

Il materiale che troverai in questo documento è completamente gratuito. Per quanto riguarda il contenuto, dato che non è stato rivisto da "esperti", mi preme precisare che NON garantisco in alcun modo la correttezza totale.

Il documento è rilasciato sotto i termini della licenza Creative Commons Attribuzione - Non commerciale - Condividi allo stesso modo 3.0 Italia. Per visionare una copia completa della licenza, visita <http://creativecommons.org/licenses/by-nc-sa/3.0/it/legalcode>.

Lo scopo di tutto ciò non è tanto arricchire il mio curriculum personale, bensì condividere le mie conoscenze con il maggiore numero di persone possibile. Considera questo documento anche come un invito a pubblicare e condividere i tuoi saperi, perché il fondamento di una vera formazione è il libero accesso al sapere in tutte le sue forme.

"Information is power. But like all power, there are those who want to keep it for themselves.

...

Those with access to these resources - students, librarians, scientists - you have been given a privilege.

...

*Meanwhile, those who have been locked out are not standing idly by. You have been sneaking through holes and climbing over fences, liberating the information locked up by the publishers and sharing them with your friends. But all of this action goes on in the dark, hidden underground. **It's called stealing or piracy, as if sharing a wealth of knowledge were the moral equivalent of plundering a ship and murdering its crew.** But sharing isn't immoral - it's a moral imperative. Only those blinded by greed would refuse to let a friend make a copy.*

...

There is no justice in following unjust laws. It's time to come into the light and, in the grand tradition of civil disobedience, declare our opposition to this private theft of public culture.

...

With enough of us, around the world, we'll not just send a strong message opposing the privatization of knowledge - we'll make it a thing of the past. Will you join us?"

Guerilla Open Access Manifesto
Aaron Swartz

Progetto Programmazione Java

Servizio di Messaggistica Istantanea

Rambod Rahmani

rambodrahmani@autistici.org

Relentlessly pursue knowledge and selflessly share it with others.

June 15, 2016

Abstract

Il presente documento contiene la documentazione prodotta durante la realizzazione del Progetto "Servizio di Messaggistica Istantanea" per l'esame di "Programmazione Java" - Prof. Mario G.C.A. Cimino - presso l'università degli studi di Pisa - Ingegneria Informatica.

Il progetto Java è disponibile open source su GitHub: <https://github.com/rambodrahmani/esame-programmazione-java>

Keywords: programmazione , java , esame , unipi , rambod , rahmani

Dedicato a mia madre Sara, mio padre Magid e mio fratello gemello Ramtin per esserci stati quando ho avuto bisogno e per avermi reso la persona che sono oggi.

Indice

1	Introduzione	5
2	Analisi	6
2.1	Server di Log	6
2.2	Client	7
3	Progetto	12
3.1	Server di Log	12
3.2	Client	14
3.3	Classi Comuni	17
4	Manuale d'uso	19
4.1	Server di Log	19
4.2	Client	24
4.3	Script bat	33

1 Introduzione

Il progetto che ho sviluppato per l'esame di Programmazione Java è un software di messaggistica istantanea che permetto di scambio di messaggi di testo tra due Client connessi contemporaneamente. Il progetto comprende anche un Server di Log che riceve messaggi di Log, relativi all'utilizzo della GUI, da parte dei vari Client connessi al servizio di messaggistica.

Il documento si sviluppa secondo le varie fasi di sviluppo del Progetto. Le fasi di lavoro sono state preimpostate dalle consegne d'esame pubblicate dal professore.

Il codice sorgente del prodotto e gli altri File prodotti durante lo sviluppo sono disponibili al seguente indirizzo: GitHub - <https://github.com/rambodrahmani/esame-programmazione-java>

2 Analisi

Il progetto consiste nella realizzazione di un applicativo che permetta lo scambio di messaggi di testo.

Il progetto è composto da:

- **Server di Log:** che si occupa di ricevere i logs, relativi alla navigazione nell'interfaccia utente, dai Clients. Il Server di Log valida il singolo log XML ricevuto dal Client e lo salva in un file di testo (file di log aperto);
- **Client:** Il client è la parte del progetto che permette lo scambio di messaggi di testo. Dotato di interfaccia grafica, permette di visualizzare i Contatti connessi, aprire nuove finestre di conversazione e accedere al diagramma a torta UML.

2.1 Server di Log

Il Server di Log non ha interfaccia grafica. Si occupa di ricevere e conservare, su file di testo locale, i vari logs XML dai Clients connessi.

Vista Dinamica

1. All'avvio, il Server di Log carica da file di configurazione locale i parametri presenti.
2. Una volta avviato il Server, l'applicativo rimane in attesa di connessioni da parte dei vari Clients;
3. Valida i singoli logs XML quando ricevuti dai Clients;
4. Appende su file di testo locale (file di log aperto) i singoli XML validati;
5. Alla chiusura, disconnette i vari Clients connessi e termina.

File di Configurazione Locale in XML

All'avvio del sistema, il Server carica dal file di configurazione locale:

1. La porta da utilizzare per avviare il Server;
2. La locazione del file di Log locale;
3. La locazione del file XSD da utilizzare per validare i singoli log.

Messaggio di Log

Ogni messaggio di Log contiene:

1. L'indirizzo email del Contatto che lo ha inviato;

2. Il tipo di evento;
3. Data e ora dell'invio da parte del Contatto.

File di Log di Testo

Il Sistema riceve e registra un log per i seguenti eventi:

1. Eventi relativi all'utilizzo dell'interfaccia grafica, inviati da parte dei Client.

2.2 Client

Il Client ha interfaccia grafica. Permette all'utente di visualizzare i Contatti connessi, aprire nuove conversazioni con questi, inviare e ricevere messaggi ed accedere al diagramma a Torta nella finestra Statistiche.

Vista Dinamica

1. All'avvio dell'applicativo, il Client legge dal file di configurazione locale i parametri presenti;
2. Quando l'utente preme il pulsante "Connetti", il Client si connette al Server di Log e si registra nella Base di Dati con l'email inserita;
3. Il Client legge da Base di Dati i Contatti connessi e li carica nella Tabella "Contatti";
4. Il Client attende le connessioni da parte dei vari Contatti;
5. A intervalli di tempo determinati, il Client aggiorna la lista Contatti connessi;
6. Lo stato di connessione viene aggiornato impostando il testo della Stringa "Stato" che si trova sotto la lista dei Contatti;
7. L'utente può aprire una nuova finestra di conversazione con un Contatto facendo doppio Click sul nome del contatto nella lista Contatti;
8. La finestra di conversazione aperta permette lo scambio di messaggi con il Contatto selezionato;
9. L'utente può utilizzare il menu "Strumenti" per accedere alla finestra "Statistiche";
10. La finestra statistiche contiene un diagramma a torta che rappresenta le ore di attività dei 5 Contatti più connessi;
11. Alla chiusura del Client, il Contatto viene rimosso dalla Base di Dati, si disconnette dal Server di Log e termina;
12. Alla disconnessione di un Client dal servizio di messaggistica:

- (a) Vengono notificati della sua disconnessione, oltre che il Server di Log, tutti i Client che hanno una conversazione aperta con esso.
- (b) Vengono aggiornate le ore di attività nella tabella "storico" della Base di dati.

File di Configurazione Locale in XML

All'avvio del sistema, il Client carica dal file di configurazione locale:

1. Indirizzo e porta del Server di Log a cui connettersi;
2. La porta su cui avviare il Client;
3. L'indirizzo, la porta, il nome utente e la password da utilizzare per accedere alla Base di dati;
4. Il numero di ultimi messaggi di ciascuna conversazioni da salvare in Cache.

Cache Locale

Alla chiusura di una finestra di conversazione con un Contatto, il Sistema salva su file binario gli ultimi N messaggi scambiati. All'avvio della successiva conversazione con il Contatto, carica dal suddetto file gli ultimi N messaggi.

Base di Dati

Il Sistema archivia le seguenti informazioni su base di dati:

1. Connessione e disconnessione dei vari Client (email, indirizzo IP, data e ora);
2. I messaggi inviati dai vari Client (mittente, destinatario, messaggio, data e ora).
3. Il numero di ore che ogni Contatto spende connesso al Servizio di Messaggistica.

Struttura Tabelle Base di Dati

Contatti

```
contatti (email , indirizzo_ip , data)
email: VARCHAR(45) , PRIMARY KEY,
indirizzo_ip: VARCHAR(45) ,
data: TIMESTAMP
```

La tabella Contatti contiene gli indirizzi Email dei vari Clients connessi e l'indirizzo IP a questi associato.

Ogni contatto è caratterizzato da un indirizzo Email, da un indirizzo IP e dall'ora in cui si è connesso.

Non è possibile che due o più contatti con lo stesso indirizzo Email possano connettersi.

Messaggi

<pre>messaggi (id , mittente , destinatario , testo , data) id: INT(45) , PRIMARY KEY, mittente: VARCHAR(45) , destinatario: VARCHAR(45) , testo: MEDIUMTEXT, data: TIMESTAMP</pre>

La tabella Messaggi contiene i messaggi scambiati tra i vari Contatti.

Ogni messaggio è composto da un id numerico (ho preferito utilizzare una colonna chiave dedicata perchè altrimenti si dovrebbe coinvolgere tutte le colonne della tabella per ottenere una chiave univoca), l'indirizzo email del mittente, l'indirizzo email del destinatario, il testo del messaggio e la data e l'ora in cui è stato spedito.

Storico

<pre>storico (email , durata) email: VARCHAR(45) , PRIMARY KEY, durata: DOUBLE</pre>
--

La tabella Storico contiene, per ciascun Client, l'email e il numero di ore spese connesso. Viene utilizzata dai vari Clients per generare il diagramma a Torta.

Ogni volta che un Client si disconnette, viene utilizzata la colonna data della tabella contatti per calcolare il numero di ore che il Client è rimasto connesso.

File di Log remoto

Il Sistema invia un log per i seguenti eventi:

1. Quando l'utente preme il pulsante "Connetti" per avviare il Client;
2. Quando l'utente preme il pulsante "Disconnetti" per fermare il Client;
3. Quando l'utente accede al menu "Strumenti";
4. Quando l'utente accede al sotto-menu "Statistiche" (Strumenti → Statistiche) aprendo il frame Statistiche;

5. Quando l'utente apre una nuova finestra di conversazione con doppio click sulla tabella "Contatti";
6. Quando un Client invia un messaggio premendo sul pulsante "Invia" del Frame conversazione.

Interfaccia grafica disegnata per il Client:

Client GUI

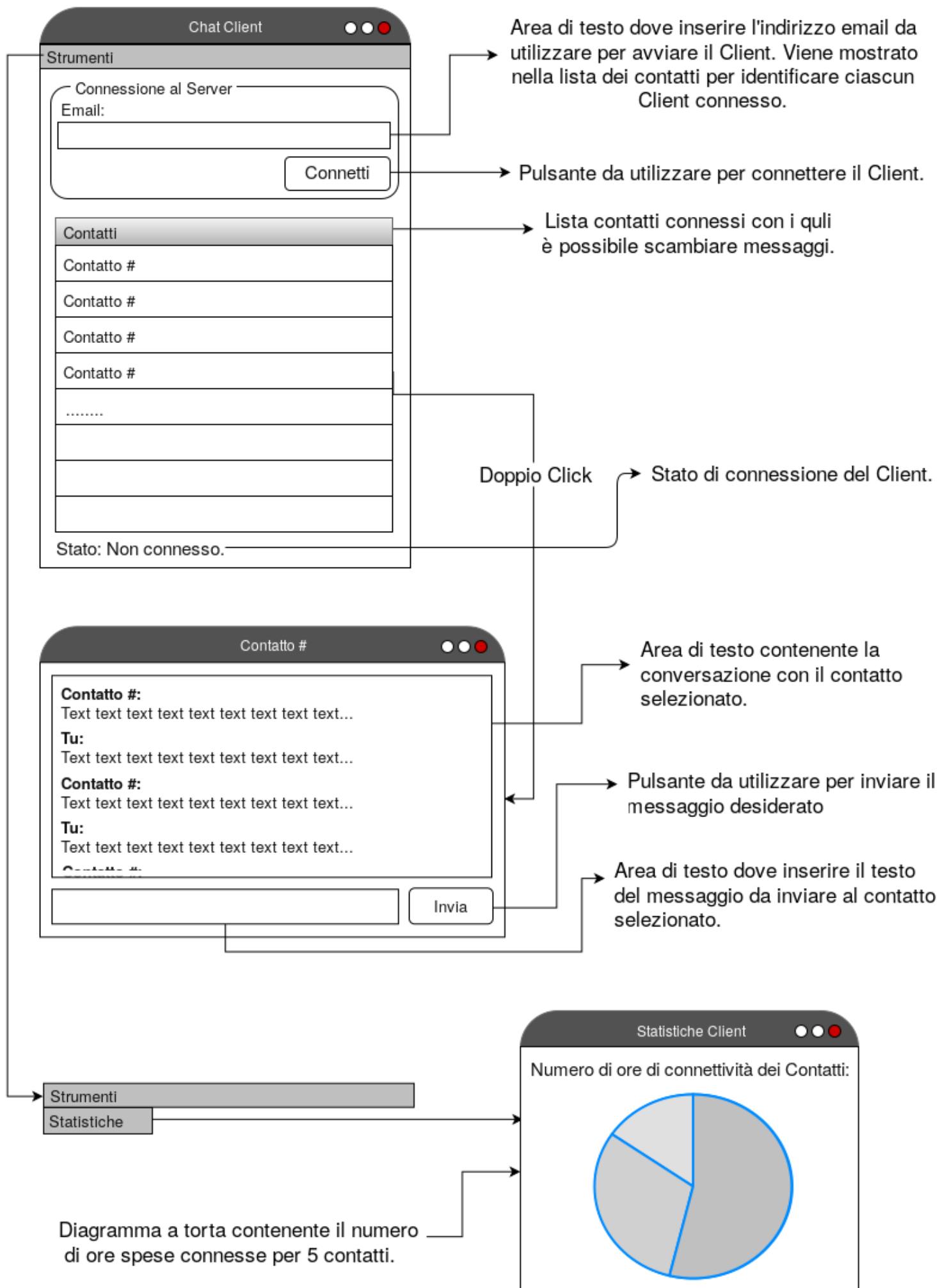


Figura 1: Interfaccia Grafica Client

3 Progetto

3.1 Server di Log

Le classi del Server di Log, come mostrate graficamente nel diagramma UML, sono le seguenti:

Classe Server

Classe contenente la funzione main() del progetto.

Gestisce l'Input/Output da terminale per la gestione (avvio, stampa logs, stampa lista Client connessi, arresto) del Server di Log.

Classe ParametriConfigurazione

Contiene i parametri di configurazione letti dal file di configurazione .xml locale.

Utilizzato dalla classe GestoreParametriConfigurazioneXML.

Serializzata/Deserializzata tramite XMLStream.

Classe GestoreParametriConfigurazioneXML

Legge i parametri di configurazione dal file di configurazione .xml locale.

Valida i parametri tramite XML Schema con file .xsd locale.

Salva i parametri di configurazione in una istanza statica della classe ParametriConfigurazione.

Classe GestoreLogsXML

Gestisce l'accesso sincronizzato di lettura e scrittura sul file di log .txt locale.

Valida i log ricevuti tramite XML Schema con file .xsd locale.

Classe ThreadServerSocket

Estende Thread.

Gestisce il ServerSocket in attesa delle connessioni da parte dei Client.

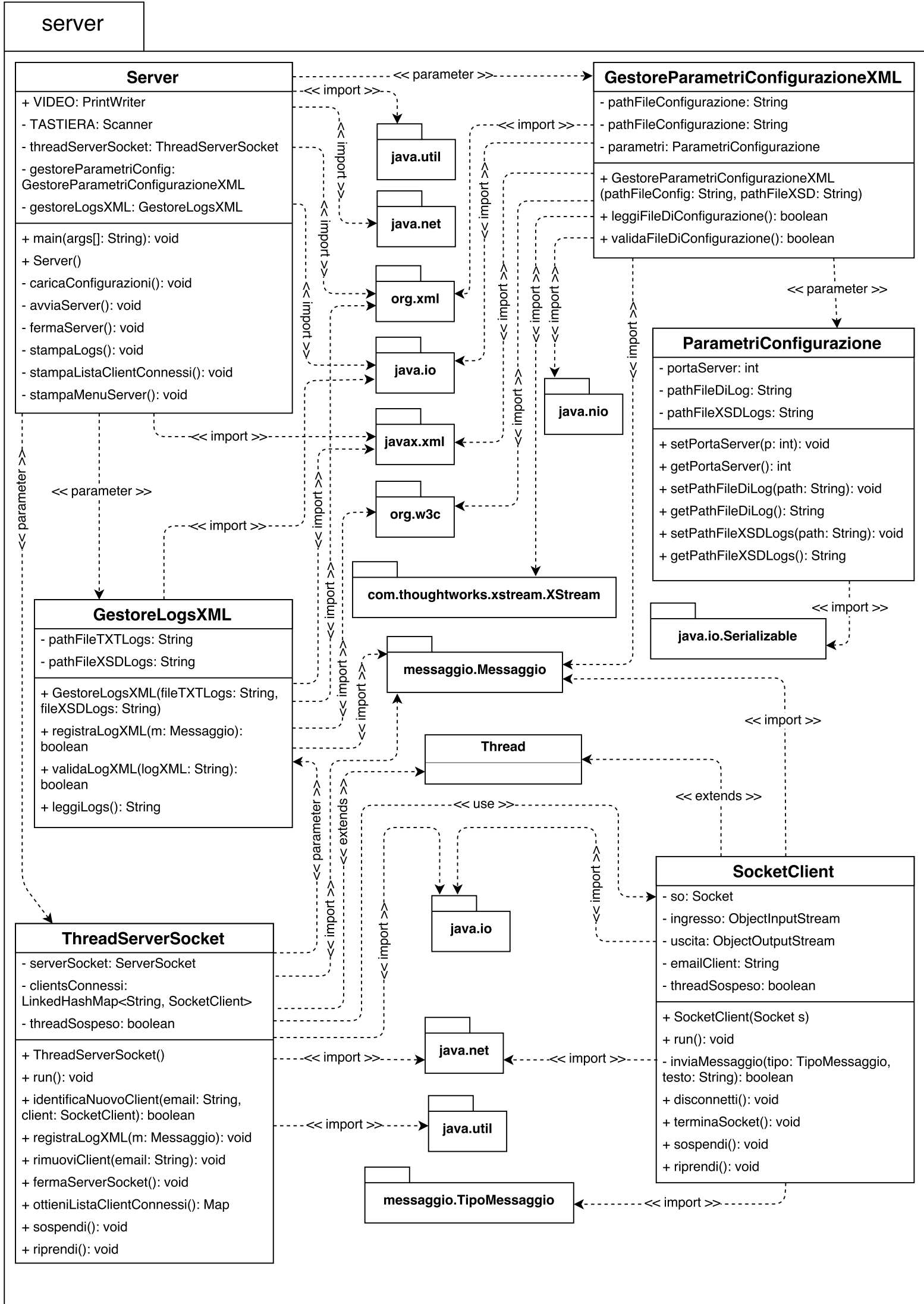
Classe SocketClient

Estende Thread.

Gestisce il Socket di comunicazione tra Server di Log e ciascun Client connesso.

Diagramma UML:

N.B: I file PDF dei diagrammi UML sono disponibili singolarmente nella cartella "2 - Progetto".



3.2 Client

Le classi del Client, come mostrate graficamente nel diagramma UML, sono le seguenti:

Classe Client

Classe contenente la funzione main() del progetto.

Carica i parametri di configurazione utilizzando la classe GestoreParametriDiConfigurazione.

Se la lettura dei parametri di configurazione termina con success, istanzia una variabile di tipo FrameContatti contenente la GUI principale del progetto.

Classe Contatto

Contiene le informazioni di ciascun Contatto connesso.

Utilizzata dalla classe GestoreBaseDiDati per ottenere la lista dei contatti connessi.

Classe FrameContatti

Contiene il JFrame con la GUI principale (area connessione, lista contatti connessi, menu strumenti).

Istanza una variabile di tipo ServerSocketConversazioni che avvia un ServerSocket che rimane in ascolto delle connessione da parte degli altri Contatti.

Classe FrameConversazione

Contiene il JFrame con la GUI della finestra di conversazione.

Istanza una variabile di tipo SocketConversazione che si connette al ServerSocketConversazioni del contatto con cui si desidera conversare.

Classe FrameStatistiche

Contiene il JFrame con la GUI della finestra delle statistiche.

Carica il pannello JavaFX contenente il diagramma a Torta delle statistiche.

Classe GestoreBaseDiDati

Esegue tutte le query sulla Base di Dati.

Classe GestoreCacheConversazioni

Si occupa di salvare e prelevare da file binario locale gli ultimi messaggi scambiati con ciascun Contatto.

Classe GestoreParametriConfigurazioneXML

Legge i parametri di configurazione dal file di configurazione .xml locale.
Valida i parametri tramite XML Schema con file .xsd locale.

Classe ParametriConfigurazione

Contiene i parametri di configurazione letti dal file di configurazione .xml locale dalla classe GestoreParametriConfigurazioneXML.
Serializzata/Deserializzata tramite XMLStream.

Classe ModelloTabellaContatti

Estende AbstractTableModel.
Implementa i metodi per AbstractTableModel.
Gestisce i dati della tabella contenente la lista dei Contatti connessi.

Classe ServerSocketConversazioni

Gestisce il ServerSocket in attesa delle connessioni da parte degli altri Contatti.
Estende Thread.

Classe SocketConversazione

Gestisce il Socket di comunicazione tra due Clients connessi che scambiano messaggi.
Estende Thread.

Classe SocketServerDiLog

Gestisce il Socket di comunicazione con il Server di Log.
Estende Thread.

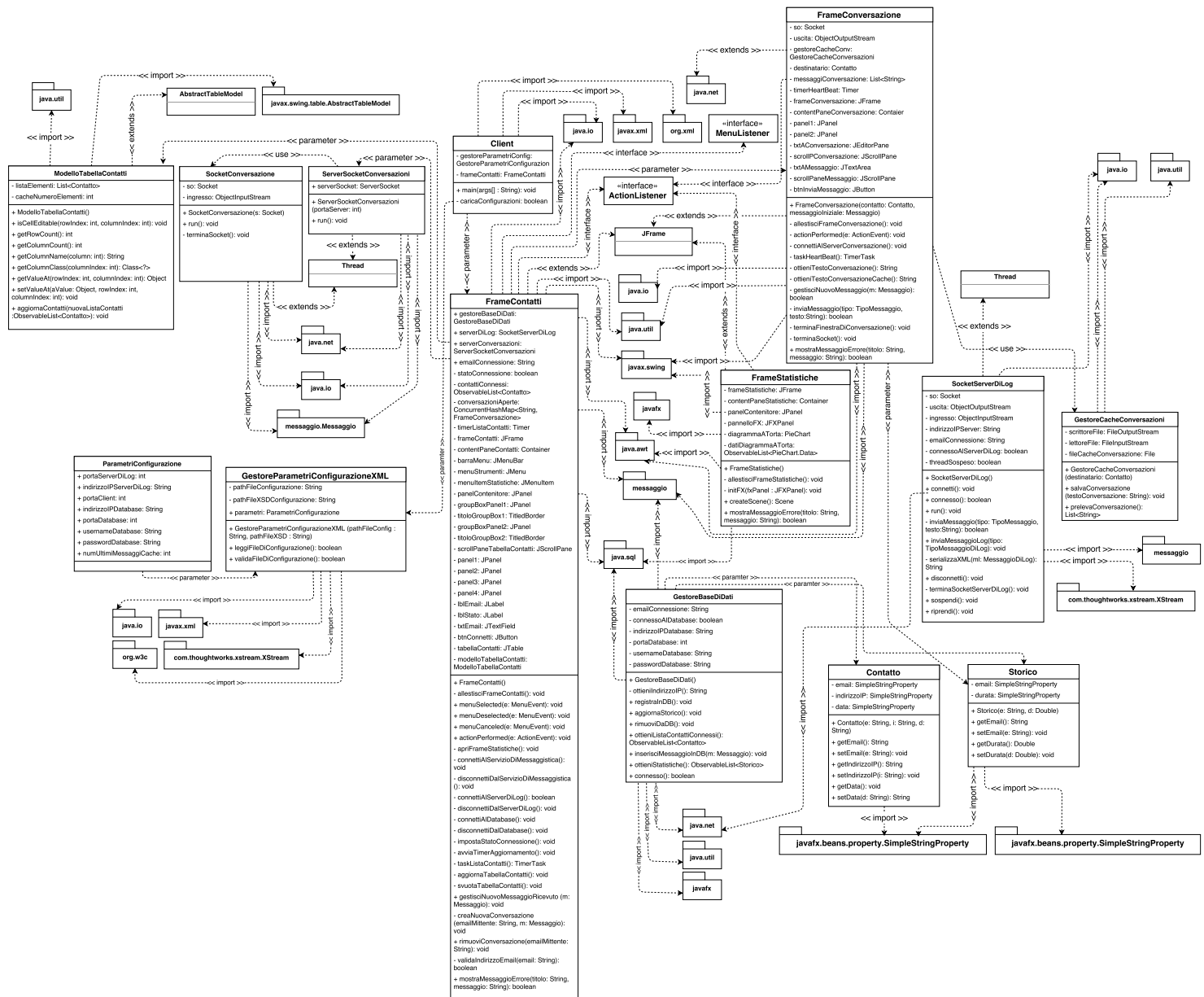
Classe Storico

Contiene le informazioni riguardanti il numero di ore di connessione per ciascun Contatto.
Utilizzata dalla classe GestoreBaseDiDati per ottenere la lista delle ore spese connesse per ciascun contatti.

Diagramma UML:

N.B: I file PDF dei diagrammi UML sono disponibili singolarmente nella cartella "2 - Progetto".

client



3.3 Classi Comuni

Client e Server di Log hanno in comune il package messaggio, che contiene le classi Messaggio, MessaggioDiLog, TipoMessaggio e TipoMessaggioDiLog, e che verrà sviluppato sotto forma di una Java Class Library, in modo da ottenere un .jar da includere nei due progetti che ne necessitano.

Essendo messaggio l'oggetto serializzato che viene scritto e letto da ObjectOutputStream e ObjectInputStream, conviene avere un unico file .jar da includere nel progetto Server e Client in modo da non incorrere in errori di incongruenza.

Classe Messaggio

Implementa Serializable.

Contiene i campi di ciascun messaggio scambiato tra Server e Client o tra due Client.

Classe TipoMessaggio

Implementa Serializable.

Enumerazione contenente i tipi di messaggio che possono essere creati.

Classe MessaggioDiLog

Implementa Serializable.

Contiene i campi di ciascun messaggio di log inviato dai Client al Server di Log.

Classe TipoMessaggioDiLog

Implementa Serializable.

Enumerazione contenente i tipi di messaggio di log che possono essere creati.

Diagramma UML:

N.B: I file PDF dei diagrammi UML sono disponibili singolarmente nella cartella "2 - Progetto".

messaggio

Messaggio

- tipo: TipoMessaggio
- mittente: String
- destinatario: String
- testo: String

+ Messaggio(t: TipoMessaggio, m: String, d: String, te: String)
+ setTipo(t: TipoMessaggio): void
+ getTipo(): TipoMessaggio
+ setMittente(m: String): void
+ getMittente(): String
+ setDestinatario(d: String): void
+ getDestinatario(): String
+ setTesto(te: String): void
+ getTesto(): String

<< Enumeration >>
TipoMessaggio

MESSAGGIO_NORMALE
MESSAGGIO_LOG
MESSAGGIO_HEARTBEAT
IDENTIFICAZIONE_CLIENT
IDENTIFICAZIONE_CLIENT_FALLITA
CLIENT_DISCONNESSO
SERVER_LOG_ARRESTATO

tipo

<< import >>

java.io.Serializable

<< import >>

<< interface >>

<<interface>>
Serializable

<< Interface >>

MessaggioDiLog

- tipo: TipoMessaggioDiLog
- mittente: String
- data: String

+ MessaggioDiLog(m: String, t: TipoMessaggio)
+ setTipo(t: TipoMessaggioDiLog): void
+ getTipo(): TipoMessaggioDiLog
+ setMittente(m: String): void
+ getMittente(): String
+ setData(d: String): void
+ getData(): String

<< Enumeration >>
TipoMessaggioDiLog

CLICK_PULSANTE_CONNETTI
CLICK_PULSANTE_DISCONNETTI
CLICK_MENU_STRUMENTI
CLICK_MENU_STATISTICHE
CLICK_PULSANTE_INVIA
DOPPIO_CLICK_LISTA_CONTATTI

tipo

<< import >>

java.io.Serializable

<< import >>

<< interface >>

<<interface>>
Serializable

<< Interface >>

<< import >>

java.sql.Timestamp

4 Manuale d'uso

Il presente manuale d'uso spiega come utilizzare il Server di Log e il Client del progetto.

Il Server di Log non ha interfaccia grafica e può essere gestito utilizzando la linea di comando, mentre il Client ha un'interfaccia grafica composta da differenti finestre di controllo.

4.1 Server di Log

Prima di avviare il Server di Log assicurarsi che:

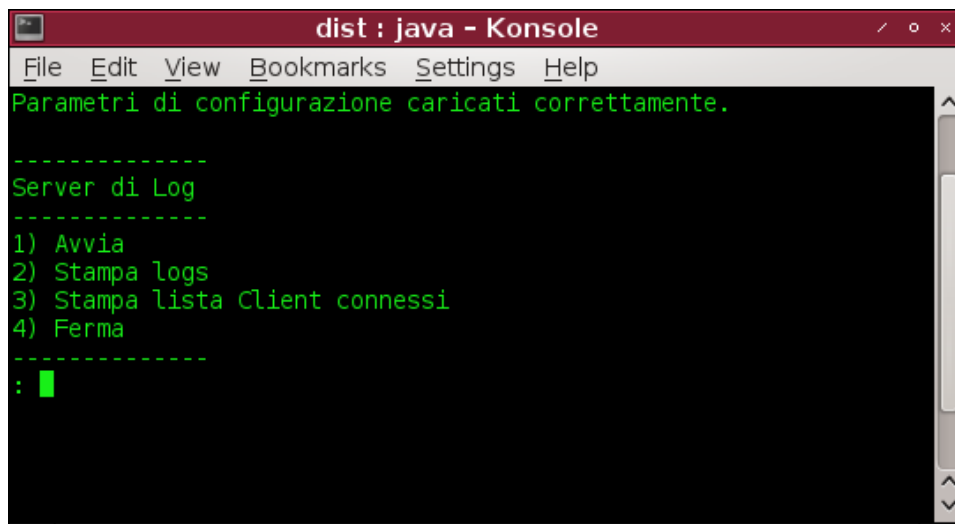
1. Il file configurazione.xml sia presente nella cartella da cui viene eseguito il Server di Log.
2. Il file configurazione.xsd sia presente nella cartella da cui viene eseguito il Server di Log.
3. Il file validaLogs.xsd sia presente nella cartella da cui viene eseguito il Server di Log.
4. Siano stati inseriti i parametri corretti nel file configurazioni.xml.

Il Server di Log può essere avviato da linea di comando con il seguente comando:

```
$ java -jar Server.jar
```

Oppure è possibile utilizzare lo script .bat, vedi sezione 4.3.

Una volta avviato il Server di Log da linea di comando, verrà stampato il seguente Menu su terminale:

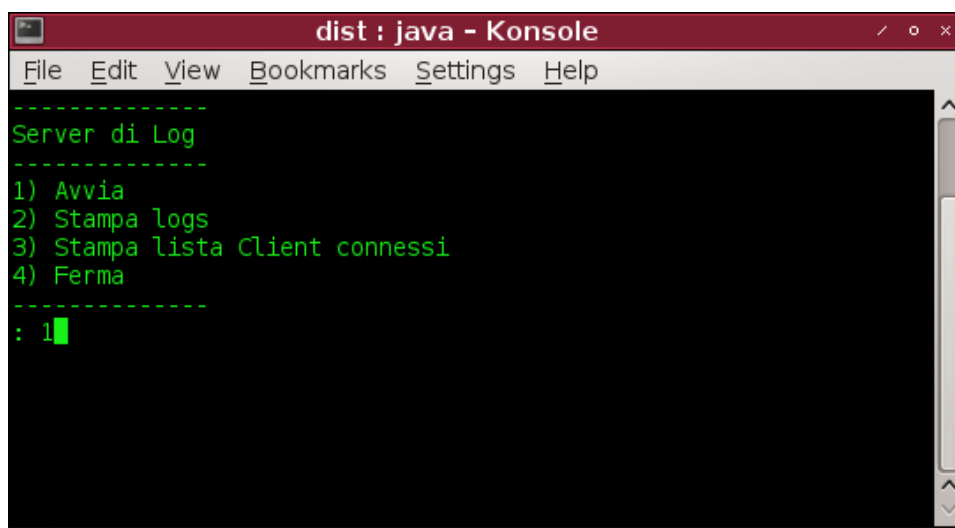


```
dist : java - Konsole
File Edit View Bookmarks Settings Help
Parametri di configurazione caricati correttamente.

-----
Server di Log
-----
1) Avvia
2) Stampa logs
3) Stampa lista Client connessi
4) Ferma
-----
: █
```

Figura 2: Avvio Server di Log

Il Server di Log può essere avviato con input 1:



```
dist : java - Konsole
File Edit View Bookmarks Settings Help
-----
Server di Log
-----
1) Avvia
2) Stampa logs
3) Stampa lista Client connessi
4) Ferma
-----
: 1 █
```

Figura 3: Avvio Server di Log

```
dist : java - Konsole
File Edit View Bookmarks Settings Help
3) Stampa lista Client connessi
4) Ferma
-----
: 1
Server avviato. [redacted]/127.0.1.1 : 8181
-----
Server di Log
-----
1) Avvia
2) Stampa logs
3) Stampa lista Client connessi
4) Ferma
-----
: 
```

Figura 4: Avvio Server di Log

In questo stato il Server di Log rimane in attesa di comandi e stampa messaggi relativi agli eventi che si verificano:

```
dist : java - Konsole
File Edit View Bookmarks Settings Help
2) Stampa logs
3) Stampa lista Client connessi
4) Ferma
-----
: Identificato nuovo Client: utente1@prova.com
: Messaggio di Log da utente1@prova.com registrato.
: Messaggio di Log da utente1@prova.com registrato.
: Messaggio di Log da utente1@prova.com registrato.
: Identificato nuovo Client: utente2@prova.com
: Messaggio di Log da utente2@prova.com registrato.
: Messaggio di Log da utente2@prova.com registrato.
: Rimosso Client: utente1@prova.com
: Messaggio di Log da utente2@prova.com registrato.
: Rimosso Client: utente2@prova.com
-----
: 
```

Figura 5: Eventi Server di Log

Con l'input 2 possiamo stampare tutti i messaggi di Log ricevuti dai vari Client dal momento dell'avvio del Server di Log:

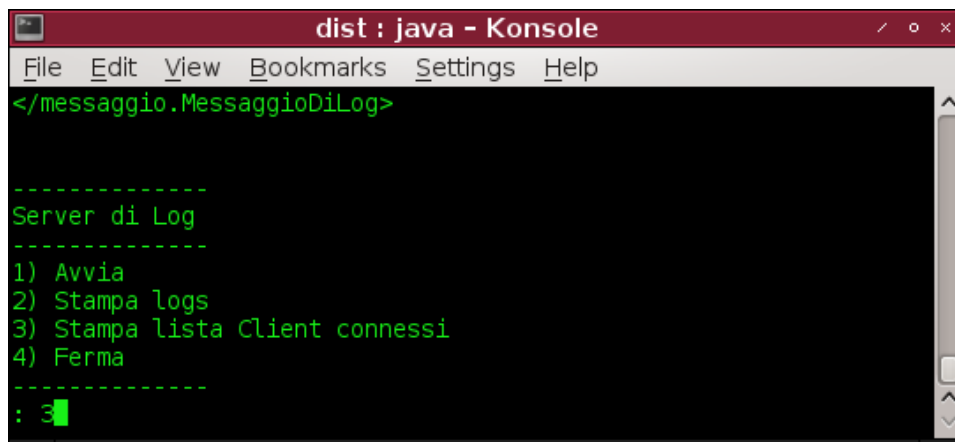

```
dist : java - Konsole
File Edit View Bookmarks Settings Help
: Messaggio di Log da utente4@prova.com registrato.
: Messaggio di Log da utente4@prova.com registrato.
: Rimosso Client: utente3@prova.com
: Identificato nuovo Client: utente5@prova.com
: Messaggio di Log da utente5@prova.com registrato.
: Messaggio di Log da utente5@prova.com registrato.
: Messaggio di Log da utente5@prova.com registrato.
: Rimosso Client: utente5@prova.com
: Identificato nuovo Client: utente6@prova.com
: Messaggio di Log da utente6@prova.com registrato.
: Messaggio di Log da utente6@prova.com registrato.
: Messaggio di Log da utente6@prova.com registrato.
: Rimosso Client: utente4@prova.com
: Rimosso Client: utente6@prova.com
: █
```

```
dist : java - Konsole
File Edit View Bookmarks Settings Help
<data>2016-06-11 14:07:00.905</data>
</messaggio.MessaggioDiLog>
<messaggio.MessaggioDiLog>
  <tipo>CLICK_MENU_STATISTICHE</tipo>
  <mittente>utente5@prova.com</mittente>
  <data>2016-06-11 14:07:01.462</data>
</messaggio.MessaggioDiLog>
<messaggio.MessaggioDiLog>
  <tipo>CLICK_PULSANTE_CONNETTI</tipo>
  <mittente>utente6@prova.com</mittente>
  <data>2016-06-11 14:07:15.234</data>
</messaggio.MessaggioDiLog>
<messaggio.MessaggioDiLog>
  <tipo>CLICK_MENU_STRUMENTI</tipo>
  <mittente>utente6@prova.com</mittente>
  <data>2016-06-11 14:07:16.007</data>
</messaggio.MessaggioDiLog>
<messaggio.MessaggioDiLog>
  <tipo>CLICK_MENU_STATISTICHE</tipo>
  <mittente>utente6@prova.com</mittente>
  <data>2016-06-11 14:07:16.997</data>
</messaggio.MessaggioDiLog>

-----
Server di Log
-----
1) Avvia
2) Stampa logs
3) Stampa lista Client connessi
4) Ferma
-----
: █
```

Figura 6: Stampa Logs Server di Log

Con l'input 3 è possibile stampare la lista dei Client connessi al Server di Log al momento:



```
dist : java - Konsole
File Edit View Bookmarks Settings Help
</messaggio.MessaggioDiLog>

-----
Server di Log
-----
1) Avvia
2) Stampa logs
3) Stampa lista Client connessi
4) Ferma
-----
: 3
```

Figura 7: Stampa Client connessi Server di Log



```
dist : java - Konsole
File Edit View Bookmarks Settings Help

-----
: 3

Client attualmente connessi al Server di Log:
utente1@prova.com
utente2@prova.com

-----
Server di Log
-----
1) Avvia
2) Stampa logs
3) Stampa lista Client connessi
4) Ferma
-----
: 3
```

Figura 8: Stampa Client connessi Server di Log

Con l'input 4 possiamo arrestare il Server di Log (viene richiesta conferma prima di procedere):

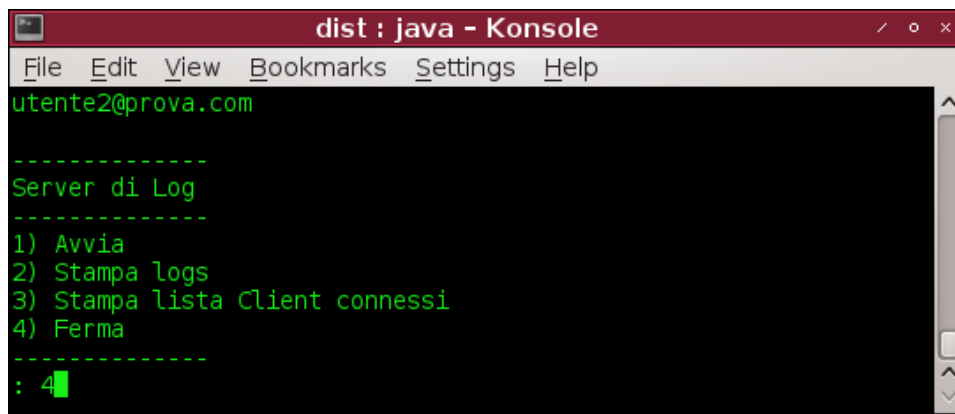


Figura 9: Arresto Server di Log

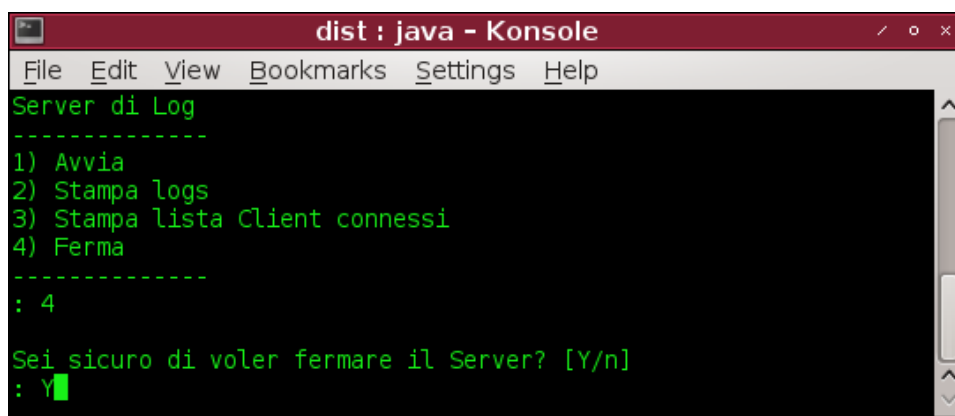


Figura 10: Arresto Server di Log

Con l'input Y il Server di Log viene terminato. L'applicativo Client è pensato per continuare a funzionare anche se il Server di Log viene fermato.

4.2 Client

Prima di avviare il Client assicurarsi che:

1. Il file configurazione.xml sia presente nella cartella da cui viene eseguito il Server di Log.
2. Il file configurazione.xsd sia presente nella cartella da cui viene eseguito il Server di Log.
3. Siano stati inseriti i parametri corretti nel file configurazioni.xml.

Il Client può essere avviato da linea di comando con il seguente comando:

```
$ java -jar Client.jar
```

Oppure è possibile utilizzare lo script .bat, vedi sezione 4.3.

Una volta avviato il Client, la prima finestra che viene mostrata all'utente è la seguente:

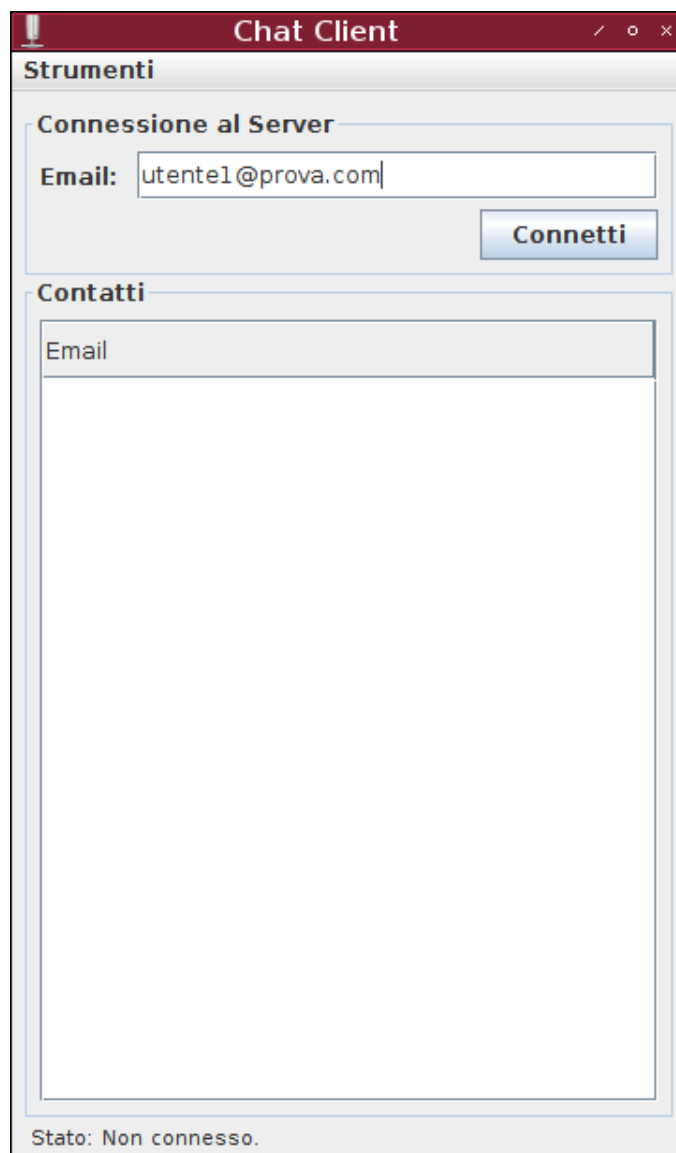


Figura 11: Avvio Client

L'utente inserisce l'indirizzo email da utilizzare e si connette al servizio di messaggistica utilizzando il pulsante "Connetti":

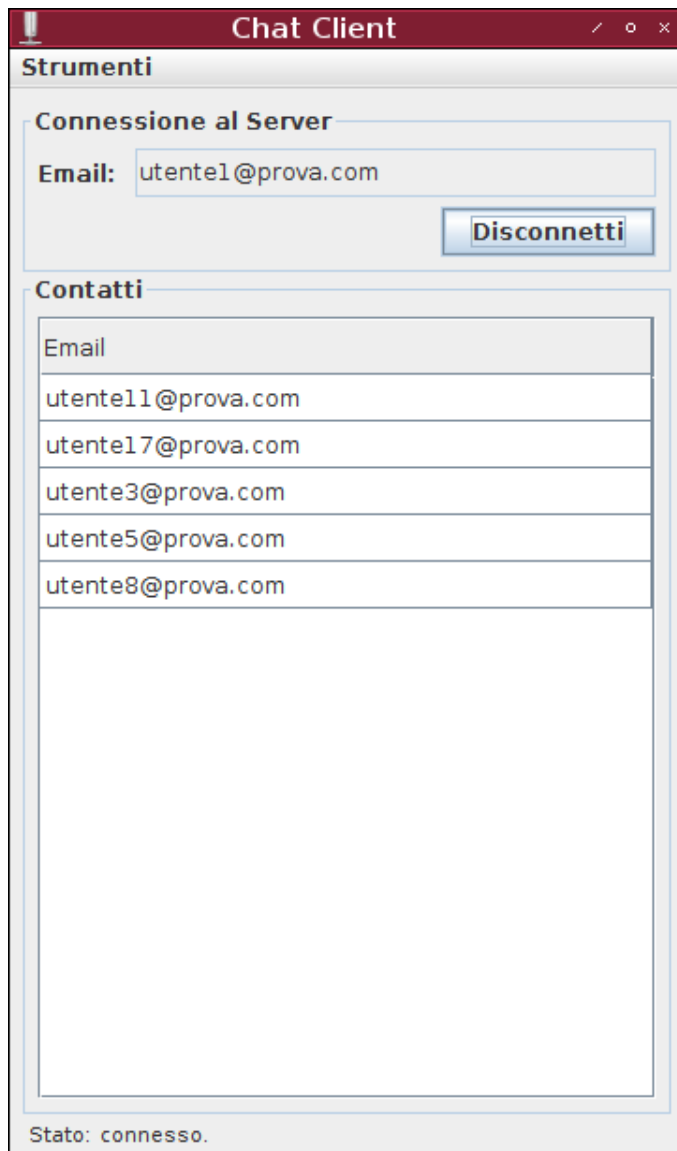
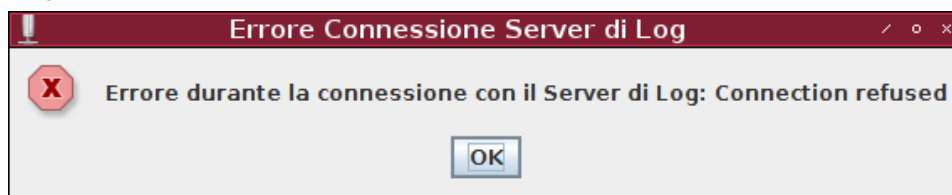


Figura 12: Client connesso

Non è possibile connettersi al servizio di messaggistica se il Server di Log non è stato avviato:



Quando un nuovo Contatto si connette al servizio di messaggistica, viene aggiornata la lista dei Contatti connessi:

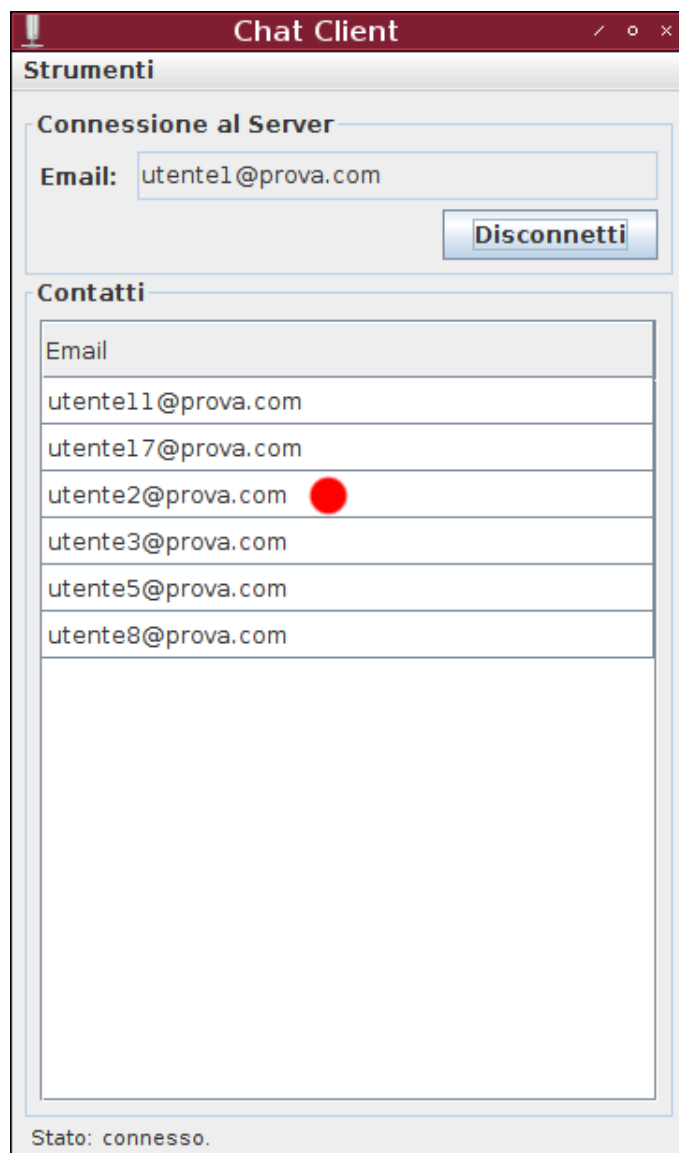


Figura 13: Aggiornamento lista Contatti connessi

Con doppio click sul nome del Contatto è possibile aprire una nuova finestra di conversazione con esso:

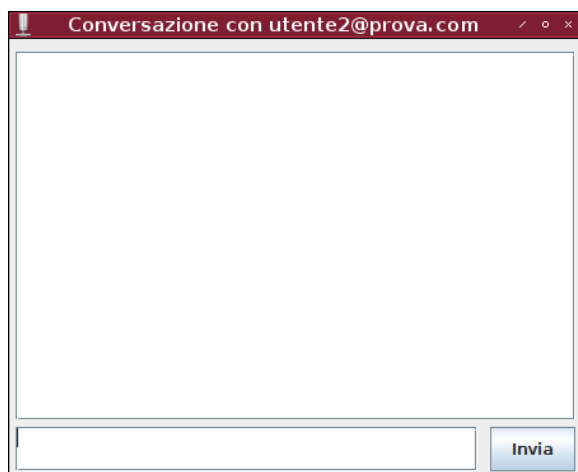
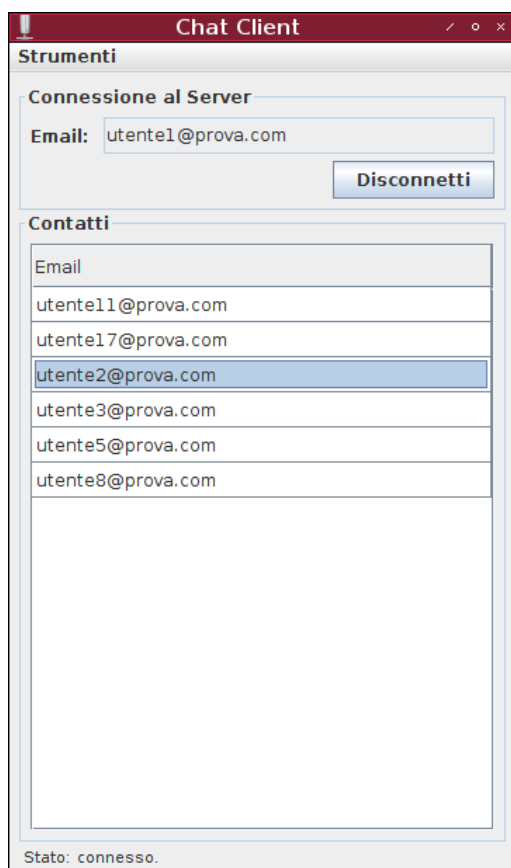


Figura 14: Apertura Nuova conversazione

Ora i due utente possono iniziare lo scambio di messaggi utilizzando l'area di testo e il pulsante "Invia":

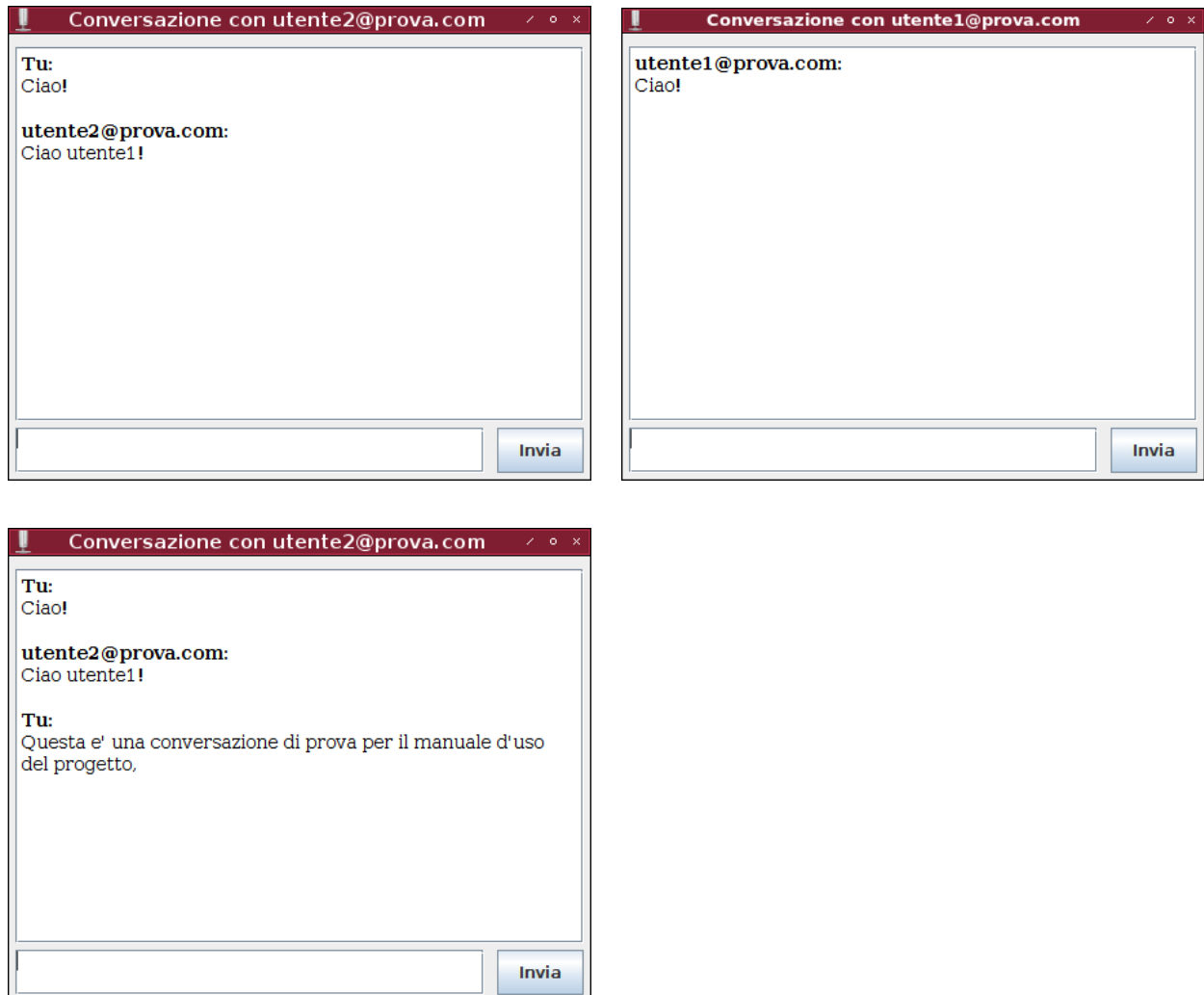


Figura 15: Scambio di messaggi tra due Contatti

Quando viene chiusa una finestra di conversazione, vengono salvati in cache gli ultimi n messaggi scambiati, come indicato nel file di configurazione locale:

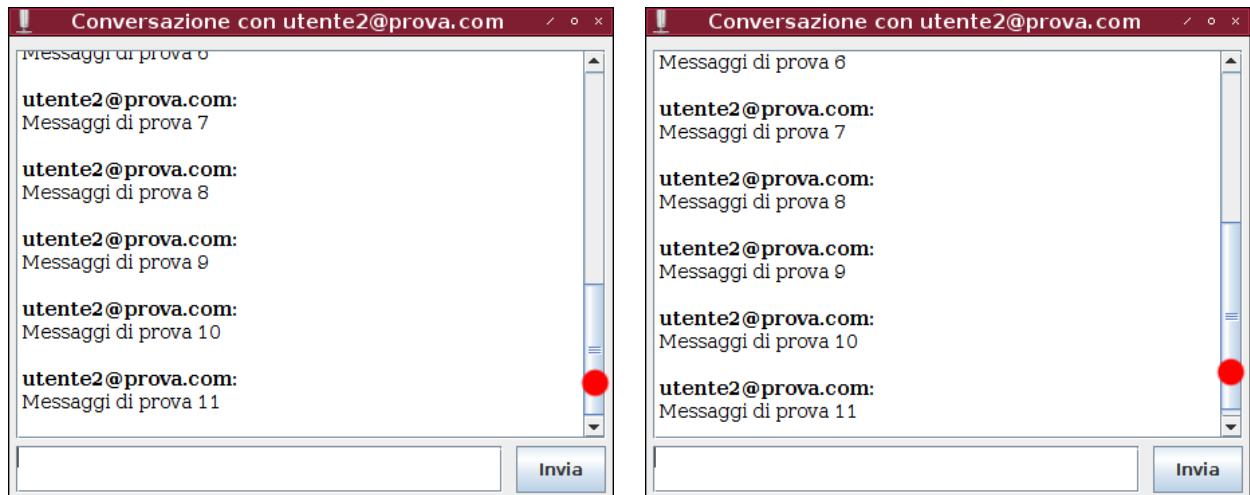


Figura 16: Caching messaggi scambiati

Quando il contatto con cui si conversa si disconnette, non è più possibile inviare messaggi nella finestra di conversazione:

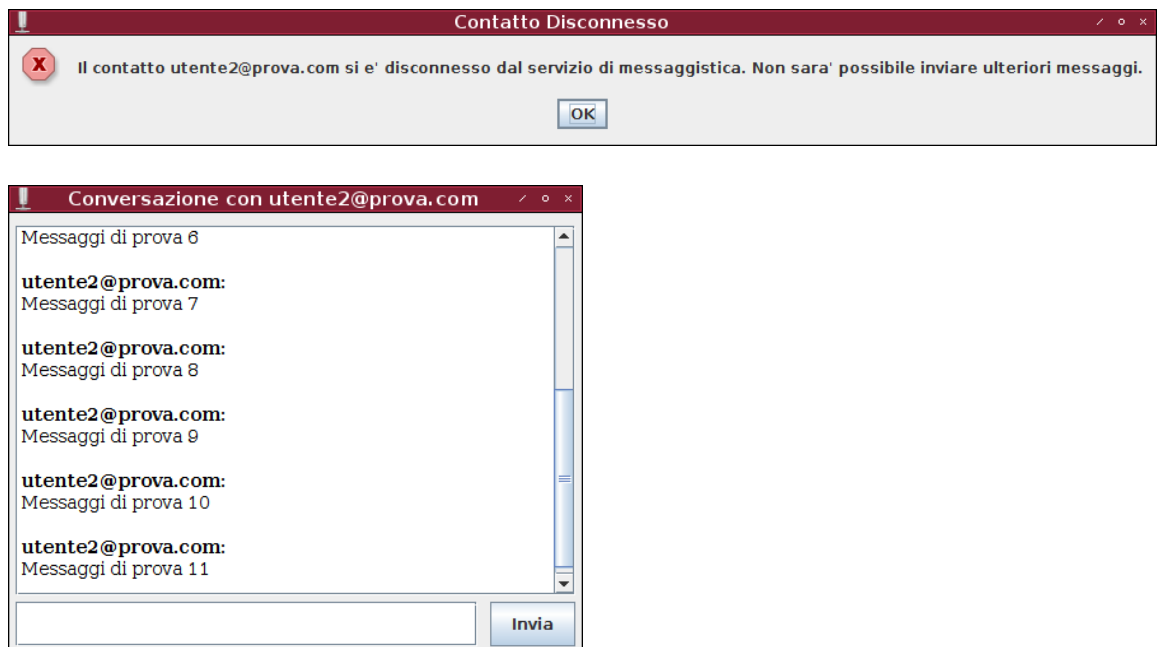


Figura 17: Disconnessione Contatto

Tramite il menu Strumenti → Statistiche, è possibile accedere alla finestra delle statistiche:

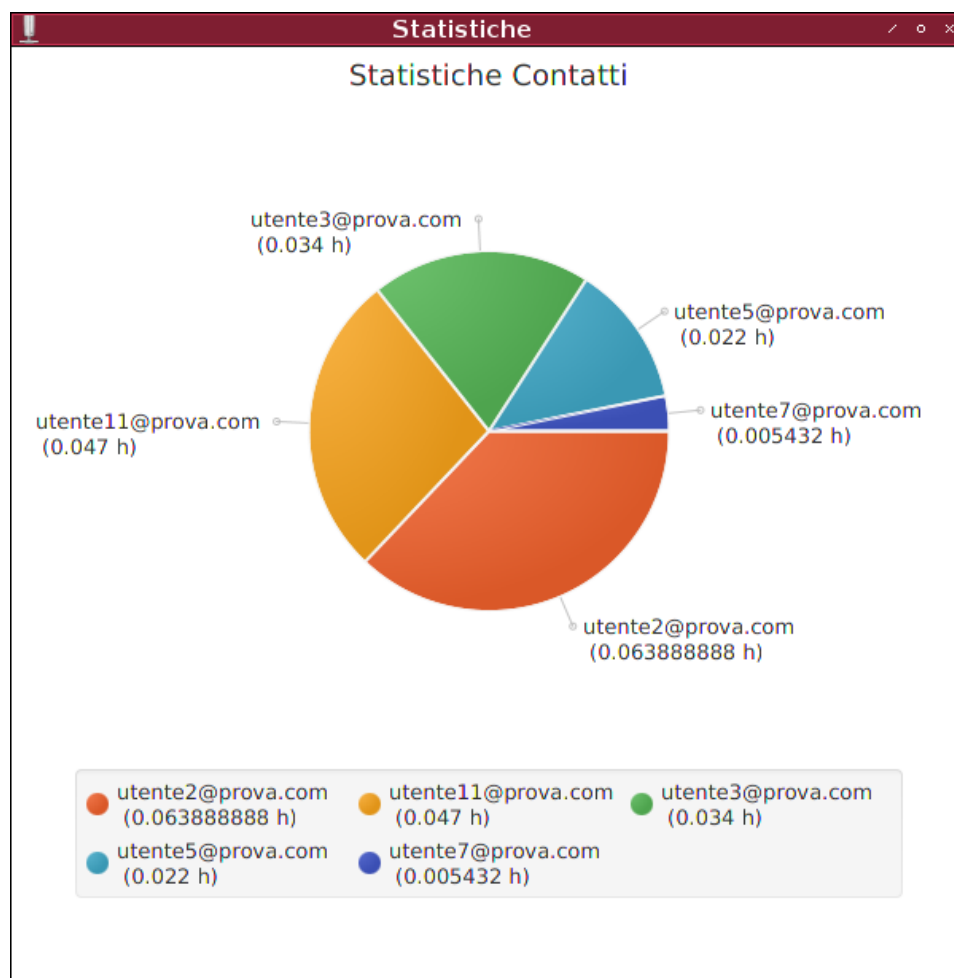


Figura 18: Finestra Statistiche

Non è possibile accedere alla finestra delle statistiche se non si è connessi al servizio di messaggistica:

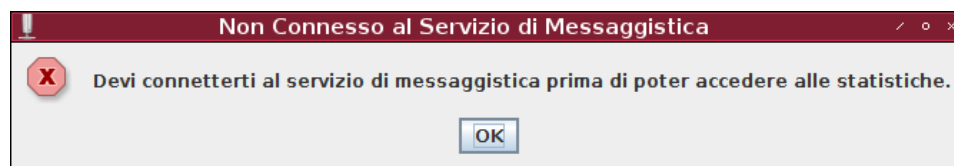


Figura 19: Finestra Statistiche

Utilizzando il pulsante "Disconnetti" è possibile disconnettersi dal servizio di messaggistica.

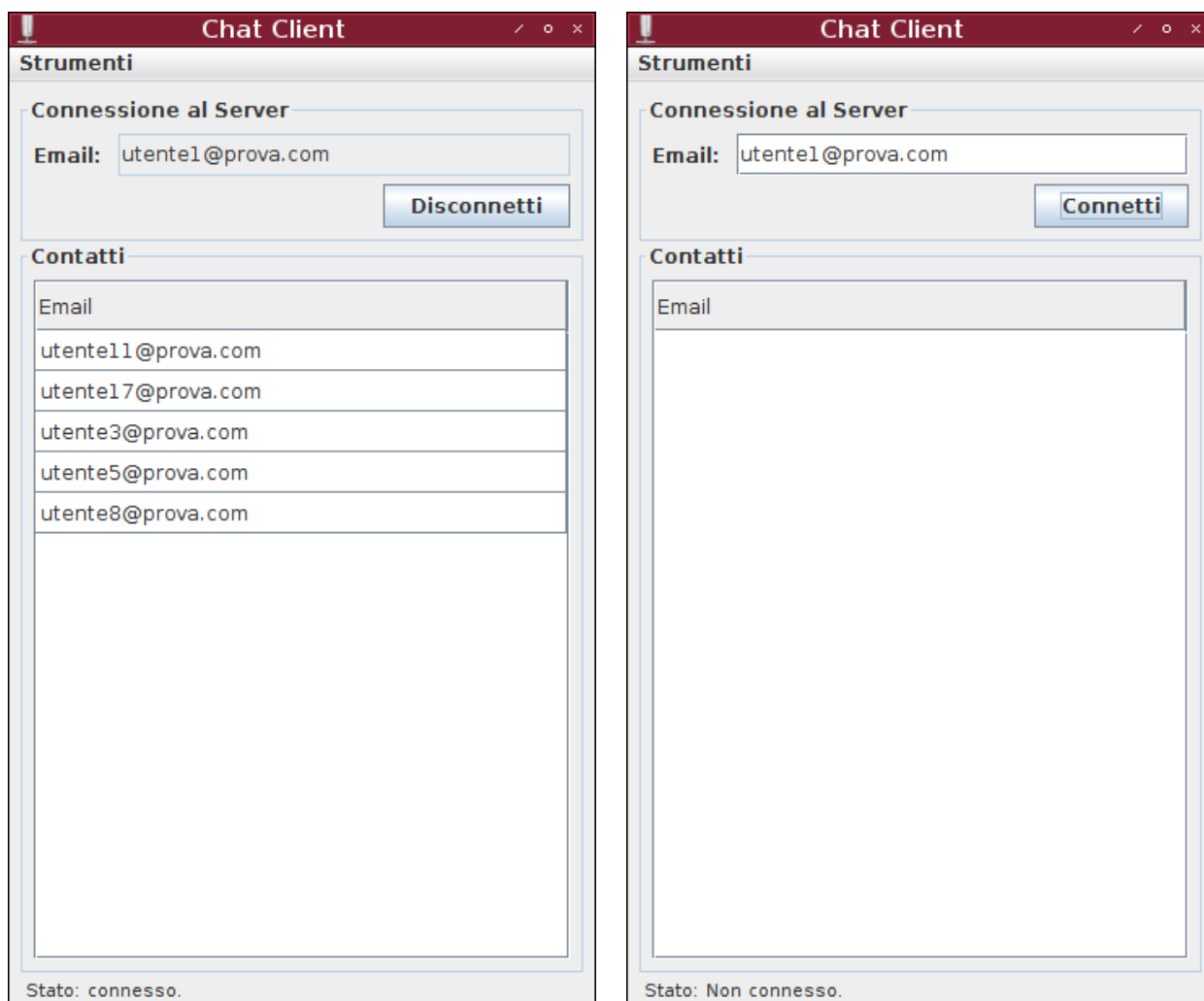


Figura 20: Disconnessione dal servizio di messaggistica

Prima di chiudere la finestra principale del Client, viene chiesta conferma:

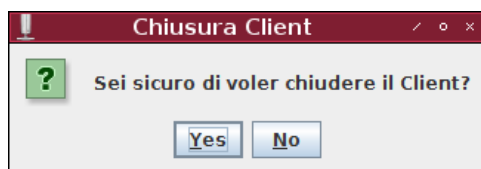
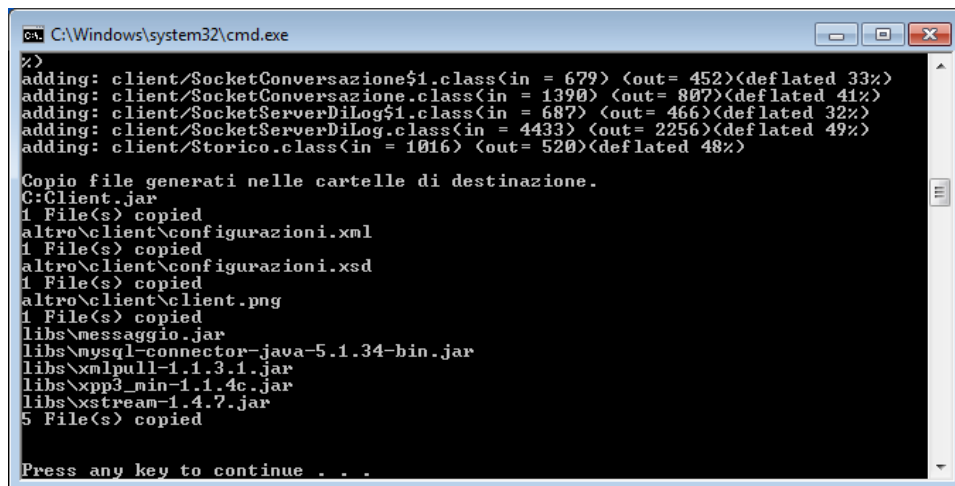


Figura 21: Finestra Statistiche

4.3 Script bat

Lo script bat "programmazione_java.bat" può essere utilizzato su macchine con sistema operativo Windows per compilare i file sorgenti dei tre progetti e ottenere due file .jar che vengono poi eseguiti.

Di seguito le screenshot relative all'utilizzo dello script .bat:

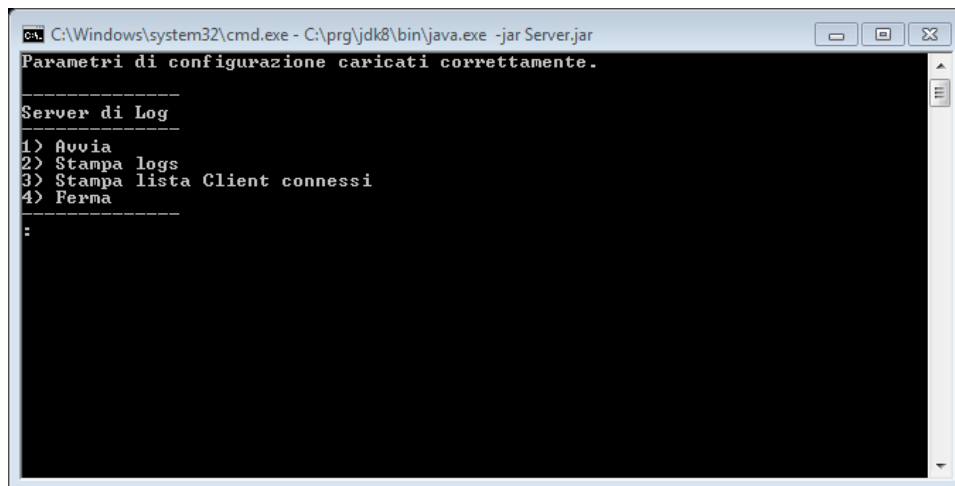


```
C:\Windows\system32\cmd.exe
%>
adding: client/SocketConversazione$1.class(in = 679) (out= 452)(deflated 33%)
adding: client/SocketConversazione.class(in = 1390) (out= 807)(deflated 41%)
adding: client/SocketServerDiLog$1.class(in = 687) (out= 466)(deflated 32%)
adding: client/SocketServerDiLog.class(in = 4433) (out= 2256)(deflated 49%)
adding: client/Storico.class(in = 1016) (out= 520)(deflated 48%)

Copia file generati nelle cartelle di destinazione.
C:\Client.jar
1 File(s) copied
altro\client\configurazioni.xml
1 File(s) copied
altro\client\configurazioni.xsd
1 File(s) copied
altro\client\client.png
1 File(s) copied
libs\messaggio.jar
libs\mysql-connector-java-5.1.34-bin.jar
libs\xmlpull-1.1.3.1.jar
libs\xpp3_min-1.1.4c.jar
libs\xstream-1.4.7.jar
5 File(s) copied

Press any key to continue . . .
```

Figura 22: Esecuzione Script bat



```
C:\Windows\system32\cmd.exe - C:\prg\jdk8\bin\java.exe -jar Server.jar
Parametri di configurazione caricati correttamente.

-----
Server di Log
-----
1) Avvia
2) Stampa logs
3) Stampa lista Client connessi
4) Ferma
-----
:
```

Figura 23: Script bat: Avvio Server di Log

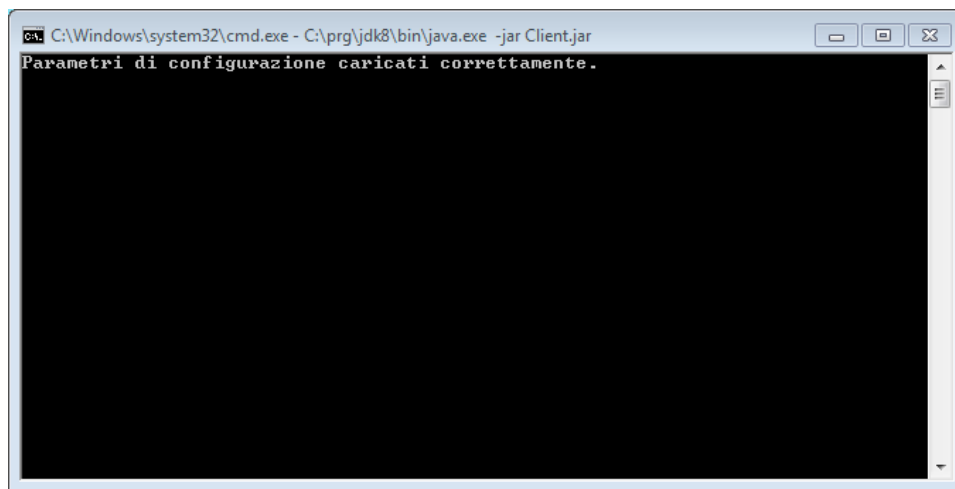


Figura 24: Script bat: Avvio Client

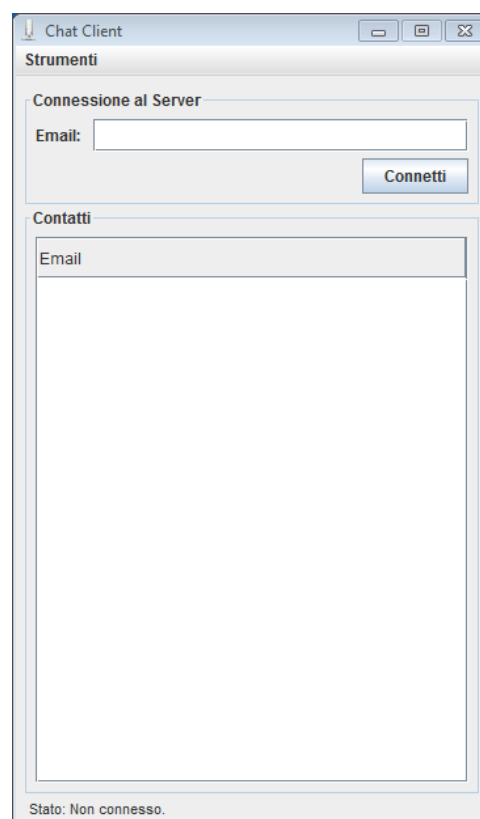


Figura 25: Script bat: Avvio Client

Bibliografia

1. D. Holmes K. Arnold, J. Gosling. *The Java Programming Language*. Addison-Wesley Professional, 4th edition, 2005.
2. A. Vecchio G. Frosini. *Programmare in Java Vol. I, II, III*. Edizioni ETS, 2005.
3. Oracle Corporation. The Java SE 8 Documentation. <https://docs.oracle.com/javase/8/>, 2014. Accessed: 2016-05-31.
4. Oracle Corporation. The Java FX 2 Documentation. <https://docs.oracle.com/javafx/2/>, 2014. Accessed: 2016-05-31.
5. Oracle Corporation. The Java FX 2 API. <https://docs.oracle.com/javase/8/docs/api/>, 2014. Accessed: 2016-05-31.
6. Oracle Corporation. The Java FX 2 API. <https://docs.oracle.com/javafx/2/api/>, 2014. Accessed: 2016-05-31.