broker*.
2. Hanya perintah - perintah MATLAB sederhana yang diuji coba.
3. MATLAB *client* dan MATLAB *server* diimplementasikan pada *localhost*.
4. RabbitMQ *message broker* diimplementasikan pada *virtual machine* yang terkoneksi dengan MATLAB *client* dan MATLAB *server* via *Local Area Network* (LAN).
5. Pada tahap uji coba, MATLAB *Client*, MATLAB *Server*, dan RabbitMQ *message broker* berada pada komputer yang terpisah dan saling terhubung via *Local Area Network* (LAN).

### 1.3 Tujuan

Tujuan dilakukannya penelitian ini adalah untuk menunjukkan metoda implementasi MATLAB *client - server* dengan arsitektur *messaging* yang menggunakan RabbitMQ sebagai *middleware / message broker*. RabbitMQ *message broker* menampung perintah MATLAB yang dikirim oleh MATLAB *client* dan MATLAB *server* melakukan proses komputasi berdasarkan perintah MATLAB yang tertampung pada RabbitMQ *message broker*. Pada penelitian ini, komunikasi antar MATLAB *client - server* menggunakan teknik *Remote Procedure Call* (RPC) yang disediakan oleh RabbitMQ.

### 1.4 Metodologi Penelitian

Laporan tugas akhir ini dilaksanakan dengan mengikuti tahapan-tahapan sebagai berikut.

#### 1. Studi Literatur

Pada tahap ini, studi literatur dilakukan untuk mempelajari arsitektur aplikasi *client* dan *server* dengan RabbitMQ sebagai *message broker*, teknik komunikasi RPC, akses MATLAB untuk kebutuhan MATLAB *server*, dan cara pembuatan MATLAB *client* dengan Java Swing (*desktop*).

#### 2. Requirement Gathering

Pada tahap ini, proses pengumpulan *requirement* dilakukan. Pada MATLAB *Client* dan MATLAB *Server*, *requirement* dikategorikan menjadi dua yaitu *functional requirement* dan *non-functional requirement*.

#### 3. Perancangan

Pada tahap ini, perancangan arsitektur aplikasi dilakukan. Rancangan berupa proses komunikasi *client* dan *server* dengan RabbitMQ *message broker*, *user interface* pada MATLAB *client*, dan proses pengaturan *session* dan komputasi MATLAB pada MATLAB *server*. Selain itu, hasil *requirement* yang sudah terkumpul sebelumnya diproses dan dirancang *use-case diagram* dan *activity diagram*.

#### 4. Implementasi

Pada tahap ini, arsitektur aplikasi yang sudah dirancang pada tahap sebelumnya diimplementasikan. Selama proses implementasi, perubahan rancangan pada tahap sebelumnya masih dapat dilakukan.

## 5. Pengujian

Pada tahap ini, aplikasi yang sudah diimplementasikan dilakukan uji coba. Jenis pengujian yang dilakukan adalah *scenario testing*.

## 1.5 Sistematika Penulisan

Sistematika penulisan laporan adalah sebagai berikut:

- Bab 1 PENDAHULUAN  
Bab 1 berisi penjelasan mengenai latar belakang penelitian, rumusan masalah yang dihadapi, tujuan penelitian yang ingin dicapai, batasan dan ruang lingkup penelitian, metodologi penelitian, dan sistematika penulisan.
- Bab 2 LANDASAN TEORI  
Bab 2 menjelaskan teori - teori penunjang yang digunakan dalam penelitian ini.
- Bab 3 PERANCANGAN DAN IMPLEMENTASI  
Bab 3 berisi proses perancangan dan pengembangan aplikasi MATLAB *client* dan *server* dengan *messaging* yang menggunakan RabbitMQ sebagai *middle-ware / message broker*.
- Bab 4 PENGUJIAN  
Bab 4 berisi skema uji coba yang digunakan untuk pengujian terhadap hasil implementasi aplikasi.
- Bab 5 PENUTUP  
Bab 5 berisi kesimpulan dan saran dari hasil penelitian yang telah dilakukan.

## BAB 2

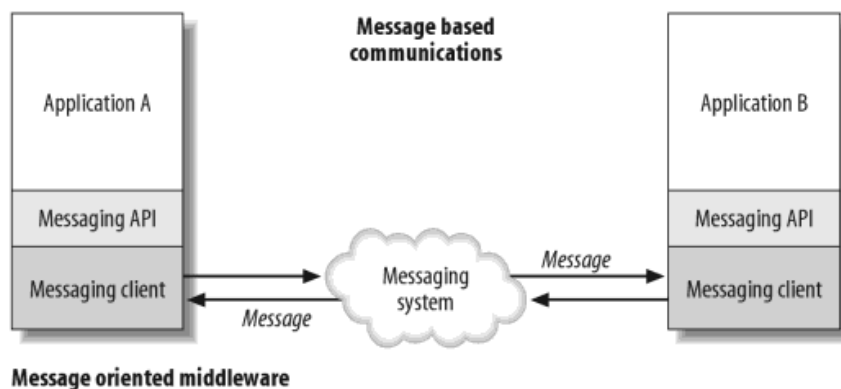
### LANDASAN TEORI

Bab ini berisi penjelasan mengenai landasan teori yang digunakan dalam tugas akhir ini. Teori - teori yang digunakan adalah *message-oriented middleware*, RabbitMQ, MATLAB, dan *remote procedure call*.

#### 2.1 *Message-Oriented Middleware*

*Message-Oriented Middleware* (MOM) adalah konsep pertukaran data antar aplikasi yang menggunakan kanal komunikasi dengan unit informasi berupa *message* [3]. Dengan mekanisme ini, aplikasi saling berkomunikasi secara abstrak dan *decouple* sehingga antara pengirim dan penerima *message* tidak perlu saling *aware*. Dengan kata lain, pengirim dan penerima *message* cukup mengakses sistem *messaging* untuk saling berkomunikasi secara *asynchronous*

Pada sistem *messaging*, aplikasi menggunakan API (yang disediakan oleh *vendor* MOM) untuk berkomunikasi. Gambar 2.1 menunjukkan mekanisme komunikasi *message-oriented middleware*.



**Gambar 2.1:** Komunikasi *message-oriented middleware* [3]

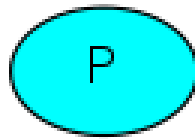
Sistem *messaging* berfungsi untuk mengatur *connection point* antar *client* dan mengatur *channel* antar *connection point*. Umumnya, sistem *messaging* diimplementasikan sebagai *message server* atau *message broker*.

## 2.2 RabbitMQ

RabbitMQ adalah aplikasi *open source* berbasis Erlang yang berfungsi sebagai *message broker* atau *message-oriented middleware* [7]. Implementasi RabbitMQ mengacu pada *application layer protocol* yaitu *Advanced Message Queuing Protocol* (AMQP). AMQP menyediakan standar protokol yang *interoperable* antar *vendor* untuk mengatur pertukaran *message* pada sistem berskala *enterprise* [8].

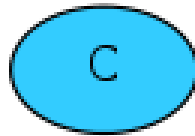
Terminologi yang dipakai dalam *messaging pattern* adalah sebagai berikut [9]:

- *Producer* adalah entitas yang berperan mengirim *message*.



**Gambar 2.2:** Notasi *producer* [9]

- *Consumer* adalah entitas yang berperan menerima *message*.



**Gambar 2.3:** Notasi *consumer* [9]

- *Queue* adalah entitas yang berperan menampung *message-message* yang dikirim oleh *producer*. Kemudian, *consumer* yang akan mengambil *message-message* tersebut.

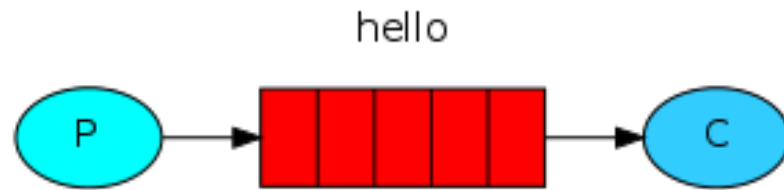


**Gambar 2.4:** Notasi *queue* [9]

Beberapa *messaging pattern* yang dapat diimplementasikan pada RabbitMQ adalah sebagai berikut :

- *Simple Queues*

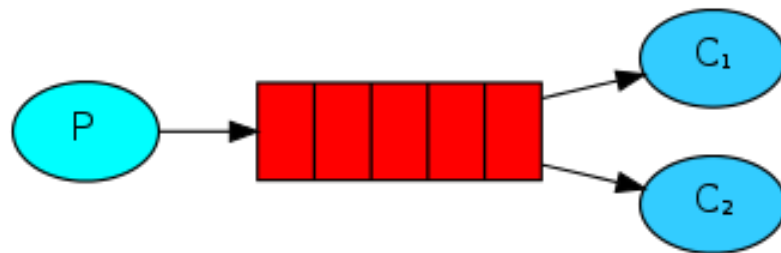
*Producer* mengirim *message* ke *queue* dan di-consume oleh *consumer*.



**Gambar 2.5:** Simple Queues [9]

- *Work Queues*

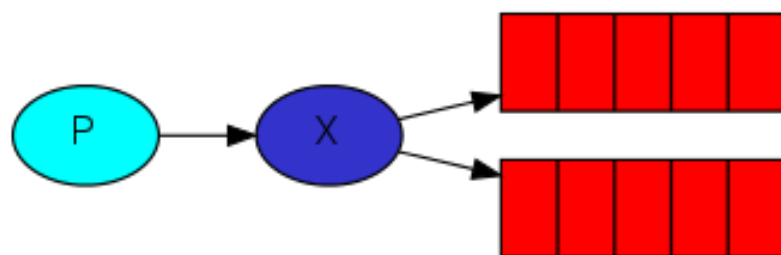
Pada metode ini, *message* pada *queue* digunakan oleh lebih dari satu *consumer*. Tujuannya adalah untuk menghindari *task* (pada *message*) yang membutuhkan *resource* yang intensif sehingga *message* yang masih berada di *queue* dapat di-*consume* oleh *consumer* lain [10].



**Gambar 2.6:** Work Queues [10]

- *Publish/Subscribe*

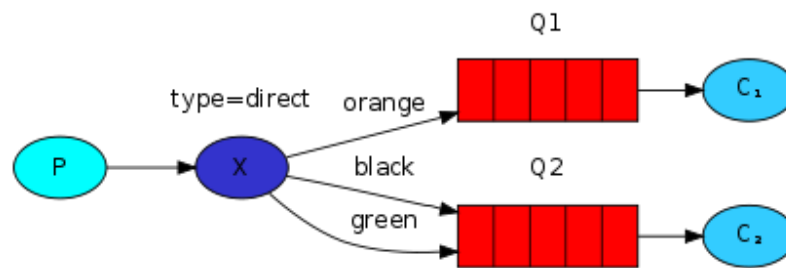
Pada metode ini, *message* dapat dikirim ke banyak *consumer* dengan *queue* yang berbeda untuk masing - masing *consumer* [11].



**Gambar 2.7:** Publish/Subscribe [11]

- *Routing*

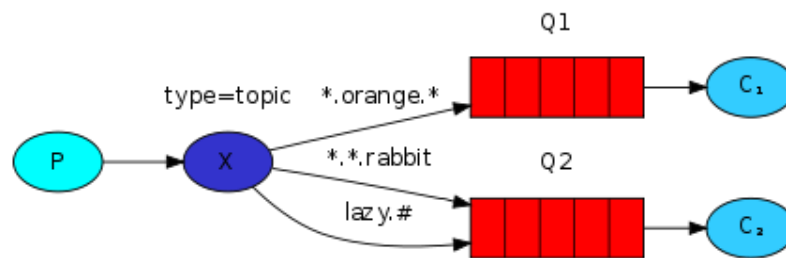
Pada metode ini, *message* dapat dikirim ke banyak *consumer* dengan *queue* yang berbeda untuk masing - masing *consumer*. Selain itu, *message* dapat diarahkan ke *queue* tertentu berdasarkan *routing key* [12].



Gambar 2.8: Routing [12]

- *Topics*

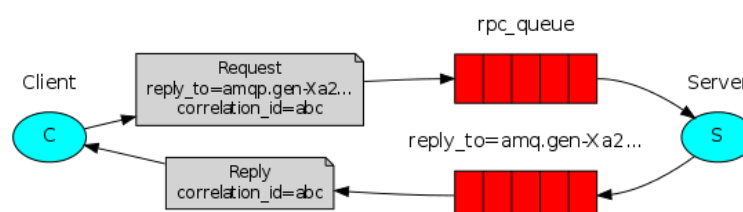
Pada metode ini, *message* dapat dikirim ke banyak *consumer* dengan *queue* yang berbeda untuk masing - masing *consumer*. Selain itu, *message* dapat diarahkan ke *queue* tertentu berdasarkan *routing key* yang dapat terdiri dari satu kata atau lebih [13].



Gambar 2.9: Topics [13]

- *Remote Procedure Call (RPC)*

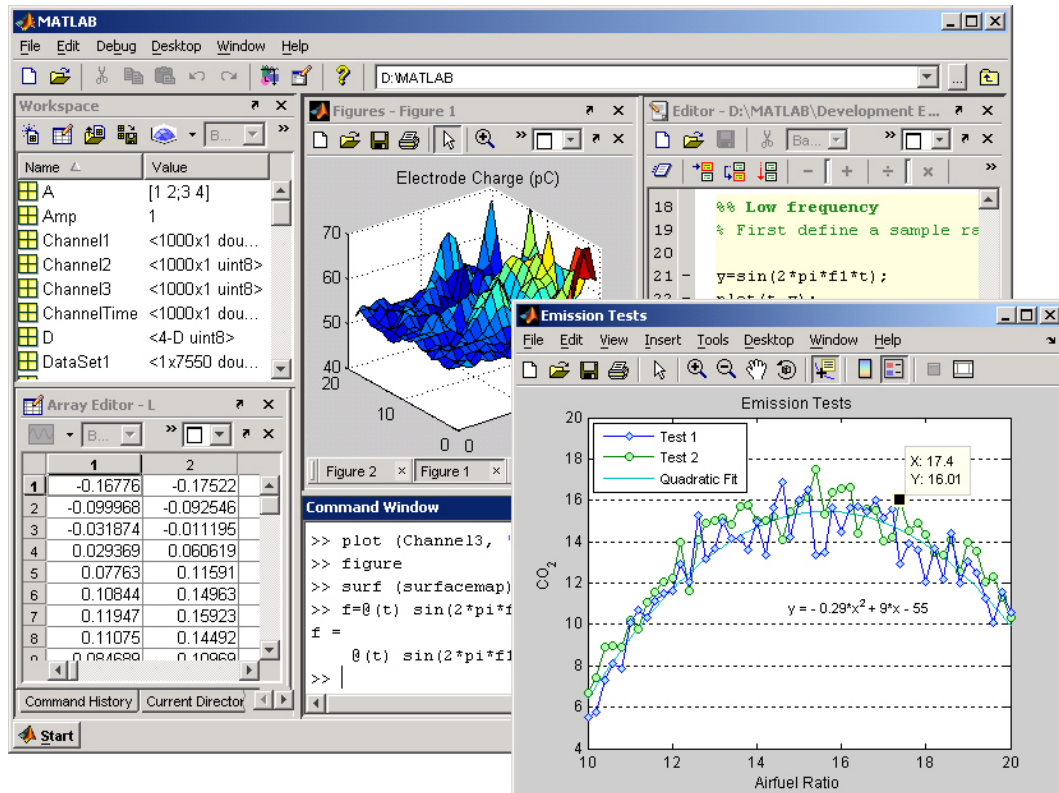
Metode ini digunakan ketika *client* menjalankan suatu fungsi dan menunggu hasilnya [14]. Dengan cara ini, teknik komunikasi menjadi bersifat *synchronous*.



Gambar 2.10: Remote Procedure Call [14]

## 2.3 MATLAB

MATLAB (*Matrix Laboratory*) adalah bahasa komputasi teknis tingkat tinggi dan lingkungan interaktif untuk pengembangan algoritma, visualisasi data, analisis data, dan komputasi numerik. Dengan aplikasi ini, masalah komputasi teknis dapat diselesaikan lebih cepat dibandingkan dengan penggunaan bahasa pemrograman tradisional seperti C, C++, dan Fortran [1].



Gambar 2.11: MATLAB [1]

### 2.3.1 JMatLink

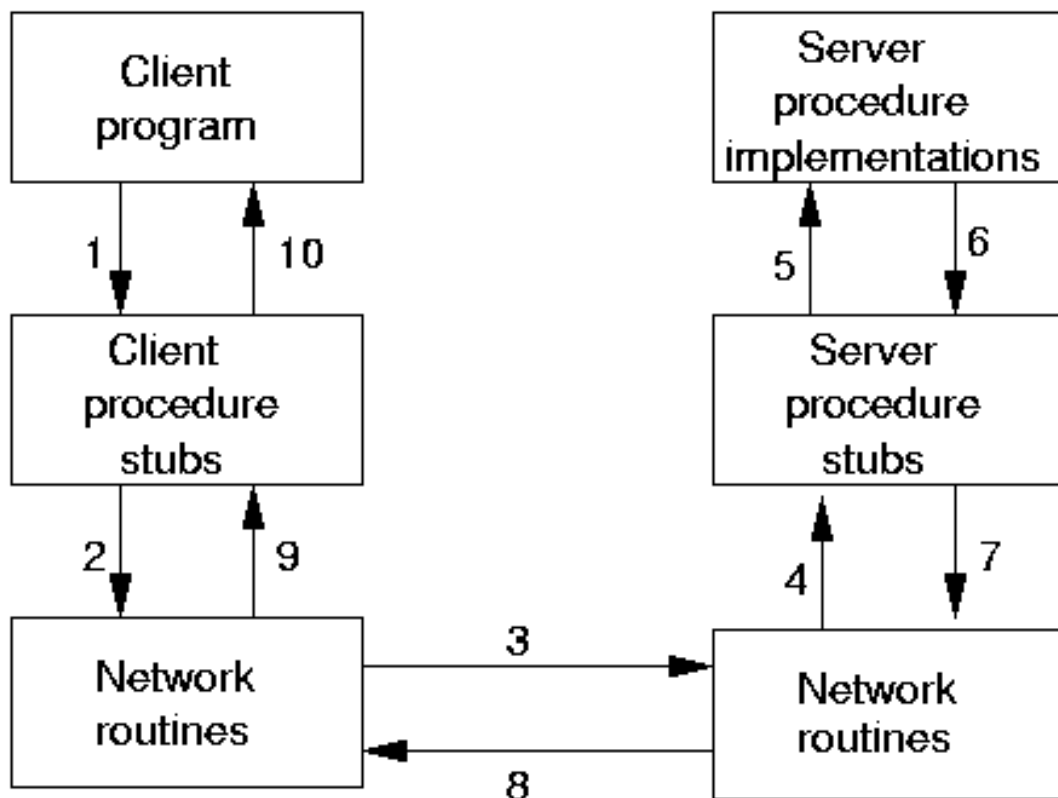
JMatLink adalah *library* yang menyediakan fungsionalitas MATLAB untuk digunakan dalam bahasa pemrograman Java [15]. JMatLink menghubungkan Java dan MATLAB dengan menggunakan *Java Native Interface* (JNI).

## 2.4 Remote Procedure Call (RPC)

*Remote Procedure Call* (RPC) adalah mekanisme komunikasi pada suatu program yang dapat memanggil *subroutine* atau *procedure* pada program lain yang berada di *address space* / komputer yang berbeda. RPC biasa digunakan sebagai



mekanisme komunikasi antar program komputer yang berada dalam suatu jaringan. Komunikasi RPC dimulai oleh *client* yang mengirim pesan ke *remote server* untuk melakukan eksekusi terhadap *procedure* tertentu. Karena bersifat *synchronous*, *client* menunggu hingga *remote server* selesai memroses kemudian *client* menerima respon dari hasil eksekusi *procedure* tersebut. Keuntungan dari RPC adalah kemudahan *developer* dalam mengeksekusi *procedure* pada program lain (*remote call*) seolah-olah seperti memanggil *local procedure* (*local call*). Namun, tidak seperti *local call*, isu mengenai masalah jaringan perlu diperhatikan dalam RPC. Gambar 2.12 menunjukkan diagram alur eksekusi pada RPC secara umum.



Gambar 2.12: Diagram alur eksekusi pada RPC

## 2.5 Teknologi Java

Teknologi Java terdiri dari bahasa pemrograman dan *platform*. Bahasa pemrograman Java adalah bahasa pemrograman yang bersifat *object-oriented*, *portable*, *multithreaded*, dan *robust*. *Platform* Java terdiri dari dua komponen yaitu *Java Virtual Machine* dan *Java Application Programming Interface (API)*. *Java Virtual Machine* adalah *virtual machine* yang menjalankan *bytecodes* yang dihasilkan oleh *compiler*. *Java Application Programming Interface (API)* adalah kumpulan komponen *software* dengan berbagai macam fungsionalitas [16].

## 2.6 *Software Development Life Cycle*

*Software Development Life Cycle* (SDLC) adalah kerangka kerja yang mendefinisikan tahapan - tahapan yang terlibat dalam pengembangan perangkat lunak komputer [17]. Secara umum, pengembangan perangkat lunak terdiri dari tiga tahapan [18]:

### 1. *Definition Stage*

Tahapan ini terdiri dari *requirement phase* dan *specification phase*. Pada tahapan ini, *user requirement* dikumpulkan dan dianalisis untuk mendapatkan spesifikasi perangkat lunak.

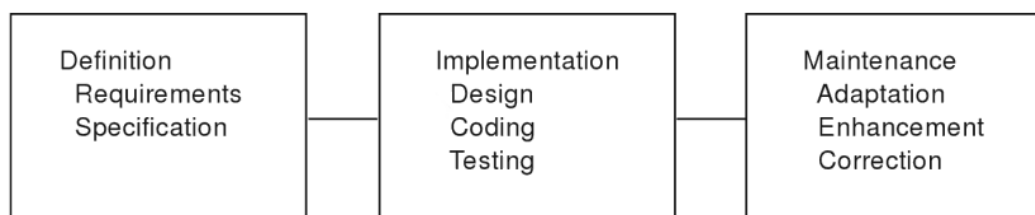
### 2. *Implementation Stage*

Pada tahapan ini, *design phase* dilakukan untuk menjabarkan spesifikasi perangkat lunak sehingga mendapatkan basis dalam implementasi konkret perangkat lunak. Setelah itu, *coding* dan pengetesan dilakukan sehingga sesuai dengan spesifikasi.

### 3. *Maintenance Stage*

Tahapan ini mengatur perubahan perangkat lunak untuk memperbaiki *error* atau menambah fitur pada perangkat lunak. Perubahan dapat berupa perubahan pada *requirement*, spesifikasi, *design*, atau implementasi perangkat lunak.

Gambar 2.13 menunjukkan tiga tahapan SDLC secara umum.



**Gambar 2.13:** Tiga tahapan umum SDLC [18]

## BAB 3

### PERANCANGAN DAN IMPLEMENTASI

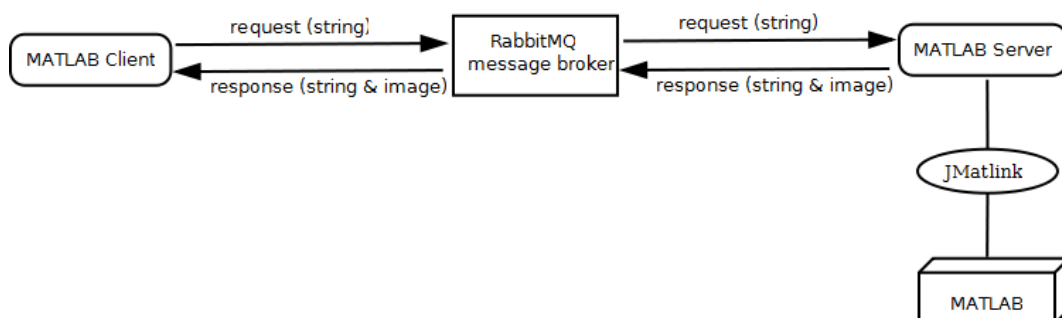
Pada bagian ini akan dijelaskan mengenai rancangan arsitektur sistem yang akan dibangun. Sistem yang akan dirancang terdiri dari MATLAB *client* dan MATLAB *server*. Pada akhir bab, RabbitMQ diinstal sebagai *message broker*.

#### 3.1 Arsitektur Sistem

Arsitektur sistem yang akan diimplementasikan terdiri dari tiga komponen utama :

- MATLAB *Client*
- MATLAB *Server*
- RabbitMQ *Message Broker*

Dalam arsitektur ini, MATLAB *Client* dan MATLAB *Server* saling berkomunikasi melalui RabbitMQ *message broker*. Keduanya mengakses RabbitMQ *message broker* untuk mengambil dan mengirim pesan. Perintah - perintah MATLAB akan dijalankan oleh MATLAB *Server* dengan menggunakan JMatlink untuk mengakses mesin komputasi MATLAB. Arsitektur sistem ditunjukkan pada Gambar 3.1.



**Gambar 3.1:** Arsitektur sistem

##### 3.1.1 Request Format Data

Pesan *request* yang dikirim ke RabbitMQ *message broker* (oleh MATLAB *Client*) dan diambil MATLAB *Server* menggunakan format berupa *string* dengan pemisah (*delimiter*) berupa “|” Ada tiga macam format pesan *request* berdasarkan jenis *request* yaitu :

- Eksekusi *Query*

```
query | <token> | <query>
```

- Ambil Gambar

```
getimage | <token>
```

- Tutup Koneksi

```
close | <token>
```

Penjelasan tiap parameter dijabarkan pada Tabel 3.1.

**Tabel 3.1:** Deskripsi paramater pada pesan *request*

Parameter	Deskripsi	Contoh
<token>	Tanda pengenal unik & <i>random</i> yang dimiliki oleh tiap MATLAB <i>Client</i>	4b54-9e02
<query>	<i>Query</i> perintah MATLAB yang akan dieksekusi	a = 45 + 56;

### 3.1.2 Response Format Data

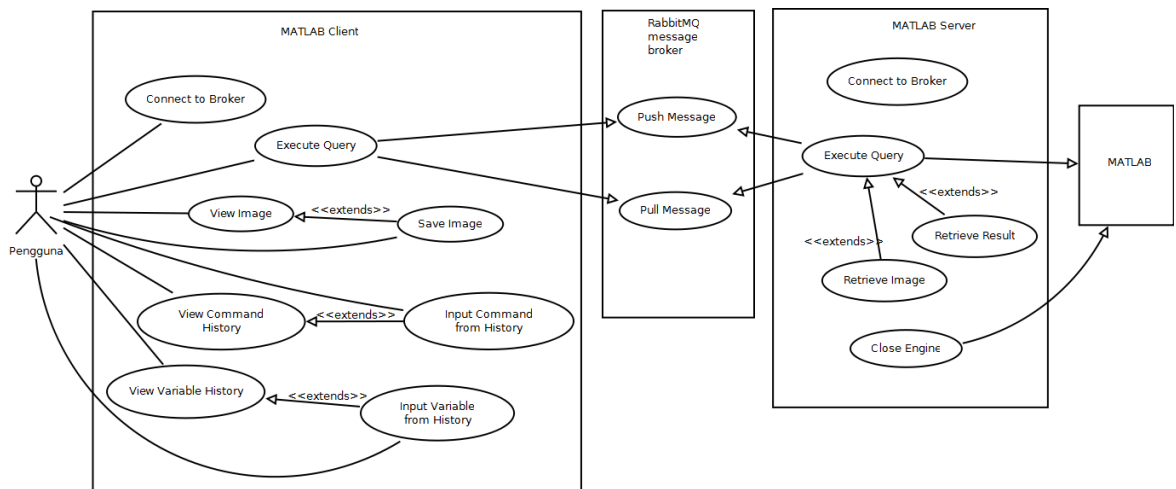
Pesan *request* yang dikirim oleh MATLAB *Client* akan diterima dan diolah oleh MATLAB *Server*. Hasil komputasi dan pengolahan tersebut akan dikirim balik oleh MATLAB *Server* menuju RabbitMQ *message broker* untuk diterima oleh MATLAB *Client*.

Pesan *response* tersebut dibedakan menjadi dua jenis berdasarkan jenis pesan *request* yang akan dikirim oleh MATLAB *Client*. Jika pesan *request* adalah perintah berupa eksekusi *query* (*query*), format pesan *response* adalah berupa *string*. Sedangkan, jika pesan *request* adalah perintah berupa pengambilan gambar (*getimage*), format pesan *response* adalah berupa gambar dalam bentuk *byte array*. Contoh pesan *request* dan *response* dapat dilihat pada Tabel 3.2.

**Tabel 3.2:** Contoh pesan *request* dan *response*

Pesan <i>request</i>	Pesan <i>response</i> yang dihasilkan
query   4b54-9e02   a = 45 + 56	a = 101
getimage   4b54-9e02	11001110101010101

## 3.2 Use Case Diagram



Gambar 3.2: Use Case Diagram

Gambar 3.2 menunjukkan *use-case diagram* yang dirancang pada arsitektur sistem. *Use-case diagram* terdiri dari tiga buah komponen utama *use-case* yang saling berhubungan. *Use-case* yang dipilih adalah representasi *functional requirement* yang dimiliki oleh MATLAB Client dan MATLAB Server.

## 3.3 MATLAB Client

### 3.3.1 Functional Requirement

Berikut ini adalah fungsi-fungsi utama yang dimiliki oleh MATLAB Client.

#### 3.3.1.1 Connect to Message Broker

Fungsi ini digunakan untuk mengkoneksikan MATLAB Client dengan RabbitMQ message broker.

#### 3.3.1.2 Execute Query

Fungsi ini digunakan untuk mengeksekusi *query* perintah MATLAB.

#### 3.3.1.3 View Image

Fungsi ini menyatakan bahwa aplikasi dapat menampilkan gambar jika *query* MATLAB yang dieksekusi menghasilkan gambar atau grafik.

#### **3.3.1.4 *Save Image***

Fungsi ini digunakan untuk menyimpan gambar yang ditampilkan oleh aplikasi.

#### **3.3.1.5 *View Variable History***

Fungsi ini digunakan untuk menampilkan variabel - variabel dari perintah MATLAB yang sebelumnya pernah dieksekusi. Tujuannya adalah memudahkan pengguna untuk melihat nilai variabel - variabel yang sebelumnya pernah dieksekusi.

#### **3.3.1.6 *Input Variable From History***

Fungsi ini digunakan untuk meng-*input* kembali variabel yang terdapat pada *variable history* ke dalam *query editor*. Tujuannya adalah memudahkan pengguna untuk menggunakan kembali variabel - variabel yang sebelumnya pernah dieksekusi.

#### **3.3.1.7 *View Command History***

Fungsi ini digunakan untuk menampilkan perintah - perintah MATLAB yang sebelumnya pernah dieksekusi.

#### **3.3.1.8 *Input Command From History***

Fungsi ini digunakan untuk meng-*input* kembali perintah yang terdapat pada *command history* ke dalam *query editor*. Tujuannya adalah memudahkan pengguna untuk menggunakan kembali perintah - perintah yang sebelumnya pernah dieksekusi.

### **3.3.2 *Nonfunctional Requirement***

#### **3.3.2.1 *User Interface Requirement***

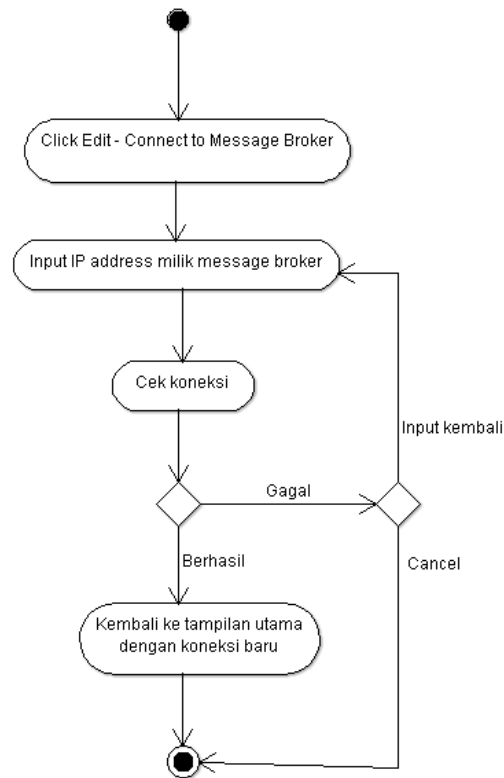
Aplikasi diharapkan memiliki tampilan yang mudah untuk dioperasikan oleh pengguna.

#### **3.3.2.2 *Error Handling Requirement***

Aplikasi dapat menangani *error* jika terdapat kesalahan pengoperasian dari pengguna. Aplikasi diharapkan dapat memberikan respon berdasarkan *error* yang terjadi.

### 3.3.3 Activity Diagram

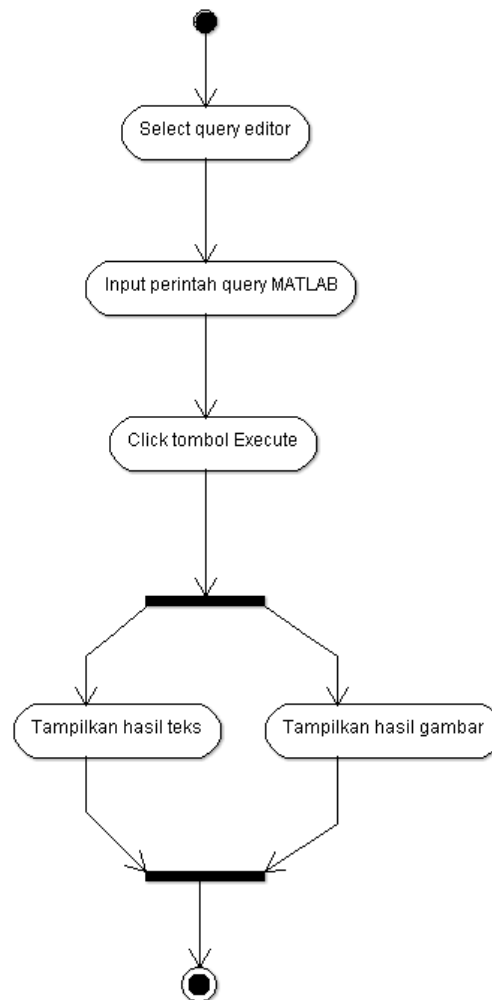
#### 1. Connect to Message Broker



**Gambar 3.3:** Activity Diagram untuk Connect to Message Broker

Gambar 3.3 menunjukkan langkah - langkah bagaimana pengguna dapat terhubung dengan *message broker host* (RabbitMQ). Pada langkah awal, sistem akan memunculkan *dialog input*. Kemudian, pengguna memasukkan *ip address* dari *message broker host* yang ingin diakses. Jika sistem berhasil tersambung dengan *message broker*, pengguna dapat mengakses menu utama. Namun, jika gagal, sistem akan memunculkan kembali *dialog input*.

## 2. Execute Query

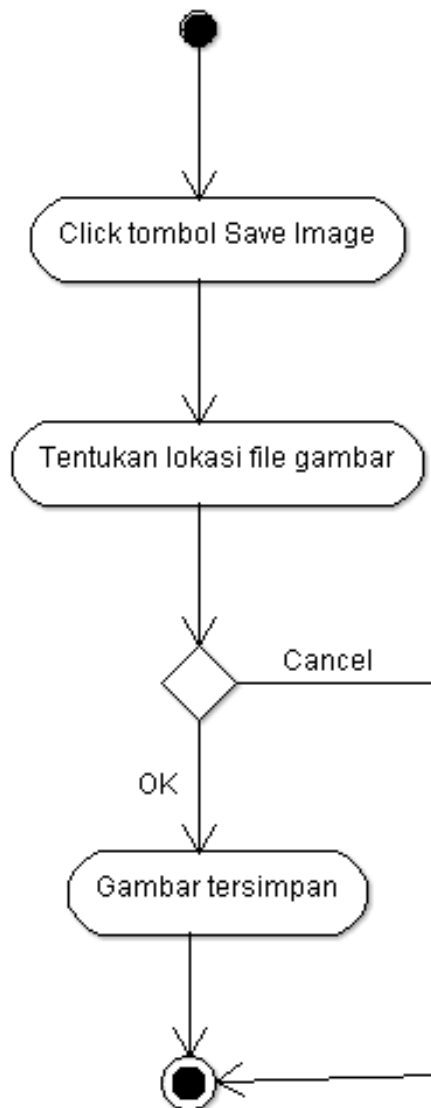


**Gambar 3.4:** Activity Diagram untuk Execute Query

Gambar 3.4 menunjukkan langkah - langkah bagaimana pengguna dapat memberi perintah / *query* MATLAB untuk dieksekusi oleh MATLAB Server. Pada langkah awal, pengguna meng-*input query* MATLAB pada *console panel*. Hasil respon berupa *string* akan ditampilkan pada *console panel* dan gambar akan ditampilkan pada *image panel*.



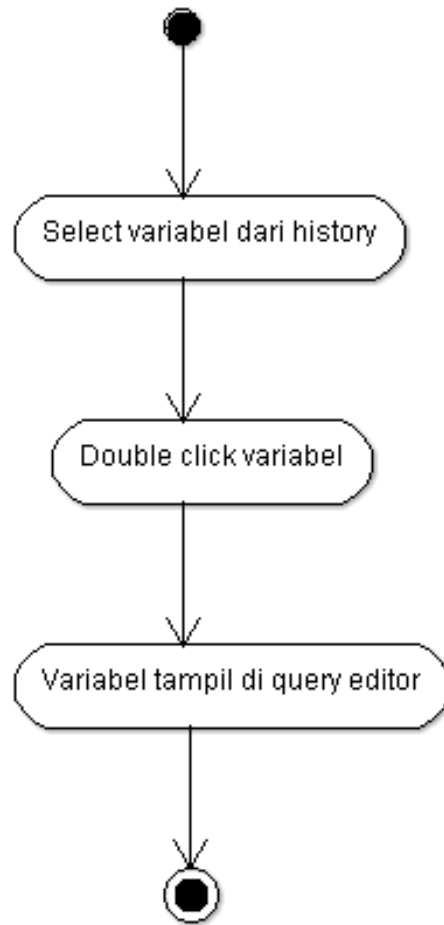
### 3. Save Image



**Gambar 3.5:** Activity Diagram untuk Save Image

Gambar 3.5 menunjukkan langkah - langkah bagaimana pengguna dapat menyimpan gambar yang ditampilkan pada MATLAB Client. Pada langkah awal, pengguna mengklik tombol “Save Image” pada *image panel*. Kemudian, sistem menampilkan *dialog* untuk menentukan lokasi penyimpanan file. Gambar akan disimpan jika pengguna sudah menentukan lokasi penyimpanan dan mengklik tombol “OK”.

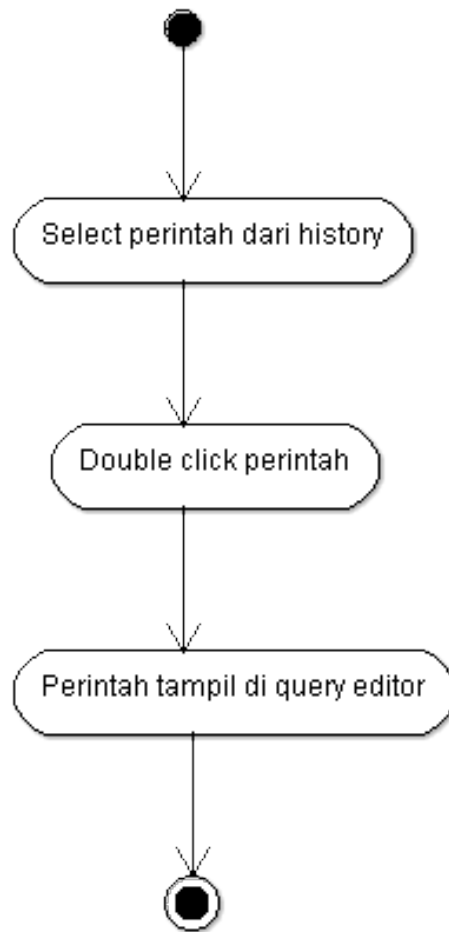
#### 4. *Input Variable from History*



**Gambar 3.6:** *Activity Diagram untuk Input Variable from History*

Gambar 3.6 menunjukkan langkah - langkah bagaimana pengguna dapat menginput kembali variabel - variabel yang sebelumnya sudah memiliki nilai. Pada langkah awal, pengguna memilih variabel yang berada pada *variable history*. Kemudian, melakukan *double click* pada variabel tersebut. Setelah itu, variabel tersebut akan tampil pada *console panel*.

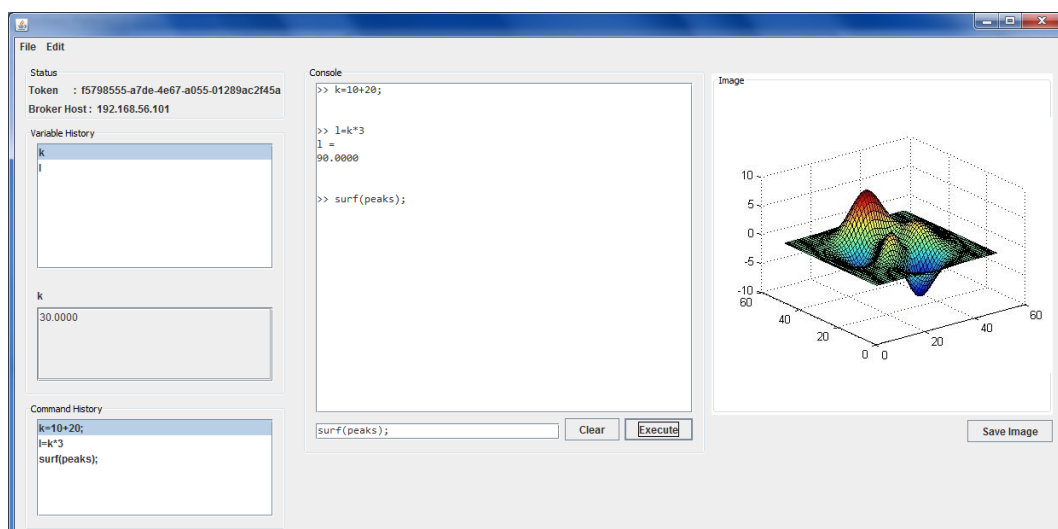
### 5. *Input Command from History*



**Gambar 3.7:** *Activity Diagram untuk Input Command from History*

Gambar 3.7 menunjukkan langkah - langkah bagaimana pengguna dapat menginput kembali perintah - perintah yang sebelumnya pernah dieksekusi. Pada langkah awal, pengguna memilih perintah yang berada pada *command history*. Kemudian, melakukan *double click* pada perintah tersebut. Setelah itu, perintah tersebut akan tampil pada *console panel*.

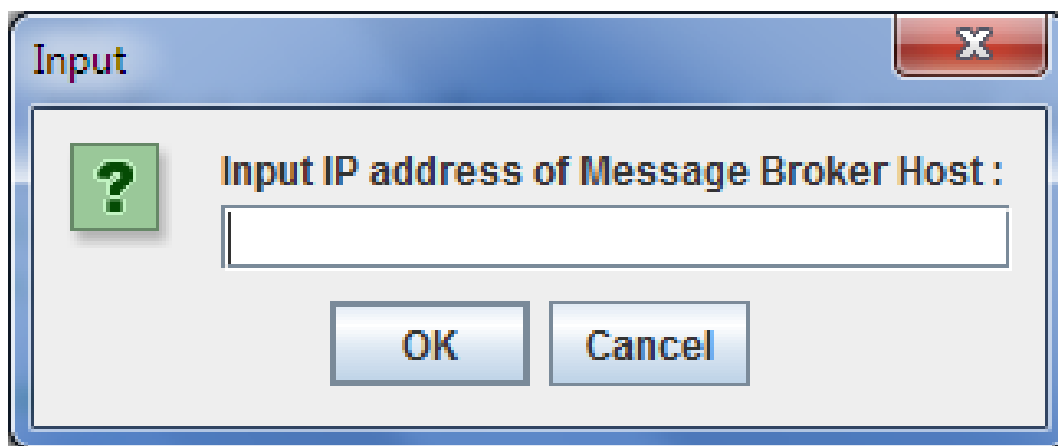
### 3.3.4 User Interface



**Gambar 3.8:** Tampilan utama MATLAB Client

User interface pada MATLAB Client dibuat dengan menggunakan Java Swing. Desain user interface dipermudah dengan bantuan Swing GUI Builder (Matisse) yang dimiliki oleh Netbeans IDE. Gambar 3.8 adalah tampilan utama dari MATLAB Client. Berikut ini adalah penjelasan dari masing-masing bagian tampilan yang dikembangkan pada MATLAB Client.

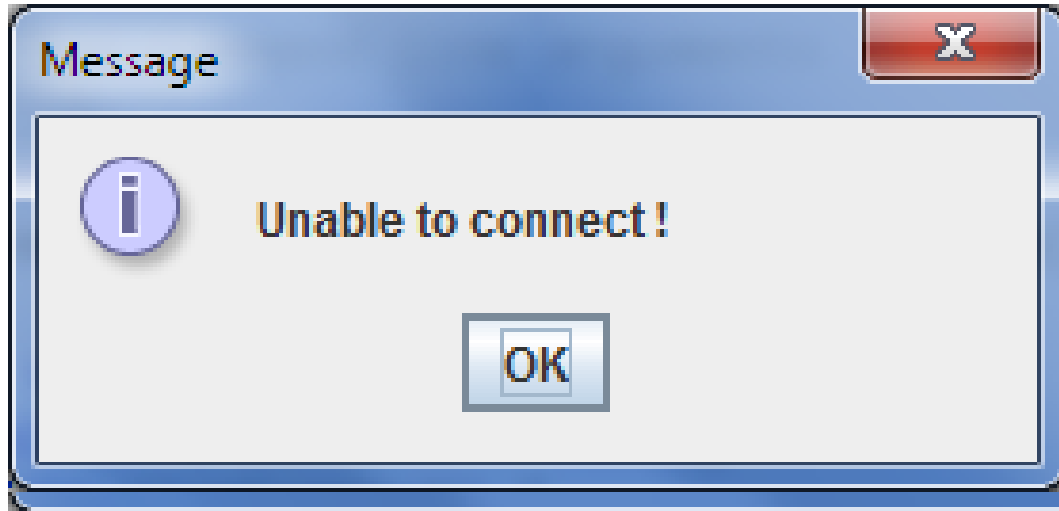
#### 3.3.4.1 Message Broker Host Input Dialog



**Gambar 3.9:** Message Broker Host Input Dialog

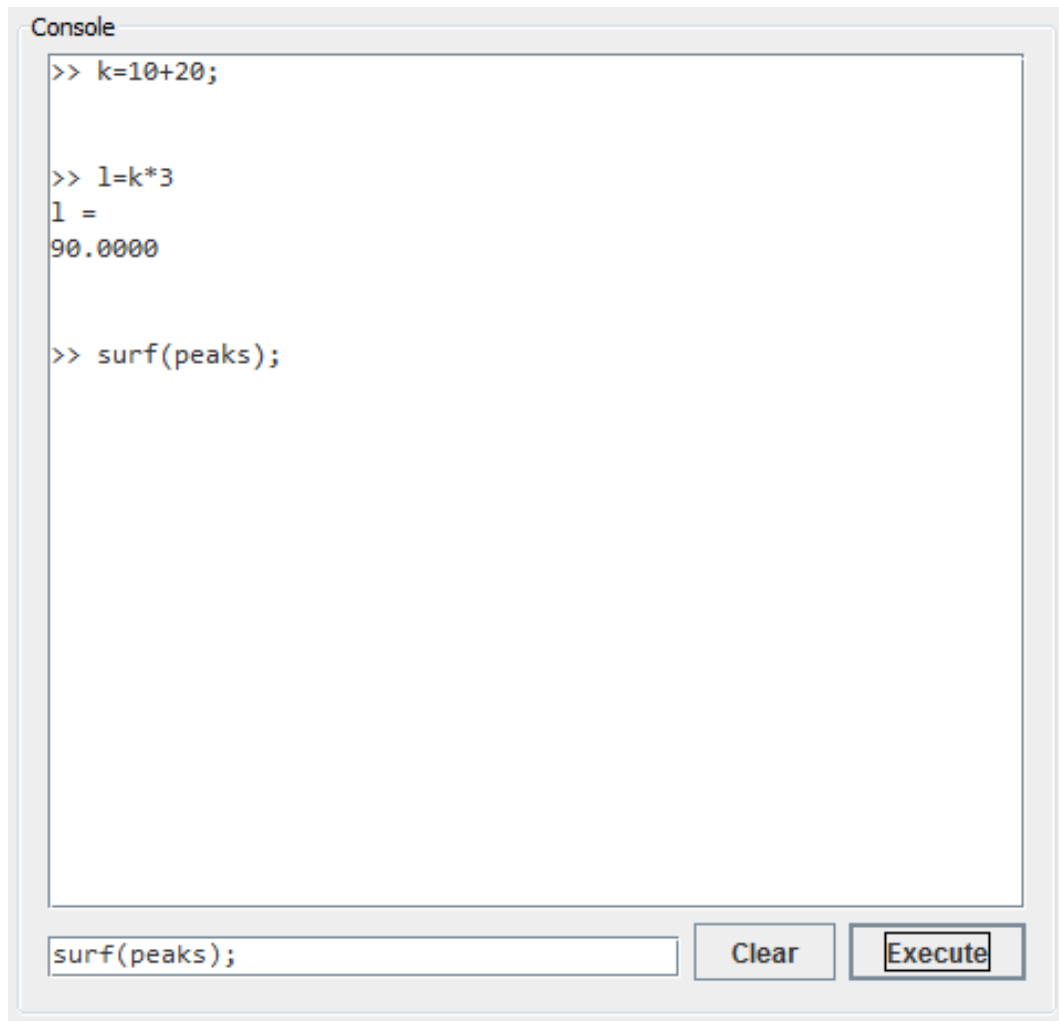
Gambar 3.9 adalah tampilan untuk memasukkan IP address message broker yang akan terkoneksi dengan MATLAB Client. User interface ini akan tampil pertama kali sebelum pengguna mengakses tampilan utama MATLAB Client. Tampilan ini

juga dapat diakses melalui menu Edit - Connect to Message Broker Host... Jika aplikasi tidak mendapatkan koneksi dari *message broker host*, tampilan pada Gambar 3.10 akan muncul.



**Gambar 3.10:** *Message Broker Host Error Input Dialog*

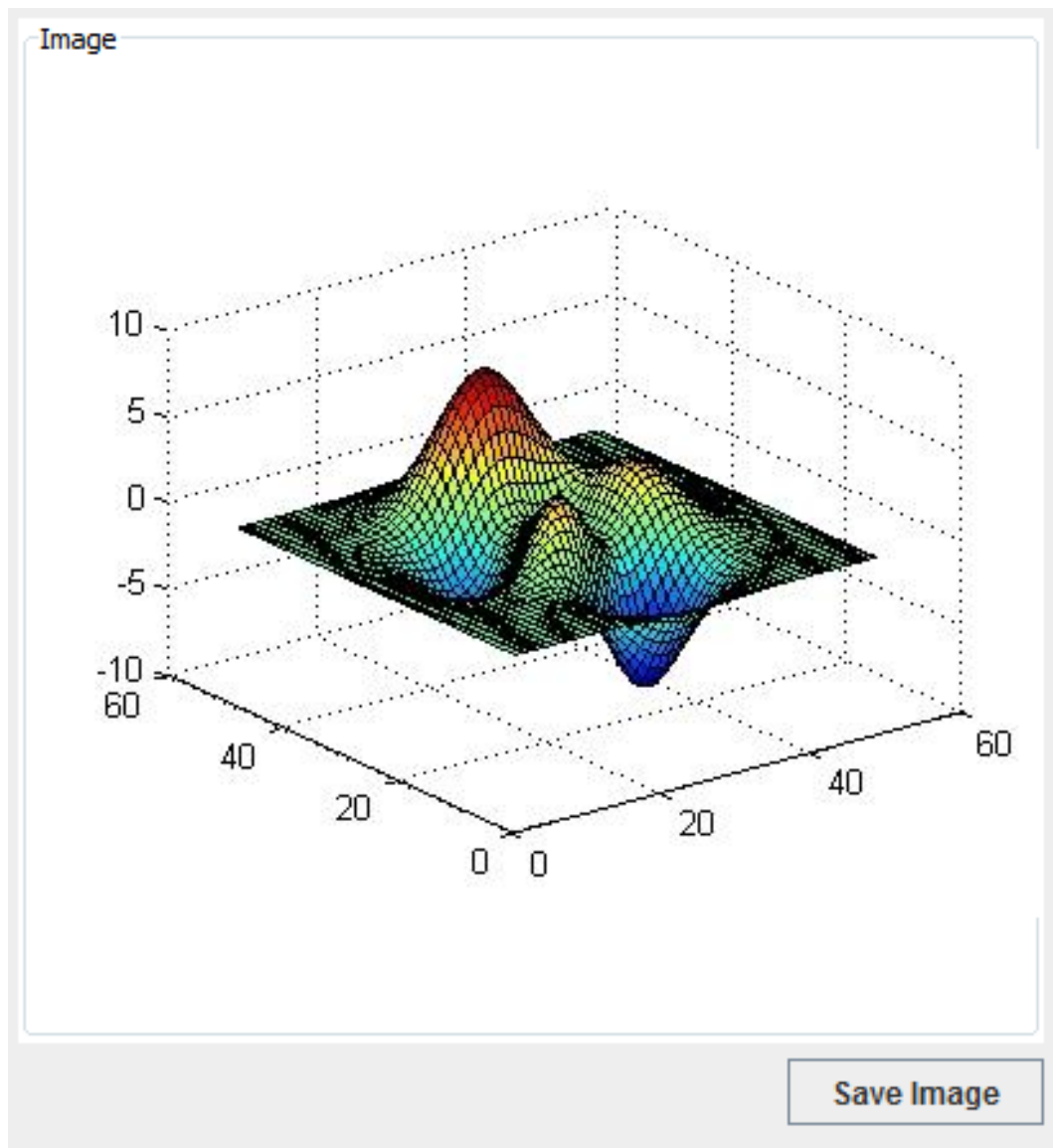
### 3.3.4.2 Console Panel



**Gambar 3.11:** Console Panel

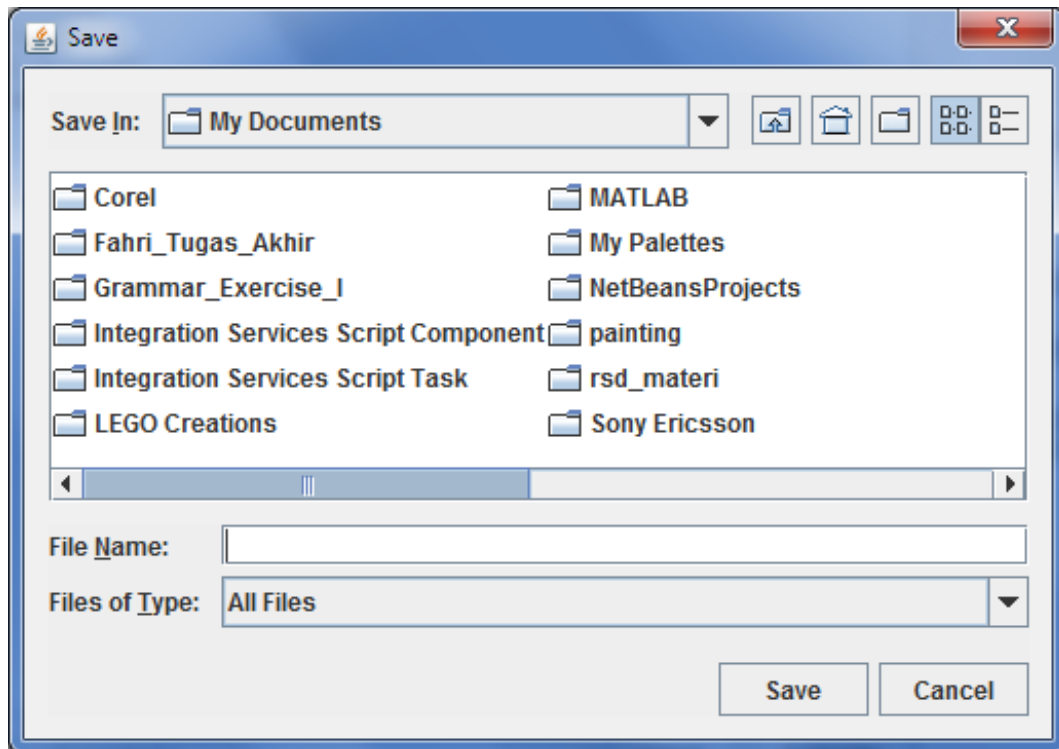
Gambar 3.11 adalah *console* yang menampilkan *query* perintah MATLAB yang di-*input* oleh pengguna dan respon dari hasil komputasi MATLAB berdasarkan perintah tersebut. Pengguna mengisi *query* perintah MATLAB pada *text field* kemudian mengklik tombol Execute untuk mengirim dan mendapatkan respon dari MATLAB. Tombol Clear berguna untuk mengosongkan *text field*.

### 3.3.4.3 Image Panel



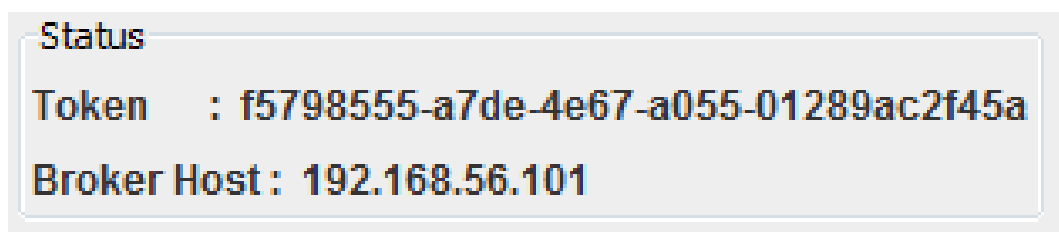
**Gambar 3.12:** *Image Panel*

Gambar 3.12 adalah *panel* yang menampilkan gambar jika perintah MATLAB yang dieksekusi memberikan respon berupa gambar / grafik. *Dialog* pada Gambar 3.13 akan muncul untuk menentukan nama dan lokasi *file* gambar yang ingin disimpan.



**Gambar 3.13:** *Save Image Dialog*

#### 3.3.4.4 *Status Panel*

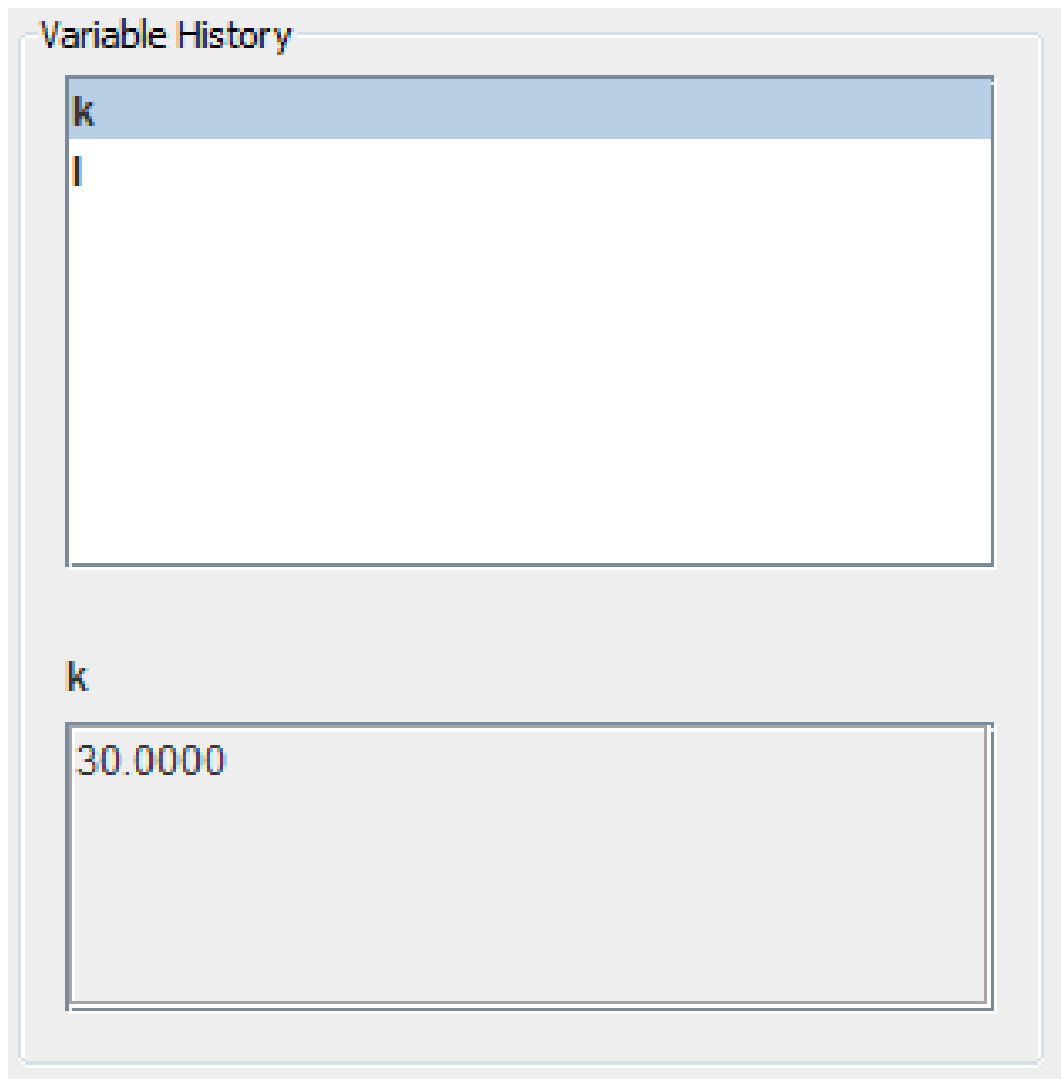


**Gambar 3.14:** *Status Panel*

Gambar 3.14 adalah *panel* yang menampilkan status *token* dan *IP address* milik *message broker host*. *Token* adalah tanda pengenal bagi *MATLAB Client* yang digunakan oleh *MATLAB Server* untuk mengatur *session* tiap *MATLAB Client*.



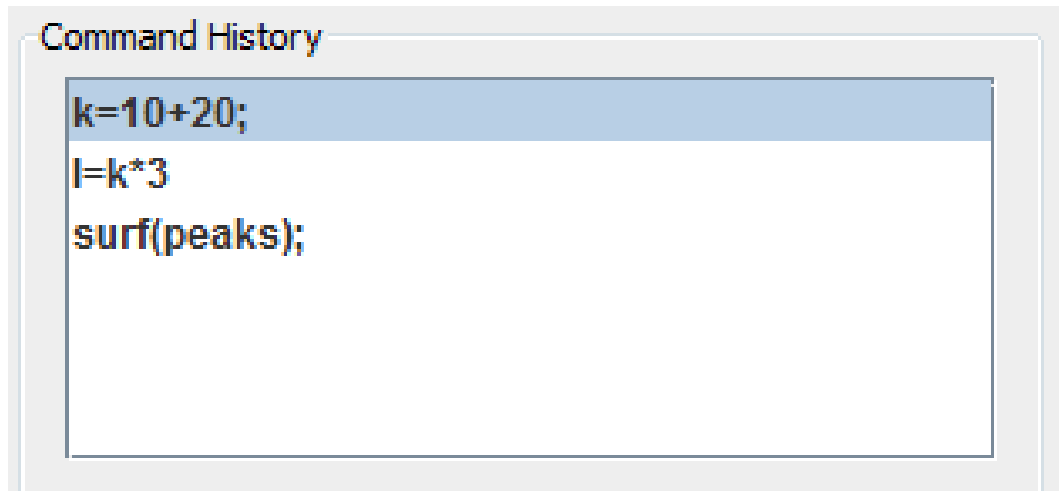
### 3.3.4.5 Variable History Panel



**Gambar 3.15:** *Variable History Panel*

Gambar 3.15 adalah daftar variabel hasil dari respon perintah MATLAB yang sebelumnya pernah dieksekusi. Bagian bawah *panel* menampilkan nilai dari variabel yang di-*select*. Jika pengguna melakukan *double click* pada variabel yang dipilih, variabel tersebut akan dimasukkan kedalam *text field* pada *console* untuk dipakai sebagai bagian dari perintah MATLAB yang akan dieksekusi.

### 3.3.4.6 Command History Panel



**Gambar 3.16:** *Command History Panel*

Gambar 3.16 adalah daftar perintah MATLAB yang sebelumnya pernah dieksekusi. Jika pengguna melakukan *double click* pada perintah yang dipilih, perintah tersebut akan dimasukkan kedalam *text field* pada *console* untuk dipakai sebagai bagian dari perintah MATLAB yang akan dieksekusi.

### 3.3.5 Teknik dan Algoritma

#### 3.3.5.1 Akses RabbitMQ Message Broker

Untuk mengakses RabbitMQ *message broker*, MATLAB *Client* perlu menginisialisasi koneksi dengan memanfaatkan beberapa *class* yang sudah disediakan oleh RabbitMQ. Teknik komunikasi yang digunakan antara MATLAB *Client* dan RabbitMQ *message broker* adalah *remote procedure call* (RPC). Penanganan koneksi dan teknik RPC ini dikerjakan oleh *class* MatlabRPC. Kelas - kelas yang dibutuhkan oleh MatlabRPC untuk penanganan koneksi adalah sebagai berikut :

- `com.rabbitmq.client.ConnectionFactory`
- `com.rabbitmq.client.Connection`
- `com.rabbitmq.client.Channel`
- `com.rabbitmq.client.RpcClient`

Pada langkah awal, objek dari `ConnectionFactory` dibuat dengan menyertakan parameter *ip address* milik RabbitMQ *message broker*. *Instance* dari

ConnectionFactory tersebut akan memberikan objek `Connection` sebagai representasi koneksi dengan RabbitMQ *message broker*. Untuk membuat teknik komunikasi berbasis RPC, objek dari `RpcClient` dibentuk dengan parameter `Channel` dan *string* nama pengenalan. Potongan kode dapat dilihat di bawah ini.

```
import com.rabbitmq.client.Channel;
import com.rabbitmq.client.Connection;
import com.rabbitmq.client.ConnectionFactory;
import com.rabbitmq.client.RpcClient;

//...//

public class MatlabRPC {

    private String HOST_IP;
    private Connection connection;
    private Channel channel;
    private RpcClient service;
    private ConnectionFactory cfconn;

    //...//

    public MatlabRPC(String host) throws IOException {
        this.HOST_IP = host;

        cfconn = new ConnectionFactory();
        cfconn.setHost(HOST_IP);

        connection = cfconn.newConnection();
        channel = connection.createChannel();
        service = new RpcClient(channel, "", "MATLAB");
    }

    //...//
}
```

### Algoritma 3.1 Inisialisasi koneksi pada MatlabRPC

Kemudian, *MATLAB Client* membuat objek `MatlabRPC` setelah pengguna mengisi *ip address* RabbitMQ *message broker*. Potongan kode tertera di bawah ini.

```
import com.fahri.matlab.client.MatlabRPC;
//...//
```

```

public class MatlabClient extends javax.swing.JFrame {

    private static MatlabRPC matlabRpc;
    //...//

    public static void main(String args[]) {
        java.awt.EventQueue.invokeLater(new Runnable() {

            public void run() {
                boolean onConnect = false;
                while (!onConnect) {
                    String ipAddress = JOptionPane.showInputDialog
                        ("Input IP address of Message Broker Host :
                        ");
                    try {
                        matlabRpc = new MatlabRPC(ipAddress);
                        onConnect = true;
                    } catch (IOException ex) {
                        JOptionPane.showMessageDialog(null, "
                        Unable to connect !");
                        onConnect = false;
                    }
                }
                new MatlabClient().setVisible(true);
            }
        });
    }

    //...//
}

```

**Algoritma 3.2** Pemakaian MatlabRPC oleh MATLAB *Client*

### 3.3.5.2 Pengiriman Perintah dan Penerimaan Respon

Setelah pengguna mengisi *query* perintah MATLAB dan mengklik tombol “Execute”, MATLAB *Client* akan mengonstruksi pesan *request* berdasarkan format data yang telah ditetapkan. Kemudian, pesan *request* tersebut dikirim dengan bantuan MatlabRPC. Respon akan diterima dan disimpan pada variabel dengan tipe *string* dan *byte array*. Potongan kode dapat dilihat di bawah ini.

```

private void btnExeActionPerformed(java.awt.event.ActionEvent evt)
{

```

```

//...//
String queryTxt = txtQuery.getText();
//...//

String cmdQuery = "query|" + matlabRpc.getToken() + "|" +
    queryTxt;
String cmdImage = "getImage|" + matlabRpc.getToken();

String responseQuery = "";
byte[] responseImage = null;
try {
    responseQuery = matlabRpc.sendStringMessage(cmdQuery);
    responseImage = matlabRpc.sendByteMessage(cmdImage.getBytes
        ());
} catch (IOException ex) {
    Logger.getLogger(MatlabClient.class.getName()).log(Level.
        SEVERE, null, ex);
} catch (ShutdownSignalException ex) {
    Logger.getLogger(MatlabClient.class.getName()).log(Level.
        SEVERE, null, ex);
}

//...//
}

```

**Algoritma 3.3** Pengiriman *request* dan penerimaan *response* pada MATLAB  
*Client*

### 3.3.5.3 Variable History

Setelah MATLAB *Client* mendapatkan respon, jika respon tersebut adalah nilai berdasarkan *query* yang menyatakan *assignment* pada suatu variabel, variabel dan nilai tersebut disimpan di dalam `DefaultListModel`. Dalam penyimpanan tersebut, suatu variabel direpresentasikan sebagai objek `Variable`.

```

private void btnExeActionPerformed(java.awt.event.ActionEvent evt)
{

//...//

Variable var = new Variable();

```

```

if (queryTxt.contains("=")) {
    var.setName(queryTxt.split("=")[0]);
}
//...//

responseQuery = matlabRpc.sendStringMessage(cmdQuery);

//...//

var.setValue(responseQuery);
if (listVariableModel.contains(var)) {
    listVariableModel.removeElement(var);
}
listVariableModel.addElement(var);

//...//
}

```

**Algoritma 3.4** *Variable History* pada MATLAB Client

#### 3.3.5.4 Command History

Setelah perintah (*command*) MATLAB dijalankan, MATLAB Client menyimpan *syntax* dari *command* / *query* tersebut. *Command* disimpan pada *DefaultListModel*.

```

private void btnExeActionPerformed(java.awt.event.ActionEvent evt)
{

    //...//

    listCommandModel.addElement(queryTxt);

    //...//

}

```

**Algoritma 3.5** *Command History* pada MATLAB Client

#### 3.3.5.5 Menampilkan Gambar

Respon berupa gambar ditampilkan dengan memanfaatkan komponen Swing *JLabel*. Respon gambar yang bertipe *byte array* diubah menjadi tipe

BufferedImage sebagai parameter di dalam *constructor* ImageIcon. Objek dari ImageIcon tersebut dijadikan parameter pada *method* setIcon untuk di-render sebagai gambar oleh JLabel.

```
private void btnExeActionPerformed(java.awt.event.ActionEvent evt)
{
    //...//

    responseImage = matlabRpc.sendByteMessage(cmdImage.getBytes());

    //...//

    InputStream in = new ByteArrayInputStream(responseImage);
    responseBufferedImage = null;
    try {
        responseBufferedImage = ImageIO.read(in);
    } catch (IOException ex) {
        Logger.getLogger(MatlabClient.class.getName()).log(Level.
            SEVERE, null, ex);
    }

    if (responseBufferedImage != null) {
        lblImage.setIcon(new ImageIcon(responseBufferedImage));
        lblImage.repaint();
    }
}
```

### Algoritma 3.6 Menampilkan gambar pada MATLAB Client

#### 3.3.5.6 Menyimpan Gambar

Gambar disimpan dengan menggunakan *method* ImageIO.write() yang membutuhkan parameter berupa BufferedImage, tipe gambar, dan File. Proses penyimpanan gambar terjadi jika pengguna mengklik tombol “Save Image” dan memilih lokasi *file* gambar yang ingin disimpan.

```
private void btnSaveImageActionPerformed(java.awt.event.
    ActionEvent evt) {
    int returnVal = fileChooser.showSaveDialog(MainGUI.this);
    if (returnVal == JFileChooser.APPROVE_OPTION) {
        File file = fileChooser.getSelectedFile();

        if (responseBufferedImage != null) {
```

```

        try {
            ImageIO.write(responseBufferedImage, "png", file);
        } catch (IOException ex) {
            Logger.getLogger(MainGUI.class.getName()).log(
                Level.SEVERE, null, ex);
        }
    }
}
}
}

```

### Algoritma 3.7 Menyimpan gambar pada MATLAB *Client*

## 3.4 MATLAB *Server*

Berikut ini adalah fungsi-fungsi utama yang dimiliki oleh MATLAB *Server*.

### 3.4.1 *Functional Requirement*

#### 3.4.1.1 *Connect to Message Broker*

Fungsi ini digunakan untuk mengkoneksikan MATLAB *Server* dengan RabbitMQ *message broker*.

#### 3.4.1.2 *Client Session Management*

Fungsi ini digunakan oleh MATLAB *Server* untuk mengatur *session* bagi tiap pengguna MATLAB *Client*. Dengan fungsi ini, MATLAB *Server* mengalokasikan *workspace* dan *session* untuk masing - masing pengguna MATLAB *Client* sehingga perintah dan nilai variabel antar pengguna MATLAB *Client* tidak saling membaaur dan *ter-override*.

#### 3.4.1.3 *Execute Query*

Fungsi ini digunakan oleh MATLAB *Server* untuk mengeksekusi perintah MATLAB dengan bantuan JMatlink.

#### 3.4.1.4 *Retrieve Image*

Fungsi ini digunakan oleh MATLAB *Server* untuk mendapatkan gambar jika perintah MATLAB yang dieksekusi menghasilkan respon berupa gambar.



### 3.4.1.5 *Close Matlab Engine*

Fungsi ini digunakan untuk mengakhiri penggunaan mesin komputasi MATLAB.

## 3.4.2 Teknik dan Algoritma

### 3.4.2.1 Akses RabbitMQ *message broker*

Sama halnya dengan MATLAB *Client*, untuk mengakses RabbitMQ *message broker*, MATLAB *Server* perlu menginisialisasi beberapa *class* yang sudah disediakan oleh RabbitMQ. Teknik komunikasi yang digunakan antara MATLAB *Server* dan RabbitMQ *message broker* adalah *remote procedure call* (RPC).

Kelas - kelas yang dibutuhkan oleh MATLAB *Server* untuk penanganan koneksi adalah sebagai berikut :

- `com.rabbitmq.client.ConnectionFactory`
- `com.rabbitmq.client.Connection`
- `com.rabbitmq.client.Channel`
- `com.rabbitmq.client.RpcClient`

Pada langkah awal, objek dari `ConnectionFactory` dibuat dengan menyertakan parameter *ip address* milik RabbitMQ *message broker*. *Instance* dari `ConnectionFactory` tersebut akan memberikan objek `Connection` sebagai representasi koneksi dengan RabbitMQ *message broker*. Untuk membuat teknik komunikasi berbasis RPC, objek dari `RpcClient` dibentuk dengan parameter `Channel` dan *string* nama pengenalan. Potongan kode dapat dilihat di bawah ini.

```
import com.rabbitmq.client.Channel;
import com.rabbitmq.client.Connection;
import com.rabbitmq.client.ConnectionFactory;

public class MatlabServer{

    public static void main(String[] args){
        //...//
        ConnectionFactory connFactory = new ConnectionFactory();
        connFactory.setHost("192.168.56.101");
        Connection conn = connFactory.newConnection();
        final Channel ch = conn.createChannel();

        ch.queueDeclare(QUEUE_NAME, false, false, false, null);
```

```

ch.basicQos(1);
QueueingConsumer consumer = new QueueingConsumer(ch);
ch.basicConsume(QueueName, false, consumer);

while (true) {
    QueueingConsumer.Delivery delivery = consumer.
        nextDelivery();

    BasicProperties props = delivery.getProperties();
    BasicProperties replyProps = new BasicProperties();
    replyProps.setCorrelationId(props.getCorrelationId());

    String requestMsg = new String(delivery.getBody());

    //... olah response ...//

    ch.basicPublish("", props.getReplyTo(), replyProps,
        response.getBytes());
    ch.basicAck(delivery.getEnvelope().getDeliveryTag(),
        false);

}
}
}

```

### Algoritma 3.8 Akses RabbitMQ *message broker* pada MATLAB Server

#### 3.4.2.2 Session Management

Dalam mengatur *session*, MATLAB Server akan mengalokasikan dan membuat *engine* MATLAB untuk masing - masing pengguna MATLAB Client. Tiap pengguna MATLAB Client mendapatkan nomor *id* dari *engine* yang sudah terbentuk. MATLAB Server menyimpan nomor *id engine* tersebut dalam bentuk *HashMap* dengan *key* berupa *token* dari MATLAB Client. Ketika evaluasi dan eksekusi *query* MATLAB dilakukan, MATLAB Server menggunakan *engine* yang sudah dimiliki oleh MATLAB Client.

```

public class MatlabServer {

    //...//

    public static void main(String[] args) {
        JMatLink engine = new JMatLink();
    }
}

```

```

Map<String, Long> sessionId = new HashMap<String, Long>();
//...//

while (true) {
    //...//
    long id = 0;
    if (sessionId.containsKey(token)) {
        id = sessionId.get(token);
    } else {
        id = engine.engOpenSingleUse();
        sessionId.put(token, id);
    }
    //...//
}

//...//
}
}

```

### Algoritma 3.9 *Session Management* pada MATLAB Server

#### 3.4.2.3 Evaluasi *Query* MATLAB dan Pengambilan Hasil

Ketika evaluasi dan eksekusi *query* MATLAB dilakukan, MATLAB *Server* menggunakan *engine* yang sudah dimiliki oleh MATLAB *Client* pada langkah sebelumnya berdasarkan nomor *id engine* yang di-*map* terhadap *token* MATLAB *Client*. Evaluasi dan pengambilan hasil memanfaatkan *method* `engEvalString` dan `engGetArray` yang dimiliki oleh `JMatlink`.

```

public class MatlabServer {

    //...//

    public static void main(String[] args) {
        JMatLink engine = new JMatLink();
        //...//

        while (true) {
            //...//
            engine.engEvalString(id, query);
            //...//
            double[][] arrayResult = engine.engGetArray(id, var);
            //...//
        }
    }
}

```

```

    }

    //...//
}

```

**Algoritma 3.10** Evaluasi *query* dan pengambilan hasil pada MATLAB Server

#### 3.4.2.4 Pengambilan Gambar

Pengambilan respon berupa gambar memanfaatkan *method* `engGetFigure` yang dimiliki oleh `JMatLink`. Gambar tersebut diubah menjadi *byte array* untuk dikirim menuju RabbitMQ *message broker*.

```

public class MatlabServer {

    //...//

    public static void main(String[] args) {
        JMatLink engine = new JMatLink();
        //...//

        while (true) {
            //...//
            Image im = engine.engGetFigure(id, 1, 400, 350);
            BufferedImage bufferedImage = new BufferedImage(im.
                getWidth(null), im.getHeight(null), BufferedImage.
                TYPE_INT_RGB);
            Graphics bg = bufferedImage.getGraphics();
            bg.drawImage(im, 0, 0, null);
            bg.dispose();

            //convert BufferedImage to byte array
            ByteArrayOutputStream baos = new ByteArrayOutputStream
                ();
            ImageIO.write(bufferedImage, "jpg", baos);
            baos.flush();
            resultImage = baos.toByteArray();
            baos.close();
            //...//
        }

        //...//
    }
}

```

```
}

```

### Algoritma 3.11 Pengambilan gambar pada MATLAB Server

#### 3.4.2.5 Tutup Koneksi

Koneksi pada *engine* MATLAB ditutup dengan menggunakan *method* `engClose` dan nomor *id* milik pengguna MATLAB *Client* sebagai parameternya.

```
public class MatlabServer {

    //...//

    public static void main(String[] args) {
        JMatLink engine = new JMatLink();
        //...//

        while (true) {
            //...//
            if (sessionId.containsKey(token)) {
                long id = sessionId.get(token);
                engine.engClose(id);
            }
            //...//
        }

        //...//
    }
}
```

### Algoritma 3.12 Tutup koneksi *engine* MATLAB pada MATLAB Server

## BAB 4

### PENGUJIAN

Pada bab ini, pengujian akan dilakukan terhadap sistem yang telah selesai dibuat. Pengujian ditekankan pada aplikasi MATLAB *Client* dengan melakukan *scenario testing*. Platform pengujian yang digunakan oleh MATLAB *Server*, MATLAB *Client*, dan RabbitMQ *message broker* ditunjukkan pada Tabel 4.1, Tabel 4.2, dan Tabel 4.3. Ketiganya saling terhubung dalam *Local Area Network* (LAN).

**Tabel 4.1:** Platform pengujian MATLAB *Server*

Komponen	Deskripsi
CPU	Intel Core i5 M450 @ 2.40 GHz
RAM	2 GB
Harddisk	500 GB
Sistem Operasi	MS Windows 7 Ultimate 32-bit
IP Address	192.168.1.100

**Tabel 4.2:** Platform pengujian MATLAB *Client*

Komponen	Deskripsi
CPU	Intel Core2 Duo E7400 @ 2.80GHz
RAM	4 GB
Harddisk	80 GB
Sistem Operasi	Ubuntu (Linux) 32-bit
IP Address	192.168.1.102

**Tabel 4.3:** Platform pengujian RabbitMQ *message broker*

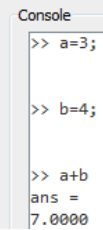
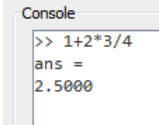
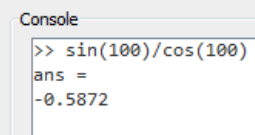
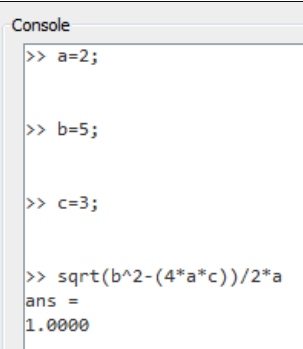
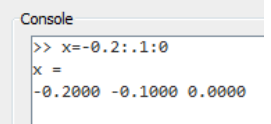
Komponen	Deskripsi
CPU	Intel Celeron 560 @ 2.13 GHz
RAM	1.5 GB
Harddisk	120 GB
Sistem Operasi	Debian (Linux) 32-bit
IP Address	192.168.1.101

Sistem diuji dengan tiga macam skenario yaitu pengujian dengan hasil *string*, pengujian dengan hasil gambar, dan pengujian *multi-user* dengan *session* yang berbeda.

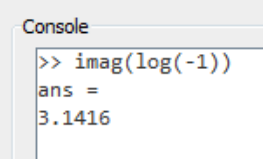
#### 4.1 Pengujian dengan Hasil *String*

Dalam pengujian ini, MATLAB *Client* mengirim *query* MATLAB yang menghasilkan respon nilai berupa *string*. Nilai respon tersebut (*actual result*) akan dibandingkan dengan nilai respon dari *console* MATLAB (*expected result*). Hasil pengujian ditunjukkan pada Tabel 4.4.

**Tabel 4.4:** Pengujian dengan Hasil *String*

<i>Command</i>	<i>Actual Result</i>	<i>Expected Result</i>
<pre>a = 3; b = 4; a + b</pre>	 <pre>Console &gt;&gt; a=3;  &gt;&gt; b=4;  &gt;&gt; a+b ans = 7.0000</pre>	<pre>» a=3; » b=4; » a+b  ans =  7</pre>
<pre>1+2*3/4</pre>	 <pre>Console &gt;&gt; 1+2*3/4 ans = 2.5000</pre>	<pre>» 1+2*3/4  ans =  2.5000</pre>
<pre>sin(100)/cos(100)</pre>	 <pre>Console &gt;&gt; sin(100)/cos(100) ans = -0.5872</pre>	<pre>» sin(100)/cos(100)  ans =  -0.5872</pre>
<pre>a=2; b=5; c=3; sqrt(b^2-(4*a*c))/2*a</pre>	 <pre>Console &gt;&gt; a=2;  &gt;&gt; b=5;  &gt;&gt; c=3;  &gt;&gt; sqrt(b^2-(4*a*c))/2*a ans = 1.0000</pre>	<pre>» a=2; » b=5; » c=3; » sqrt(b^2-(4*a*c))/2*a  ans =  1</pre>
<pre>x=-0.2:.1:0</pre>	 <pre>Console &gt;&gt; x=-0.2:.1:0 x = -0.2000 -0.1000 0.0000</pre>	<pre>» x=-0.2:.1:0  x =  -0.2000 -0.1000 0</pre>
Lanjut ke halaman berikutnya		

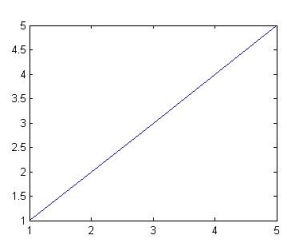
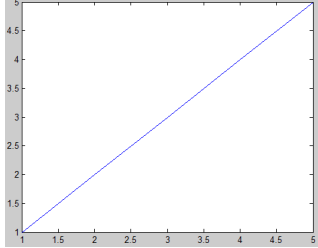
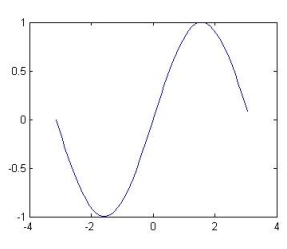
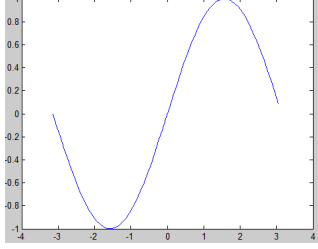
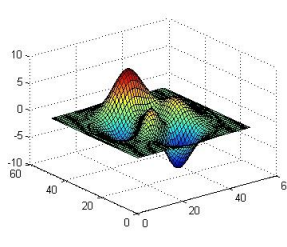
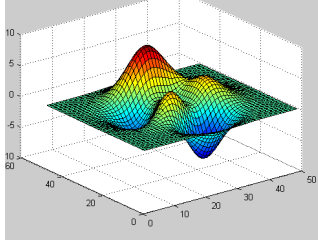
**Tabel 4.4** Pengujian dengan Hasil *String* (lanjutan)

Command	Actual Result	Expected Result
<code>imag(log(-1)) </code>		<pre>» imag(log(-1))  ans =      3.1416</pre>

## 4.2 Pengujian dengan Hasil Gambar

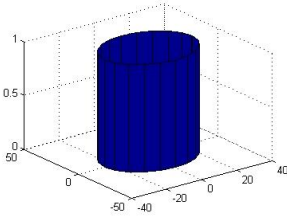
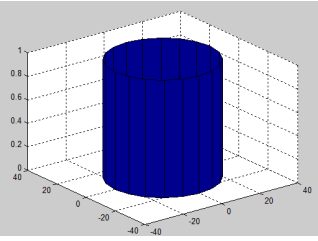
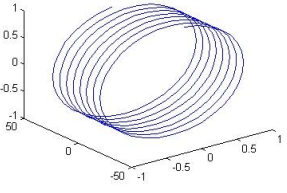
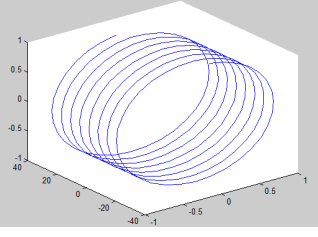
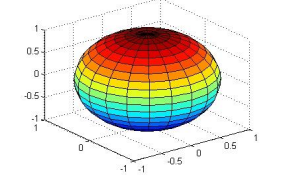
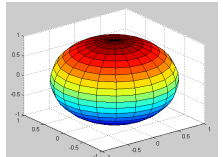
Dalam pengujian ini, *MATLAB Client* mengirim *query* MATLAB yang menghasilkan respon berupa gambar. Gambar tersebut (*actual result*) akan dibandingkan dengan gambar respon dari *console* MATLAB (*expected result*). Hasil pengujian ditunjukkan pada Tabel 4.5.

**Tabel 4.5:** Pengujian dengan Hasil Gambar

Command	Actual Result	Expected Result
<code>plot(1:5); </code>		
<code>x=-pi:.1:pi; y=sin(x); plot(x,y); </code>		
<code>surf(peaks); </code>		
Lanjut ke halaman berikutnya		



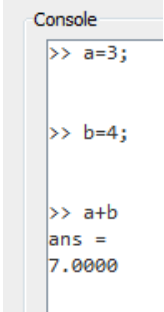
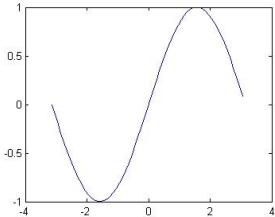
**Tabel 4.5** Pengujian dengan Hasil Gambar (lanjutan)

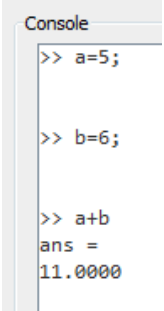
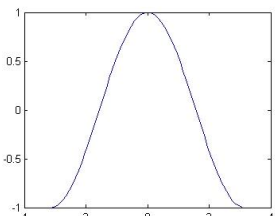
<i>Command</i>	<i>Actual Result</i>	<i>Expected Result</i>
<code>cylinder(25);</code>		
<code>t=-25:.2:25; plot(sin(t),t,cos(t));</code>		
<code>sphere;</code>		

### 4.3 Pengujian *Multi-User* dengan *Session* yang Berbeda

Sistem diuji dalam hal penanganan *session* jika terdapat banyak pengguna MATLAB *Client* yang mengakses secara bersamaan (*multi-user*). Dalam pengujian ini, terdapat dua pengguna MATLAB *Client* (Pengguna A dan Pengguna B) yang mengakses sistem secara bersamaan. Tiap pengguna akan melakukan perintah MATLAB yang mirip tetapi memiliki perbedaan nilai. Kemudian, sistem akan memberikan respon yang berbeda berdasarkan perintah dari pengguna. Dengan kata lain, sistem tidak mencampuradukkan atau meng-*override* perintah dan nilai variabel yang dimiliki oleh masing - masing pengguna. Hasil pengujian ditunjukkan pada Tabel 4.6.

**Tabel 4.6:** Pengujian *Multi-User* dengan *Session* yang Berbeda

Perintah Pengguna A	Respon untuk Pengguna A
<pre>a=3; b=4; a+b</pre>	 <pre>Console &gt;&gt; a=3;  &gt;&gt; b=4;  &gt;&gt; a+b ans = 7.0000</pre>
<pre>x=-pi:.1:pi; y=sin(x); plot(x,y);</pre>	

Perintah Pengguna B	Respon untuk Pengguna B
<pre>a=5; b=6; a+b</pre>	 <pre>Console &gt;&gt; a=5;  &gt;&gt; b=6;  &gt;&gt; a+b ans = 11.0000</pre>
<pre>x=-pi:.1:pi; y=cos(x); plot(x,y);</pre>	

## **BAB 5**

### **KESIMPULAN DAN SARAN**

Pada bab ini, penulis membuat kesimpulan berdasarkan hasil yang telah didapatkan selama proses pengerjaan tugas akhir ini. Selain itu, penulis juga menuliskan beberapa saran untuk penelitian berikutnya.

#### **5.1 Kesimpulan**

Kesimpulan dari hasil tugas akhir ini adalah sebagai berikut :

- Sistem MATLAB *Client* dan MATLAB *Server* dapat dibangun dengan arsitektur *messaging* yang menggunakan RabbitMQ sebagai *message broker*.
- Teknik komunikasi yang digunakan antar MATLAB *Client* dan MATLAB *Server* adalah *remote procedure call* (RPC).
- MATLAB *Server* dapat menangani perintah - perintah dasar MATLAB, respon berupa gambar, dan *session management* untuk tiap MATLAB *Client* dengan bantuan JMatlink sebagai penghubung mesin komputasi MATLAB.

#### **5.2 Saran**

Untuk penelitian lebih lanjut, penulis menyarankan beberapa hal berikut :

- Arsitektur sistem dapat dicoba dengan teknik komunikasi lain, selain RPC, yang didukung oleh RabbitMQ.
- Format pesan dapat menggunakan format data yang lebih *universal* dan standar seperti XML.
- MATLAB *Server* perlu dikembangkan lebih lanjut untuk penanganan perintah - perintah MATLAB yang lebih kompleks dan dapat menangani tipe data lain seperti matriks.
- Sistem perlu diuji dengan *performance testing* untuk mengukur kinerja sistem jika diakses oleh banyak MATLAB *Client* secara bersamaan.
- Adanya pengujian sistem untuk mengukur keakuratan data dan tipe data dari nilai hasil komputasi yang dilakukan oleh sistem.

## DAFTAR REFERENSI

- [1] Mathworks. *Introduction*. 2011. 30 Juni 2011.  
⟨<http://www.mathworks.com/products/matlab/description1.html>⟩.
- [2] D. J. Higham and N. J. Higham. *MATLAB Guide, 2nd Edition*. SIAM: Society for Industrial and Applied Mathematics, 2005.
- [3] Chappell, David A. *Enterprise Service Bus*. O'Reilly Media, 2004.
- [4] G. Hohpe and B. Woolf. *Enterprise Integration Patterns: Designing, Building, and Deploying Messaging Solutions, 1st Edition*. Addison-Wesley Professional, 2003.
- [5] Garnock-Jones, Tony. *Achieving Scale with Messaging & the Cloud*. 2011. 21 Agustus 2011.  
⟨[http://dev.lshift.net/tonyg/Achieving Scale with Messaging and the Cloud \(with notes\).pdf](http://dev.lshift.net/tonyg/Achieving%20Scale%20with%20Messaging%20and%20the%20Cloud%20(with%20notes).pdf)⟩.
- [6] Vmware. *How To with RabbitMQ*. 2011. 21 Agustus 2011.  
⟨<http://www.rabbitmq.com/how.html>⟩.
- [7] Wikipedia. *RabbitMQ*. 2011. 30 Juni 2011.  
⟨<http://http://en.wikipedia.org/wiki/RabbitMQ/>⟩.
- [8] Vmware. *Messaging That Just Works*. 2011. 30 Juni 2011.  
⟨<http://www.rabbitmq.com/>⟩.
- [9] Vmware. *Introduction*. 2011. 30 Juni 2011.  
⟨<http://www.rabbitmq.com/tutorials/tutorial-one-java.html>⟩.
- [10] Vmware. *Work Queues*. 2011. 30 Juni 2011.  
⟨<http://www.rabbitmq.com/tutorials/tutorial-two-java.html>⟩.
- [11] Vmware. *Publish / Subscribe*. 2011. 30 Juni 2011.  
⟨<http://www.rabbitmq.com/tutorials/tutorial-three-java.html>⟩.
- [12] Vmware. *Routing*. 2011. 30 Juni 2011.  
⟨<http://www.rabbitmq.com/tutorials/tutorial-four-java.html>⟩.
- [13] Vmware. *Topics*. 2011. 30 Juni 2011.  
⟨<http://www.rabbitmq.com/tutorials/tutorial-five-java.html>⟩.

- [14] Vmware. *Remote procedure call (RPC)*. 2011. 30 Juni 2011.  
⟨<http://www.rabbitmq.com/tutorials/tutorial-six-java.html>⟩.
- [15] Ying Bai. *Applications Interface Programming Using Multiple Languages: A Windows Programmer's Guide*. Prentice Hall, 2003.
- [16] Oracle. *About the Java Technology*. 2011. 30 Juni 2011.  
⟨<http://download.oracle.com/javase/tutorial/getStarted/intro/definition.html>⟩.
- [17] Lewis, Jeremy. *SDLC 100 Success Secrets - Software Development Life Cycle (SDLC) 100 Most Asked Questions, SDLC Methodologies, Tools, Process and Business Models*. Emereo Publishing, 2008.
- [18] Saleh, Kassem A. *Software Engineering*. J. Ross Publishing, 2009.
- [19] Reese, George. *Database Programming with JDBC and Java, 2nd Edition*. O'Reilly Media, 2000.
- [20] Larman, Craig. *Applying UML and Patterns : An Introduction to Object-Oriented Analysis and Design and the Unified Process, 2nd Edition*. Prentice Hall. 2001.

## LAMPIRAN 1

### Lampiran 1 JMatLink Method Summary

Return Value	Method Signature	Description
void	engClose()	Close the connection to Matlab.
void	engClose(long epI)	Close a specified connection to an instance of Matlab.
void	engCloseAll()	Close all connections to Matlab.
void	engEvalString(long epI, java.lang.String evalS)	Evaluate an expression in a specified workspace.
void	engEvalString(java.lang.String evalS)	Evaluate an expression in Matlab's workspace.
double[][]	engGetArray(long epI, java.lang.String arrayS)	Get an array from a specified instance/workspace of Matlab.
double[][]	engGetArray(java.lang.String arrayS)	Get an array from Matlab's workspace.
java.lang.String[]	engGetCharArray(long epI, java.lang.String arrayS)	Get an 'char' array (string) from a specified instance/workspace of Matlab.
java.lang.String[]	engGetCharArray(java.lang.String arrayS)	Get an 'char' array (string) from Matlab's workspace.
java.awt.Image	engGetFigure(int figure, int dx, int dy)	Return image of figure from Matlab.
java.awt.Image	engGetFigure(long epI, int figure, int dx, int dy)	Return image of figure from Matlab.
java.lang.String	engGetOutputBuffer()	Return the outputs of previous commands in Matlab's workspace.

<code>java.lang.String</code>	<code>engGetOutputBuffer(long epI)</code>	Return the outputs of previous commands from a specified instance/workspace of Matlab.
<code>double</code>	<code>engGetScalar(long epI, java.lang.String arrayS)</code>	Get a scalar value from a specified workspace.
<code>double</code>	<code>engGetScalar(java.lang.String arrayS)</code>	Get a scalar value from Matlab's workspace.
<code>boolean</code>	<code>engGetVisible(long epI)</code>	Return the visibility status of the Matlab's window.
<code>void</code>	<code>engOpen()</code>	Open engine.
<code>void</code>	<code>engOpen(java.lang.String startCmdS)</code>	Open engine.
<code>long</code>	<code>engOpenSingleUse()</code>	Open engine for single use.
<code>long</code>	<code>engOpenSingleUse(java.lang.String startCmdS)</code>	Open engine for single use.
<code>int</code>	<code>engOutputBuffer()</code>	Return the outputs of previous commands from Matlab's workspace.
<code>int</code>	<code>engOutputBuffer(long epI)</code>	Return the outputs of previous commands from a specified instance/workspace of Matlab.
<code>int</code>	<code>engOutputBuffer(long epI, int buflenI)</code>	Return the outputs of previous commands from a specified instance/workspace of Matlab.
<code>void</code>	<code>engPutArray(long epI, java.lang.String arrayS, double valueD)</code>	Put an array into a specified instance/workspace of Matlab.
<code>void</code>	<code>engPutArray(long epI, java.lang.String arrayS, double[] valuesD)</code>	Put an array (1 dimensional) into a specified instance/workspace of Matlab.

void	<code>engPutArray(long epI, java.lang.String arrayS, double[][] valuesDD)</code>	Put an array (2 dimensional) into a specified instance/workspace of Matlab.
void	<code>engPutArray( java.lang.String arrayS, double valueD)</code>	Put an array into Matlab's workspace.
void	<code>engPutArray( java.lang.String arrayS, double[] valuesD)</code>	Put an array (1 dimensional) into Matlab's workspace.
void	<code>engPutArray( java.lang.String arrayS, double[][] valuesD)</code>	Put an array (2 dimensional) into Matlab's workspace.
void	<code>engPutArray( java.lang.String arrayS, int valueI)</code>	Put an array into Matlab's workspace.
void	<code>engSetVisible(long epI, boolean visB)</code>	Set the visibility of the Matlab's window.
java.lang.String	<code>getVersion()</code>	Returns the current version of JMatLink.
void	<code>kill()</code>	Obsolete method.
void	<code>setDebug(boolean debugB)</code>	Switch on or disable debug information printed to standard output.



## LAMPIRAN 2

### Lampiran 2 Instalasi MATLAB *Server* dan MATLAB *Client* (untuk *End User*)

#### Instalasi pada *Host RabbitMQ*

1. Unduh dan *install* RabbitMQ (<https://www.rabbitmq.com/download.html>).

#### Instalasi pada *Host MATLAB Server*

1. *Install* MATLAB versi 2009a.
2. Unduh dan *install* Java SE Runtime Environment 6 (JRE 6) (<http://www.oracle.com/technetwork/java/javase/downloads/index.html>).
3. *Copy* MULTIMEDIA\distribution\MatlabServer\JMatLink.dll ke C:\Windows\JMatLink.dll .
4. Jalankan MATLAB *Server* pada *command prompt* :

- `cd MULTIMEDIA\distribution\MatlabServer`
- `java -jar matlab-server.jar 192.168.1.101`

*Catatan:* Ganti 192.168.1.101 dengan *ip address host RabbitMQ*.

#### Instalasi pada *Host MATLAB Client*

1. Unduh dan *install* Java SE Runtime Environment 6 (JRE 6) (<http://www.oracle.com/technetwork/java/javase/downloads/index.html>).
2. Jalankan MATLAB *Client* pada *command prompt* :

- `cd MULTIMEDIA\distribution\MatlabClient`
- `java -jar matlab-client.jar`

## LAMPIRAN 3

### Lampiran 3 Tahapan Kompilasi Kode Sumber MATLAB *Server* dan MATLAB *Client* (untuk *Developer*)

#### Persiapan pada *Host* RabbitMQ

1. Unduh dan *install* RabbitMQ (<https://www.rabbitmq.com/download.html>).

#### Kompilasi pada *Host* MATLAB *Server*

1. *Install* MATLAB versi 2009a.
2. Unduh dan *install* Java SE Development Kit 6 (JDK 6) (<http://www.oracle.com/technetwork/java/javase/downloads/index.html>).
3. Unduh dan *install* Apache Maven (<http://maven.apache.org/download.cgi>).
4. *Copy* MULTIMEDIA\source\_code\MatlabServer\JMatLink.dll ke C:\Windows\JMatLink.dll .

5. *Compile* dan jalankan MATLAB *Server* pada *command prompt* :

- cd MULTIMEDIA\source\_code\MatlabServer
- mvn clean install exec:java -D192.168.1.101  
*Catatan:* Ganti 192.168.1.101 dengan *ip address* host RabbitMQ.

6. *Packaging* menjadi JAR (Java ARchive) pada *command prompt* :

- cd MULTIMEDIA\source\_code\MatlabServer
- mvn clean compile assembly:single
- Hasil *packaging* berada di target\matlab-server.jar

**Kompilasi pada *Host MATLAB Client***

1. Unduh dan *install* Java SE Development Kit 6 (JDK 6) (<http://www.oracle.com/technetwork/java/javase/downloads/index.html>).
2. Unduh dan *install* Apache Maven (<http://maven.apache.org/download.cgi>).
3. *Compile* dan jalankan MATLAB *Client* pada *command prompt* :
  - `cd MULTIMEDIA\source_code\MatlabClient`
  - `mvn clean install exec:java`
4. *Packaging* menjadi JAR (Java ARchive) pada *command prompt* :
  - `cd MULTIMEDIA\source_code\MatlabClient`
  - `mvn clean compile assembly:single`
  - Hasil *packaging* berada di `target\matlab-client.jar`