Word Quizzle

v1.5.8

Author:
Luca Canessa (516639)

Generated by Doxygen 1.8.13

Contents

1 WQGame 2

2 Namespace Index 3

2.1 Packages . . . 3

3 Hierarchical Index 4

3.1 Class Hierarchy . . . 4

4 Class Index 5

4.1 Class List . . . 5

5 File Index 5

5.1 File List . . . 5

6 Namespace Documentation 6

6.1 Package client . . . 6

6.2 Package server . . . 6

7 Class Documentation 7

7.1 client.ACK Enum Reference . . . 7

7.1.1 Detailed Description . . . 8

7.1.2 Member Data Documentation . . . 9

7.2 server.ACK Enum Reference . . . 13

7.2.1 Detailed Description . . . 14

7.2.2 Member Data Documentation . . . 14

7.3 client.CGUI Class Reference . . . 19

7.3.1 Constructor & Destructor Documentation . . . 22

7.3.2 Member Function Documentation . . . 23

7.3.3 Member Data Documentation . . . 28

7.4 server.Challenge Class Reference . . . 34

7.4.1 Detailed Description . . . 36

7.4.2 Constructor & Destructor Documentation . . . 36

7.4.3 Member Function Documentation . . . 36
7.4.4 Member Data Documentation . . . 39
7.5 client.ClientMSG Enum Reference . . . 40
7.5.1 Detailed Description . . . 41
7.5.2 Member Data Documentation . . . 41
7.6 server.ClientMSG Enum Reference . . . 43
7.6.1 Detailed Description . . . 44
7.6.2 Member Data Documentation . . . 44
7.7 server.ClientTasks Class Reference . . . 46
7.7.1 Detailed Description . . . 48
7.7.2 Constructor & Destructor Documentation . . . 48
7.7.3 Member Function Documentation . . . 49
7.7.4 Member Data Documentation . . . 55
7.8 client.comUDP Class Reference . . . 57
7.8.1 Detailed Description . . . 59
7.8.2 Constructor & Destructor Documentation . . . 59
7.8.3 Member Function Documentation . . . 60
7.8.4 Member Data Documentation . . . 61
7.9 client.ConfParser Class Reference . . . 62
7.9.1 Detailed Description . . . 63
7.9.2 Constructor & Destructor Documentation . . . 63
7.9.3 Member Function Documentation . . . 64
7.9.4 Member Data Documentation . . . 67
7.10 server.ConfParser Class Reference . . . 68
7.10.1 Detailed Description . . . 69
7.10.2 Constructor & Destructor Documentation . . . 69
7.10.3 Member Function Documentation . . . 70
7.10.4 Member Data Documentation . . . 77
7.11 server.DBsWriter Class Reference . . . 77
7.11.1 Detailed Description . . . 78

7.11.2 Constructor & Destructor Documentation . . . 79
7.11.3 Member Function Documentation . . . 79
7.11.4 Member Data Documentation . . . 80
7.12 server.Friendships Class Reference . . . 81
7.12.1 Detailed Description . . . 82
7.12.2 Constructor & Destructor Documentation . . . 82
7.12.3 Member Function Documentation . . . 83
7.12.4 Member Data Documentation . . . 90
7.13 client.Game Class Reference . . . 91
7.13.1 Detailed Description . . . 93
7.13.2 Constructor & Destructor Documentation . . . 93
7.13.3 Member Function Documentation . . . 93
7.13.4 Member Data Documentation . . . 94
7.14 server.Main Class Reference . . . 95
7.14.1 Detailed Description . . . 96
7.14.2 Member Function Documentation . . . 96
7.14.3 Member Data Documentation . . . 102
7.15 client.Main Class Reference . . . 104
7.15.1 Detailed Description . . . 105
7.15.2 Member Function Documentation . . . 105
7.15.3 Member Data Documentation . . . 108
7.16 client.MyActionListener Class Reference . . . 109
7.16.1 Detailed Description . . . 110
7.16.2 Member Function Documentation . . . 110
7.17 server.RegRMImplementation Class Reference . . . 124
7.17.1 Detailed Description . . . 125
7.17.2 Constructor & Destructor Documentation . . . 126
7.17.3 Member Function Documentation . . . 126
7.17.4 Member Data Documentation . . . 127
7.18 server.RegRMInterface Class Reference . . . 128

7.18.1 Detailed Description . . . 129
7.18.2 Member Function Documentation . . . 129
7.19 client.Tasks Class Reference . . . 130
7.19.1 Detailed Description . . . 131
7.19.2 Member Function Documentation . . . 131
7.20 server.User Class Reference . . . 140
7.20.1 Detailed Description . . . 142
7.20.2 Constructor & Destructor Documentation . . . 143
7.20.3 Member Function Documentation . . . 143
7.20.4 Member Data Documentation . . . 150
7.21 server.Users Class Reference . . . 152
7.21.1 Detailed Description . . . 154
7.21.2 Constructor & Destructor Documentation . . . 154
7.21.3 Member Function Documentation . . . 154
7.21.4 Member Data Documentation . . . 161

8 File Documentation **162**

8.1 resources/doxyhome.md File Reference . . . 162
8.2 src/client/ACK.java File Reference . . . 162
8.3 src/server/ACK.java File Reference . . . 162
8.4 src/client/CGUI.java File Reference . . . 163
8.5 src/client/comUDP.java File Reference . . . 163
8.6 src/client/ConfParser.java File Reference . . . 163
8.7 src/server/ConfParser.java File Reference . . . 163
8.8 src/client/Game.java File Reference . . . 164
8.9 src/client/Main.java File Reference . . . 164
8.10 src/server/Main.java File Reference . . . 164
8.11 src/client/MyActionListener.java File Reference . . . 164
8.12 src/client/Tasks.java File Reference . . . 164
8.13 src/server/Challenge.java File Reference . . . 165
8.14 src/server/ClientTasks.java File Reference . . . 165
8.15 src/server/Friendships.java File Reference . . . 165
8.16 src/server/RegRMImplementation.java File Reference . . . 165
8.17 src/server/RegRMInterface.java File Reference . . . 166
8.18 src/server/User.java File Reference . . . 166
8.19 src/server/Users.java File Reference . . . 166

Index 167

1 WQGame

Requesiti

Descrizione del Problema

Il progetto consiste nell'implementazione di un sistema di sfide di traduzione italiano-inglese tra utenti registrati al servizio. Gli utenti registrati possono sfidare i propri amici ad una gara il cui scopo è quello di tradurre in inglese il maggiore numero di parole italiane proposte dal servizio. Il sistema consente inoltre la gestione di una rete sociale tra gli utenti iscritti. L'applicazione è implementata secondo una architettura client server.

Specifica delle Operazioni

Di seguito sono specificate le operazioni offerte dal servizio WQ. In sede di implementazione è possibile aggiungere ulteriori parametri se necessario.

Registrazione di un utente:
Per inserire un nuovo utente, il server mette a disposizione una operazione *registra_utente(nickUtente,password)*. Il server risponde con un codice che può indicare l'avvenuta registrazione, oppure, se il nickname è già presente, o se la password è vuota, restituisce un messaggio d'errore. Come specificato in seguito, le registrazioni sono tra le informazioni da persistere.

login(nickUtente, password):
Login di un utente già registrato per accedere al servizio. Il server risponde con un codice che può indicare l'avvenuto login, oppure, se l'utente ha già effettuato la login o la password è errata, restituisce un messaggio d'errore.

logout(nickUtente):
Effettua il logout dell'utente dal servizio.

aggiungi_amico (nickUtente, nickAmico):
Registrazione di un'amicizia: aggiungere un amico alla cerchia di amici di un utente. Viene creato un arco non orientato tra i due utenti (se A è amico di B, B è amico di A). Il Server risponde con un codice che indica l'avvenuta registrazione dell'amicizia oppure con un codice di errore, se il nickname del nodo destinazione/sorgente della richiesta non esiste, oppure se è stato richiesto di creare una relazione di amicizia già esistente. Non è necessario che il server richieda l'accettazione dell'amicizia da parte di nickAmico.

lista_amici(nickUtente):
Utilizzata da un utente per visualizzare la lista dei propri amici, fornendo le proprie generalità. Il server restituisce un oggetto JSON che rappresenta la lista degli amici.

sfida(nickUtente, nickAmico):
L'utente *nickUtente* intende sfidare l'utente di nome *nickAmico*. Il server controlla che *nickAmico* appartenga alla lista di amicizie di *nickUtente*, in caso negativo restituisce un codice di errore e l'operazione termina. In caso positivo, il server invia a nickAmico una richiesta di accettazione della sfida e, solo dopo che la richiesta è stata accettata, la sfida può avere inizio (se la risposta non è stata ricevuta entro un intervallo di tempo *T1* si considera la sfida come non accettata). La sfida riguarda la traduzione di una lista di parole italiane in parole inglesi, nel minimo tempo possibile. Il server sceglie, in modo casuale, *K* parole da un dizionario contenente *N* parole italiane da inviare successivamente, una alla volta, ai due sfidanti. La partita può durare al massimo un intervallo di tempo *T2*.

Il server invia ai partecipanti la prima parola. Quando il giocatore invia la traduzione (giusta o sbagliata), il server invia la parola successiva a quel giocatore. Il gioco termina quando entrambi i giocatori hanno inviato le traduzioni alle *K* parole o quando scade il timer. La correttezza della traduzione viene controllata dal server utilizzando un servizio esterno, come specificato nella sezione seguente. Ogni traduzione corretta assegna *X* punti al giocatore; ogni traduzione sbagliata assegna *Y* punti negativi; il giocatore con più punti vince la sfida ed ottiene *Z* punti extra. Per ogni risposta non inviata (a causa della scadenza del timer) si assegnano 0 punti. Il punteggio ottenuto da ciascun partecipante alla fine della partita viene chiamato punteggio partita. I valori espressi come *K, N, T1, T2, X, Y* e *Z* sono a discrezione dello studente.

mostra_punteggio(nickUtente):
Il server restituisce il punteggio di *nickUtente* (chiamato "punteggio utente") totalizzato in base ai punteggi partita ottenuti in tutte le sfide che ha effettuato.

mostra_classifica(nickUtente):
Il server restituisce in formato JSON la classifica calcolata in base ai punteggi utente ottenuti da *nickUtente* e dai suoi amici.

Specifiche di Implementazione

Nella realizzazione del progetto devono essere utilizzate molte tecnologie illustrate durante il corso.

In particolare:

- la fase di registrazione viene implementata mediante RMI.
- La fase di login deve essere effettuata come prima operazione dopo aver instaurato una connessione TCP con il server. Su questa connessione TCP, dopo previa login effettuata con successo, avvengono le interazioni client- server (richieste/risposte).
- Il server inoltra la richiesta di sfida originata da nickUtente all'utente nickAmico usando la comunicazione UDP.
- Il server può essere realizzato multithreaded oppure può effettuare il multiplexing dei canali mediante NIO.
- Il server gestisce un dizionario di N parole italiane, memorizzato in un file. Durante la fase di setup di una sfida fra due utenti il server seleziona K parole a caso su N parole presenti nel dizionario. Prima dell'inizio della partita, ma dopo che ha ricevuto l'accettazione della sfida da parte dell'amico, il server chiede, tramite una chiamata HTTP GET, la traduzione delle parole selezionate al servizio esterno accessibile alla URL `https://mymemory.translated.net/doc/spec.php` . Le traduzioni vengono memorizzate per tutta la durata della partita per verificare la correttezza delle risposte inviate dal client.
- L'utente interagisce con WQ mediante un client che può utilizzare una semplice interfaccia grafica, oppure una interfaccia a linea di comando, definendo un insieme di comandi, presentati in un menu.
- Il server persiste le informazioni di registrazione, relazioni di amicizia e punteggio degli utenti su file JSON.

2 Namespace Index

2.1 Packages

Here are the packages with brief descriptions (if available):

client **6**

server 6

3 Hierarchical Index

3.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

client.ACK 7

server.ACK 13

client.ClientMSG 40

server.ClientMSG 43

client.ConfParser 62

server.ConfParser 68

server.Friendships 81
JFrame

- **client.CGUI** 19

server.Main 95

client.Main 104

client.MyActionListener 109
RemoteServer

- **server.RegRMImplementation** 124
Runnable

- **client.comUDP** 57
- **client.Game** 91
- **server.Challenge** 34
- **server.ClientTasks** 46
- **server.DBsWriter** 77

client.Tasks 130
Remote

- **server.RegRMInterface** 128
  - **server.RegRMImplementation** 124
Serializable

- **server.User** 140
- **server.Users** 152

4 Class Index

4.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

client.ACK 7

server.ACK 13

client.CGUI 19

server.Challenge 34

client.ClientMSG 40

server.ClientMSG 43

server.ClientTasks 46

client.comUDP 57

client.ConfParser 62

server.ConfParser 68

server.DBsWriter 77

server.Friendships 81

client.Game 91

server.Main 95

client.Main 104

client.MyActionListener 109

server.RegRMImplementation 124

server.RegRMInterface 128

client.Tasks 130

server.User 140

server.Users 152

5 File Index

5.1 File List

Here is a list of all files with brief descriptions:

src/client/ACK.java 162

src/client/CGUI.java 163

src/client/comUDP.java 163

src/client/ConfParser.java 163

src/client/Game.java 164

src/client/Main.java 164

src/client/MyActionListener.java 164

src/client/Tasks.java 164

src/server/ACK.java 162

src/server/Challenge.java 165

src/server/ClientTasks.java 165

src/server/ConfParser.java 163

src/server/Friendships.java 165

src/server/Main.java 164

src/server/RegRMImplementation.java 165

src/server/RegRMInterface.java 166

src/server/User.java 166

src/server/Users.java 166

6 Namespace Documentation

6.1 Package client

Classes

- enum ACK
- class CGUI
- enum ClientMSG
- class comUDP
- class ConfParser
- class Game
- class Main
- class MyActionListener
- class Tasks

6.2 Package server

Classes

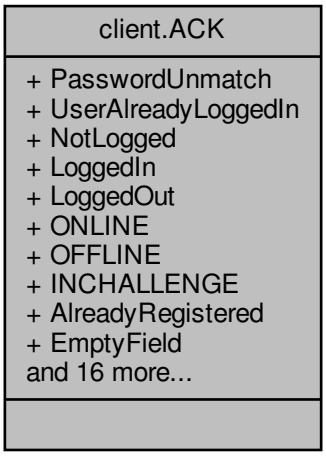
- enum ACK
- class Challenge
- enum ClientMSG

- class ClientTasks
- class ConfParser
- class DBsWriter
- class Friendships
- class Main
- class RegRMImplementation
- class RegRMInterface
- class User
- class Users

7 Class Documentation

7.1 client.ACK Enum Reference

Collaboration diagram for client.ACK:



Public Attributes

- PasswordUnmatch

  password non corrisponde a quella del User
- UserAlreadyLoggedIn

  User già entrato.
- NotLogged

  < User non loggato
- LoggedIn

  User loggato.
- LoggedOut

  User uscito.
- ONLINE

stato dell'User online

- OFFLINE

  stato dell'User offline
- INCHALLENGE

  stato dell'User in sfida
- AlreadyRegistered

  User già registrato.
- EmptyField

  campi utente vuoti
- UserRegistered

  User registrato.
- RegistrationError

  problema generale durante la registrazione dell'User
- UserFound

  User trovato nel database.
- UserNotFound

  User non trovato nel database.
- UserDeleted

  User deregistrato.
- UserNotDeleted

  User non deregistrato.
- AlreadyFriends

  User(s) già amici.
- FriendNotFound

  User amico non trovato.
- FriendAdded

  User registrato come amico.
- UserAdded

  User inserito nel database.
- AlreadyExisting
- FriendRemoved

  User amico rimosso dalla lista degli amici.
- Accepted

  sfida accettata
- Rejected

  sfida rifiutata
- OperationUnknown

  operazione richiesta non riconosciuta
- OK

  azione eseguita

7.1.1 Detailed Description

In questo file vengono implementati i messaggi ricevuti dal server a termine delle operazioni richieste

Author

Luca Canessa (Mat. 516639)

Version

1.0

Since

1.0

7.1.2 Member Data Documentation

7.1.2.1 Accepted

`client.ACK.Accepted`

sfida accettata

7.1.2.2 AlreadyExisting

`client.ACK.AlreadyExisting`

7.1.2.3 AlreadyFriends

`client.ACK.AlreadyFriends`

User(s) già amici.

7.1.2.4 AlreadyRegistered

`client.ACK.AlreadyRegistered`

User già registrato.

7.1.2.5 EmptyField

`client.ACK.EmptyField`

campi utente vuoti

7.1.2.6 FriendAdded

`client.ACK.FriendAdded`

User registrato come amico.

7.1.2.7 FriendNotFound

`client.ACK.FriendNotFound`

User amico non trovato.

7.1.2.8 FriendRemoved

`client.ACK.FriendRemoved`

User amico rimosso dalla lista degli amici.

7.1.2.9 INCHALLENGE

`client.ACK.INCHALLENGE`

stato dell'User in sfida

7.1.2.10 LoggedIn

`client.ACK.LoggedIn`

User loggato.

7.1.2.11 LoggedOut

`client.ACK.LoggedOut`

User uscito.

7.1.2.12 NotLogged

`client.ACK.NotLogged`

< User non loggato

7.1.2.13 OFFLINE

`client.ACK.OFFLINE`

stato dell'User offline

7.1.2.14 OK

`client.ACK.OK`

azione eseguita

7.1.2.15 ONLINE

`client.ACK.ONLINE`

stato dell'User online

7.1.2.16 OperationUnknown

`client.ACK.OperationUnknown`

operazione richiesta non riconosciuta

7.1.2.17 PasswordUnmatch

`client.ACK.PasswordUnmatch`

password non corrisponde a quella del User

7.1.2.18 RegistrationError

`client.ACK.RegistrationError`

problema generale durante la registrazione dell'User

7.1.2.19 Rejected

`client.ACK.Rejected`

sfida rifiutata

7.1.2.20 UserAdded

`client.ACK.UserAdded`

User inserito nel database.

7.1.2.21 UserAlreadyLoggedIn

`client.ACK.UserAlreadyLoggedIn`

User già entrato.

7.1.2.22 UserDeleted

`client.ACK.UserDeleted`

User deregistrato.

7.1.2.23 UserFound

`client.ACK.UserFound`

User trovato nel database.

7.1.2.24 UserNotDeleted

`client.ACK.UserNotDeleted`

User non deregistrato.

7.1.2.25 UserNotFound

`client.ACK.UserNotFound`

User non trovato nel database.

7.1.2.26 UserRegistered

`client.ACK.UserRegistered`

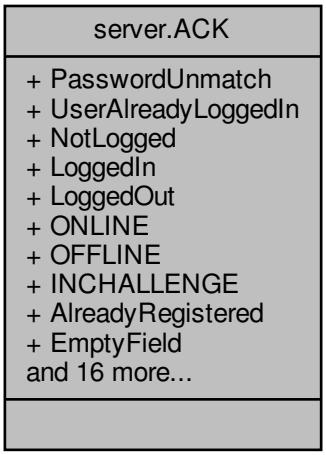
User registrato.

The documentation for this enum was generated from the following file:

- src/client/ACK.java

7.2 server.ACK Enum Reference

Collaboration diagram for server.ACK:



Public Attributes

- PasswordUnmatch

  password non corrisponde a quella del User
- UserAlreadyLoggedIn

  User già entrato.
- NotLogged

  < User non loggato
- LoggedIn

  User loggato.
- LoggedOut

  User uscito.
- ONLINE

  stato dell'User online
- OFFLINE

  stato dell'User offline
- INCHALLENGE

  stato dell'User in sfida
- AlreadyRegistered

  User già registrato.
- EmptyField

  campi utente vuoti
- UserRegistered

  User registrato.
- RegistrationError

  problema generale durante la registrazione dell'User

- UserFound

  User trovato nel database.
- UserNotFound

  User non trovato nel database.
- UserDeleted

  User deregistrato.
- UserNotDeleted

  User non deregistrato.
- AlreadyFriends

  User(s) già amici.
- FriendNotFound

  User amico non trovato.
- FriendAdded

  User registrato come amico.
- UserAdded

  User inserito nel database.
- AlreadyExisting
- FriendRemoved

  User amico rimosso dalla lista degli amici.
- Accepted

  sfida accettata
- Rejected

  sfida rifiutata
- OperationUnknown

  operazione richiesta non riconosciuta
- OK

  azione eseguita

7.2.1 Detailed Description

In questo file vengono implementati i messaggi da inviare al client a termine delle operazioni richieste

Author

Luca Canessa (Mat. 516639)

Version

1.6

Since

1.0

7.2.2 Member Data Documentation

7.2.2.1 Accepted

`server.ACK.Accepted`

sfida accettata

7.2.2.2 AlreadyExisting

`server.ACK.AlreadyExisting`

7.2.2.3 AlreadyFriends

`server.ACK.AlreadyFriends`

User(s) già amici.

7.2.2.4 AlreadyRegistered

`server.ACK.AlreadyRegistered`

User già registrato.

7.2.2.5 EmptyField

`server.ACK.EmptyField`

campi utente vuoti

7.2.2.6 FriendAdded

`server.ACK.FriendAdded`

User registrato come amico.

7.2.2.7 FriendNotFound

`server.ACK.FriendNotFound`

User amico non trovato.

7.2.2.8 FriendRemoved

`server.ACK.FriendRemoved`

User amico rimosso dalla lista degli amici.

7.2.2.9 INCHALLENGE

`server.ACK.INCHALLENGE`

stato dell'User in sfida

7.2.2.10 LoggedIn

`server.ACK.LoggedIn`

User loggato.

7.2.2.11 LoggedOut

`server.ACK.LoggedOut`

User uscito.

7.2.2.12 NotLogged

`server.ACK.NotLogged`

< User non loggato

7.2.2.13 OFFLINE

`server.ACK.OFFLINE`

stato dell'User offline

7.2.2.14 OK

`server.ACK.OK`

azione eseguita

7.2.2.15 ONLINE

`server.ACK.ONLINE`

stato dell'User online

7.2.2.16 OperationUnknown

`server.ACK.OperationUnknown`

operazione richiesta non riconosciuta

7.2.2.17 PasswordUnmatch

`server.ACK.PasswordUnmatch`

password non corrisponde a quella del User

7.2.2.18 RegistrationError

`server.ACK.RegistrationError`

problema generale durante la registrazione dell'User

7.2.2.19 Rejected

`server.ACK.Rejected`

sfida rifiutata

7.2.2.20 UserAdded

`server.ACK.UserAdded`

User inserito nel database.

7.2.2.21 UserAlreadyLoggedIn

`server.ACK.UserAlreadyLoggedIn`

User già entrato.

7.2.2.22 UserDeleted

`server.ACK.UserDeleted`

User deregistrato.

7.2.2.23 UserFound

`server.ACK.UserFound`

User trovato nel database.

7.2.2.24 UserNotDeleted

`server.ACK.UserNotDeleted`

User non deregistrato.

7.2.2.25 UserNotFound

`server.ACK.UserNotFound`

User non trovato nel database.

7.2.2.26 UserRegistered

`server.ACK.UserRegistered`

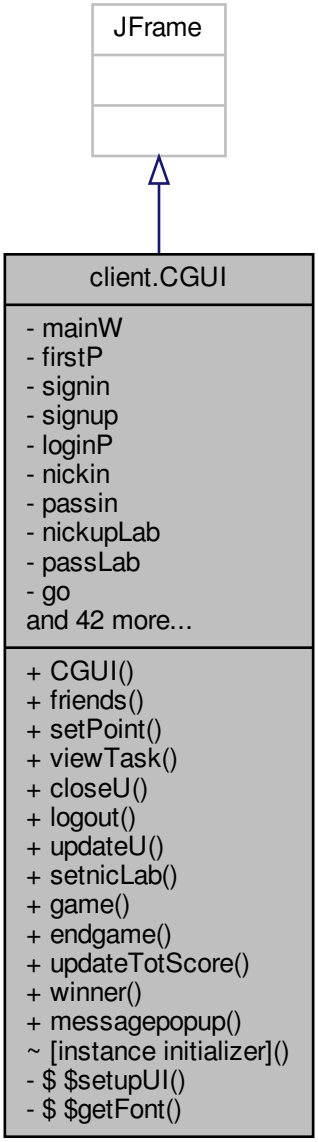
User registrato.

The documentation for this enum was generated from the following file:

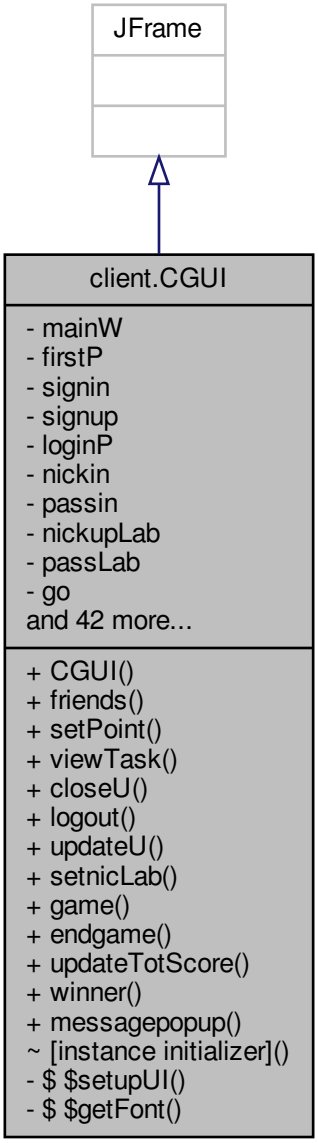
- src/server/ACK.java

7.3 client.CGUI Class Reference

Inheritance diagram for client.CGUI:



Collaboration diagram for client.CGUI:



Public Member Functions

- CGUI (String title) throws HeadlessException
- void friends (String[ ] friend, String numF)
- void setPoint (String point)
- void viewTask ()
- void closeU ()
- void logout ()
- void updateU ()
- void setnicLab (String Nickname)

- void game ()
- void endgame ()
- void updateTotScore ()

Static Public Member Functions

- static void winner ()
- static void messagepopup (String Message)

Package Functions

- [instance initializer]

Private Member Functions

- void $ $setupUI ()
- Font $ $getFont (String fontName, int style, int size, Font currentFont)

Private Attributes

- JFrame mainW
- JPanel firstP
- JButton signin
- JButton signup
- JPanel loginP
- JTextField nickin
- JPasswordField passin
- JLabel nickupLab
- JLabel passLab
- JButton go
- JButton back1
- JLabel nickLab
- JPanel signupP
- JButton back2
- JPasswordField passup
- JButton goup
- JPanel SignupP
- JTextField nickup
- JLabel passupLab
- JTextField surnameup
- JLabel nameupLab
- JLabel surnameupLab
- JTextField nameup
- JPanel tasks
- JSplitPane split
- JButton addfrd
- JLabel friend
- JList listf
- JLabel num
- JButton rmfrd
- JButton upfrd

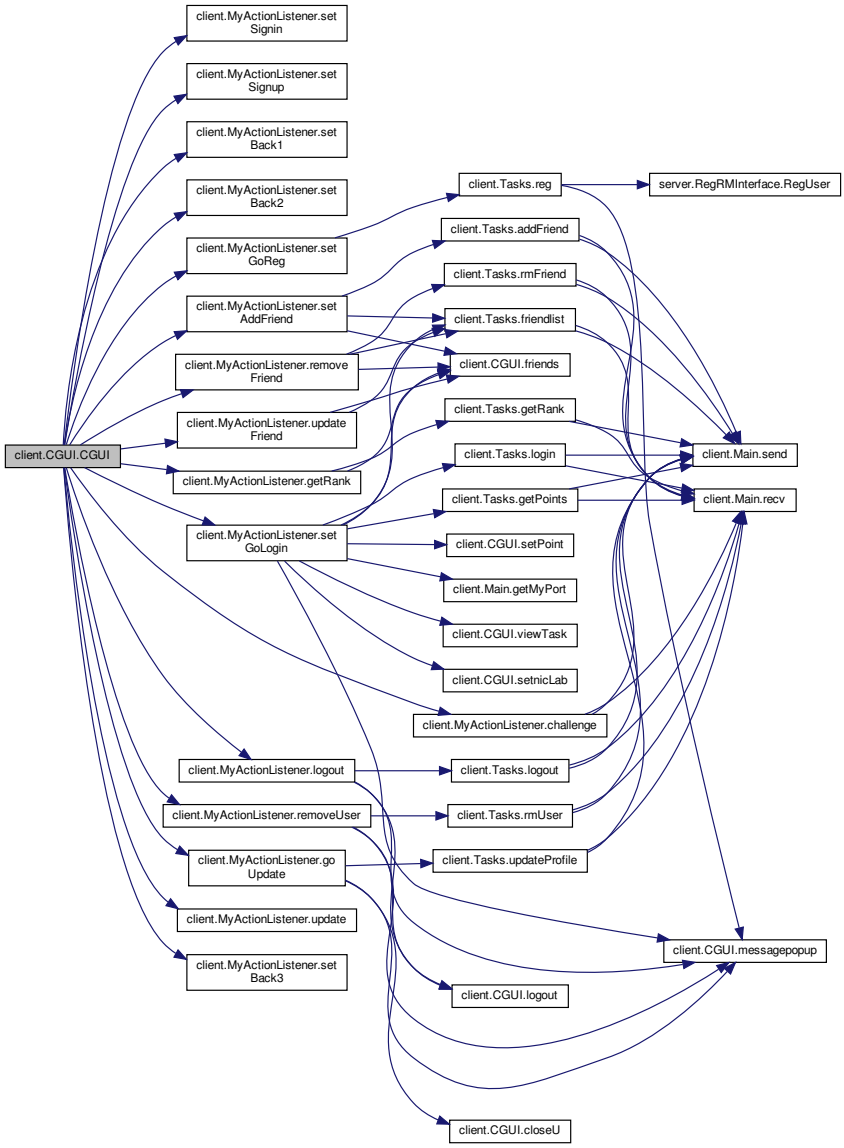
- JButton chal
- JButton upprof
- JLabel totpoint
- JButton getrk
- JButton rmusr
- JButton logoutButton
- JPanel updateU
- JButton back3
- JTextField nameU
- JTextField snameU
- JPasswordField passU
- JButton go3
- JLabel nameLU
- JLabel snameUL
- JLabel passLU
- JLabel helloNick
- JPanel gameP
- JTextField word
- JButton sendW
- JLabel getW
- JLabel counter

7.3.1 Constructor & Destructor Documentation

7.3.1.1 CGUI()

```
client.CGUI.CGUI (
            String title ) throws HeadlessException
```

Here is the call graph for this function:



7.3.2 Member Function Documentation

7.3.2.1 $ $getFont()

```
Font client.CGUI.$ $getFont (
            String fontName,
            int style,
            int size,
            Font currentFont )  [private]
```

ALL

7.3.2.2 $ $setupUI()

```
void client.CGUI.$ $setupUI ( )  [private]
```

Method generated by IntelliJ IDEA GUI Designer

IMPORTANT!! <<<

DO NOT edit this method OR call it in your code!

ALL

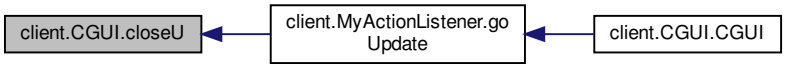
7.3.2.3 [instance initializer]()

```
client.CGUI.[instance initializer] ( )  [package]
```

7.3.2.4 closeU()

```
void client.CGUI.closeU ( )
```

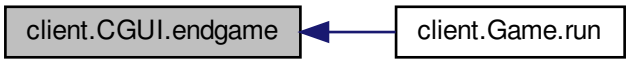
Here is the caller graph for this function:



7.3.2.5 endgame()

```
void client.CGUI.endgame ( )
```

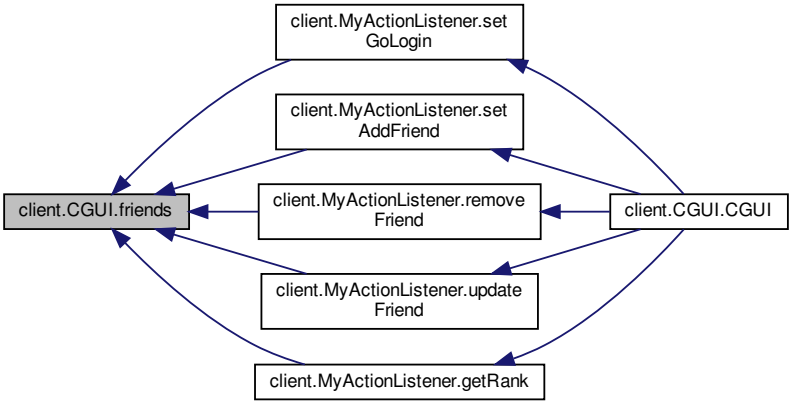
Here is the caller graph for this function:



7.3.2.6 friends()

```
void client.CGUI.friends (
            String [] friend,
            String numF )
```

Here is the caller graph for this function:



7.3.2.7 game()

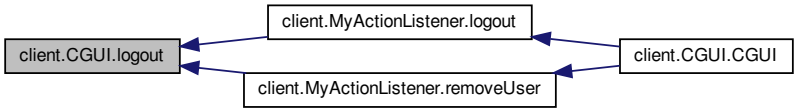
```
void client.CGUI.game ( )
```

Here is the caller graph for this function:

7.3.2.8 logout()

```
void client.CGUI.logout ( )
```

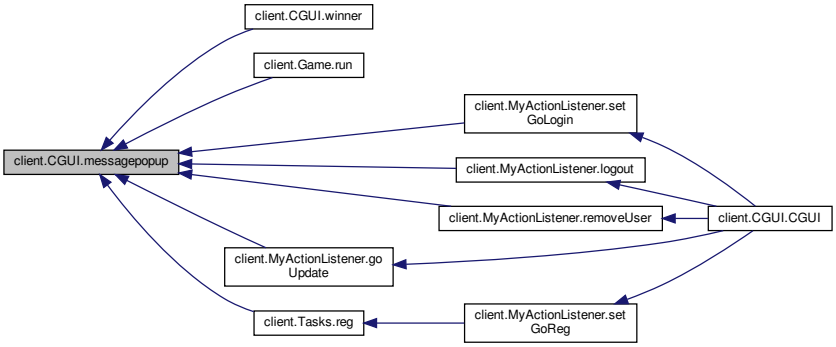
Here is the caller graph for this function:



7.3.2.9 messagepopup()

```
static void client.CGUI.messagepopup (
            String Message )  [static]
```

Here is the caller graph for this function:



7.3.2.10 setnicLab()

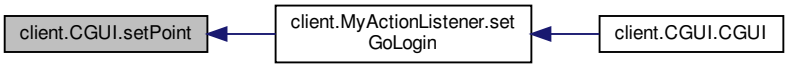
```
void client.CGUI.setnicLab (
            String Nickname )
```

Here is the caller graph for this function:

7.3.2.11 setPoint()

```
void client.CGUI.setPoint (
            String point )
```

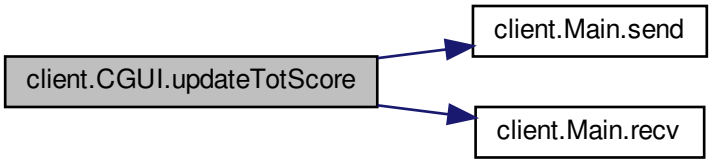
Here is the caller graph for this function:



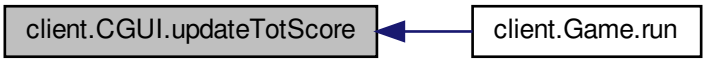
7.3.2.12 updateTotScore()

```
void client.CGUI.updateTotScore ( )
```

Here is the call graph for this function:



Here is the caller graph for this function:



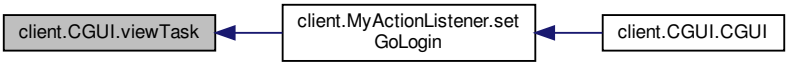
7.3.2.13 updateU()

```
void client.CGUI.updateU ( )
```

7.3.2.14 viewTask()

```
void client.CGUI.viewTask ( )
```

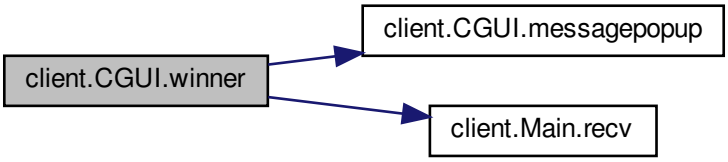
Here is the caller graph for this function:



7.3.2.15 winner()

```
static void client.CGUI.winner ( )  [static]
```

Here is the call graph for this function:



7.3.3 Member Data Documentation

7.3.3.1 addfrd

```
JButton client.CGUI.addfrd  [private]
```

7.3.3.2 back1

```
JButton client.CGUI.back1  [private]
```

7.3.3.3 back2

```
JButton client.CGUI.back2  [private]
```

7.3.3.4 back3

```
JButton client.CGUI.back3  [private]
```

7.3.3.5 chal

```
JButton client.CGUI.chal  [private]
```

7.3.3.6 counter

```
JLabel client.CGUI.counter  [private]
```

7.3.3.7 firstP

```
JPanel client.CGUI.firstP  [private]
```

7.3.3.8 friend

```
JLabel client.CGUI.friend  [private]
```

7.3.3.9 gameP

```
JPanel client.CGUI.gameP  [private]
```

7.3.3.10 getrk

```
JButton client.CGUI.getrk  [private]
```

7.3.3.11 getW

```
JLabel client.CGUI.getW  [private]
```

7.3.3.12 go

```
JButton client.CGUI.go  [private]
```

7.3.3.13 go3

```
JButton client.CGUI.go3  [private]
```

7.3.3.14 goup

```
JButton client.CGUI.goup  [private]
```

7.3.3.15 helloNick

```
JLabel client.CGUI.helloNick  [private]
```

7.3.3.16 listf

```
JList client.CGUI.listf  [private]
```

7.3.3.17 loginP

```
JPanel client.CGUI.loginP  [private]
```

7.3.3.18 logoutButton

```
JButton client.CGUI.logoutButton  [private]
```

7.3.3.19 mainW

```
JFrame client.CGUI.mainW  [private]
```

7.3.3.20 nameLU

```
JLabel client.CGUI.nameLU  [private]
```

7.3.3.21 nameU

```
JTextField client.CGUI.nameU  [private]
```

7.3.3.22 nameup

```
JTextField client.CGUI.nameup  [private]
```

7.3.3.23 nameupLab

```
JLabel client.CGUI.nameupLab  [private]
```

7.3.3.24 nickin

```
JTextField client.CGUI.nickin  [private]
```

7.3.3.25 nickLab

```
JLabel client.CGUI.nickLab  [private]
```

7.3.3.26 nickup

```
JTextField client.CGUI.nickup  [private]
```

7.3.3.27 nickupLab

```
JLabel client.CGUI.nickupLab  [private]
```

7.3.3.28 num

```
JLabel client.CGUI.num  [private]
```

7.3.3.29 passin

```
JPasswordField client.CGUI.passin  [private]
```

7.3.3.30 passLab

```
JLabel client.CGUI.passLab  [private]
```

7.3.3.31 passLU

```
JLabel client.CGUI.passLU  [private]
```

7.3.3.32 passU

```
JPasswordField client.CGUI.passU  [private]
```

7.3.3.33 passup

```
JPasswordField client.CGUI.passup  [private]
```

7.3.3.34 passupLab

```
JLabel client.CGUI.passupLab  [private]
```

7.3.3.35 rmfrd

```
JButton client.CGUI.rmfrd  [private]
```

7.3.3.36 rmusr

```
JButton client.CGUI.rmusr  [private]
```

7.3.3.37 sendW

```
JButton client.CGUI.sendW  [private]
```

7.3.3.38 signin

```
JButton client.CGUI.signin  [private]
```

7.3.3.39 signup

```
JButton client.CGUI.signup  [private]
```

7.3.3.40 signupP

```
JPanel client.CGUI.signupP  [private]
```

7.3.3.41 SignupP

```
JPanel client.CGUI.SignupP  [private]
```

7.3.3.42 snameU

```
JTextField client.CGUI.snameU  [private]
```

7.3.3.43 snameUL

```
JLabel client.CGUI.snameUL  [private]
```

7.3.3.44 split

```
JSplitPane client.CGUI.split  [private]
```

7.3.3.45 surnameup

```
JTextField client.CGUI.surnameup  [private]
```

7.3.3.46 surnameupLab

```
JLabel client.CGUI.surnameupLab  [private]
```

7.3.3.47 tasks

```
JPanel client.CGUI.tasks  [private]
```

7.3.3.48 totpoint

```
JLabel client.CGUI.totpoint  [private]
```

7.3.3.49 updateU

```
JPanel client.CGUI.updateU  [private]
```

7.3.3.50 upfrd

```
JButton client.CGUI.upfrd  [private]
```

7.3.3.51 upprof

```
JButton client.CGUI.upprof  [private]
```

7.3.3.52 word

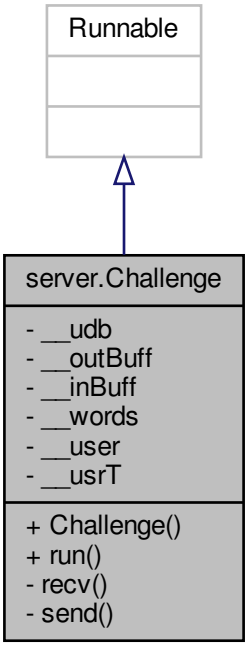
```
JTextField client.CGUI.word  [private]
```

The documentation for this class was generated from the following file:

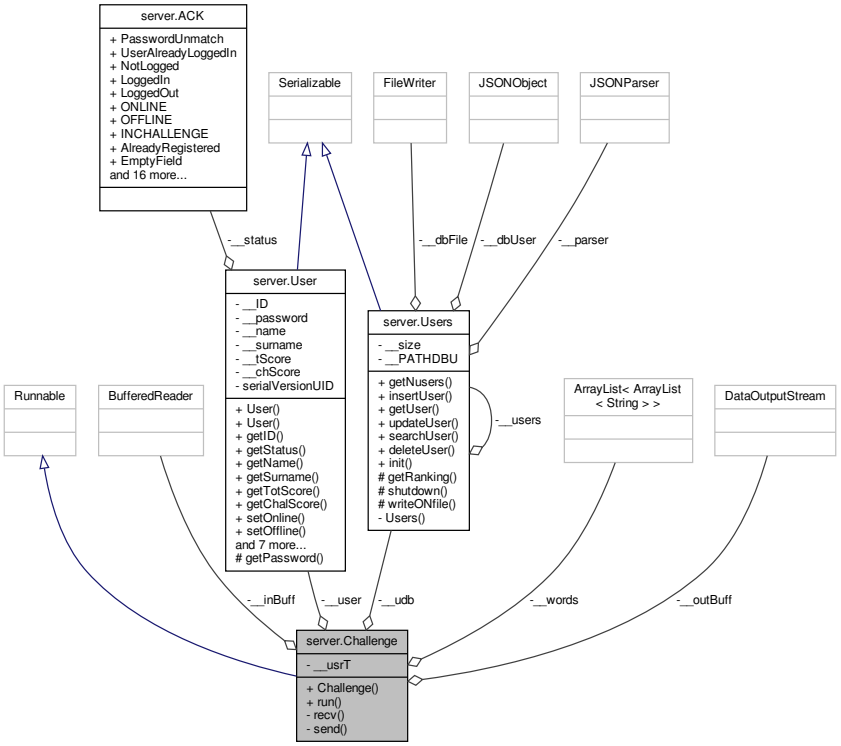
- src/client/CGUI.java

7.4 server.Challenge Class Reference

Inheritance diagram for server.Challenge:



Collaboration diagram for server.Challenge:



Public Member Functions

- Challenge (BufferedReader inputb, DataOutputStream outputb, ArrayList< ArrayList< String >> Words, User usr, Users udb, Thread T)
- void run ()

Private Member Functions

- String recv ()
- void send (String str)

Private Attributes

- Users __udb = null

  database degli utenti
- DataOutputStream __outBuff = null

  stampa sulla socket
- BufferedReader __inBuff = null

  legge sulla socket
- ArrayList< ArrayList< String > > __words = null

  matrice con parole italiane e parole inglesi
- User __user = null

  User che sta giocando.
- Thread __usrT = null

  thread principale del client

7.4.1 Detailed Description

Classe che modella l'oggetto Challenge di un User per gestire la sua sfida implementando metodi e variabili utili per controllare e gestire la sfida tra i client inviato e ricevendo le parole rispettivamente in italiano e in inglese

Author

Luca Canessa (Mat. 516639)

Version

1.1

Since

1.0

7.4.2 Constructor & Destructor Documentation

7.4.2.1 Challenge()

```
server.Challenge.Challenge (
            BufferedReader inputb,
            DataOutputStream outputb,
            ArrayList< ArrayList< String >> Words,
            User usr,
            Users udb,
            Thread T )
```

Costruttore della sfida tra due utenti. Gestisce la sfida per ogni utente se viene lanciato da thread, inviando le parole italiane e ricevendo quelle tradotte dal client. Al termine della sfida aggiorna i parametri dell'utente con i punti guadagnati e invia al proprio client il vincitore della sfida

Parameters

inputb	buffer su cui può ricevere i dati dal client
outputb	buffer su cui può inviare i dati al client
Words	matrice con le parole e le rispettive traduzioni
usr	User che sta giocando attualmente
udb	Database
T	Thread principale del client

7.4.3 Member Function Documentation

7.4.3.1 recv()

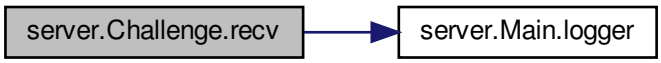
```
String server.Challenge.recv ( )  [private]
```

Rimane in attesa sulla socket per ricevere i dati dal client

Returns

il buffer (String) contenente i dati

Here is the call graph for this function:



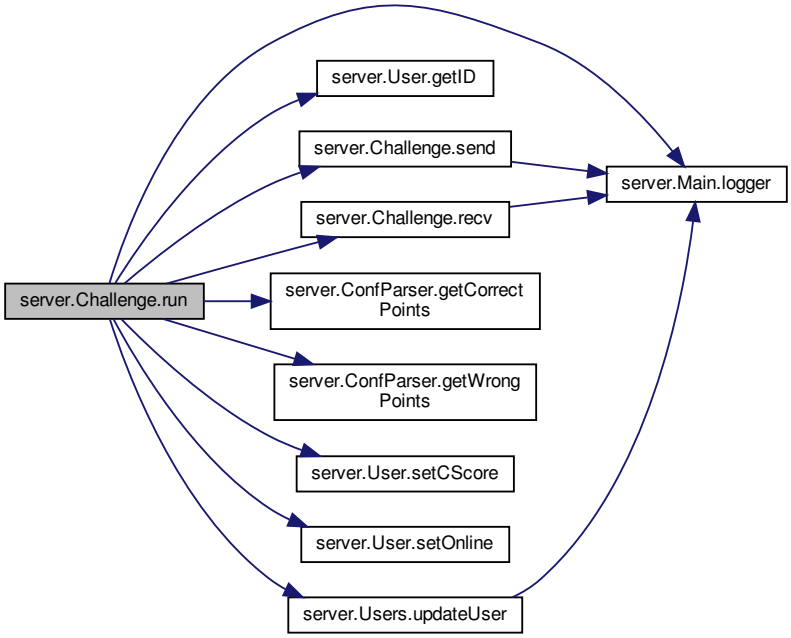
Here is the caller graph for this function:

7.4.3.2 run()

```
void server.Challenge.run ( )
```

Metodo che mette in esecuzione il thread che gesitsce il gioco di un client. Invia una parola italiana alla volta e ne riceve la corrispettiva traduzione del User dal client, ne controlla la correttezza e ne aggiorna i punti, sommando un valore se corretto o levando dal totale della partita un altro valore se traduzione errata, nessuna valutazione in caso di risposta assente. Tali valori di partita sono presi dal file di configurazione. Quando termina aggiorna i punti della

partita del User e aggiorna lo stato del User in ONLINE, terminando poi alzando un interrupt per il thread principale avvisandolo della terminazione del gioco. Here is the call graph for this function:



7.4.3.3 send()

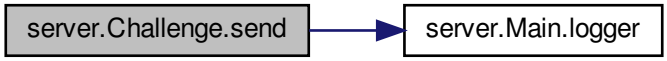
```
void server.Challenge.send (
            String str )  [private]
```

Invia al client attraverso la socket i dati

Parameters

str	stringa contenente i dati da mandare al client

Here is the call graph for this function:



Here is the caller graph for this function:

7.4.4 Member Data Documentation

7.4.4.1 __inBuff

```
BufferedReader server.Challenge.__inBuff = null  [private]
```

legge sulla socket

7.4.4.2 __outBuff

```
DataOutputStream server.Challenge.__outBuff = null  [private]
```

stampa sulla socket

7.4.4.3 __udb

```
Users server.Challenge.__udb = null  [private]
```

database degli utenti

7.4.4.4 __user

```
User server.Challenge.__user = null  [private]
```

User che sta giocando.

7.4.4.5 __usrT

```
Thread server.Challenge.__usrT = null  [private]
```

thread principale del client

7.4.4.6 __words

```
ArrayList<ArrayList<String> > server.Challenge.__words = null  [private]
```

matrice con parole italiane e parole inglesi

The documentation for this class was generated from the following file:

- src/server/Challenge.java

7.5 client.ClientMSG Enum Reference

Collaboration diagram for client.ClientMSG:

Public Attributes

- LOGIN
  richiesta di login
- LOGOUT
  richiesta di logout
- ADDFRIEND
  richiesta di aggiunta di amico alla lista
- GETFRIENDS
  richiesta lista amicizie
- GETNFRIENDS
  richiesto numero di amici
- STARTCH
  richiesta di inizio di una sfida
- ACCEPTEDCH
  accettazione della sfida arrivata

- UPDATEINFO
  richiesta di aggiornamento dei dati
- GETPOINTS
  richiesta punti totalizzati
- GETRANK
  richiesta della classifica
- RMUSER
  richiesta rimozione dell'User

7.5.1 Detailed Description

In questa enumerazione vengono implementati i messaggi che il server riceve dal client

Author

Luca Canessa (Mat. 516639)

Version

1.1

Since

1.0

7.5.2 Member Data Documentation

7.5.2.1 ACCEPTEDCH

`client.ClientMSG.ACCEPTEDCH`

accettazione della sfida arrivata

7.5.2.2 ADDFRIEND

`client.ClientMSG.ADDFRIEND`

richiesta di aggiunta di amico alla lista

7.5.2.3 GETFRIENDS

`client.ClientMSG.GETFRIENDS`

richiesta lista amicizie

7.5.2.4 GETNFRIENDS

`client.ClientMSG.GETNFRIENDS`

richiesto numero di amici

7.5.2.5 GETPOINTS

`client.ClientMSG.GETPOINTS`

richiesta punti totalizzati

7.5.2.6 GETRANK

`client.ClientMSG.GETRANK`

richiesta della classifica

7.5.2.7 LOGIN

`client.ClientMSG.LOGIN`

richiesta di login

7.5.2.8 LOGOUT

`client.ClientMSG.LOGOUT`

richiesta di logout

7.5.2.9 RMUSER

`client.ClientMSG.RMUSER`

richiesta rimozione dell'User

7.5.2.10 STARTCH

`client.ClientMSG.STARTCH`

richiesta di inizio di una sfida

7.5.2.11 UPDATEINFO

`client.ClientMSG.UPDATEINFO`

richiesta di aggiornamento dei dati

The documentation for this enum was generated from the following file:

- src/client/Main.java

7.6 server.ClientMSG Enum Reference

Collaboration diagram for server.ClientMSG:

Public Attributes

- LOGIN
  richiesta di login
- LOGOUT
  richiesta di logout
- ADDFRIEND
  richiesta di aggiunta di amico alla lista
- GETFRIENDS
  richiesta lista amicizie
- GETNFRIENDS
  richiesto numero di amici
- STARTCH
  richiesta di inizio di una sfida
- ACCEPTEDCH
  accettazione della sfida arrivata

- UPDATEINFO
  richiesta di aggiornamento dei dati
- GETPOINTS
  richiesta punti totalizzati
- GETRANK
  richiesta della classifica
- RMUSER
  richiesta rimozione dell'User

7.6.1 Detailed Description

In questa enumerazione vengono implementati i messaggi che il server riceve dal client

Author

Luca Canessa (Mat. 516639)

Version

1.1

Since

1.0

7.6.2 Member Data Documentation

7.6.2.1 ACCEPTEDCH

`server.ClientMSG.ACCEPTEDCH`

accettazione della sfida arrivata

7.6.2.2 ADDFRIEND

`server.ClientMSG.ADDFRIEND`

richiesta di aggiunta di amico alla lista

7.6.2.3 GETFRIENDS

`server.ClientMSG.GETFRIENDS`

richiesta lista amicizie

7.6.2.4 GETNFRIENDS

`server.ClientMSG.GETNFRIENDS`

richiesto numero di amici

7.6.2.5 GETPOINTS

`server.ClientMSG.GETPOINTS`

richiesta punti totalizzati

7.6.2.6 GETRANK

`server.ClientMSG.GETRANK`

richiesta della classifica

7.6.2.7 LOGIN

`server.ClientMSG.LOGIN`

richiesta di login

7.6.2.8 LOGOUT

`server.ClientMSG.LOGOUT`

richiesta di logout

7.6.2.9 RMUSER

`server.ClientMSG.RMUSER`

richiesta rimozione dell'User

7.6.2.10 STARTCH

`server.ClientMSG.STARTCH`

richiesta di inizio di una sfida

7.6.2.11 UPDATEINFO

`server.ClientMSG.UPDATEINFO`

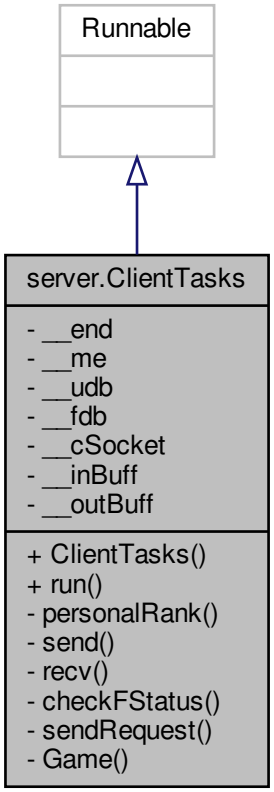
richiesta di aggiornamento dei dati

The documentation for this enum was generated from the following file:

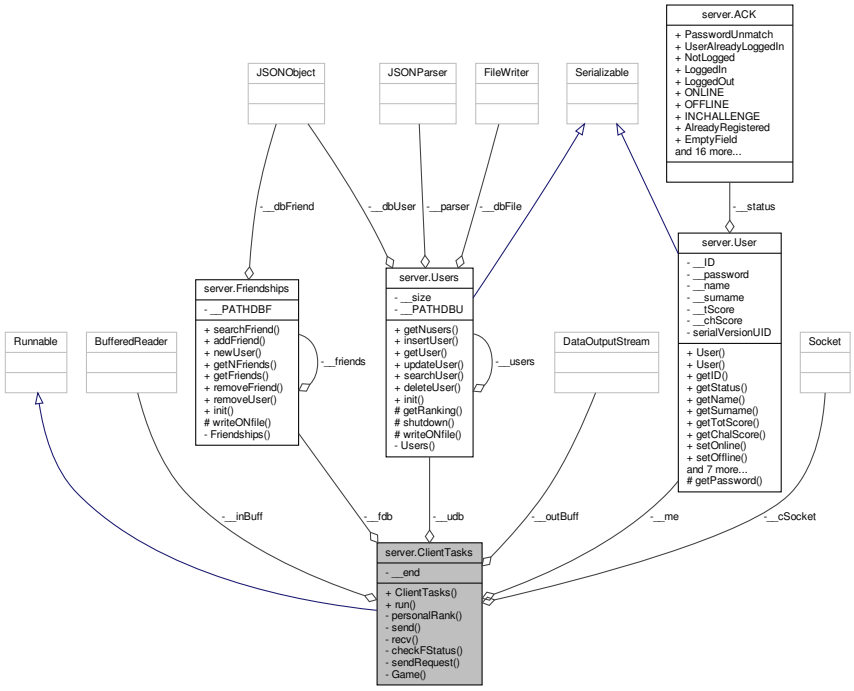
- src/server/ClientTasks.java

7.7 server.ClientTasks Class Reference

Inheritance diagram for server.ClientTasks:



Collaboration diagram for server.ClientTasks:



Public Member Functions

- ClientTasks (Users udb, Friendships fdb, Socket socket)
- void run ()

Private Member Functions

- JSONObject personalRank ()
- void send (String msg)
- String recv ()
- ACK checkFStatus (String Friend)
- ACK sendRequest (String friendName)
- void Game (String nickname, String rival)

Private Attributes

- boolean __end = false

  variabile usata per determinare la terminazione del ciclo principale 'while' nel metodo 'run'
- User __me = null

  User che si collega.
- Users __udb = null

  databse degli User
- Friendships __fdb = null

  database delle amicizie
- Socket __cSocket = null

socket di comunicazione con il client

- BufferedReader __inBuff = null

  buffer su cui scrive il client per comunicare

- DataOutputStream __outBuff = null

  buffer su cui scrive il Thread (l'oggetto ClientTask) in esecuzione per comunucare con il client

7.7.1 Detailed Description

In questa classe vengono implementati i metodi e le variabili necessari per controllare le operazioni richieste dal client, identificate dall'enumerazione ClientMSG. Per ogni operazione richiesta viene eseguito un task e inviato il risultato della terminazione del task al client

Author

Luca Canessa (Mat. 516639)

Version

1.6

Since

1.0

7.7.2 Constructor & Destructor Documentation

7.7.2.1 ClientTasks()

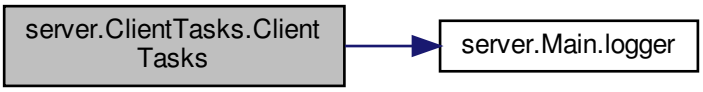
```
server.ClientTasks.ClientTasks (
            Users udb,
            Friendships fdb,
            Socket socket )
```

Costruttore del gestore dei task dell'User, si occupa di inizializzare le variabili

Parameters

udb	database degli utenti
fdb	database delle amicizie
socket	socket su cui comunicare con il client

Here is the call graph for this function:



7.7.3 Member Function Documentation

7.7.3.1 checkFStatus()

```
ACK server.ClientTasks.checkFStatus (
            String Friend )  [private]
```

Controlla lo stato di attività dell'amico ritornando un messaggio di ACK

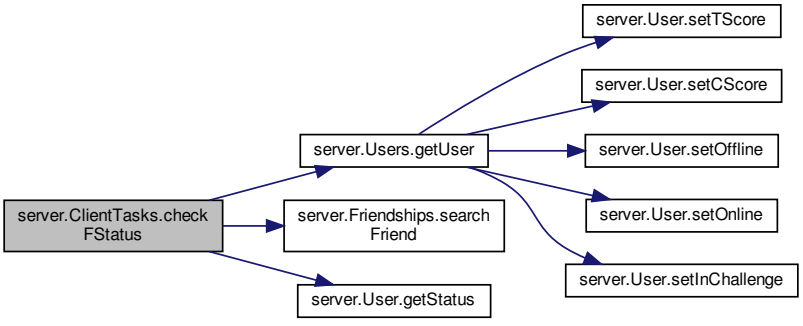
Parameters

Friend	stringa del nickname dell'amico da controllare

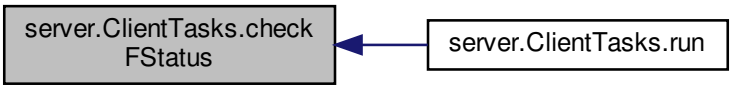
Returns

ONLINE se l'User ha effettuato il login al gioco OFFLINE se l'User non è presente al gioco INCHALLENGE se l'User sta già effettuando una sfida FriendNotFound se l'User non è tra le amicizie

Here is the call graph for this function:



Here is the caller graph for this function:



7.7.3.2 Game()

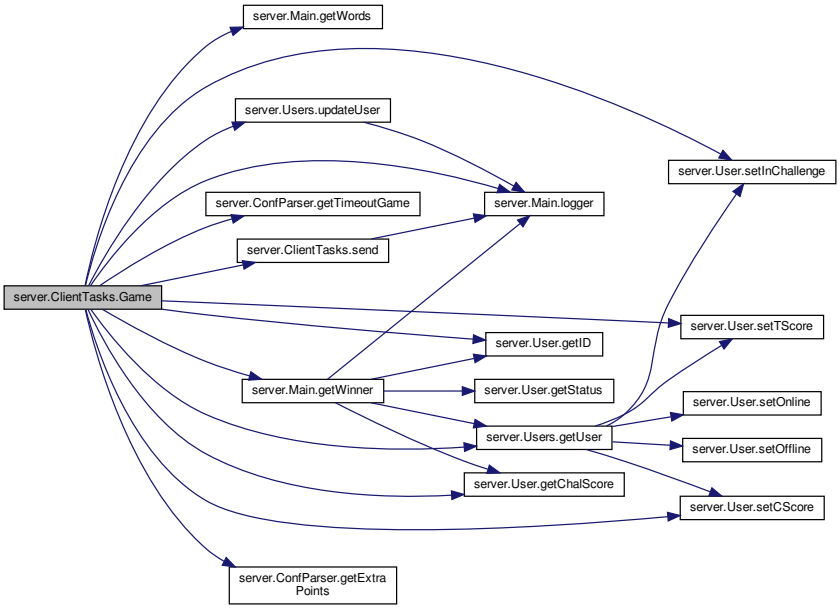
```
void server.ClientTasks.Game (
            String nickname,
            String rival )  [private]
```

Metodo che gestisce l'inizio e la fine di ogni sfida aggiornando lo stato dell'User in INCHALLENGE e preparando lista di parole da inviare prima dell'avvio della sfida e dopo aver avviato la sfida attende che questa sia finita per aggiornare i punti guadagnati all'User e decretare il vincitore della sfida. Terminata la sfida elimina i file su disco che sono serviti per la condivisione delle parole.

Parameters

nickname	Stringa da usare per decodificare la sequenza di parole da usare
rival	Stringa per determinare il nome dell'avversario

Here is the call graph for this function:



Here is the caller graph for this function:

7.7.3.3 personalRank()

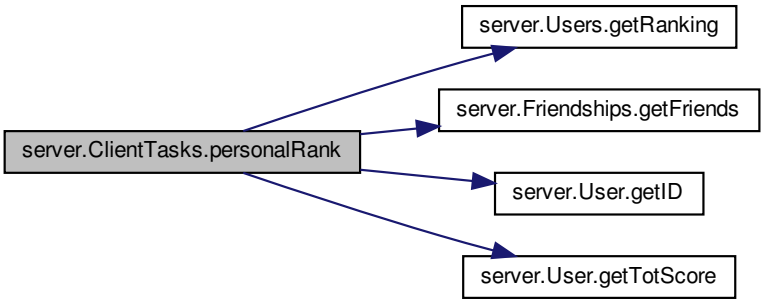
```
JSONObject server.ClientTasks.personalRank ( )  [private]
```

Genera la classifica personale del User estrapolando gli amici e i loro punti dalla classifica generale del gioco e restituendola ordinata per il valore dei punti.

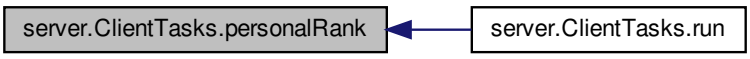
Returns

il JSON con la classifica rispetto all'User

Here is the call graph for this function:



Here is the caller graph for this function:



7.7.3.4 recv()

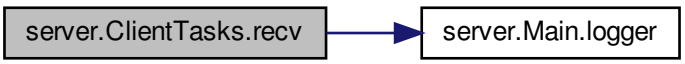
```
String server.ClientTasks.recv ( )  [private]
```

Aspetta di ricevere dati dal client sul buffer di ingresso come tipo String (more BufferedReader.readLine())

Returns

stringa di dati ricevuti dal Client

Here is the call graph for this function:



Here is the caller graph for this function:

7.7.3.5 run()

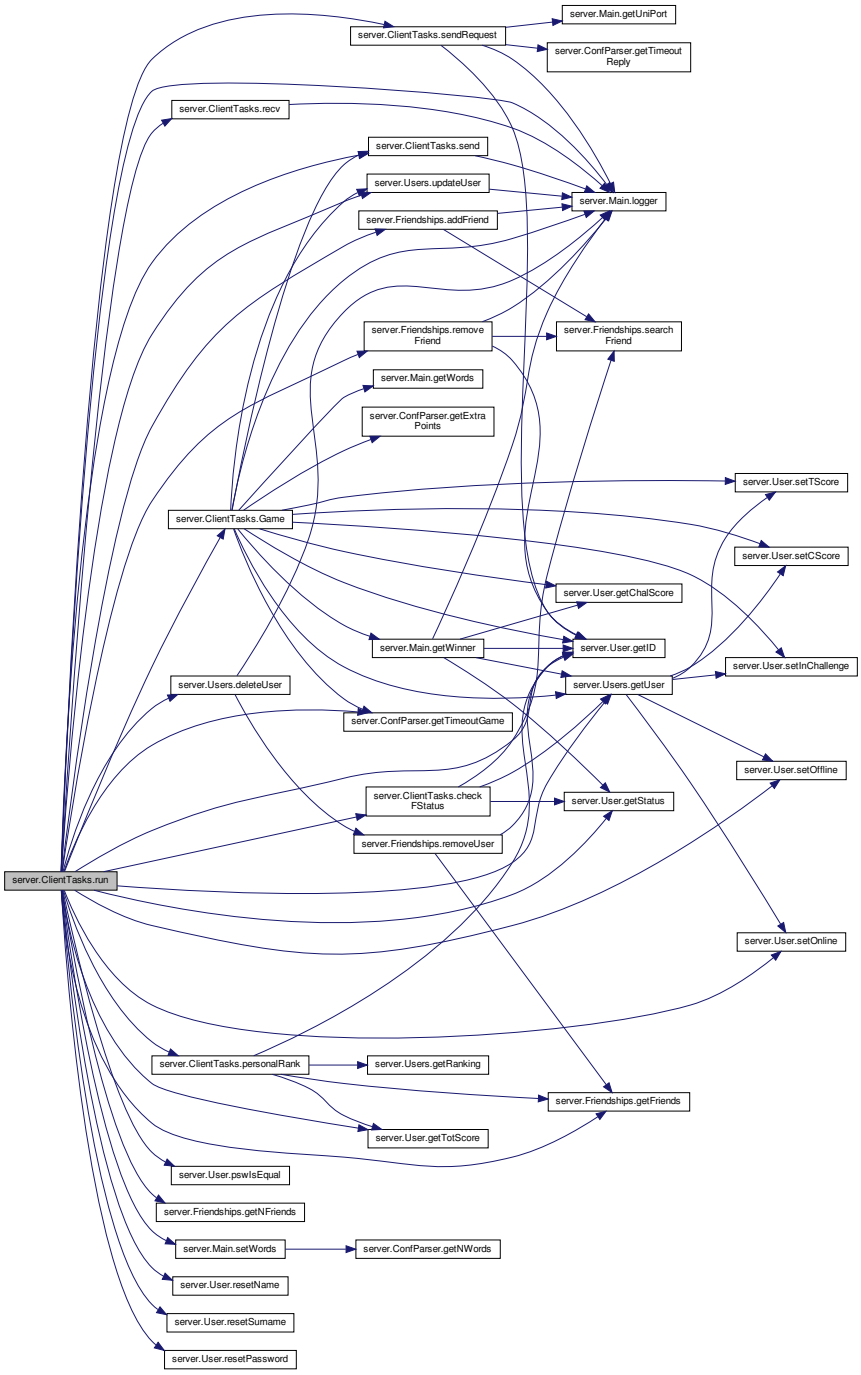
```
void server.ClientTasks.run ( )
```

Metodo, messo in esecuzione da un thread, utile alla gestione del client collegatosi. Permette di ricevere come primo pacchetto l'operazione da eseguire, che è determinata attraverso la classe Enum ClientMSG. Per ogni operazione richiesta esegue compiti differenti, mandando sempre al client come risposta l'esito dell'operazione. Le operazioni consentite dal client User sono possibili se e solo se è stata effettuata come prima operazione quella di LOGIN, in caso contrario non è possibile procedere. Le operazioni consentite al User sono:

- LOGIN: si salva aggiornano le variabili dell'User in base a chi ha chiesto l'operazione
- LOGOUT: si resettano le variabili dell'User
- ADDFRIEND: si aggiunge l'User richiesto nella lista delle amicizie se e solo se questo esiste
- GETFRIENDS: si restituisce la lista delle amicizie dell'User
- GETNFRIENDS: si restituisce il numero di amici dell'User

- STARTCH: si invia una richiesta di sfida ad un altro User se possibile. Se questo accetta si avvia
- ACCEPTEDCH: si avvia la sfida che è stata appena accettata
- UPDATEINFO: si aggiornano le informazioni dell'User sul DB
- GETPOINTS: si invia al client il numero di punti accumulati durante tutto il gioco
- GETRANK: si invia al client la ranking personale del gioco in cui vi sono solo gli amici del User e l'User stesso

Here is the call graph for this function:



7.7.3.6 send()

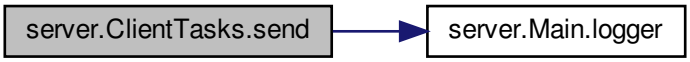
```
void server.ClientTasks.send (
            String msg )  [private]
```

Scrive un messaggio sul buffer in uscita da mandare al client come sequenza di Bytes (more DataOutputStream.↩ writeBytes())

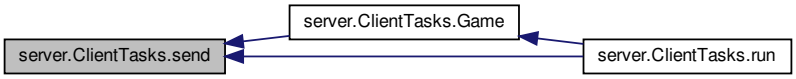
Parameters

msg	messaggio da inviare al client

Here is the call graph for this function:



Here is the caller graph for this function:



7.7.3.7 sendRequest()

```
ACK server.ClientTasks.sendRequest (
            String friendName )  [private]
```

Invia una richiesta di sfida di gioco al friend con nickname 'friendName' utilizzando il protocollo UDP e indirizzando il pacchetto verso la porta di destinazione identificata dal codice hash del nickname dell'amico. (Nickname univoco quindi bassa possibilità di collisioni). Attende la risposta del friend fino allo scadere del Timeout, poi considera la sfida rifiutata. Tempo di Timeout preso dal file di configurazione. Restituisce un valore di ACK

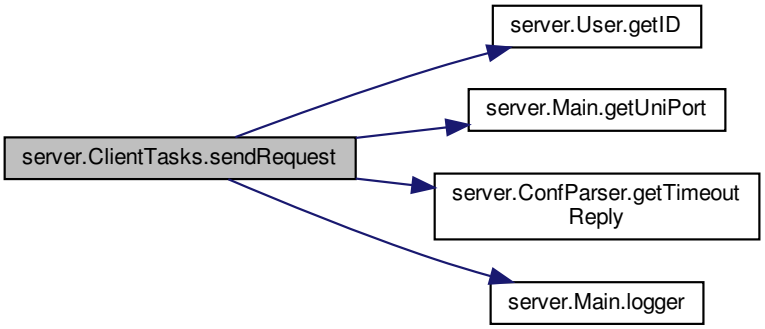
Parameters

friendName	nome dell'amico che si vuole sfidare

Returns

ACK.Accepted se l'amico accetta la sfida ACK.Rejected se l'amico rifiuta la sfida o se scade il timeout ACK.ERROR se si presentano errori sulla socket UDP

Here is the call graph for this function:



Here is the caller graph for this function:

7.7.4 Member Data Documentation

7.7.4.1 __cSocket

```
Socket server.ClientTasks.__cSocket = null  [private]
```

socket di comunicazione con il client

7.7.4.2 __end

```
boolean server.ClientTasks.__end = false  [private]
```

variabile usata per determinare la terminazione del ciclo principale 'while' nel metodo 'run'

7.7.4.3 __fdb

```
Friendships server.ClientTasks.__fdb = null  [private]
```

database delle amicizie

7.7.4.4 __inBuff

```
BufferedReader server.ClientTasks.__inBuff = null  [private]
```

buffer su cui scrive il client per comunicare

7.7.4.5 __me

```
User server.ClientTasks.__me = null  [private]
```

User che si collega.

7.7.4.6 __outBuff

```
DataOutputStream server.ClientTasks.__outBuff = null  [private]
```

buffer su cui scrive il Thread (l'oggetto ClientTask) in esecuzione per comunucare con il client

7.7.4.7 __udb

```
Users server.ClientTasks.__udb = null  [private]
```

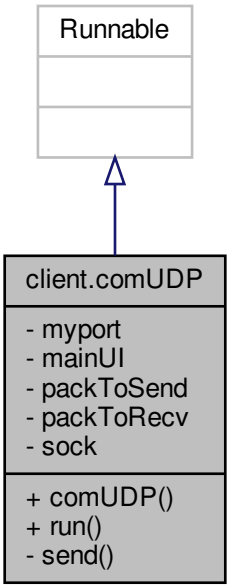
databse degli User

The documentation for this class was generated from the following file:

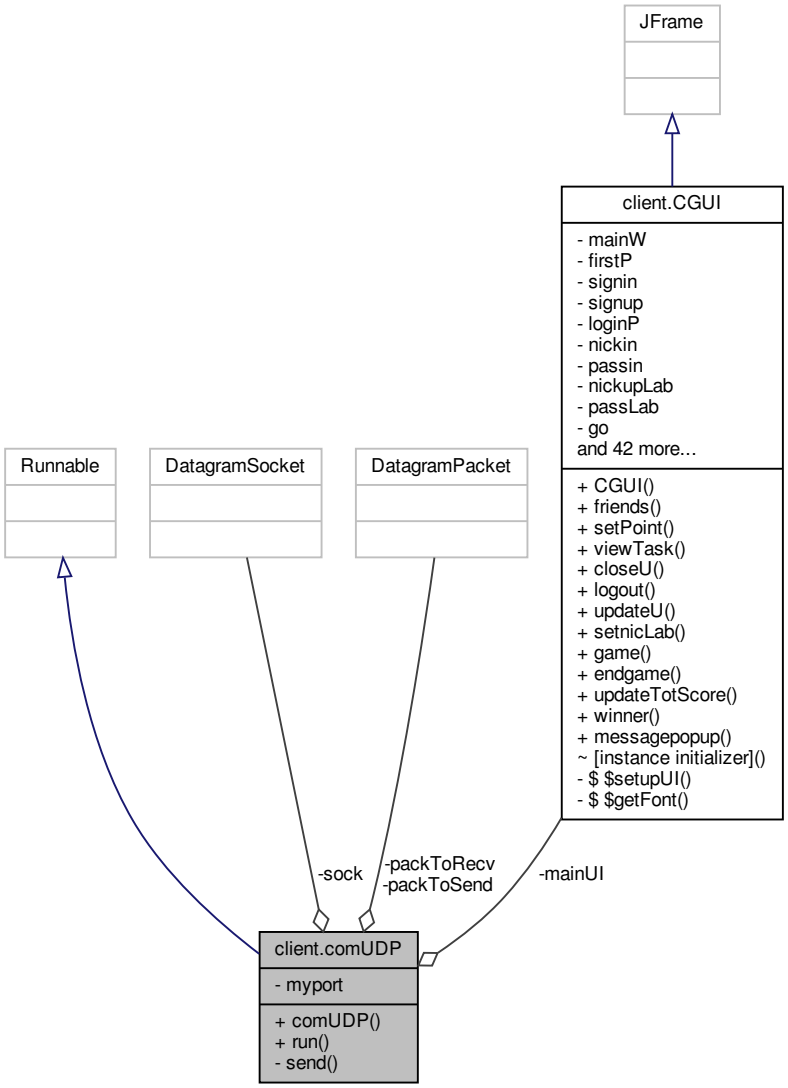
- src/server/ClientTasks.java

7.8 client.comUDP Class Reference

Inheritance diagram for client.comUDP:



Collaboration diagram for client.comUDP:



Public Member Functions

- comUDP (int port, CGUI ui)
- void run ()

Private Member Functions

- void send (String msg, InetAddress address, int port)

Private Attributes

- int myport

  numero di porta su cui il client riceve il paccetto
- CGUI mainUI

  riferimento al gestore della grafica
- DatagramPacket packToSend = null

  pacchetto per l'invio di messaggi
- DatagramPacket packToRecv = null

  pacchetto per la ricezione di messaggi
- DatagramSocket sock = null

  socket UDP per la comunicazione

7.8.1 Detailed Description

In questa classe vengono implementati i metodi necessari per la comunicazione su protocollo UDP tra server e client per la ricezione di richieste di sfida e l'invio della risposta. Tale classe deve essere eseguita da un thread.

Author

Luca Canessa (Mat. 516639)

Version

1.4

Since

1.0

7.8.2 Constructor & Destructor Documentation

7.8.2.1 comUDP()

```
client.comUDP.comUDP (
            int port,
            CGUI ui )
```

Costruttore del gestore della comunicazione su protocollo UDP

Parameters

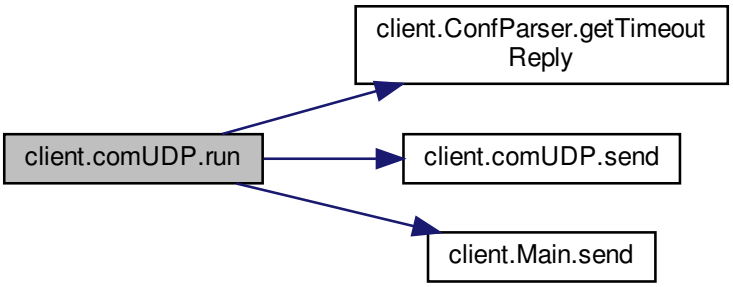
port	porta di comunicazione del client
ui	riferimento al gestore della grafica

7.8.3 Member Function Documentation

7.8.3.1 run()

```
void client.comUDP.run ( )
```

Metodo messo in esecuzione dal thread che si occupa di gestire il pacchetto in ingresso, tradurre il messaggio all'intero, interfaccaiarsi con l'utente e inviare un messaggio di risposta al server. Si aspetta che il messaggio in ingresso sia il 'Nickname' del friend che vuole una sfida. Quando il Nickname è disponibile, crea una finestra per comunicare con l'utente e invia reinvia al server la risposta alla sfida che l'utente gli ha dato. Poi chiude la finestra creata e si rimetti in ascolto sulla socket. Allo scadere di un tempo la riposta inviata di default è il rifiuto della sfida. Here is the call graph for this function:



7.8.3.2 send()

```
void client.comUDP.send (
            String msg,
            InetAddress address,
            int port )  [private]
```

Metodo che si occupa dell'invio del messaggio 'msg' all'indirizzo 'address' sulla porta 'port' del server

Parameters

msg	messaggio da inviare al server
address	indirizzo IP del server
port	numero di porta su cui comunica il server

Here is the caller graph for this function:

7.8.4 Member Data Documentation

7.8.4.1 mainUI

```
CGUI client.comUDP.mainUI  [private]
```

riferimento al gestore della grafica

7.8.4.2 myport

```
int client.comUDP.myport  [private]
```

numero di porta su cui il client riceve il paccetto

7.8.4.3 packToRecv

```
DatagramPacket client.comUDP.packToRecv = null  [private]
```

pacchetto per la ricezione di messaggi

7.8.4.4 packToSend

```
DatagramPacket client.comUDP.packToSend = null  [private]
```

pacchetto per l'invio di messaggi

7.8.4.5 sock

```
DatagramSocket client.comUDP.sock = null  [private]
```

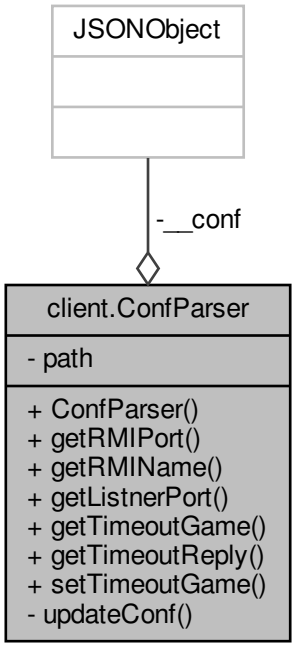
socket UDP per la comunicazione

The documentation for this class was generated from the following file:

- src/client/comUDP.java

7.9 client.ConfParser Class Reference

Collaboration diagram for client.ConfParser:



Public Member Functions

- ConfParser (String confPath)
- int getRMIPort ()
- String getRMIName ()
- int getListnerPort ()
- long getTimeoutGame ()
- long getTimeoutReply ()
- void setTimeoutGame (String time)

Private Member Functions

- void updateConf ()

Private Attributes

- JSONObject __conf = null

  configurazione completa del server
- String path = null

  percorso verso il file di configurazione

7.9.1 Detailed Description

In questa classe viene gestito il parser del file di configurazione dal quale si estraggono i valori di default del client. Di default il file di configurazione è all'interno della cartella conf.

!!! NECCESSARIO CHE NESSUN VALORE DELLE CHIAVI IN QUESTO FILE SIA VUOTO !!! controlla che tutti i campi siano consistenti appena l'oggetto viene creato

Author

Luca Canessa (Mat. 516639)

Version

1.1

Since

1.0

7.9.2 Constructor & Destructor Documentation

7.9.2.1 ConfParser()

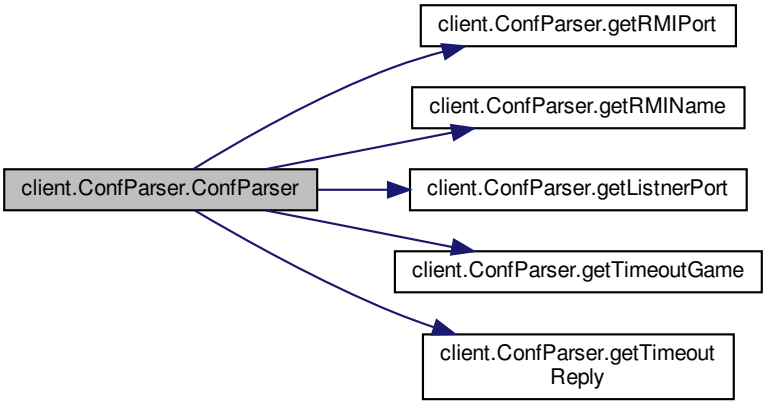
```
client.ConfParser.ConfParser (
            String confPath )
```

Costruttore del parser. Prendendo come parametro la path del file JSON su cui sono salvate le configurazioni, si preoccupa di salvare l'intero file in una struttura JSONObject per poter essere consultata

Parameters

confPath	path del file di configurazione

Here is the call graph for this function:



7.9.3 Member Function Documentation

7.9.3.1 getListnerPort()

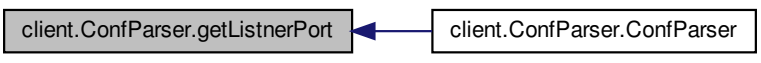
```
int client.ConfParser.getListnerPort ( )
```

Cerca all'interno della configurazione il campo 'port', estraendo il suo valore. Controlla poi se tale valore è utilizzabile, in caso negativo esce restituendo un errore.

Returns

il numero di porta da utilizzare per la socket listener su cui il server accetta le connessioni in entrata

Here is the caller graph for this function:



7.9.3.2 getRMIName()

```
String client.ConfParser.getRMIName ( )
```

Cerca all'interno della configurazione il campo 'rminame', estraendo il suo valore. Controlla poi se tale valore è utilizzabile, in caso negativo esce restituendo un errore.

Returns

il nome del server RMI

Here is the caller graph for this function:

7.9.3.3 getRMIPort()

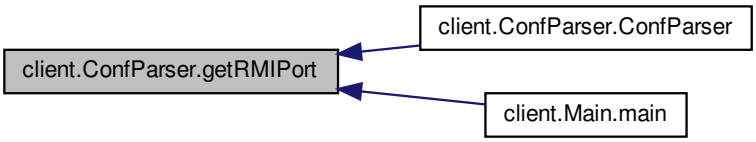
```
int client.ConfParser.getRMIPort ( )
```

Cerca all'interno della configurazione il campo 'rmiport', estraendo il suo valore. Controlla poi se tale valore è utilizzabile, in caso negativo esce restituendo un errore.

Returns

il numero di porta da utilizzare per la 'RegRMI'

Here is the caller graph for this function:



7.9.3.4 getTimeoutGame()

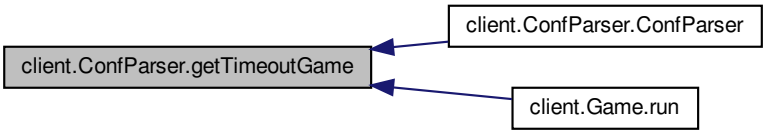
```
long client.ConfParser.getTimeoutGame ( )
```

Cerca all'interno della configurazione il campo 'timeoutgame', estraendo il suo valore. Controllapoi se tale valore è utilizzabile, in caso negativo esce restituendo un errore.

Returns

il tempo massimo per completare una sfida

Here is the caller graph for this function:



7.9.3.5 getTimeoutReply()

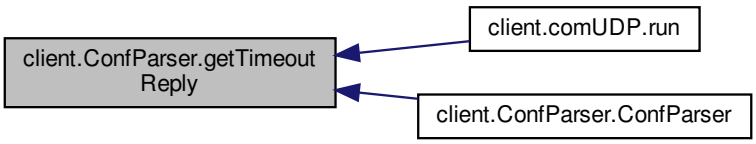
```
long client.ConfParser.getTimeoutReply ( )
```

Cerca all'interno della configurazione il campo 'timeoutreq', estraendo il suo valore. Controlla poi se tale valore è utilizzabile, in caso negativo esce restituendo un errore.

Returns

il tempo massimo per accettare una richiesa di sfida

Here is the caller graph for this function:



7.9.3.6 setTimeoutGame()

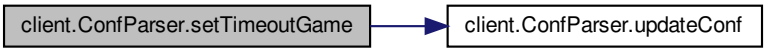
```
void client.ConfParser.setTimeoutGame (
            String time )
```

Cerca all'interno della configurazione il campo 'rminame', estraendo il suo valore. Controlla poi se tale valore è utilizzabile, in caso negativo esce restituendo un errore.

Returns

il nome del server RMI

Here is the call graph for this function:



Here is the caller graph for this function:

7.9.3.7 updateConf()

```
void client.ConfParser.updateConf ( )  [private]
```

Scrive su file l'aggiornamento della configurazione che gli è arrivata dal server. Here is the caller graph for this function:

7.9.4 Member Data Documentation

7.9.4.1 __conf

```
JSONObject client.ConfParser.__conf = null  [private]
```

configurazione completa del server

7.9.4.2 path

```
String client.ConfParser.path = null  [private]
```

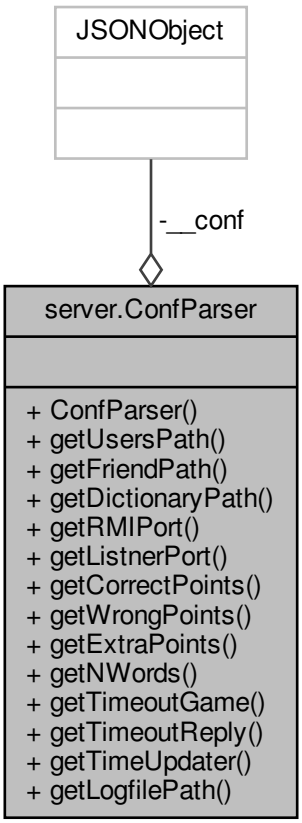
percorso verso il file di configurazione

The documentation for this class was generated from the following file:

- src/client/ConfParser.java

7.10 server.ConfParser Class Reference

Collaboration diagram for server.ConfParser:



Public Member Functions

- ConfParser (String confPath)
- String getUsersPath ()
- String getFriendPath ()
- String getDictionaryPath ()
- int getRMIPort ()
- int getListnerPort ()
- int getCorrectPoints ()
- int getWrongPoints ()
- int getExtraPoints ()
- int getNWords ()
- long getTimeoutGame ()
- long getTimeoutReply ()
- long getTimeUpdater ()
- String getLogfilePath ()

Private Attributes

- JSONObject __conf = null

  configurazione completa del server

7.10.1 Detailed Description

In questa classe viene gestito il parser del file di configurazione dal quale si estraggono i valori di default del server. Di default il file di configurazione è all'interno della cartella conf. Il file è composto da tre oggetti principali:

- SERVER: ci sono tutti i valori che servono per le configurazioni di base del server
- GAME: ci sono tutti i valori utili per settare le sfide tra gli utenti
- DB: ci sono tutti i valori necessari per impostare i database degli utenti e delle amicizie tra gli User

  !!! NECCESSARIO CHE NESSUN VALORE DELLE CHIAVI IN QUESTO FILE SIA VUOTO !!! controlla che tutti i campi siano consistenti appena l'oggetto viene creato

Author

Luca Canessa (Mat. 516639)

Version

2.0

Since

1.0

7.10.2 Constructor & Destructor Documentation

7.10.2.1 ConfParser()

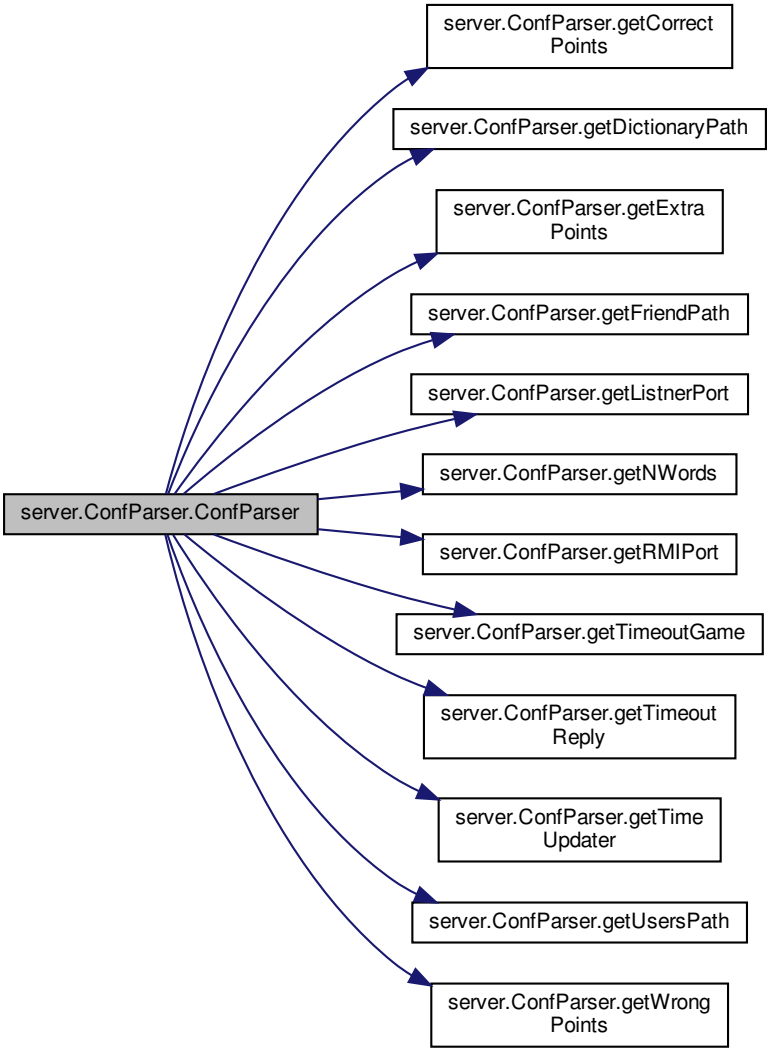
```
server.ConfParser.ConfParser (
            String confPath )
```

Costruttore del parser. Prendendo come parametro la path del file JSON su cui sono salvate le configurazioni, si preoccupa di salvare l'intero file in una struttura JSONObject per poter essere consultata. Prima di terminare la creazione dell'parser controlla che tutti i valori di configurazione nel file siano attendibili e consistenti. !!! TUTTE LE QUANTITÀ DI TEMPO SONO DA RITENERSI IN UNITÀ DI TEMPO DI MILLISECONDI !!!

Parameters

confPath	path del file di configurazione

Here is the call graph for this function:



7.10.3 Member Function Documentation

7.10.3.1 getCorrectPoints()

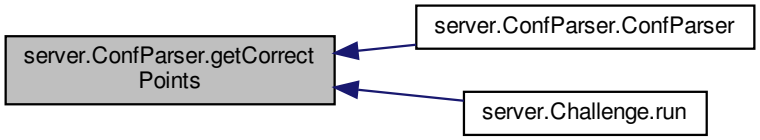
```
int server.ConfParser.getCorrectPoints ( )
```

Cerca all'interno della configurazione il campo 'correct', estrapolandolo dall'oggetto 'GAME', estraendo il suo valore. Controlla poi se tale valore è utilizzabile, in caso negativo esce restituendo un errore.

Returns

il numero di punti da assegnare ad ogni risposta corretta durante una sfida

Here is the caller graph for this function:



7.10.3.2 getDictionaryPath()

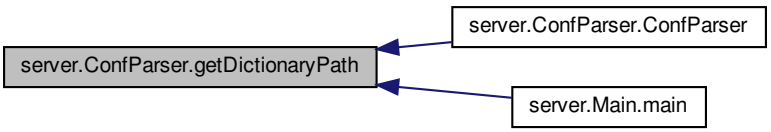
```
String server.ConfParser.getDictionaryPath ( )
```

Cerca all'interno della configurazione il campo 'dictionary', estrapolandolo dall'oggetto 'SERVER', estraendo il suo valore. Controlla poi se tale valore è utilizzabile, in caso negativo esce restituendo un errore.

Returns

il percorso al dizionario delle parole italiane

Here is the caller graph for this function:



7.10.3.3 getExtraPoints()

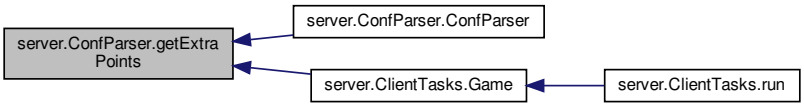
```
int server.ConfParser.getExtraPoints ( )
```

Cerca all'interno della configurazione il campo 'extra', estrapolandolo dall'oggetto 'GAME', estraendo il suo valore. Controlla poi se tale valore è utilizzabile, in caso negativo esce restituendo un errore.

Returns

il numero dei punti extra da assegnare al vincitore di una partita

Here is the caller graph for this function:



7.10.3.4 getFriendPath()

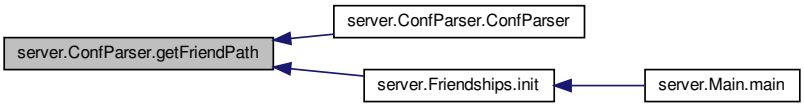
```
String server.ConfParser.getFriendPath ( )
```

Cerca all'interno della configurazione il campo 'friendpath', estrapolandolo dall'oggetto 'DB', estraendo il suo valore. Controlla poi se tale valore è utilizzabile, in caso negativo esce restituendo un errore.

Returns

il percorso al DB delle amicizie degli utenti

Here is the caller graph for this function:



7.10.3.5 getListnerPort()

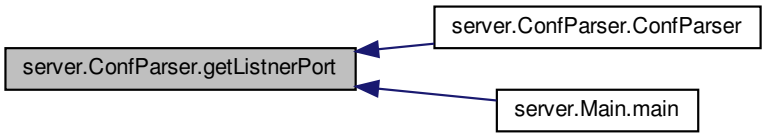
```
int server.ConfParser.getListnerPort ( )
```

Cerca all'interno della configurazione il campo 'port', estrapolandolo dall'oggetto 'SERVER', estraendo il suo valore. Controlla poi se tale valore è utilizzabile, in caso negativo esce restituendo un errore.

Returns

il numero di porta da utilizzare per la socket listener su cui il server accetta le connessioni in entrata

Here is the caller graph for this function:



7.10.3.6 getLogfilePath()

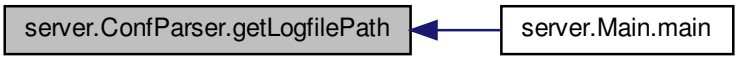
```
String server.ConfParser.getLogfilePath ( )
```

Cerca all'interno della configurazione il campo 'logpath', estrapolandolo dall'oggetto 'SERVER', estraendo il suo valore. Controlla poi se tale valore è utilizzabile, in caso negativo esce restituendo un errore.

Returns

il percorso al file di log del server

Here is the caller graph for this function:



7.10.3.7 getNWords()

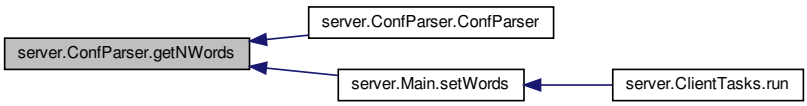
```
int server.ConfParser.getNWords ( )
```

Cerca all'interno della configurazione il campo 'words', estrapolandolo dall'oggetto 'GAME', estraendo il suo valore. Controlla poi se tale valore è utilizzabile, in caso negativo esce restituendo un errore.

Returns

il numero di parole che devono essere inviate agli sfidanti per una partita

Here is the caller graph for this function:



7.10.3.8 getRMIPort()

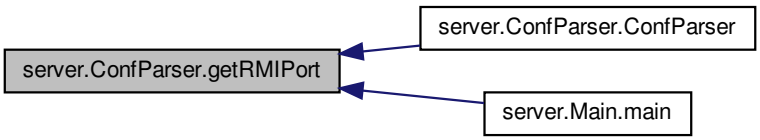
```
int server.ConfParser.getRMIPort ( )
```

Cerca all'interno della configurazione il campo 'rmiport', estrapolandolo dall'oggetto 'SERVER', estraendo il suo valore. Controlla poi se tale valore è utilizzabile, in caso negativo esce restituendo un errore.

Returns

il numero di porta da utilizzare per la 'RegRMI'

Here is the caller graph for this function:



7.10.3.9 getTimeoutGame()

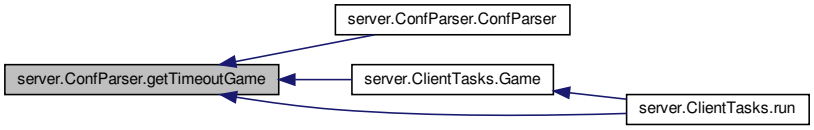
```
long server.ConfParser.getTimeoutGame ( )
```

Cerca all'interno della configurazione il campo 'timeoutgame', estrapolandolo dall'oggetto 'GAME', estraendo il suo valore. Controlla poi se tale valore è utilizzabile, in caso negativo esce restituendo un errore.

Returns

il tempo massimo per completare una sfida

Here is the caller graph for this function:



7.10.3.10 getTimeoutReply()

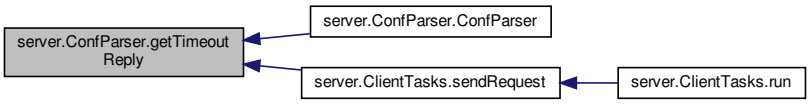
```
long server.ConfParser.getTimeoutReply ( )
```

Cerca all'interno della configurazione il campo 'timeoutreq', estrapolandolo dall'oggetto 'SERVER', estraendo il suo valore. Controlla poi se tale valore è utilizzabile, in caso negativo esce restituendo un errore.

Returns

il tempo massimo per accettare una richiesa di sfida

Here is the caller graph for this function:



7.10.3.11 getTimeUpdater()

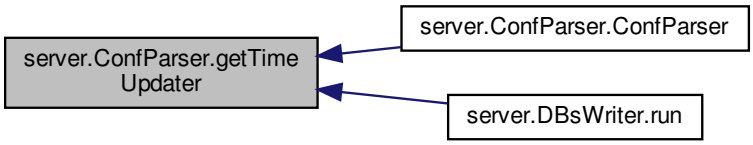
```
long server.ConfParser.getTimeUpdater ( )
```

Cerca all'interno della configurazione il campo 'updaterdb', estrapolandolo dall'oggetto 'SERVER', estraendo il suo valore. Controlla poi se tale valore è utilizzabile, in caso negativo esce restituendo un errore.

Returns

ogni quanti millisecondi il database viene aggiornato su file

Here is the caller graph for this function:



7.10.3.12 getUsersPath()

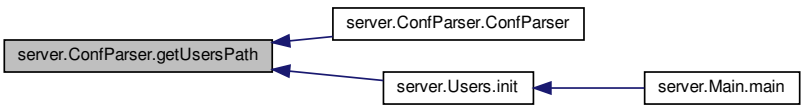
```
String server.ConfParser.getUsersPath ( )
```

Cerca all'interno della configurazione il campo 'userpath', estrapolandolo dall'oggetto 'DB', estraendo il suo valore. Controlla poi se tale valore è utilizzabile, in caso negativo esce restituendo un errore.

Returns

il percorso al DB degli utenti

Here is the caller graph for this function:



7.10.3.13 getWrongPoints()

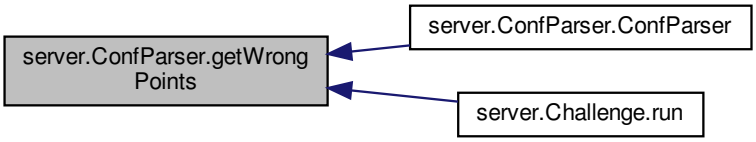
```
int server.ConfParser.getWrongPoints ( )
```

Cerca all'interno della configurazione il campo 'wrong', estrapolandolo dall'oggetto 'GAME', estraendo il suo valore. Controlla poi se tale valore è utilizzabile, in caso negativo esce restituendo un errore.

Returns

il numero di punti che devono essere sottratti ad ogni risposta sbagliata durante una sfida

Here is the caller graph for this function:



7.10.4 Member Data Documentation

7.10.4.1 __conf

```
JSONObject server.ConfParser.__conf = null  [private]
```

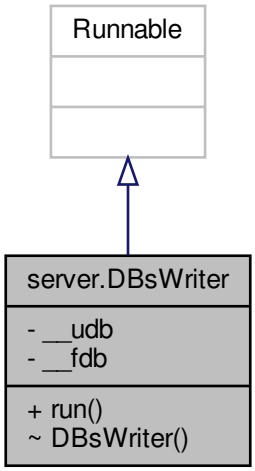
configurazione completa del server

The documentation for this class was generated from the following file:

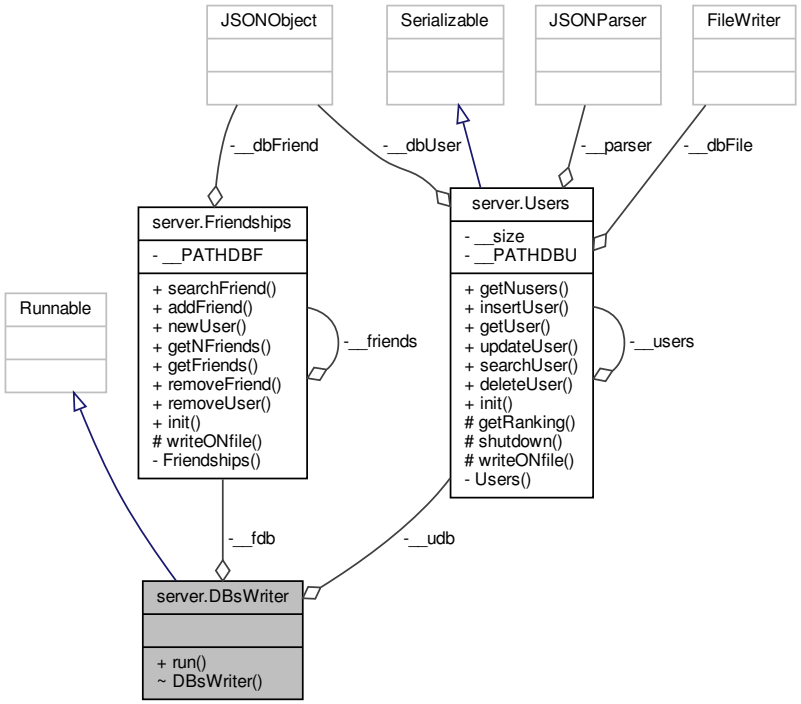
- src/server/ConfParser.java

7.11 server.DBsWriter Class Reference

Inheritance diagram for server.DBsWriter:



Collaboration diagram for server.DBsWriter:



Public Member Functions

- void run ()

Package Functions

- DBsWriter (Users UDB, Friendships FDB)

Private Attributes

- Users __udb = null

  database degli User
- Friendships __fdb = null

  database delle amicizie tra gli User

7.11.1 Detailed Description

In questa classe viene implementato l'eseguibile di un thread utile per l'aggioramento dei database sui rispettivi files json salvati su disco. Il thread messo in esecuzione aggiornerà i files allo scadere di un determinato tempo, settato nel file di configurazione.

Author

Luca Canessa (Mat. 516639)

Version

1.0

Since

1.0

7.11.2 Constructor & Destructor Documentation

7.11.2.1 DBsWriter()

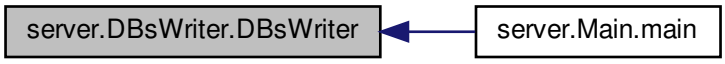
```
server.DBsWriter.DBsWriter (
            Users UDB,
            Friendships FDB )  [package]
```

Costruttore dello scrittore su disco dei database

Parameters

UDB	database degli utentu
FDB	database delle amicizie degli utenti

Here is the caller graph for this function:

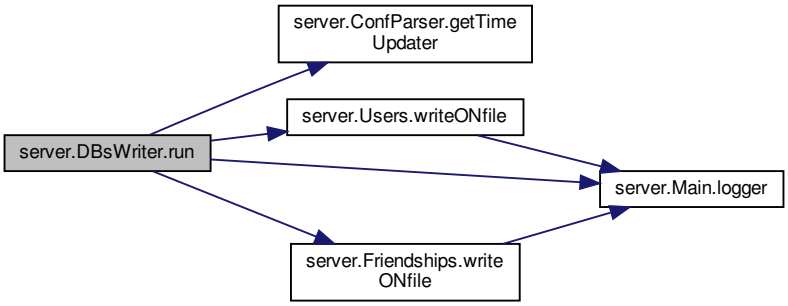


7.11.3 Member Function Documentation

7.11.3.1 run()

```
void server.DBsWriter.run ( )
```

Scrive su file i database attraverso i rispettivi metodi predisposti per tale funzione, al termine della scrittura si mette in 'sleep' fino allo scadere del tempo che è stato impostato nel file di configuazione Here is the call graph for this function:



7.11.4 Member Data Documentation

7.11.4.1 __fdb

```
Friendships server.DBsWriter.__fdb = null  [private]
```

database delle amicizie tra gli User

7.11.4.2 __udb

```
Users server.DBsWriter.__udb = null  [private]
```

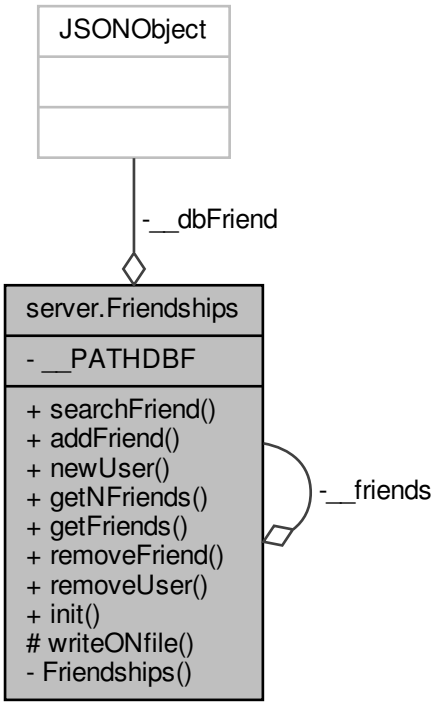
database degli User

The documentation for this class was generated from the following file:

- src/server/Main.java

7.12 server.Friendships Class Reference

Collaboration diagram for server.Friendships:



Public Member Functions

- synchronized ACK searchFriend (User a, User b)
- synchronized ACK addFriend (User a, User b)
- synchronized ACK newUser (User a)
- synchronized long getNFriends (User a)
- synchronized JSONArray getFriends (User a)
- synchronized ACK removeFriend (User a, User b)
- synchronized void removeUser (User a)

Static Public Member Functions

- static Friendships init ()

Protected Member Functions

- synchronized void writeONfile ()

Private Member Functions

- Friendships ()

Static Private Attributes

- static String __PATHDBF = null

  path del db delle amicizie
- static Friendships __friends = null

  singleton
- static JSONObject __dbFriend = null

  struttra dati del db in memoria

7.12.1 Detailed Description

In questa classe viene gestito il database delle amicizie tra gli utenti. Questo database viene recuperato, se esistente, da un file JSON salvato sulla macchina, di default nella cartella '.data'. Tale file JSON è composto da una serie di oggetti che rappresentano l'User registrato a gioco. Ogni oggetto è composto da un JSONArray contenente tutte le amicizie di quell'User.

Author

Luca Canessa (Mat. 516639)

Version

1.3

Since

1.0

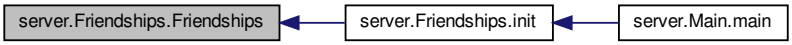
7.12.2 Constructor & Destructor Documentation

7.12.2.1 Friendships()

```
server.Friendships.Friendships ( )  [private]
```

Costruttore del database delle amicizie tra utenti registrati al gioco. Il compito è di aprire e copiare il db in una struttura dati JSON se esistente, altrimenti di crearlo. Costruttore privato perché oggetto implementato come singoletto.
Here is the call graph for this function:

Here is the caller graph for this function:



7.12.3 Member Function Documentation

7.12.3.1 addFriend()

```
synchronized ACK server.Friendships.addFriend (
            User a,
            User b )
```

Aggiunge l'User a alla lista delle amicizie dell'User b e viceversa se già non presenti. Ritorna un valore di ACK

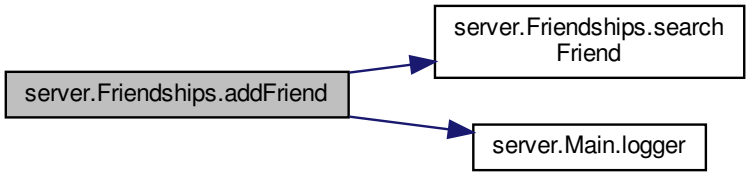
Parameters

a	User
b	User

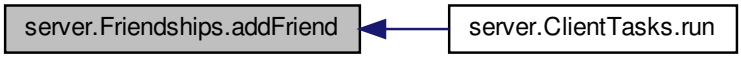
Returns

FriendAdded se aggiunti alle liste AlreadyFriends se non aggiunti

Here is the call graph for this function:



Here is the caller graph for this function:



7.12.3.2 getFriends()

```
synchronized JSONArray server.Friendships.getFriends (
            User a )
```

Restituisce un JSONArray contenente il tutti gli amici dell'User

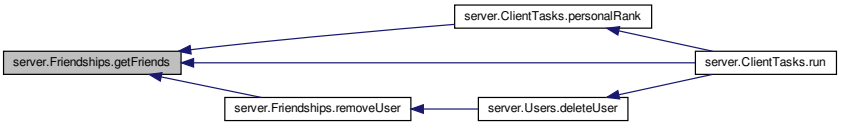
Parameters

a	User che richiede la lista delle amicizie

Returns

lista degli amici (JSONArray)

Here is the caller graph for this function:



7.12.3.3 getNFriends()

```
synchronized long server.Friendships.getNFriends (
            User a )
```

Restituisce il numero di amici di un User

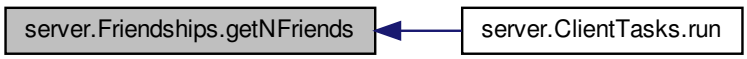
Parameters

a	User di cui sapere il numero di amici

Returns

un numero $>$ 0 se amici presenti, 0 se nessun amico presente

Here is the caller graph for this function:



7.12.3.4 init()

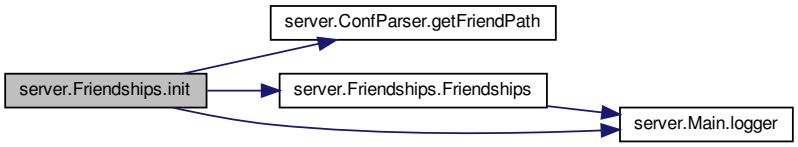
```
static Friendships server.Friendships.init ( )  [static]
```

Metodo che deve essere chiamato per istanziare il db delle amicizie. Metodo necessario per creare un singleton e avere la possibilità di operare solo su un oggetto senza avere problemi di ridondanze di dati e/o inconsistenze

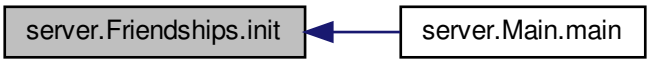
Returns

l'oggetto contenente il db

Here is the call graph for this function:



Here is the caller graph for this function:



7.12.3.5 newUser()

```
synchronized ACK server.Friendships.newUser (
            User a )
```

Metodo chiamato alla creazione di un User per poterlo inserire anche nel database delle amicizie. Restituisce un valore di ACK.

Parameters

a	User da aggiungere al db

Returns

UserAdded se l'utente è stato inserito AlreadyExisting se l'utente era già presente

7.12.3.6 removeFriend()

```
synchronized ACK server.Friendships.removeFriend (
            User a,
            User b )
```

Rimuove gli User dalla reciproca lista delli amici. Restituisce un valore di ACK

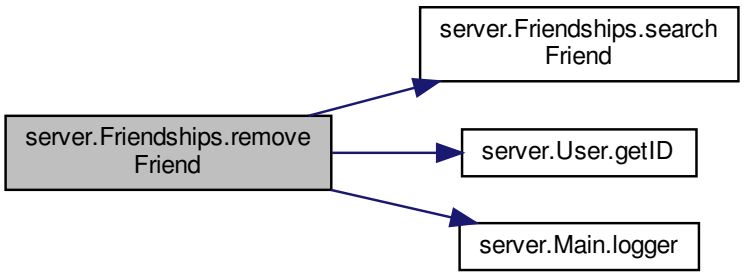
Parameters

a	User
b	User

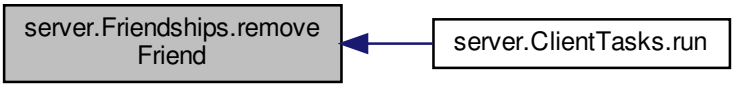
Returns

FriendRemoved se gli User sono stati rimossi AlreadyFriend se non sono stati rimossi

Here is the call graph for this function:



Here is the caller graph for this function:



7.12.3.7 removeUser()

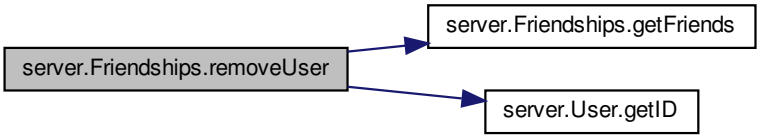
```
synchronized void server.Friendships.removeUser (
            User a )
```

Rimuove l'User dalla lista di tutti i suoi amici e poi lo cancella dal database

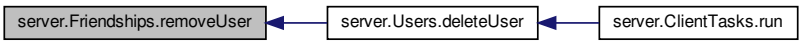
Parameters

a	User da eliminare

Here is the call graph for this function:



Here is the caller graph for this function:



7.12.3.8 searchFriend()

```
synchronized ACK server.Friendships.searchFriend (
            User a,
            User b )
```

Cerca all'interno del db delle amicizie se due User sono amici tra loro restituendo un valore di ACK al termine

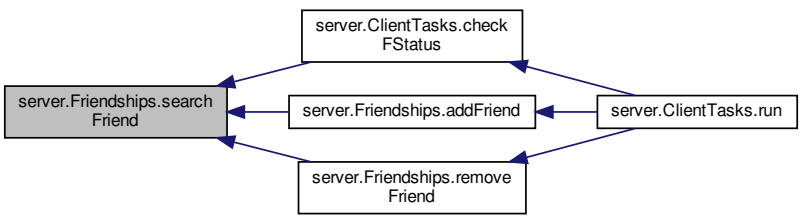
Parameters

a	User
b	User

Returns

AlreadyFriends se i due utenti sono già amici FriendNotFound se i due utenti ancora non sono amici

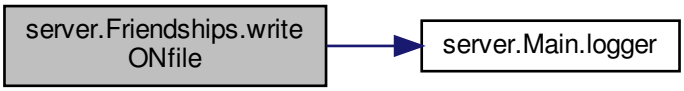
Here is the caller graph for this function:



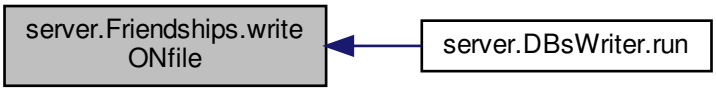
7.12.3.9 writeONfile()

```
synchronized void server.Friendships.writeONfile ( )  [protected]
```

Metodo utilizzato per scrivere il db delle amicizie su file Here is the call graph for this function:



Here is the caller graph for this function:



7.12.4 Member Data Documentation

7.12.4.1 __dbFriend

```
JSONObject server.Friendships.__dbFriend = null  [static], [private]
```

struttra dati del db in memoria

7.12.4.2 __friends

```
Friendships server.Friendships.__friends = null  [static], [private]
```

singleton

7.12.4.3 __PATHDBF

```
String server.Friendships.__PATHDBF = null  [static], [private]
```

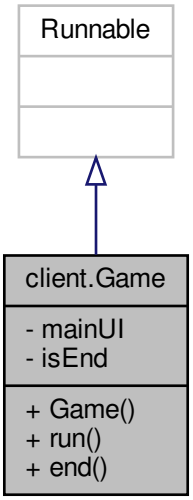
path del db delle amicizie

The documentation for this class was generated from the following file:

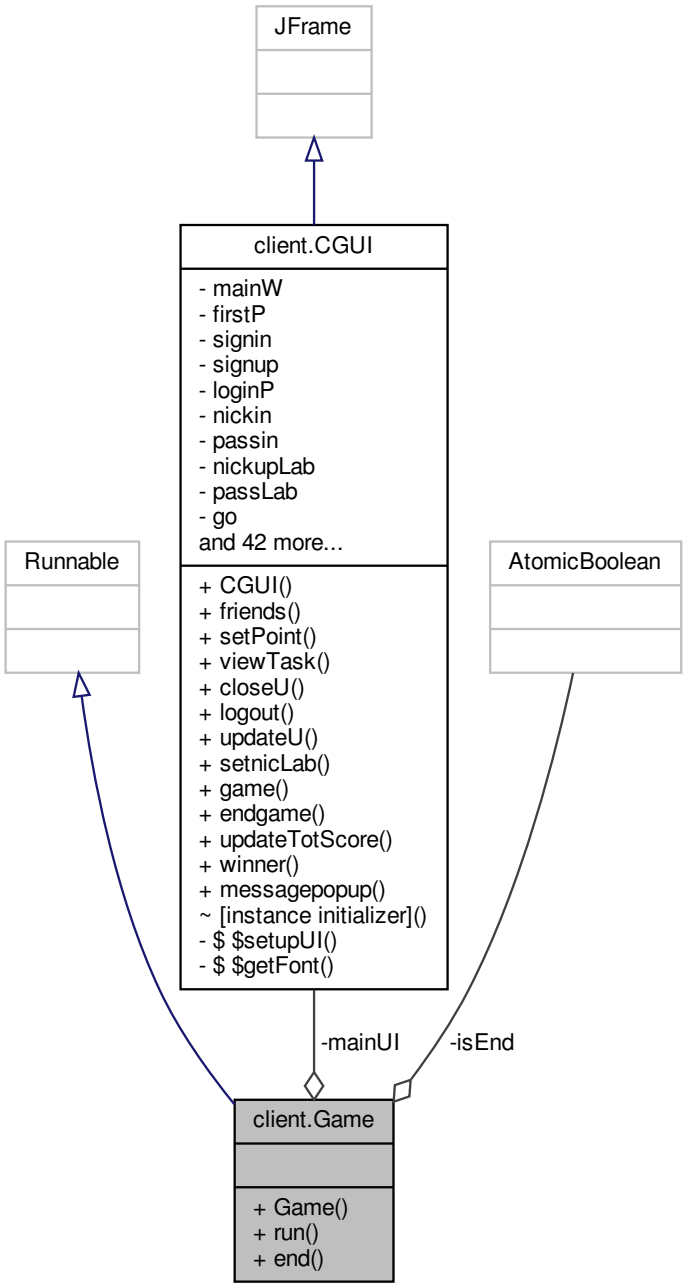
- src/server/Friendships.java

7.13 client.Game Class Reference

Inheritance diagram for client.Game:



Collaboration diagram for client.Game:



Public Member Functions

- Game (CGUI ui)
- void run ()

Static Public Member Functions

- static void end ()

Private Attributes

- CGUI mainUI

  riferimento al gestore della grafica

Static Private Attributes

- static AtomicBoolean isEnd

  flag usato per determinare quando terminare la partita

7.13.1 Detailed Description

In questa classe vengono implementati i metodi per gestire il gioco quando entrambi i giocatori hanno accettato la sfida. Viene creata una nuova finestra in cui vengono visualizzate le parole da tradurre e dove l'utente può scrivere la traduzione secondo lui corretta. Al termine delle parole o allo scadere del Timeout, viene gestita la chiusura della sfida.

Author

Luca Canessa (Mat. 516639)

Version

1.5

Since

1.0

7.13.2 Constructor & Destructor Documentation

7.13.2.1 Game()

```
client.Game.Game (
            CGUI ui )
```

Costruttore del gestore della sfida

Parameters

ui	riferimento al gestore della grafica

7.13.3 Member Function Documentation

7.13.3.1 end()

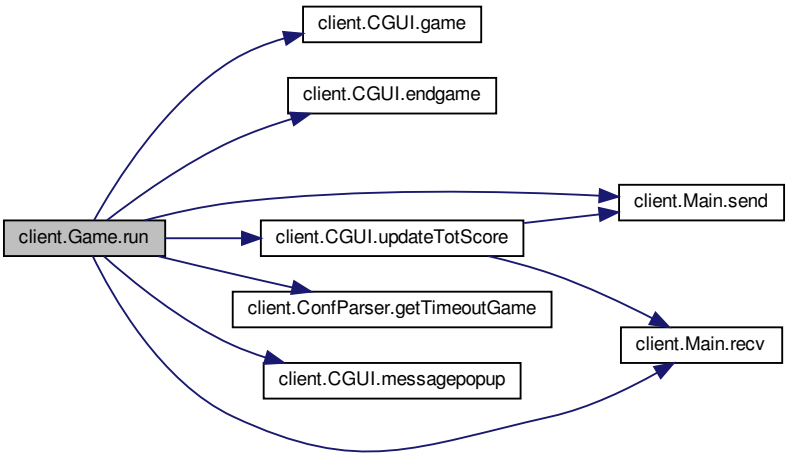
```
static void client.Game.end ( )  [static]
```

Metodo incaricato di settare la variabile di controllo di terminazione a true, per concludere il match

7.13.3.2 run()

```
void client.Game.run ( )
```

Messo in esecuzione da un thread, tale metodo ha il compito di creare una finestra per interfacciare l'utente con il gioco e nascondere la schermata principale. Creata la finestra scrive la parola italiana ricevuta e aspetta che l'utente premi il bottone di invio che permette di recapitare al server la parola che l'utente ha scritto come traduzione di quella visualizzata. Allo scadere del tempo richiesto per la sfida o al termine delle parole da tradurre, il metodo si occupa di aggiornare la grafica nascondendo la finestra di gioco e rivisualizzando la schermata principale, di scrivere un un popup di sistema il nome del vincitore e aggiornare i punti totali dell'utente sulla schermata principale. Here is the call graph for this function:



7.13.4 Member Data Documentation

7.13.4.1 isEnd

```
AtomicBoolean client.Game.isEnd  [static], [private]
```

flag usato per determinare quando terminare la partita

7.13.4.2 mainUI

```
CGUI client.Game.mainUI  [private]
```

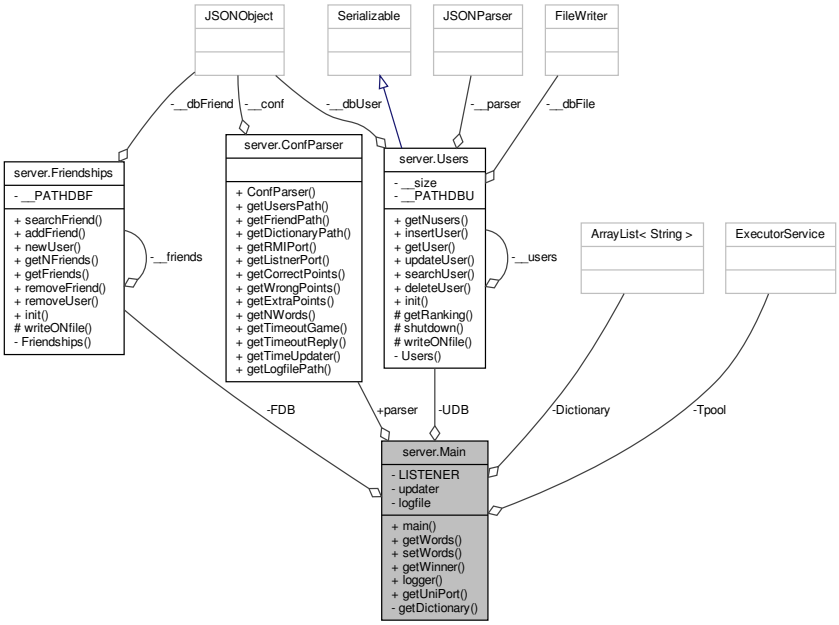
riferimento al gestore della grafica

The documentation for this class was generated from the following file:

- src/client/Game.java

7.14 server.Main Class Reference

Collaboration diagram for server.Main:



Static Public Member Functions

- static void main (String[ ] args)
- static ArrayList< ArrayList< String > > getWords (String nickname)
- static synchronized void setWords (String sender) throws IOException, ParseException
- static String getWinner (User U1, User U2, Users udb)
- static void logger (String log)
- static int getUniPort (String nickname)

Static Public Attributes

- static ConfParser parser = null

  parser del file di configurazione

Static Private Member Functions

- static ArrayList< String > getDictionary (String filepath)

Static Private Attributes

- static Users UDB = null

  database degli User
- static Friendships FDB = null

  database delle amicizie tra gli User
- static ExecutorService Tpool = null

  thread pool usato per i thread che gestiscono le richieste dei clients
- static ServerSocket LISTENER = null

  socket predisposta per l'ascolto dei clients che vogliono collegarsi e per la loro accettazione
- static ArrayList< String > Dictionary = null

  dizionario delle parole italiane disponibili
- static Thread updater = null

  thread per l'esecuzione del DBsWriter
- static PrintStream logfile = null

  scrittore su file di un log del server

7.14.1 Detailed Description

Classe principale di esecuzione del server. Tale classe rappresenta il server in toto.

Author

Luca Canessa (Mat. 516639)

Version

1.8

Since

1.0

7.14.2 Member Function Documentation

7.14.2.1 getDictionary()

```
static ArrayList<String> server.Main.getDictionary (
            String filepath )  [static], [private]
```

Metodo statico predisposto per l'analisi e la scrittura in memoria del dizionario delle parole italiane. Legge una parola per riga e la inserisce in un 'ArrayList', saltando le righe commentate usando il carattere '#'. Restituisce il dizionario salvato in memoria

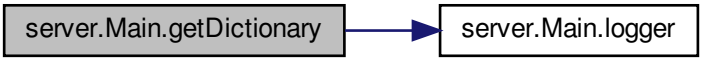
Parameters

filepath	percorso al dizionario salvato su disco

Returns

result ArrayList di stringhe contenente il dizionario null in caso di errore

Here is the call graph for this function:



7.14.2.2 getUniPort()

```
static int server.Main.getUniPort (
            String nickname )  [static]
```

Metodo statico utile per la creazione della porta univoca corrispondente al client. Per la sua creazione viene usato l'hash code del suo nome, e modificato opportunamente per essere usato.

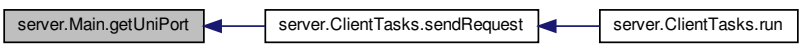
Parameters

nickname	nickname dell'utente di cui si vuole conoscere la porta di comunicazione

Returns

port un numero compreso tra 1024 e 65535 corrispondente alla porta del client

Here is the caller graph for this function:



7.14.2.3 getWinner()

```
static String server.Main.getWinner (
            User U1,
            User U2,
            Users udb )  [static]
```

Aspetta che i giocatori abbiano terminato la partita per poter controllare i loro rispettivi punti guadagnati e decretare il vincitore del match

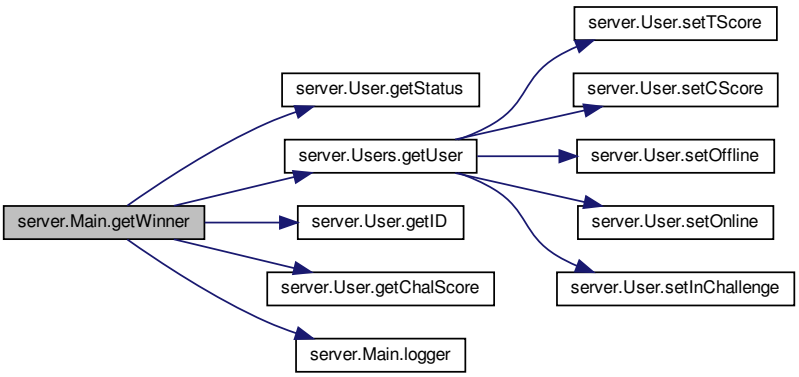
Parameters

U1	User 1 appartenente alla sfida in analisi
U2	User 2 appartenente alla sfida in analisi
udb	database degli utenti

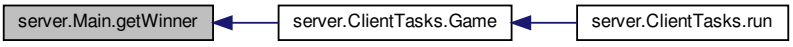
Returns

il nickname del vincitore della sfida 'TIE' nel caso in cui il match termini in parità

Here is the call graph for this function:



Here is the caller graph for this function:



7.14.2.4 getWords()

```
static ArrayList<ArrayList<String> > server.Main.getWords (
            String nickname )  [static]
```

Estrazione dal file salvato su filesystem dei sets di parole necessarie per avviare la sfida, contrassegnati dal hash code del nickname del giocatore sfidante. Entrambi i set vengono salvati in un 'ArrayList' contenente due 'ArrayList' con ripsettivamente le parole italiane e quelle inglesi

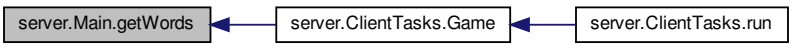
Parameters

nickname	nickname del giocatore sfidante usato per la creazione dei file su disco

Returns

words 'ArrayList' di 'ArrayList' in cui vi sono le parole da inviare ai giocatori e le loro traduzioni

Here is the caller graph for this function:



7.14.2.5 logger()

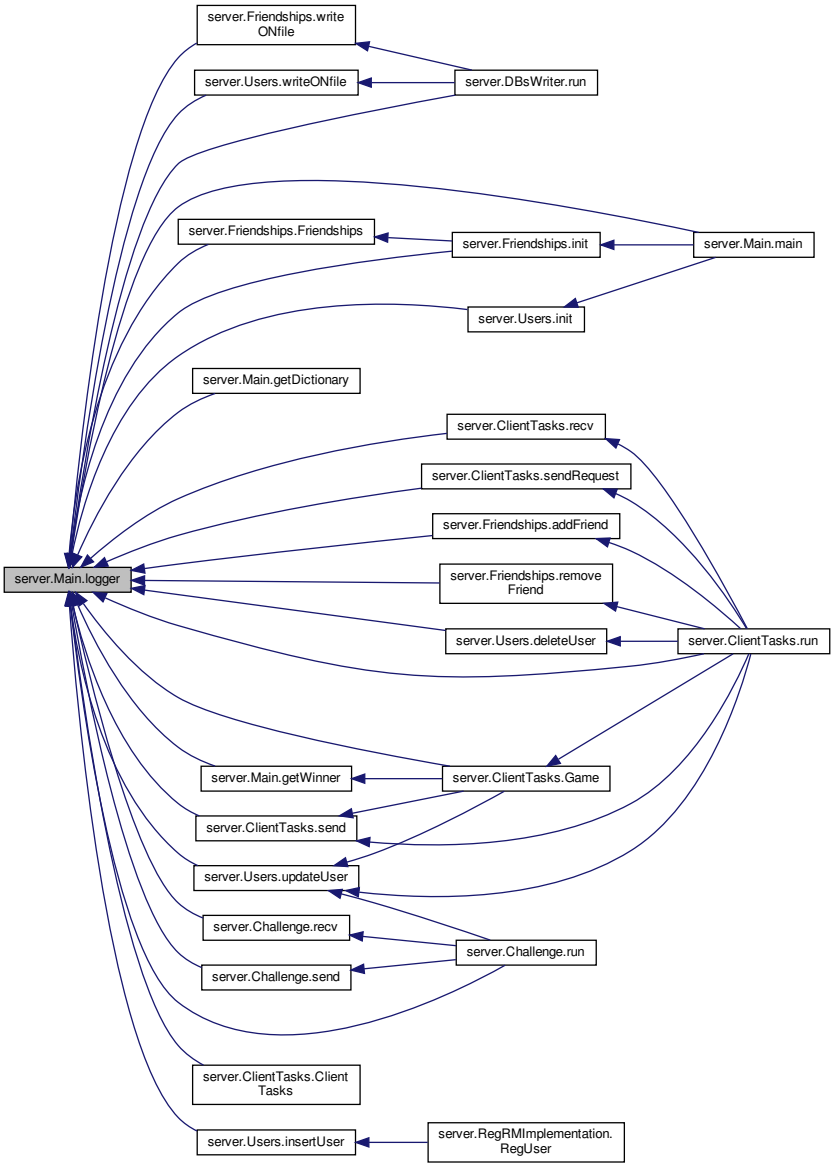
```
static void server.Main.logger (
            String log )  [static]
```

Metodo predisposto per il settaggio del messaggio da scrivere nel file di log. Viene impostata l'ora, i minuti, i secondi e i millisecondi del giorno nel mese nell'anno in cui si scrive il messaggio sul file.

Parameters

log	messaggio da scrivere su file

Here is the caller graph for this function:



7.14.2.6 main()

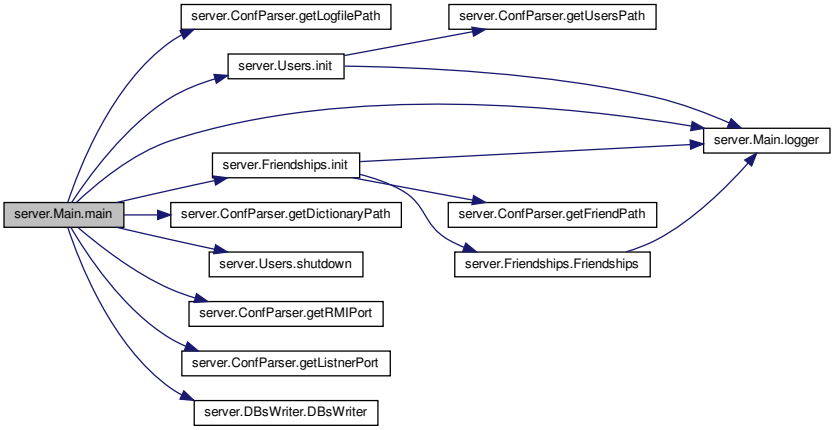
```
static void server.Main.main (
            String [] args )  [static]
```

Metodo principale di esecuzione del server. All'avvio del server controlla che gli venga passato il percorso al file di configurazione del server; inizializza il parser di configurazione; inizializza lo scrittore su file del log del server; inizializza i database 'users' e 'friendship'; inizializza il dizionario delle parole italiane; inizializza un handler dei segnali di terminazione, in modo che il server, quando viene chiuso, abbia sempre i dati consistenti e la situazione degli User aggiornata; inizializza il threadpool per la gestione dei tasks dei clients; inizializza la socket di ascolto per le richieste di connessione; inizializza e setta RMI per poter essere usato dai client; inizia il ciclo infinito che lo mette in ascolto di richieste di connessione

Parameters

args	percorso al file di configurazione

Here is the call graph for this function:



7.14.2.7 setWords()

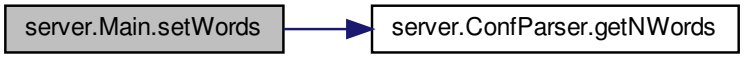
```
static synchronized void server.Main.setWords (
            String sender ) throws IOException, ParseException  [static]
```

Metodo statico predisposto per la selezione random di un numero di parole dal dizionario e la loro traduzione, per poter essere condivise con gli utenti che le hanno richieste. Estrapola quindi un numero di parole (numero impostato nel file di configurazione) dal dizionario, scelte a caso e controllando che non siano mai uguali. Queste parole vengono salvate in un 'ArrayList'. Per ogni parole nel 'ArrayList' viene eseguita la traduzione usando il servizio esterno 'mymemory.translated.net' attraverso le API che mette a disposizione. Tale servizio restituisce un JSON che viene analizzato e dal quale viene estratta la traduzione (in inglese) della parole richiesta. Tutte le parole tradotte vengono salvate in un altro 'ArrayList'. Entrambi gli 'ArrayList' contenenti le parole, vengono trascritti su un file nella cartella '/tmp' del filesystem, utilizzando come nome del file il codice hash del nickname del richiedente del set di parole.

Parameters

sender	nickname dell'User che richiede il set di parole

Here is the call graph for this function:



Here is the caller graph for this function:

7.14.3 Member Data Documentation

7.14.3.1 Dictionary

```
ArrayList<String> server.Main.Dictionary = null  [static], [private]
```

dizionario delle parole italiane disponibili

7.14.3.2 FDB

```
Friendships server.Main.FDB = null  [static], [private]
```

database delle amicizie tra gli User

7.14.3.3 LISTENER

```
ServerSocket server.Main.LISTENER = null  [static], [private]
```

socket predisposta per l'ascolto dei clients che vogliono collegarsi e per la loro accettazione

7.14.3.4 logfile

```
PrintStream server.Main.logfile = null  [static], [private]
```

scrittore su file di un log del server

7.14.3.5 parser

```
ConfParser server.Main.parser = null  [static]
```

parser del file di configurazione

7.14.3.6 Tpool

```
ExecutorService server.Main.Tpool = null  [static], [private]
```

thread pool usato per i thread che gestiscono le richieste dei clients

7.14.3.7 UDB

```
Users server.Main.UDB = null  [static], [private]
```

database degli User

7.14.3.8 updater

```
Thread server.Main.updater = null  [static], [private]
```

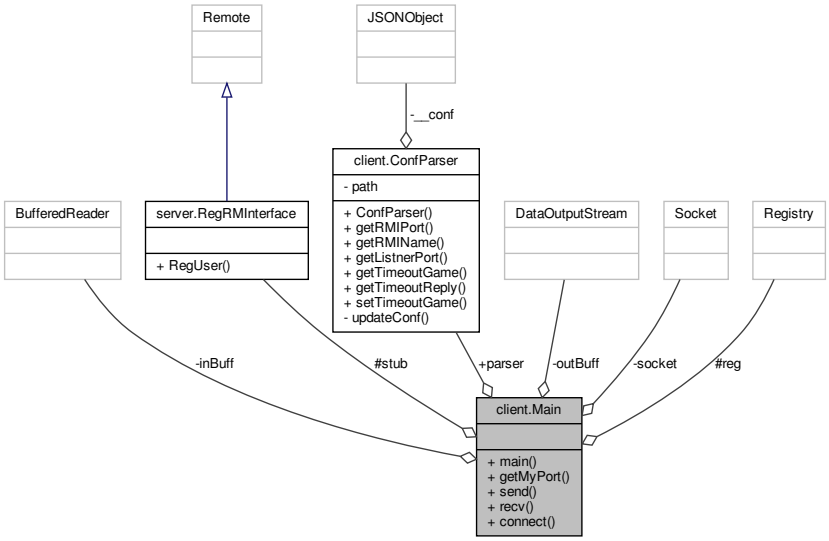
thread per l'esecuzione del DBsWriter

The documentation for this class was generated from the following file:

- src/server/Main.java

7.15 client.Main Class Reference

Collaboration diagram for client.Main:



Static Public Member Functions

- static void main (String[ ] args) throws IOException
- static int getMyPort (String myname)
- static void send (String toSend) throws IOException
- static String recv () throws IOException
- static void connect ()

Static Public Attributes

- static ConfParser parser = null

  parser della configurazione

Static Protected Attributes

- static Registry reg = null

  registro dei serivizi del server
- static RegRMInterface stub = null

  interfaccia RMI del server

Static Private Attributes

- static Socket socket = null

  socket con cui connettersi al server
- static BufferedReader inBuff = null

  buffer di ingresso per la socket
- static DataOutputStream outBuff = null

  buffer di uscita per la socket

7.15.1 Detailed Description

Classe principale che permette di eseguire il client

Author

Luca Canessa (Mat. 516639)

Version

1.6

Since

1.0

7.15.2 Member Function Documentation

7.15.2.1 connect()

```
static void client.Main.connect ( )  [static]
```

Metodo con il compito di instaurare la connessione con il server usando il protocollo di trasporto TCP, e di inizializzare i buffer di ingresso e uscita. Here is the caller graph for this function:

7.15.2.2 getMyPort()

```
static int client.Main.getMyPort (
            String myname )  [static]
```

Metodo statico usato per individuare il numero di porta su cui comunicare quando viene usato il protocollo UDP. Numero ricavato attraverso l'hash code del nickname dell'utente

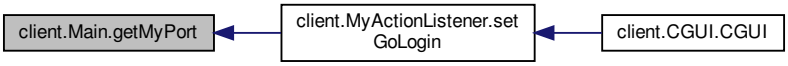
Parameters

myname	nickname dell'utente

Returns

port numero di porta da utilizzare per la socket UDP

Here is the caller graph for this function:



7.15.2.3 main()

```
static void client.Main.main (
            String [] args ) throws IOException  [static]
```

Metodo principale per l'avvio del client. Tale metodo ha il compito di settare il registro dei servizi su RMI del server e il servizio desiderato (iscrizione dell'utente); di inizializzare la socket e i buffers di comunicazione con il server ed avviare la connessione; di inizializzare il gestore della grafica; e di aggioranare il parametro di timeout sul file di configurazione, dopo che lo ha ricevuto dal server. Tutti i valori necessari per l'inizializzazione vengono prelevati dal file di configurazione salvato nel filesystem

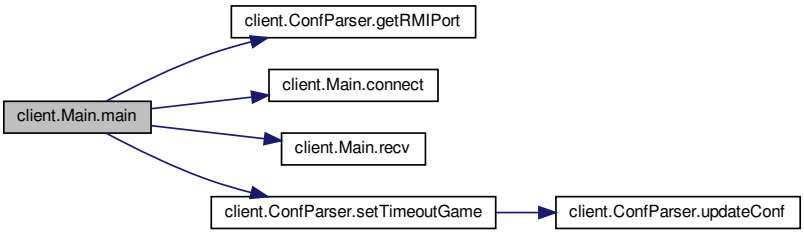
Parameters

args	percorso al file di configurazione

Exceptions

IOException	nel caso di problemi con RMI

Here is the call graph for this function:



7.15.2.4 recv()

```
static String client.Main.recv ( ) throws IOException  [static]
```

Ha il compito di ricevere dati dal server e trascriverli in una stringa per poter essere utlilizzati in altre parti del codice

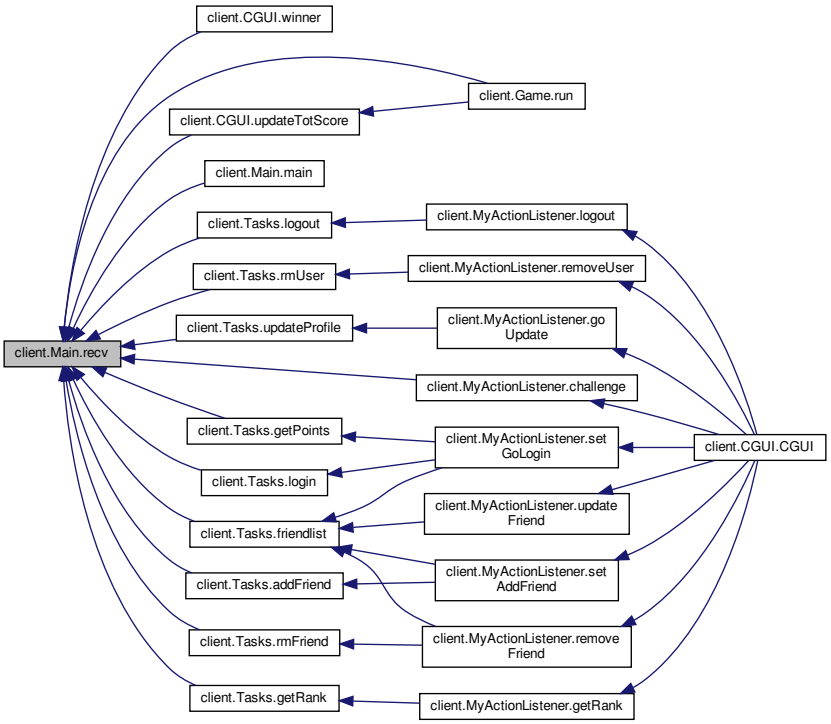
Returns

s String ottenuta leggendo il buffer di ingresso della socket

Exceptions

IOException	

Here is the caller graph for this function:



7.15.2.5 send()

```
static void client.Main.send (
            String toSend ) throws IOException  [static]
```

Ha il compito di inviare il messaggio passatogli come parametro al server attraverso la socket di comunicazione.

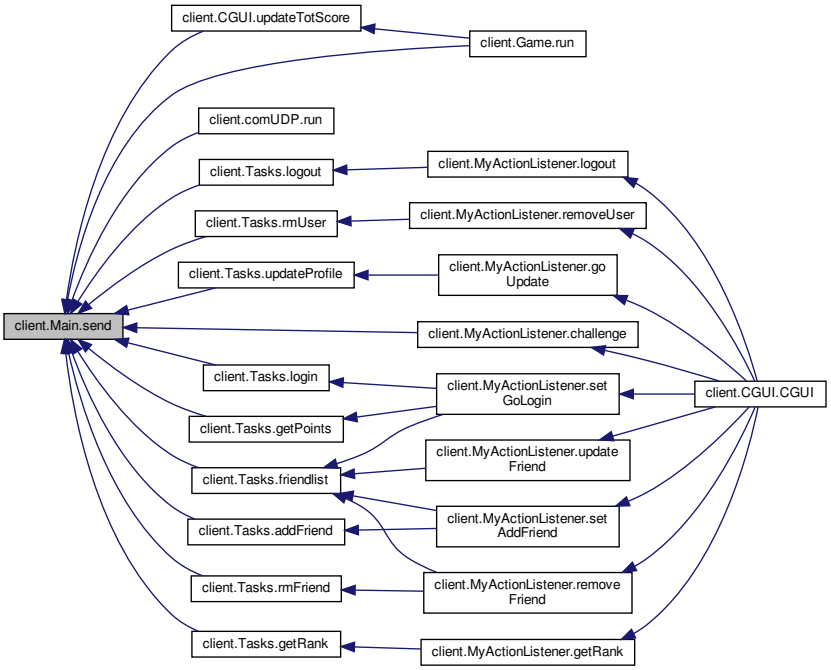
Parameters

toSend	messaggio da inviare al server

Exceptions

IOException	

Here is the caller graph for this function:



7.15.3 Member Data Documentation

7.15.3.1 inBuff

```
BufferedReader client.Main.inBuff = null  [static], [private]
```

buffer di ingresso per la socket

7.15.3.2 outBuff

```
DataOutputStream client.Main.outBuff = null  [static], [private]
```

buffer di uscita per la socket

7.15.3.3 parser

```
ConfParser client.Main.parser = null  [static]
```

parser della configurazione

7.15.3.4 reg

```
Registry client.Main.reg = null  [static], [protected]
```

registro dei serivizi del server

7.15.3.5 socket

```
Socket client.Main.socket = null  [static], [private]
```

socket con cui connettersi al server

7.15.3.6 stub

```
RegRMInterface client.Main.stub = null  [static], [protected]
```

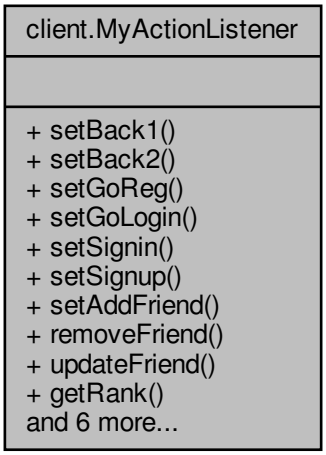
interfaccia RMI del server

The documentation for this class was generated from the following file:

- src/client/Main.java

7.16 client.MyActionListener Class Reference

Collaboration diagram for client.MyActionListener:



Static Public Member Functions

- static void setBack1 (JButton back1, JPanel loginP, JFrame mainW, JPanel firstP)
- static void setBack2 (JButton back2, JPanel signupP, JFrame mainW, JPanel firstP)
- static void setGoReg (JButton goup, JTextField nick, JPasswordField pass, JTextField name, JTextField surname)
- static void setGoLogin (JButton go, JTextField nick, JPasswordField pass, CGUI gui)
- static void setSignin (JButton signin, JFrame mainW, JPanel firstP, JPanel loginP)
- static void setSignup (JButton signup, JFrame mainW, JPanel firstP, JPanel signupP)
- static void setAddFriend (JButton addfrd, CGUI ui)
- static void removeFriend (JButton rm, CGUI ui)
- static void updateFriend (JButton update, CGUI ui)
- static void getRank (JButton rank, CGUI ui)
- static void logout (JButton logout, CGUI ui)
- static void removeUser (JButton remove, CGUI ui)
- static void update (JButton up, CGUI ui)
- static void setBack3 (JButton back3, JPanel tasks, JFrame mainW, JPanel updateU)
- static void goUpdate (JButton go, JTextField name, JTextField surname, JPasswordField pass, CGUI ui)
- static void challenge (JButton sendCh, CGUI ui)

7.16.1 Detailed Description

Classe con metodi statici utile per gestire gli input che l'utente invia attraverso la GUI

Author

Luca Canessa (Mat. 516639)

Version

1.3

Since

1.0

7.16.2 Member Function Documentation

7.16.2.1 challenge()

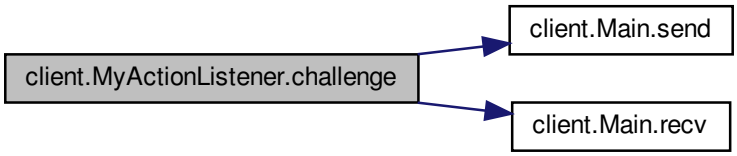
```
static void client.MyActionListener.challenge (
            JButton sendCh,
            CGUI ui )  [static]
```

Metodo predisposto per gestire l'invio della richiesta di sfida e rispettiva risposta. Crea una finestra in cui indicare l'amico da sfidare e poter richiedere la sfida al server, e nel caso di risposta positiva dall'amico fa partire il thread per la gestiore del gioco e chiude la finestra appea utilizzata

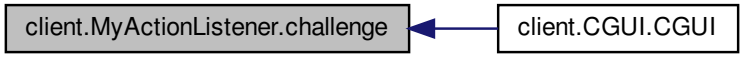
Parameters

sendCh	bottone per richiedere l'operazione di invio richiesta sfida
ui	riferimento al gestore della GUI

Here is the call graph for this function:



Here is the caller graph for this function:



7.16.2.2 getRank()

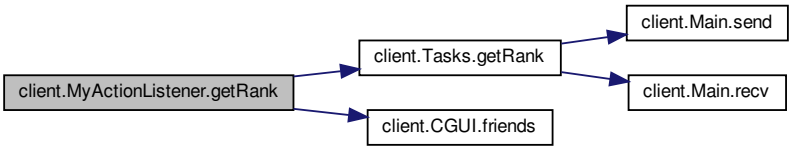
```
static void client.MyActionListener.getRank (
            JButton rank,
            CGUI ui )  [static]
```

Metodo che richiede la classifica dell'utente al server e la visualizza nella GUI

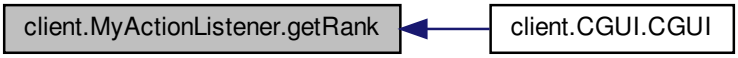
Parameters

rank	bottone per richiedere la visualizzazione della classifica
ui	riferimento al gestore della GUI

Here is the call graph for this function:



Here is the caller graph for this function:



7.16.2.3 goUpdate()

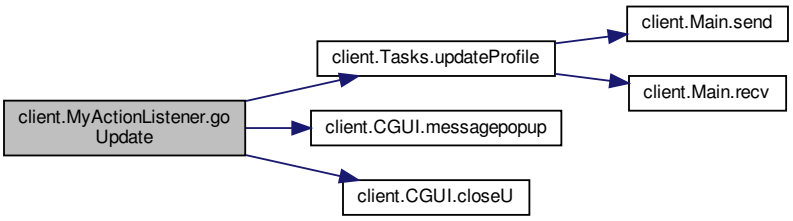
```
static void client.MyActionListener.goUpdate (
            JButton go,
            JTextField name,
            JTextField surname,
            JPasswordField pass,
            CGUI ui )  [static]
```

Metodo per la gestione dell'aggiornamento del profilo, invia al server i dati da aggiornare e visualizza un popup di sistema con il messaggio di risposta del server

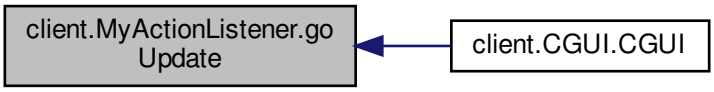
Parameters

go	bottone per richiedere l'invio dei dati al server
name	campo dove prelevare il nome dell'utente
surname	campo dove prelevare il cognome dell'utente
pass	campodove prelevare la password dell'utente
ui	riferimento al gestore della GUI

Here is the call graph for this function:



Here is the caller graph for this function:



7.16.2.4 logout()

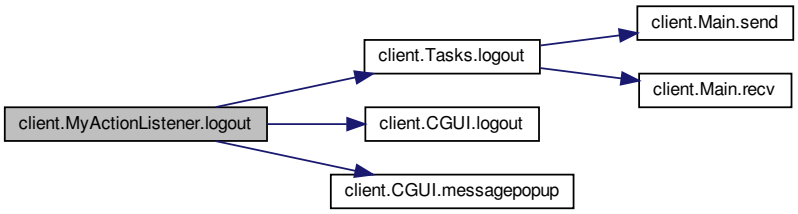
```
static void client.MyActionListener.logout (
            JButton logout,
            CGUI ui )  [static]
```

Metodo che effettua il logout quando riceve risposta positiva dal server cambiando contesto.

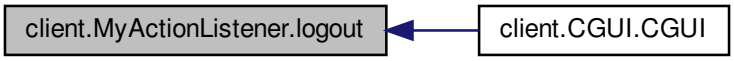
Parameters

logout	bottone per richiedere il logout dell'utente
ui	riferimento al gestore GUI

Here is the call graph for this function:



Here is the caller graph for this function:



7.16.2.5 removeFriend()

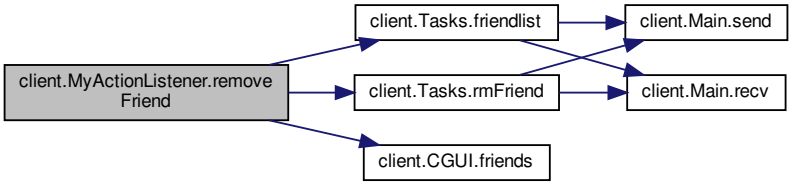
```
static void client.MyActionListener.removeFriend (
            JButton rm,
            CGUI ui )  [static]
```

Metodo per gestire la richiesta di rimozione di un amico dalla lista Per la rimozione dell'amico viene creata una nuova finestra in cui si può inserire il Nickname desiderato e inviarlo al server e visualizzare la risposta all'operazione richiesta.

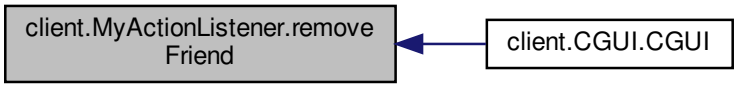
Parameters

rm	bottone che riceve l'input per la richiesta della rimozione dell'amico
ui	riferimento al gestore della GUI

Here is the call graph for this function:



Here is the caller graph for this function:



7.16.2.6 removeUser()

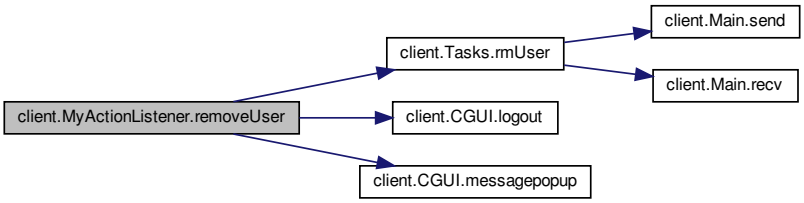
```
static void client.MyActionListener.removeUser (
            JButton remove,
            CGUI ui )  [static]
```

Metodo che si preoccupa di gestire la richiesta di rimozione dell'utente dal gioco

Parameters

remove	bottone per richiedere la cancellazione dell'utente
ui	riferimento al gestore della GUI

Here is the call graph for this function:



Here is the caller graph for this function:

7.16.2.7 setAddFriend()

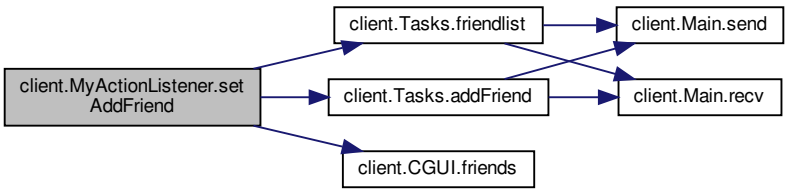
```
static void client.MyActionListener.setAddFriend (
            JButton addfrd,
            CGUI ui )  [static]
```

Metodo per gestire la richiesta di aggiunta di un amico alla lista Per l'aggiunta dell'amico viene creata una nuova finestra in cui si può inserire il Nickname desiderato e inviarlo al server e visualizzare la risposta all'operazione richiesta.

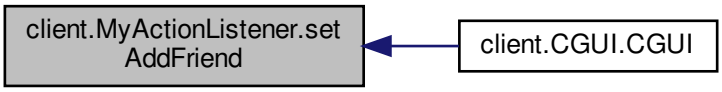
Parameters

addfrd	bottone per richiedere l'operazione di aggiunta dell'amico
ui	riferimento al gestore della GUI

Here is the call graph for this function:



Here is the caller graph for this function:



7.16.2.8 setBack1()

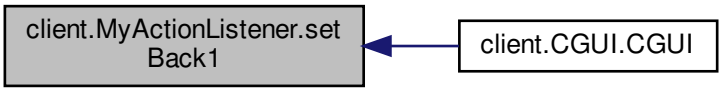
```
static void client.MyActionListener.setBack1 (
            JButton back1,
            JPanel loginP,
            JFrame mainW,
            JPanel firstP )  [static]
```

Metodo che effettua il cambio di scenario da pannello di Login a pannello con il menù principale

Parameters

back1	bottone che riceve l'input
loginP	pannello di login
mainW	finestra principale su cui cambiare pannelli
firstP	pannello con menù iniziale

Here is the caller graph for this function:



7.16.2.9 setBack2()

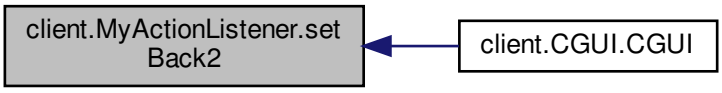
```
static void client.MyActionListener.setBack2 (
            JButton back2,
            JPanel signupP,
            JFrame mainW,
            JPanel firstP )  [static]
```

Metodo che effettua il cambio di scenario da pannello di registrazione a pannello con il menù principale

Parameters

back2	bottone che riceve input
signupP	pannello di registrazione dell'utente
mainW	finestra principale
firstP	pannello con menù principale

Here is the caller graph for this function:



7.16.2.10 setBack3()

```
static void client.MyActionListener.setBack3 (
            JButton back3,
            JPanel tasks,
```

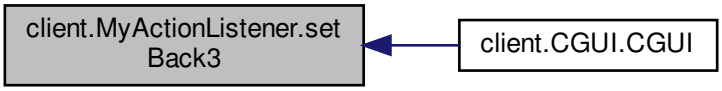
```
            JFrame mainW,
            JPanel updateU )  [static]
```

Metodo che effettua il cambio di scenario da pannello di aggiornamento del profilo a pannello con i tasks possibili e richiedibili

Parameters

back3	bottone per richiedere il cambio di contesto
tasks	pannello con i tasks possibili
mainW	finestra su cui effettuare il cambio contesto
updateU	pannello per l'aggiornamento profilo

Here is the caller graph for this function:



7.16.2.11 setGoLogin()

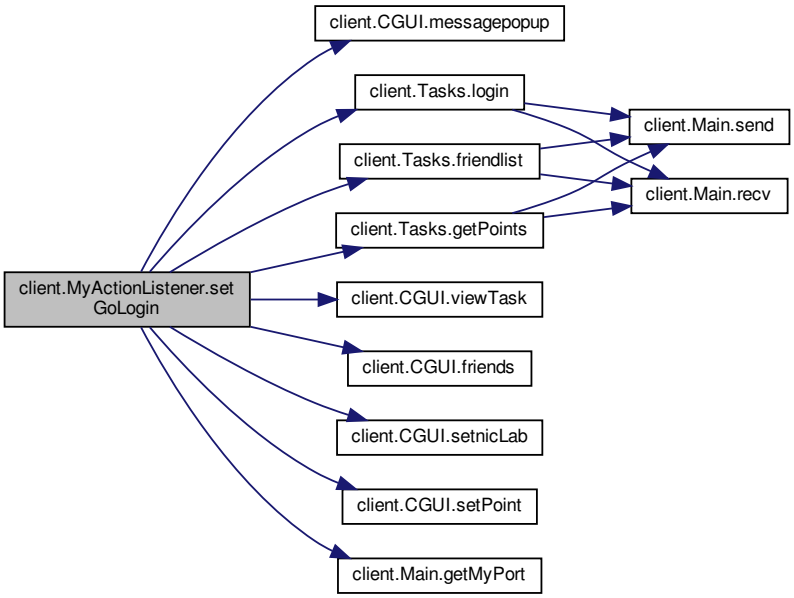
```
static void client.MyActionListener.setGoLogin (
            JButton go,
            JTextField nick,
            JPasswordField pass,
            CGUI gui )  [static]
```

Metodo per effettuare il login dell'utente e visualizzare i tasks possibili e richiedibili al server

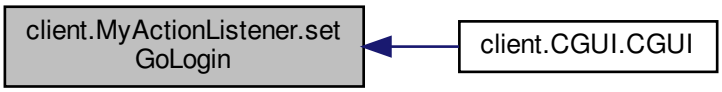
Parameters

go	bottone che riceve l'input da gestire
nick	nickname dell'utente
pass	password dell'utente
gui	riferimento al gestore della GUI

Here is the call graph for this function:



Here is the caller graph for this function:



7.16.2.12 setGoReg()

```
static void client.MyActionListener.setGoReg (
            JButton goup,
            JTextField nick,
            JPasswordField pass,
            JTextField name,
            JTextField surname )  [static]
```

Metodo predisosto alla registrazione dell'utente

Parameters

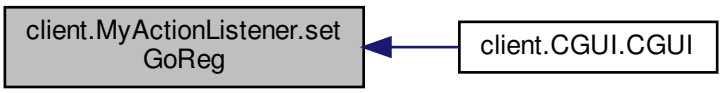
goup	bottone che riceve l'input da gestire

Parameters

nick	nickname dell'utente
pass	password dell'utente
name	nome dell'utente
surname	cognome dell'utente

Here is the call graph for this function:

Here is the caller graph for this function:



7.16.2.13 setSignin()

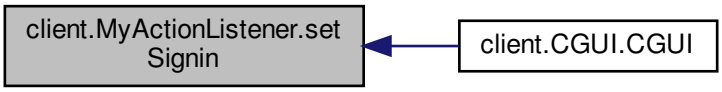
```
static void client.MyActionListener.setSignin (
            JButton signin,
            JFrame mainW,
            JPanel firstP,
            JPanel loginP )  [static]
```

Metodo per effettuare il cambio scenario da pannello principale a pannello per effettuare il login

Parameters

signin	bottone che riceve l'input da gestire
mainW	finestra principale su gestire gli scenari
firstP	pannello del menù principale
loginP	pannello per eseguire il login

Here is the caller graph for this function:



7.16.2.14 setSignup()

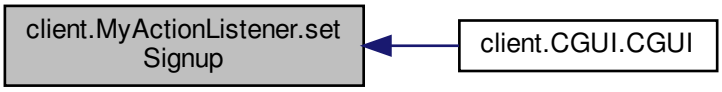
```
static void client.MyActionListener.setSignup (
            JButton signup,
            JFrame mainW,
            JPanel firstP,
            JPanel signupP )  [static]
```

Metodo per effettuare il cambio scenario da pannello principale a pannello per effettuare la registrazione utente

Parameters

signup	bottone che riceve l'input
mainW	finestra principale su cui eseguire lo scenario
firstP	pannello del menù principale
signupP	pannello per effettuare la registrazione

Here is the caller graph for this function:



7.16.2.15 update()

```
static void client.MyActionListener.update (
            JButton up,
            CGUI ui )  [static]
```

Metodo incaricato di gestire la richiesta di aggiornamento del profilo dell'utente

Parameters

up	bottone per richiedere l'aggiornamento del profilo
ui	riferimento al gestore della GUI

Here is the caller graph for this function:

7.16.2.16 updateFriend()

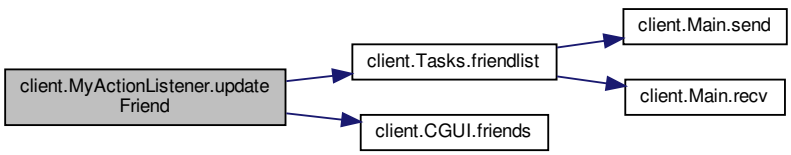
```
static void client.MyActionListener.updateFriend (
            JButton update,
            CGUI ui )  [static]
```

Metodo che richiede la lista delle amicizie al server quando l'utente preme il bottone e aggiorna la lista visualizzata.

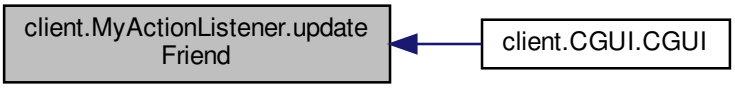
Parameters

update	bottone per richiedere l'aggiornamento lista
ui	riferimento al gestore della GUI

Here is the call graph for this function:



Here is the caller graph for this function:

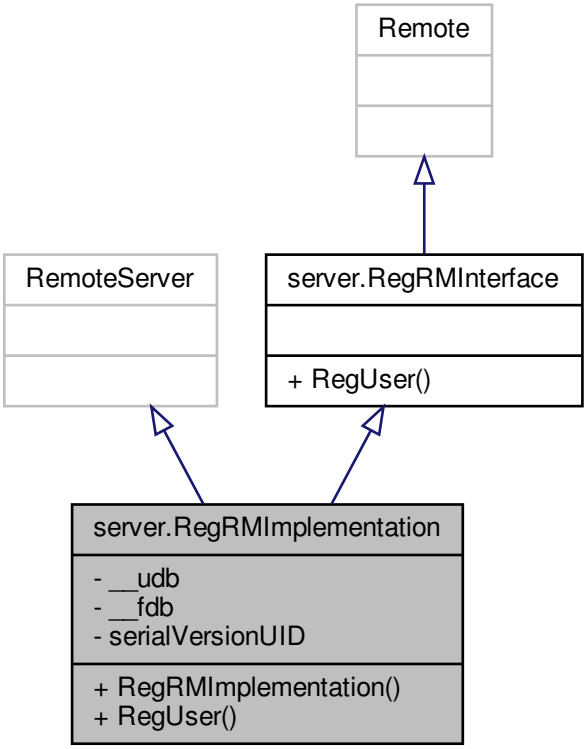


The documentation for this class was generated from the following file:

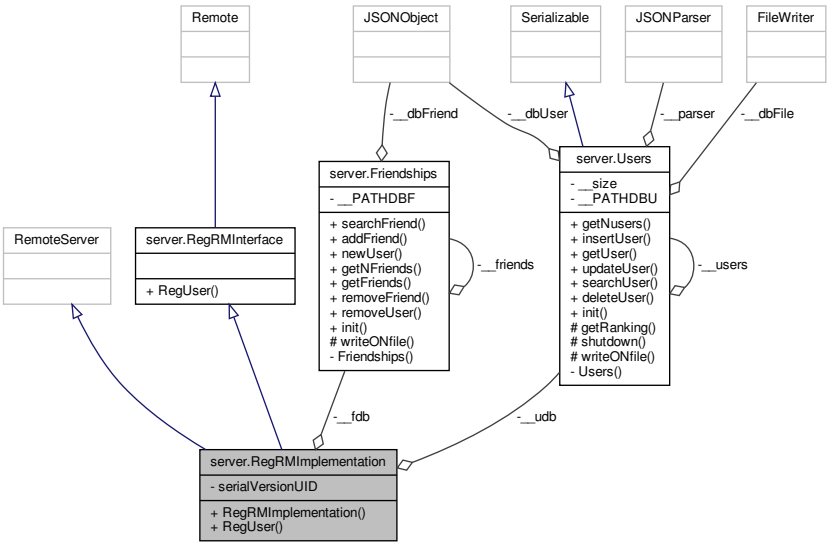
- src/client/MyActionListener.java

7.17 server.RegRMImplementation Class Reference

Inheritance diagram for server.RegRMImplementation:



Collaboration diagram for server.RegRMImplementation:



Public Member Functions

- RegRMImplementation (Users usrs, Friendships frd)
- ACK RegUser (User u) throws RemoteException

Private Attributes

- Users __udb

  database degli utenti
- Friendships __fdb

  database delle amicizie

Static Private Attributes

- static final long serialVersionUID = 1L

  variabile necessaria per la serializzazione dell'oggetto

7.17.1 Detailed Description

In questa classe viene implementata e gestita la Remote Method Invocation usata dai client per effettuare la registrazione dell'User. Utilizza il metodo inserUser della classe Users per inserire l'utente

Author

Luca Canessa (Mat. 516639)

Version

1.1

Since

1.0

7.17.2 Constructor & Destructor Documentation

7.17.2.1 RegRMImplementation()

```
server.RegRMImplementation.RegRMImplementation (
            Users usrs,
            Friendships frd )
```

Costruttore della RMI necessaria per permettere la registrazione di un utente

Parameters

usrs	database degli User
frd	database delle Friendships

7.17.3 Member Function Documentation

7.17.3.1 RegUser()

```
ACK server.RegRMImplementation.RegUser (
            User u ) throws RemoteException
```

Implementazione del metodo usato per registrare l'User. Tale metodo si basa sul metodo all'interno della classe Users predisposto per questo utilizzo. Ritorna un valore di ACK.

Parameters

u	User da inserire del database

Returns

ACK (dipende da Users.insertUser())

Exceptions

RemoteException	

Here is the call graph for this function:

7.17.4 Member Data Documentation

7.17.4.1 __fdb

```
Friendships server.RegRMImplementation.__fdb  [private]
```

database delle amicizie

7.17.4.2 __udb

```
Users server.RegRMImplementation.__udb  [private]
```

database degli utenti

7.17.4.3 serialVersionUID

```
final long server.RegRMImplementation.serialVersionUID = 1L  [static], [private]
```

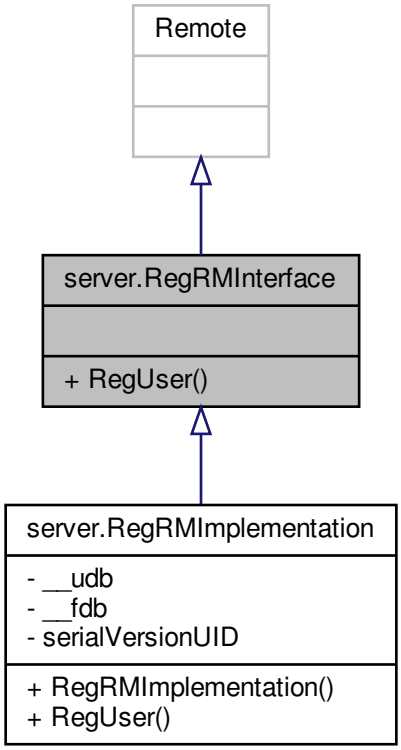
variabile necessaria per la serializzazione dell'oggetto

The documentation for this class was generated from the following file:

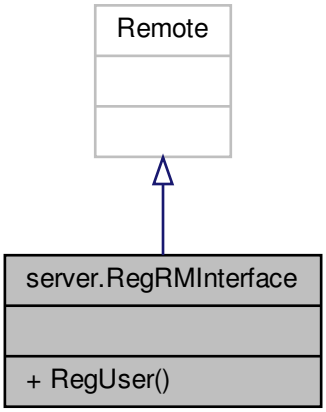
- src/server/RegRMImplementation.java

7.18 server.RegRMInterface Class Reference

Inheritance diagram for server.RegRMInterface:



Collaboration diagram for server.RegRMInterface:



Public Member Functions

- ACK RegUser (User u) throws RemoteException

7.18.1 Detailed Description

In questa interfaccia viene usata per poter usare la RMI per registrare gli User.

Author

Luca Canessa (Mat. 516639)

Version

1.1

Since

1.0

7.18.2 Member Function Documentation

7.18.2.1 RegUser()

```
ACK server.RegRMInterface.RegUser (
            User u ) throws RemoteException
```

Metodo dichiarato per registrare l'utente. Ritorna un valore di ACK

Parameters

u	User da registrare

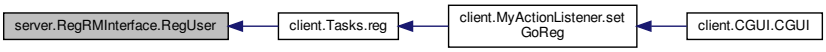
Returns

risultato dell'operazione di registrazione

Exceptions

RemoteException	

Here is the caller graph for this function:

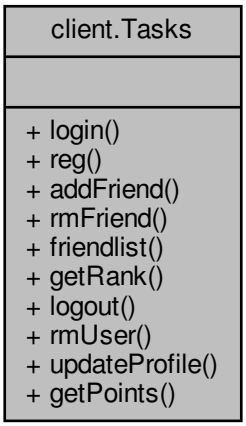


The documentation for this class was generated from the following file:

- src/server/RegRMInterface.java

7.19 client.Tasks Class Reference

Collaboration diagram for client.Tasks:



Static Public Member Functions

- static String login (String nickname, String Password)
- static void reg (String nickin, String passin, String name, String surname)
- static String addFriend (String name)
- static String rmFriend (String name)
- static String [ ] friendlist ()
- static String [ ] getRank ()
- static String logout ()
- static String rmUser ()
- static String updateProfile (String name, String surname, String password)
- static String getPoints ()

7.19.1 Detailed Description

Classe con metodi statici necessari per il compimento dei tasks richiesti dal client

Author

Luca Canessa (Mat. 516639)

Version

1.2

Since

1.0

7.19.2 Member Function Documentation

7.19.2.1 addFriend()

```
static String client.Tasks.addFriend (
            String name )  [static]
```

Metodo per la richiesta di aggiunta di un amico nella propria lista

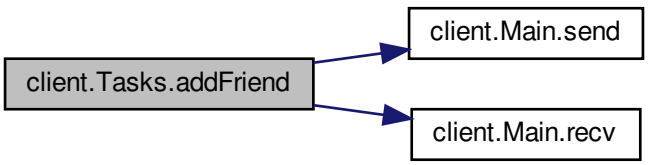
Parameters

name	nickname dell'amico

Returns

ret risultato dell'operazione

Here is the call graph for this function:



Here is the caller graph for this function:

7.19.2.2 friendlist()

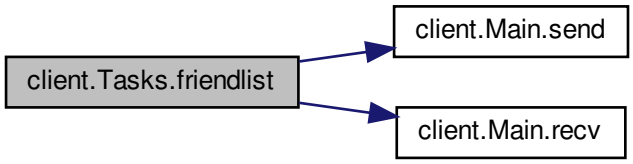
```
static String [] client.Tasks.friendlist ( )  [static]
```

Richiede al server la lista dei propri amici

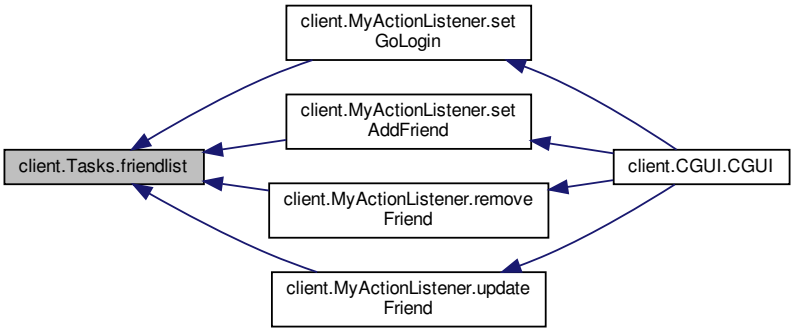
Returns

una array con i nickname degli amici

Here is the call graph for this function:



Here is the caller graph for this function:



7.19.2.3 getPoints()

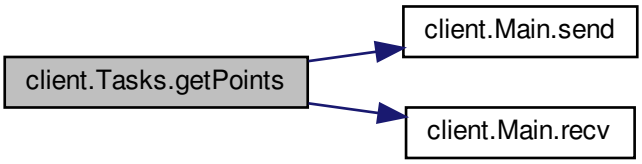
```
static String client.Tasks.getPoints ( )  [static]
```

Richiede al server il punteggio totalizzato fino a quel momento

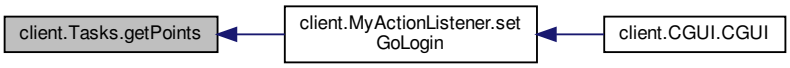
Returns

point il numero di punti guadagnati

Here is the call graph for this function:



Here is the caller graph for this function:



7.19.2.4 getRank()

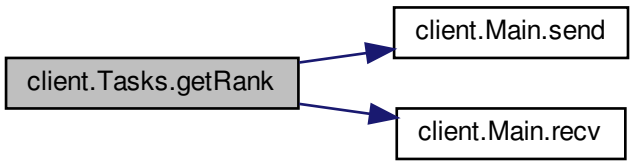
```
static String [] client.Tasks.getRank ( )  [static]
```

Richiede al server la ranking list personale.

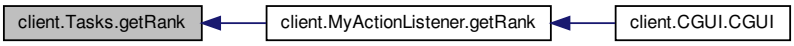
Returns

array con i nomi e il punteggio degli amici

Here is the call graph for this function:



Here is the caller graph for this function:



7.19.2.5 login()

```
static String client.Tasks.login (
            String nickname,
            String Password )  [static]
```

Metodo con il compito di richiedere al server di effettuare il login all'utente, inviandogli il suo nickname e la password corrispondente. Ritorna la rispota del server

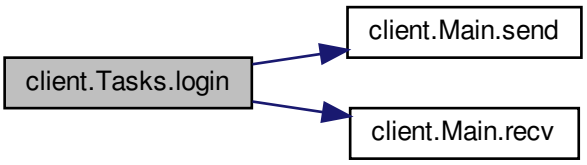
Parameters

nickname	nickname dell'utente
Password	password dell'utente

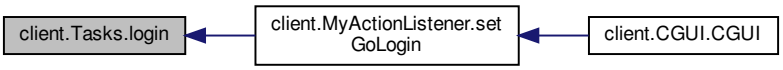
Returns

rec risposta del server all'esecuzione dell'operazione

Here is the call graph for this function:



Here is the caller graph for this function:



7.19.2.6 logout()

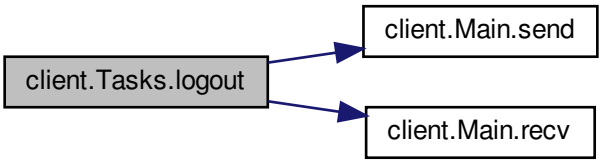
```
static String client.Tasks.logout ( )  [static]
```

Richiede al server di effettuare il logout per l'utente

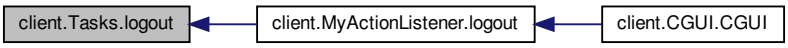
Returns

rec il risultato dell'operazione

Here is the call graph for this function:



Here is the caller graph for this function:



7.19.2.7 reg()

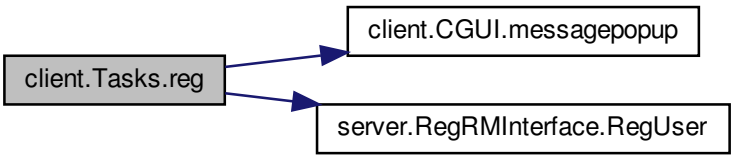
```
static void client.Tasks.reg (
            String nickin,
            String passin,
            String name,
            String surname )  [static]
```

Richiesta di registrazione di un utente attraverso il RemoteMethodInvocation. Crea popup di sistema con messaggio del risultato dell'operazione (nickname e password OBBLIGATORI)

Parameters

nickin	nickname dell'utente
passin	password dell'utente
name	nome dell'utente
surname	cognome dell'utente

Here is the call graph for this function:



Here is the caller graph for this function:

7.19.2.8 rmFriend()

```
static String client.Tasks.rmFriend (
            String name )  [static]
```

Metodo che richiede al server la rimozione di un amico dalla propria lista

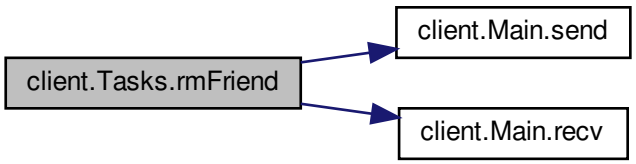
Parameters

name	nickname dell'amico

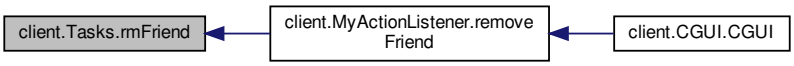
Returns

ret risultato dell'operazione

Here is the call graph for this function:



Here is the caller graph for this function:



7.19.2.9 rmUser()

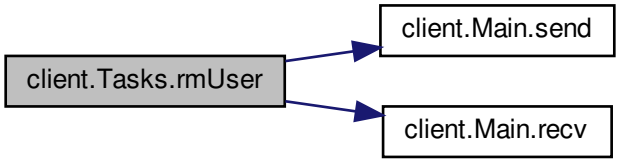
```
static String client.Tasks.rmUser ( )  [static]
```

Richiede al server la rimozione dell'utente dal gioco

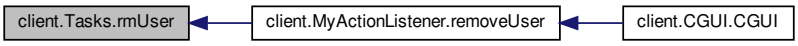
Returns

rec risultato dell'operazione

Here is the call graph for this function:



Here is the caller graph for this function:



7.19.2.10 updateProfile()

```
static String client.Tasks.updateProfile (
            String name,
            String surname,
            String password )  [static]
```

Richiede al server l'aggiornamento del profilo personale. Stampa il risultato dell'operazione su un popup di sistema

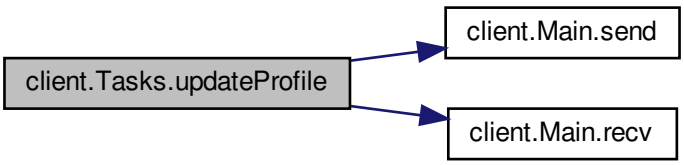
Parameters

name	nome dell'utente
surname	cognome dell'utente
password	password dell'utente

Returns

rec risultato dell'operazione

Here is the call graph for this function:



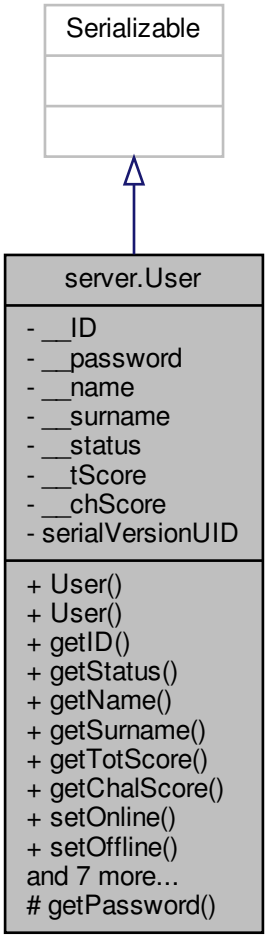
Here is the caller graph for this function:

The documentation for this class was generated from the following file:

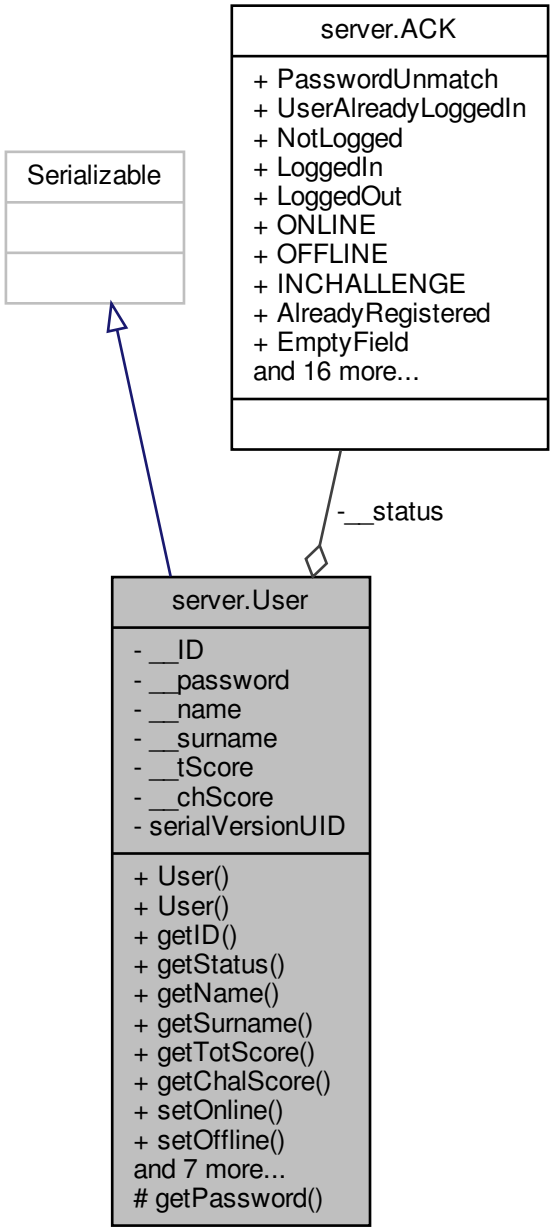
- src/client/Tasks.java

7.20 server.User Class Reference

Inheritance diagram for server.User:



Collaboration diagram for server.User:



Public Member Functions

- User (String Username, String Password)
- User (String Username, String Password, String Name, String Surname)
- String getID ()
- ACK getStatus ()
- String getName ()
- String getSurname ()

- int getTotScore ()
- int getChalScore ()
- void setOnline ()
- void setOffline ()
- void setInChallenge ()
- void setTScore (int Score)
- void setCScore (int Score)
- void resetName (String Name)
- void resetSurname (String Surname)
- void resetPassword (String Password)
- boolean pswIsEqual (String Password)

Protected Member Functions

- String getPassword ()

Private Attributes

- String __ID
  nickname dell'utente - ID del database
- String __password
  password dell'utente
- String __name
  nome dell'utente
- String __surname
  cognome dell'utente
- ACK __status
  stato sul server dell'utente
- int __tScore
  numero di punti totalizzati dall'utente
- int __chScore
  numero di punti ottenuti durante l'ultima parita dall'utente

Static Private Attributes

- static final long serialVersionUID = 1L
  variabile necessaria per rendere l'oggetto serializzabile

7.20.1 Detailed Description

In questo file vengono implementati i metodi e le variabili necessari per modellare e gestire la classe 'User' che rappresenta il client. L'User è caratterizzato da proprietà che possono essere modificate una volta che l'User è creato. L'User è identificato UNIVOCAMENTE dal suo nickname che lo distingue da tutti gli altri User

Author

Luca Canessa (Mat. 516639)

Version

1.4

Since

1.0

7.20.2 Constructor & Destructor Documentation

7.20.2.1 User() [1/2]

```
server.User.User (
            String Username,
            String Password )
```

Costruttore a due parametri che rappresentano codice univoco dell'utente e la sua password. Le altre variabili sono impostate con valore di default.

Parameters

Username	codice univoco (nickname) dell'utente
Password	password di accesso dell'utente

7.20.2.2 User() [2/2]

```
server.User.User (
            String Username,
            String Password,
            String Name,
            String Surname )
```

Costuttore a quattro parametri che rappresentano rispettivamente il codice univoco dell'utente, la sua password, il nome e il cognome le altre variabili sono settate al valore di default

Parameters

Username	codice univoco (nickname) dell'utente
Password	password di accesso dell'utente
Name	nome dell'utente
Surname	cognome dell'utente

7.20.3 Member Function Documentation

7.20.3.1 getChalScore()

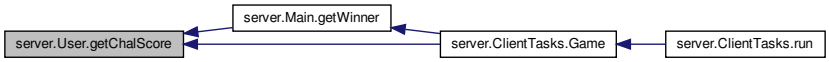
```
int server.User.getChalScore ( )
```

Restituisce il valore del punteggio guadagnato durante l'ultima partita

Returns

punteggio della partita

Here is the caller graph for this function:



7.20.3.2 getID()

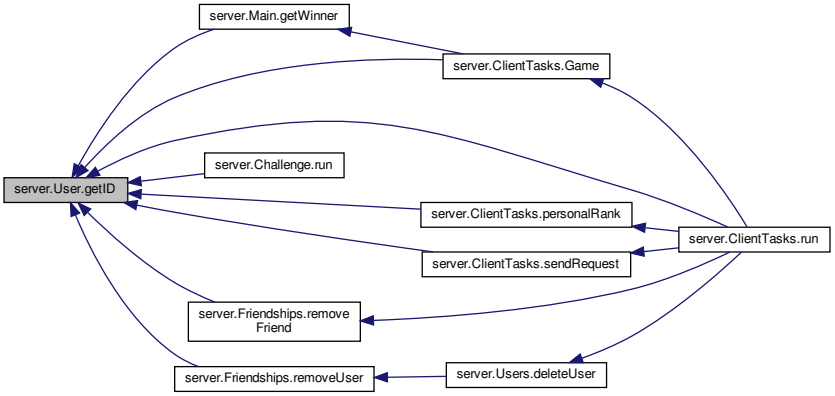
```
String server.User.getID ( )
```

Restituisce il nickname dell'User

Returns

il codice univoco (nickname) come String

Here is the caller graph for this function:



7.20.3.3 getName()

```
String server.User.getName ( )
```

Restutuisce il nome dell'utente se è stato impostato, null altrimenti

Returns

il nome dell'utente

7.20.3.4 getPassword()

```
String server.User.getPassword ( )  [protected]
```

Restituisce la password dell'utente

Returns

la password dell'utente

7.20.3.5 getStatus()

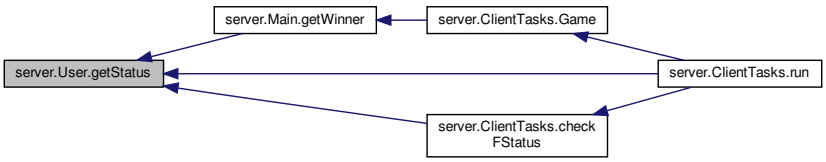
```
ACK server.User.getStatus ( )
```

Restituisce lo stato attuale dell'utente come tipo UStatus, il valore di default è OFFLINE

Returns

lo stato dell'utente

Here is the caller graph for this function:



7.20.3.6 getSurname()

```
String server.User.getSurname ( )
```

Restituisce il cognome dell'utente se impostato, null altrimenti

Returns

il cognome dell'utente

7.20.3.7 getTotScore()

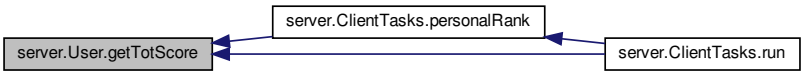
```
int server.User.getTotScore ( )
```

Restituisce il valore del punteggio attuale totale dell'utente

Returns

punteggio totale

Here is the caller graph for this function:



7.20.3.8 pswIsEqual()

```
boolean server.User.pswIsEqual (
            String Password )
```

Compara la password passatagli con quella impostata dall'utente e ne ritorna valore

Parameters

Password	password da controllare

Returns

true se le password coincidono, false altrimenti

Here is the caller graph for this function:

7.20.3.9 resetName()

```
void server.User.resetName (
            String Name )
```

Imposta il nome dell'utente

Parameters

Name	nome dell'utente

Here is the caller graph for this function:

7.20.3.10 resetPassword()

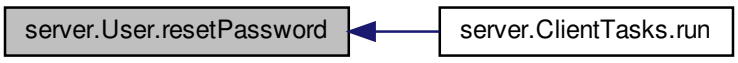
```
void server.User.resetPassword (
            String Password )
```

Reimposta password

Parameters

Password	nuova password

Here is the caller graph for this function:



7.20.3.11 resetSurname()

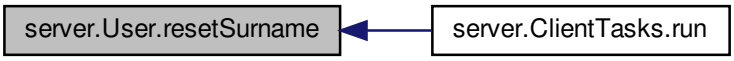
```
void server.User.resetSurname (
            String Surname )
```

Imposta il cognome dell'utente

Parameters

Surname	cognome dell'utente

Here is the caller graph for this function:



7.20.3.12 setCScore()

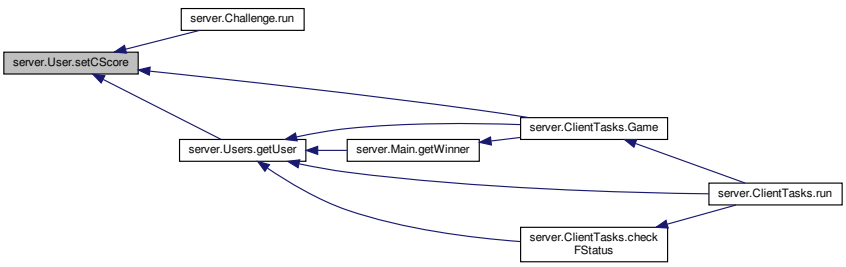
```
void server.User.setCScore (
            int Score )
```

Imposta il valore dei punti guadagnati nell'ultima partita

Parameters

Score	valore ottenuto dall'ultima partita

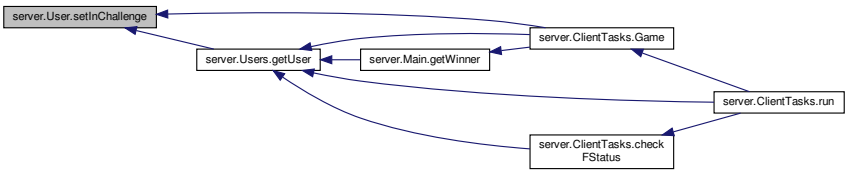
Here is the caller graph for this function:



7.20.3.13 setInChallenge()

```
void server.User.setInChallenge ( )
```

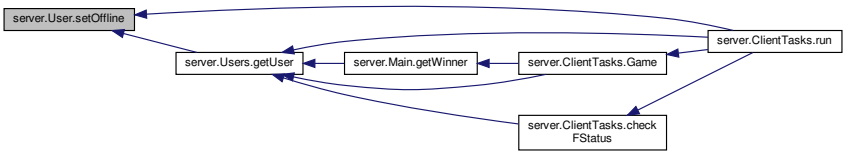
Imposta lo stato dell'utente come INCHALLENGE Here is the caller graph for this function:



7.20.3.14 setOffline()

```
void server.User.setOffline ( )
```

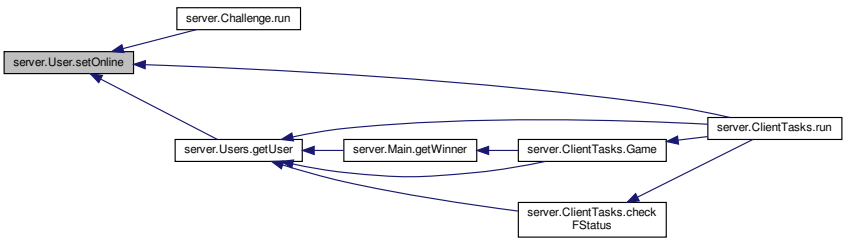
Imposta lo stato dell'utente come OFFLINE Here is the caller graph for this function:



7.20.3.15 setOnline()

```
void server.User.setOnline ( )
```

Imposta lo stato dell'utente come ONLINE Here is the caller graph for this function:



7.20.3.16 setTScore()

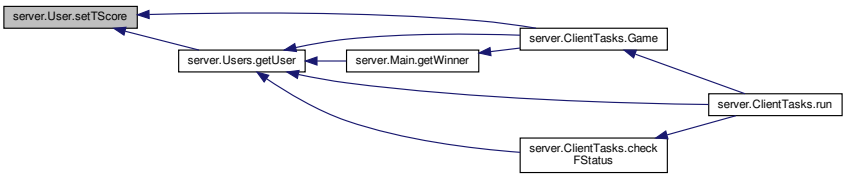
```
void server.User.setTScore (
            int Score )
```

Imposta il valore dei punti guadagnati aggiornando il valore il punteggio totale dell'utente

Parameters

Score	valore punteggio

Here is the caller graph for this function:



7.20.4 Member Data Documentation

7.20.4.1 __chScore

```
int server.User.__chScore  [private]
```

numero di punti ottenuti durante l'ultima parita dall'utente

7.20.4.2 __ID

```
String server.User.__ID  [private]
```

nickname dell'utente - ID del database

7.20.4.3 __name

```
String server.User.__name  [private]
```

nome dell'utente

7.20.4.4 __password

```
String server.User.__password  [private]
```

password dell'utente

7.20.4.5 __status

```
ACK server.User.__status  [private]
```

stato sul server dell'utente

7.20.4.6 __surname

```
String server.User.__surname  [private]
```

cognome dell'utente

7.20.4.7 __tScore

```
int server.User.__tScore  [private]
```

numero di punti totalizzati dall'utente

7.20.4.8 serialVersionUID

```
final long server.User.serialVersionUID = 1L  [static], [private]
```

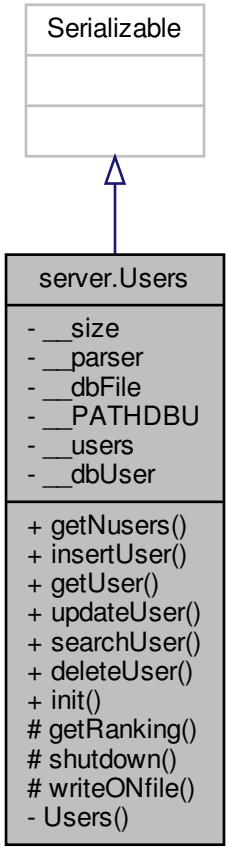
variabile necessaria per rendere l'oggetto serializzabile

The documentation for this class was generated from the following file:

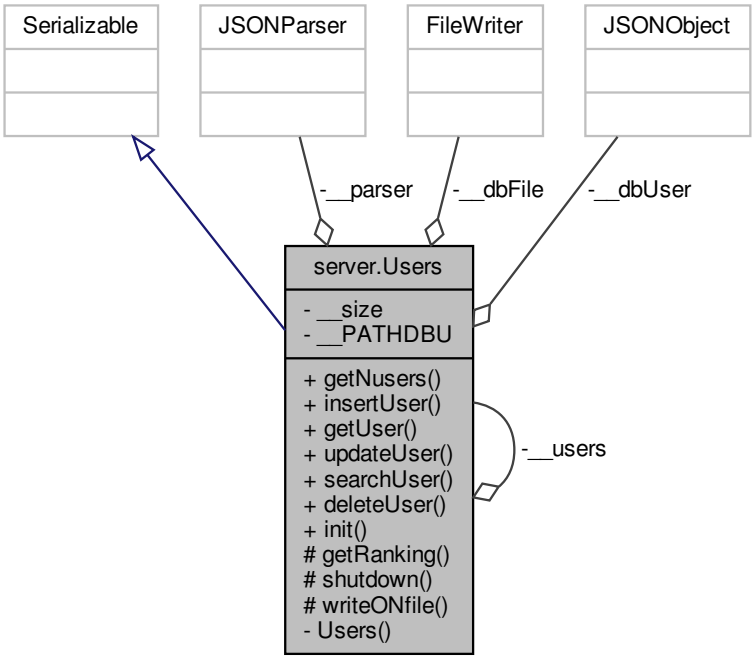
- src/server/User.java

7.21 server.Users Class Reference

Inheritance diagram for server.Users:



Collaboration diagram for server.Users:



Public Member Functions

- synchronized int getNusers ()
- synchronized ACK insertUser (User usr, Friendships f)
- synchronized User getUser (String nickname)
- synchronized ACK updateUser (User usr)
- synchronized ACK searchUser (String nickname)
- synchronized ACK deleteUser (String nickname, Friendships f)

Static Public Member Functions

- static Users init ()

Protected Member Functions

- HashMap< String, Integer > getRanking ()
- synchronized void shutdown ()
- synchronized void writeONfile ()

Private Member Functions

- Users ()

Private Attributes

- int __size = -1
  numero di utenti iscritti
- JSONParser __parser = null
  file parser
- FileWriter __dbFile = null
  scrittore del db sul file '__PATHDBU'

Static Private Attributes

- static String __PATHDBU = null
  path al file del db utent
- static Users __users = null
  singleton
- static JSONObject __dbUser = null
  il db reale

7.21.1 Detailed Description

In questo file vengono implementati i metodi e le variabili per modellare il database degli utenti usando come modello dell'utente la classe 'user'

Author

Luca Canessa (Mat. 516639)

Version

1.5

Since

1.0

7.21.2 Constructor & Destructor Documentation

7.21.2.1 Users()

```
server.Users.Users ( )  [private]
```

Costruttore del databse degli utenti. Implementato come singoletto per evitare problemi di copie inconsistenti e/o ridondanze.

7.21.3 Member Function Documentation

7.21.3.1 deleteUser()

```
synchronized ACK server.Users.deleteUser (
            String nickname,
            Friendships f )
```

Elimina un User dal db (se possibile)

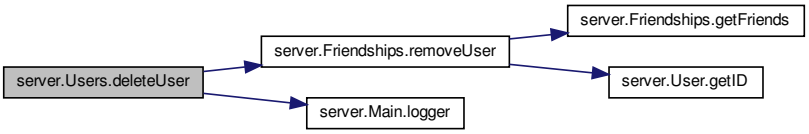
Parameters

nickname	stringa del nickname dell'User da rimuovere

Returns

UserDeleted se l'operazione è riuscita UserNotDeleted se l'operazione non ha terminato con successo

Here is the call graph for this function:



Here is the caller graph for this function:

7.21.3.2 getNusers()

```
synchronized int server.Users.getNusers ( )
```

Restituisce il numero di oggetti User all'interno del database

Returns

il numero degli utenti iscritti

7.21.3.3 getRanking()

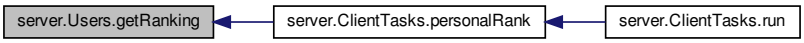
```
HashMap<String, Integer> server.Users.getRanking ( )  [protected]
```

Restituisce la ranking list del gioco ordinata per valore di punti in ordine decrescente di tutti gli utenti

Returns

una HashMap con i nickname e i punti degli utenti una HashMap vuota nel caso non vi siano utenti

Here is the caller graph for this function:



7.21.3.4 getUser()

```
synchronized User server.Users.getUser (
            String nickname )
```

Restituisce l'oggetto User se presente all'interno del db

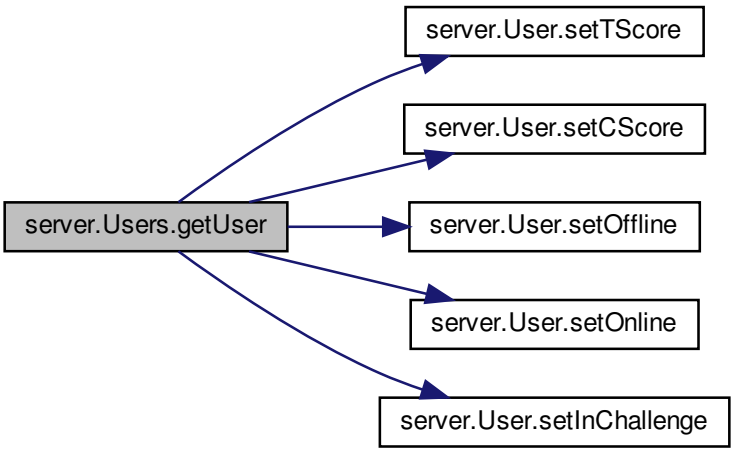
Parameters

nickname	stringa del nickname dell'User da estrapolare

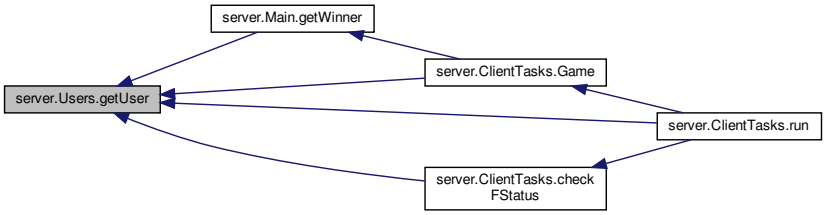
Returns

l'oggetto User se presente, null altrimenti

Here is the call graph for this function:



Here is the caller graph for this function:



7.21.3.5 init()

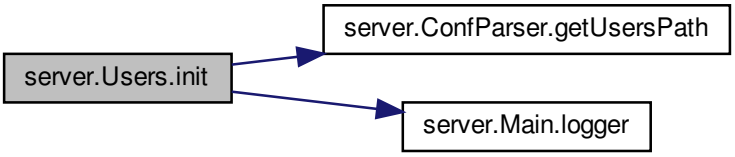
```
static Users server.Users.init ( )  [static]
```

Inizializzatore del database. Si interfaccia direttamente con il costruttore della classe

Returns

oggetto che rappresenta il database

Here is the call graph for this function:



Here is the caller graph for this function:

7.21.3.6 insertUser()

```
synchronized ACK server.Users.insertUser (
             User usr,
             Friendships f )
```

Inserisce l'utente all'interno del database estrapolando i dati dall'oggetto User passatogli restituendo un valore ACK

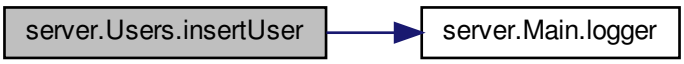
Parameters

usr	utente da inserire

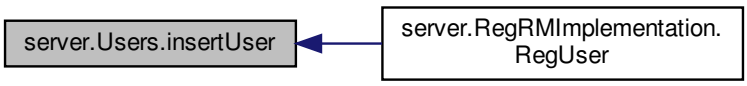
Returns

UserRegistered se l'User è stato iscritto UserFound se l'User è già presente

Here is the call graph for this function:



Here is the caller graph for this function:



7.21.3.7 searchUser()

```
synchronized ACK server.Users.searchUser (
            String nickname )
```

Cerca all'interno del db se esiste l'User con il nickname 'nickname'

Parameters

nickname	stringa del nickname dell'User da cercare

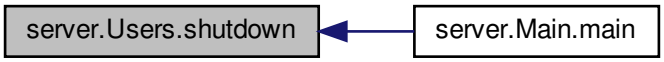
Returns

UserFound se l'User è all'interno del db UserNotFound se l'User non è stato trovato

7.21.3.8 shutdown()

```
synchronized void server.Users.shutdown ( )  [protected]
```

Here is the caller graph for this function:



7.21.3.9 updateUser()

```
synchronized ACK server.Users.updateUser (
            User usr )
```

Aggiorana tutti i valori dell'User nel database e ritorna l'esito dell'operazione come valore ACK

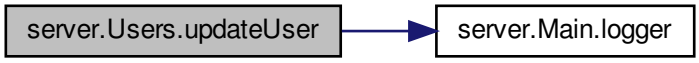
Parameters

usr	ogetto User di cui si vogliono aggiornare i valori

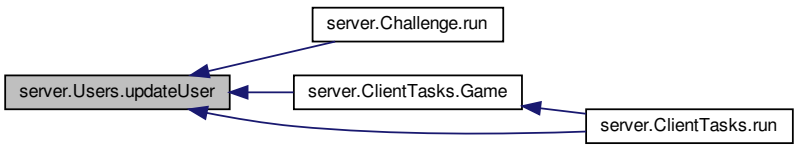
Returns

OK se l'operazione è riuscita UserNotFound se l'operazione non è terminata

Here is the call graph for this function:



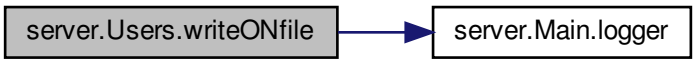
Here is the caller graph for this function:



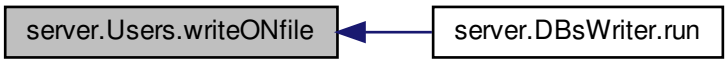
7.21.3.10 writeONfile()

```
synchronized void server.Users.writeONfile ( )  [protected]
```

Scrive sul file il db degli utenti Here is the call graph for this function:



Here is the caller graph for this function:



7.21.4 Member Data Documentation

7.21.4.1 __dbFile

```
FileWriter server.Users.__dbFile = null  [private]
```

scrittore del db sul file '__PATHDBU'

7.21.4.2 __dbUser

```
JSONObject server.Users.__dbUser = null  [static], [private]
```

il db reale

7.21.4.3 __parser

```
JSONParser server.Users.__parser = null  [private]
```

file parser

7.21.4.4 __PATHDBU

```
String server.Users.__PATHDBU = null  [static], [private]
```

path al file del db utent

7.21.4.5 __size

```
int server.Users.__size = -1  [private]
```

numero di utenti iscritti

7.21.4.6 __users

```
Users server.Users.__users = null  [static], [private]
```

singleton

The documentation for this class was generated from the following file:

- src/server/Users.java

8 File Documentation

8.1 resources/doxyhome.md File Reference

8.2 src/client/ACK.java File Reference

Classes

- enum client.ACK

Packages

- package client

8.3 src/server/ACK.java File Reference

Classes

- enum server.ACK

Packages

- package server

8.4 src/client/CGUI.java File Reference

Classes

- class client.CGUI

Packages

- package client

8.5 src/client/comUDP.java File Reference

Classes

- class client.comUDP

Packages

- package client

8.6 src/client/ConfParser.java File Reference

Classes

- class client.ConfParser

Packages

- package client

8.7 src/server/ConfParser.java File Reference

Classes

- class server.ConfParser

Packages

- package server

8.8 src/client/Game.java File Reference

Classes

- class client.Game

Packages

- package client

8.9 src/client/Main.java File Reference

Classes

- enum client.ClientMSG
- class client.Main

Packages

- package client

8.10 src/server/Main.java File Reference

Classes

- class server.DBsWriter
- class server.Main

Packages

- package server

8.11 src/client/MyActionListener.java File Reference

Classes

- class client.MyActionListener

Packages

- package client

8.12 src/client/Tasks.java File Reference

Classes

- class client.Tasks

Packages

- package client

8.13 src/server/Challenge.java File Reference

Classes

- class server.Challenge

Packages

- package server

8.14 src/server/ClientTasks.java File Reference

Classes

- enum server.ClientMSG
- class server.ClientTasks

Packages

- package server

8.15 src/server/Friendships.java File Reference

Classes

- class server.Friendships

Packages

- package server

8.16 src/server/RegRMImplementation.java File Reference

Classes

- class server.RegRMImplementation

Packages

- package server

8.17 src/server/RegRMInterface.java File Reference

Classes

- class server.RegRMInterface

Packages

- package server

8.18 src/server/User.java File Reference

Classes

- class server.User

Packages

- package server

8.19 src/server/Users.java File Reference

Classes

- class server.Users

Packages

- package server

Index

$ $getFont
    client::CGUI, 23
$ $setupUI
    client::CGUI, 23
__ID
    server::User, 150
__PATHDBF
    server::Friendships, 90
__PATHDBU
    server::Users, 161
__cSocket
    server::ClientTasks, 55
__chScore
    server::User, 150
__conf
    client::ConfParser, 67
    server::ConfParser, 77
__dbFile
    server::Users, 161
__dbFriend
    server::Friendships, 90
__dbUser
    server::Users, 161
__end
    server::ClientTasks, 55
__fdb
    server::ClientTasks, 55
    server::DBsWriter, 80
    server::RegRMImplementation, 127
__friends
    server::Friendships, 90
__inBuff
    server::Challenge, 39
    server::ClientTasks, 56
__me
    server::ClientTasks, 56
__name
    server::User, 150
__outBuff
    server::Challenge, 39
    server::ClientTasks, 56
__parser
    server::Users, 161
__password
    server::User, 150
__size
    server::Users, 162
__status
    server::User, 150
__surname
    server::User, 151
__tScore
    server::User, 151
__udb
    server::Challenge, 39
    server::ClientTasks, 56
    server::DBsWriter, 80
    server::RegRMImplementation, 127
__user
    server::Challenge, 39
__users
    server::Users, 162
__usrT
    server::Challenge, 39
__words
    server::Challenge, 39
[instance initializer]
    client::CGUI, 24

ACCEPTEDCH
    client::ClientMSG, 41
    server::ClientMSG, 44
ADDFRIEND
    client::ClientMSG, 41
    server::ClientMSG, 44
Accepted
    client::ACK, 9
    server::ACK, 14
addFriend
    client::Tasks, 131
    server::Friendships, 83
addfrd
    client::CGUI, 28
AlreadyExisting
    client::ACK, 9
    server::ACK, 15
AlreadyFriends
    client::ACK, 9
    server::ACK, 15
AlreadyRegistered
    client::ACK, 9
    server::ACK, 15

back1
    client::CGUI, 28
back2
    client::CGUI, 28
back3
    client::CGUI, 29

CGUI
    client::CGUI, 22
chal
    client::CGUI, 29
Challenge
    server::Challenge, 36
challenge
    client::MyActionListener, 110
checkFStatus
    server::ClientTasks, 49
client, 6

client.ACK, 7
client.CGUI, 19
client.ClientMSG, 40
client.comUDP, 57
client.ConfParser, 62
client.Game, 91
client.Main, 104
client.MyActionListener, 109
client.Tasks, 130
client::ACK
    Accepted, 9
    AlreadyExisting, 9
    AlreadyFriends, 9
    AlreadyRegistered, 9
    EmptyField, 9
    FriendAdded, 9
    FriendNotFound, 9
    FriendRemoved, 10
    INCHALLENGE, 10
    LoggedIn, 10
    LoggedOut, 10
    NotLogged, 10
    OFFLINE, 10
    ONLINE, 11
    OK, 10
    OperationUnknown, 11
    PasswordUnmatch, 11
    RegistrationError, 11
    Rejected, 11
    UserAdded, 11
    UserAlreadyLoggedIn, 11
    UserDeleted, 12
    UserFound, 12
    UserNotDeleted, 12
    UserNotFound, 12
    UserRegistered, 12
client::CGUI
    $ $getFont, 23
    $ $setupUI, 23
    [instance initializer], 24
    addfrd, 28
    back1, 28
    back2, 28
    back3, 29
    CGUI, 22
    chal, 29
    closeU, 24
    counter, 29
    endgame, 24
    firstP, 29
    friend, 29
    friends, 24
    game, 25
    gameP, 29
    getrk, 29
    getW, 29
    go, 29
    go3, 30
    goup, 30
    helloNick, 30
    listf, 30
    loginP, 30
    logout, 25
    logoutButton, 30
    mainW, 30
    messagepopup, 26
    nameLU, 30
    nameU, 30
    nameup, 31
    nameupLab, 31
    nickLab, 31
    nickin, 31
    nickup, 31
    nickupLab, 31
    num, 31
    passLab, 31
    passLU, 32
    passin, 31
    passU, 32
    passup, 32
    passupLab, 32
    rmfrd, 32
    rmusr, 32
    sendW, 32
    setPoint, 26
    setnicLab, 26
    signin, 32
    signup, 32
    SignupP, 33
    signupP, 33
    snameUL, 33
    snameU, 33
    split, 33
    surnameup, 33
    surnameupLab, 33
    tasks, 33
    totpoint, 33
    updateTotScore, 27
    updateU, 27, 34
    upfrd, 34
    upprof, 34
    viewTask, 27
    winner, 28
    word, 34
client::ClientMSG
    ACCEPTEDCH, 41
    ADDFRIEND, 41
    GETFRIENDS, 41
    GETNFRIENDS, 41
    GETPOINTS, 42
    GETRANK, 42
    LOGIN, 42
    LOGOUT, 42
    RMUSER, 42
    STARTCH, 42
    UPDATEINFO, 42

client::ConfParser
__conf, 67
ConfParser, 63
getListnerPort, 64
getRMIName, 64
getRMIPort, 65
getTimeoutGame, 65
getTimeoutReply, 66
path, 68
setTimeoutGame, 66
updateConf, 67
client::Game
end, 93
Game, 93
isEnd, 94
mainUI, 94
run, 94
client::Main
connect, 105
getMyPort, 105
inBuff, 108
main, 106
outBuff, 108
parser, 108
recv, 106
reg, 109
send, 107
socket, 109
stub, 109
client::MyActionListener
challenge, 110
getRank, 111
goUpdate, 112
logout, 113
removeFriend, 114
removeUser, 115
setAddFriend, 116
setBack1, 117
setBack2, 118
setBack3, 118
setGoLogin, 119
setGoReg, 120
setSignin, 121
setSignup, 122
update, 122
updateFriend, 123
client::Tasks
addFriend, 131
friendlist, 132
getPoints, 132
getRank, 133
login, 134
logout, 135
reg, 136
rmFriend, 137
rmUser, 137
updateProfile, 138
client::comUDP
comUDP, 59
mainUI, 61
myport, 61
packToRecv, 61
packToSend, 61
run, 60
send, 60
sock, 61
ClientTasks
server::ClientTasks, 48
closeU
client::CGUI, 24
comUDP
client::comUDP, 59
ConfParser
client::ConfParser, 63
server::ConfParser, 69
connect
client::Main, 105
counter
client::CGUI, 29

DBsWriter
server::DBsWriter, 79
deleteUser
server::Users, 154
Dictionary
server::Main, 102

EmptyField
client::ACK, 9
server::ACK, 15
end
client::Game, 93
endgame
client::CGUI, 24

FDB
server::Main, 102
firstP
client::CGUI, 29
friend
client::CGUI, 29
FriendAdded
client::ACK, 9
server::ACK, 15
FriendNotFound
client::ACK, 9
server::ACK, 15
FriendRemoved
client::ACK, 10
server::ACK, 15
friendlist
client::Tasks, 132
friends
client::CGUI, 24
Friendships
server::Friendships, 82

GETFRIENDS
client::ClientMSG, 41
server::ClientMSG, 44
GETNFRIENDS
client::ClientMSG, 41
server::ClientMSG, 44
GETPOINTS
client::ClientMSG, 42
server::ClientMSG, 45
GETRANK
client::ClientMSG, 42
server::ClientMSG, 45
Game
client::Game, 93
server::ClientTasks, 50
game
client::CGUI, 25
gameP
client::CGUI, 29
getChalScore
server::User, 143
getCorrectPoints
server::ConfParser, 70
getDictionary
server::Main, 96
getDictionaryPath
server::ConfParser, 71
getExtraPoints
server::ConfParser, 71
getFriendPath
server::ConfParser, 72
getFriends
server::Friendships, 84
getID
server::User, 144
getListnerPort
client::ConfParser, 64
server::ConfParser, 72
getLogfilePath
server::ConfParser, 73
getMyPort
client::Main, 105
getNFriends
server::Friendships, 85
getNWords
server::ConfParser, 73
getName
server::User, 144
getNusers
server::Users, 155
getPassword
server::User, 144
getPoints
client::Tasks, 132
getRMIName
client::ConfParser, 64
getRMIPort
client::ConfParser, 65
server::ConfParser, 74
getRank
client::MyActionListener, 111
client::Tasks, 133
getRanking
server::Users, 155
getStatus
server::User, 145
getSurname
server::User, 145
getTimeUpdater
server::ConfParser, 75
getTimeoutGame
client::ConfParser, 65
server::ConfParser, 74
getTimeoutReply
client::ConfParser, 66
server::ConfParser, 75
getTotScore
server::User, 145
getUniPort
server::Main, 97
getUser
server::Users, 156
getUsersPath
server::ConfParser, 76
getWinner
server::Main, 97
getWords
server::Main, 98
getWrongPoints
server::ConfParser, 76
getrk
client::CGUI, 29
getW
client::CGUI, 29
go
client::CGUI, 29
go3
client::CGUI, 30
goUpdate
client::MyActionListener, 112
goup
client::CGUI, 30

helloNick
client::CGUI, 30

INCHALLENGE
client::ACK, 10
server::ACK, 16
inBuff
client::Main, 108
init
server::Friendships, 85
server::Users, 157
insertUser
server::Users, 158
isEnd

client::Game, 94

LISTENER
server::Main, 102
LOGIN
client::ClientMSG, 42
server::ClientMSG, 45
LOGOUT
client::ClientMSG, 42
server::ClientMSG, 45
listf
client::CGUI, 30
logfile
server::Main, 102
LoggedIn
client::ACK, 10
server::ACK, 16
LoggedOut
client::ACK, 10
server::ACK, 16
logger
server::Main, 99
login
client::Tasks, 134
loginP
client::CGUI, 30
logout
client::CGUI, 25
client::MyActionListener, 113
client::Tasks, 135
logoutButton
client::CGUI, 30

main
client::Main, 106
server::Main, 100
mainUI
client::Game, 94
client::comUDP, 61
mainW
client::CGUI, 30
messagepopup
client::CGUI, 26
myport
client::comUDP, 61

nameLU
client::CGUI, 30
nameU
client::CGUI, 30
nameup
client::CGUI, 31
nameupLab
client::CGUI, 31
newUser
server::Friendships, 86
nickLab
client::CGUI, 31
nickin
client::CGUI, 31
nickup
client::CGUI, 31
nickupLab
client::CGUI, 31
NotLogged
client::ACK, 10
server::ACK, 16
num
client::CGUI, 31

OFFLINE
client::ACK, 10
server::ACK, 16
ONLINE
client::ACK, 11
server::ACK, 16
OK
client::ACK, 10
server::ACK, 16
OperationUnknown
client::ACK, 11
server::ACK, 17
outBuff
client::Main, 108

packToRecv
client::comUDP, 61
packToSend
client::comUDP, 61
parser
client::Main, 108
server::Main, 103
passLab
client::CGUI, 31
passLU
client::CGUI, 32
passin
client::CGUI, 31
passU
client::CGUI, 32
passup
client::CGUI, 32
passupLab
client::CGUI, 32
PasswordUnmatch
client::ACK, 11
server::ACK, 17
path
client::ConfParser, 68
personalRank
server::ClientTasks, 51
pswIsEqual
server::User, 146

RMUSER
client::ClientMSG, 42
server::ClientMSG, 45
recv

client::Main, 106
server::Challenge, 36
server::ClientTasks, 51
reg
client::Main, 109
client::Tasks, 136
RegRMImplementation
server::RegRMImplementation, 126
RegUser
server::RegRMImplementation, 126
server::RegRMInterface, 129
RegistrationError
client::ACK, 11
server::ACK, 17
Rejected
client::ACK, 11
server::ACK, 17
removeFriend
client::MyActionListener, 114
server::Friendships, 86
removeUser
client::MyActionListener, 115
server::Friendships, 88
resetName
server::User, 146
resetPassword
server::User, 147
resetSurname
server::User, 147
resources/doxyhome.md, 162
rmFriend
client::Tasks, 137
rmUser
client::Tasks, 137
rmfrd
client::CGUI, 32
rmusr
client::CGUI, 32
run
client::Game, 94
client::comUDP, 60
server::Challenge, 37
server::ClientTasks, 52
server::DBsWriter, 79

STARTCH
client::ClientMSG, 42
server::ClientMSG, 45
searchFriend
server::Friendships, 89
searchUser
server::Users, 159
send
client::Main, 107
client::comUDP, 60
server::Challenge, 38
server::ClientTasks, 53
sendRequest
server::ClientTasks, 54
sendW
client::CGUI, 32
serialVersionUID
server::RegRMImplementation, 127
server::User, 151
server, 6
server.ACK, 13
server.Challenge, 34
server.ClientMSG, 43
server.ClientTasks, 46
server.ConfParser, 68
server.DBsWriter, 77
server.Friendships, 81
server.Main, 95
server.RegRMImplementation, 124
server.RegRMInterface, 128
server.User, 140
server.Users, 152
server::ACK
Accepted, 14
AlreadyExisting, 15
AlreadyFriends, 15
AlreadyRegistered, 15
EmptyField, 15
FriendAdded, 15
FriendNotFound, 15
FriendRemoved, 15
INCHALLENGE, 16
LoggedIn, 16
LoggedOut, 16
NotLogged, 16
OFFLINE, 16
ONLINE, 16
OK, 16
OperationUnknown, 17
PasswordUnmatch, 17
RegistrationError, 17
Rejected, 17
UserAdded, 17
UserAlreadyLoggedIn, 17
UserDeleted, 17
UserFound, 18
UserNotDeleted, 18
UserNotFound, 18
UserRegistered, 18
server::Challenge
__inBuff, 39
__outBuff, 39
__udb, 39
__user, 39
__usrT, 39
__words, 39
Challenge, 36
recv, 36
run, 37
send, 38
server::ClientMSG
ACCEPTEDCH, 44

ADDFRIEND, 44
GETFRIENDS, 44
GETNFRIENDS, 44
GETPOINTS, 45
GETRANK, 45
LOGIN, 45
LOGOUT, 45
RMUSER, 45
STARTCH, 45
UPDATEINFO, 45
server::ClientTasks
__cSocket, 55
__end, 55
__fdb, 55
__inBuff, 56
__me, 56
__outBuff, 56
__udb, 56
checkFStatus, 49
ClientTasks, 48
Game, 50
personalRank, 51
recv, 51
run, 52
send, 53
sendRequest, 54
server::ConfParser
__conf, 77
ConfParser, 69
getCorrectPoints, 70
getDictionaryPath, 71
getExtraPoints, 71
getFriendPath, 72
getListnerPort, 72
getLogfilePath, 73
getNWords, 73
getRMIPort, 74
getTimeUpdater, 75
getTimeoutGame, 74
getTimeoutReply, 75
getUsersPath, 76
getWrongPoints, 76
server::DBsWriter
__fdb, 80
__udb, 80
DBsWriter, 79
run, 79
server::Friendships
__PATHDBF, 90
__dbFriend, 90
__friends, 90
addFriend, 83
Friendships, 82
getFriends, 84
getNFriends, 85
init, 85
newUser, 86
removeFriend, 86
removeUser, 88
searchFriend, 89
writeONfile, 90
server::Main
Dictionary, 102
FDB, 102
getDictionary, 96
getUniPort, 97
getWinner, 97
getWords, 98
LISTENER, 102
logfile, 102
logger, 99
main, 100
parser, 103
setWords, 101
Tpool, 103
UDB, 103
updater, 103
server::RegRMImplementation
__fdb, 127
__udb, 127
RegRMImplementation, 126
RegUser, 126
serialVersionUID, 127
server::RegRMInterface
RegUser, 129
server::User
__ID, 150
__chScore, 150
__name, 150
__password, 150
__status, 150
__surname, 151
__tScore, 151
getChalScore, 143
getID, 144
getName, 144
getPassword, 144
getStatus, 145
getSurname, 145
getTotScore, 145
pswIsEqual, 146
resetName, 146
resetPassword, 147
resetSurname, 147
serialVersionUID, 151
setCScore, 148
setInChallenge, 148
setOffline, 149
setOnline, 149
setTScore, 149
User, 143
server::Users
__PATHDBU, 161
__dbFile, 161
__dbUser, 161
__parser, 161

__size, 162
__users, 162
deleteUser, 154
getNusers, 155
getRanking, 155
getUser, 156
init, 157
insertUser, 158
searchUser, 159
shutdown, 159
updateUser, 160
Users, 154
writeONfile, 160
setAddFriend
client::MyActionListener, 116
setBack1
client::MyActionListener, 117
setBack2
client::MyActionListener, 118
setBack3
client::MyActionListener, 118
setCScore
server::User, 148
setGoLogin
client::MyActionListener, 119
setGoReg
client::MyActionListener, 120
setInChallenge
server::User, 148
setOffline
server::User, 149
setOnline
server::User, 149
setPoint
client::CGUI, 26
setSignin
client::MyActionListener, 121
setSignup
client::MyActionListener, 122
setTScore
server::User, 149
setTimeoutGame
client::ConfParser, 66
setWords
server::Main, 101
setnicLab
client::CGUI, 26
shutdown
server::Users, 159
signin
client::CGUI, 32
signup
client::CGUI, 32
SignupP
client::CGUI, 33
signupP
client::CGUI, 33
snameUL
client::CGUI, 33
snameU
client::CGUI, 33
sock
client::comUDP, 61
socket
client::Main, 109
split
client::CGUI, 33
src/client/ACK.java, 162
src/client/CGUI.java, 163
src/client/ConfParser.java, 163
src/client/Game.java, 164
src/client/Main.java, 164
src/client/MyActionListener.java, 164
src/client/Tasks.java, 164
src/client/comUDP.java, 163
src/server/ACK.java, 162
src/server/Challenge.java, 165
src/server/ClientTasks.java, 165
src/server/ConfParser.java, 163
src/server/Friendships.java, 165
src/server/Main.java, 164
src/server/RegRMImplementation.java, 165
src/server/RegRMInterface.java, 166
src/server/User.java, 166
src/server/Users.java, 166
stub
client::Main, 109
surnameup
client::CGUI, 33
surnameupLab
client::CGUI, 33

tasks
client::CGUI, 33
totpoint
client::CGUI, 33
Tpool
server::Main, 103

UDB
server::Main, 103
UPDATEINFO
client::ClientMSG, 42
server::ClientMSG, 45
update
client::MyActionListener, 122
updateConf
client::ConfParser, 67
updateFriend
client::MyActionListener, 123
updateProfile
client::Tasks, 138
updateTotScore
client::CGUI, 27
updateUser
server::Users, 160
updater

server::Main, 103
updateU
client::CGUI, 27, 34
upfrd
client::CGUI, 34
upprof
client::CGUI, 34
User
server::User, 143
UserAdded
client::ACK, 11
server::ACK, 17
UserAlreadyLoggedIn
client::ACK, 11
server::ACK, 17
UserDeleted
client::ACK, 12
server::ACK, 17
UserFound
client::ACK, 12
server::ACK, 18
UserNotDeleted
client::ACK, 12
server::ACK, 18
UserNotFound
client::ACK, 12
server::ACK, 18
UserRegistered
client::ACK, 12
server::ACK, 18
Users
server::Users, 154

viewTask
client::CGUI, 27

winner
client::CGUI, 28
word
client::CGUI, 34
writeONfile
server::Friendships, 90
server::Users, 160