WQG - WordsQuizzleGame

v1.5.8

Generated by Doxygen 1.8.13

Contents

1 WQGame 1

2 Namespace Index 3

2.1 Packages . . . 3

3 Hierarchical Index 3

3.1 Class Hierarchy . . . 3

4 Class Index 4

4.1 Class List . . . 4

5 File Index 5

5.1 File List . . . 5

6 Namespace Documentation 5

6.1 Package client . . . 5

6.2 Package server . . . 5

7 Class Documentation 6

7.1 server.ACK Enum Reference . . . 6

7.1.1 Detailed Description . . . 7

7.1.2 Member Data Documentation . . . 7

7.2 server.Challenge Class Reference . . . 11

7.2.1 Detailed Description . . . 12

7.2.2 Constructor & Destructor Documentation . . . 12

7.2.3 Member Function Documentation . . . 12

7.2.4 Member Data Documentation . . . 13

7.3 server.ClientMSG Enum Reference . . . 14

7.3.1 Detailed Description . . . 15

7.3.2 Member Data Documentation . . . 15

7.4 client.ClientMSG Enum Reference . . . 17

7.4.1 Member Data Documentation . . . 17

7.5 server.ClientTasks Class Reference . . . 18
7.5.1 Detailed Description . . . 19
7.5.2 Constructor & Destructor Documentation . . . 19
7.5.3 Member Function Documentation . . . 20
7.5.4 Member Data Documentation . . . 22
7.6 server.ConfParser Class Reference . . . 23
7.6.1 Detailed Description . . . 23
7.6.2 Constructor & Destructor Documentation . . . 24
7.6.3 Member Function Documentation . . . 24
7.6.4 Member Data Documentation . . . 27
7.7 server.DBsWriter Class Reference . . . 27
7.7.1 Constructor & Destructor Documentation . . . 28
7.7.2 Member Function Documentation . . . 28
7.7.3 Member Data Documentation . . . 28
7.8 server.Friendships Class Reference . . . 29
7.8.1 Detailed Description . . . 29
7.8.2 Constructor & Destructor Documentation . . . 30
7.8.3 Member Function Documentation . . . 30
7.8.4 Member Data Documentation . . . 33
7.9 client.Main Class Reference . . . 33
7.9.1 Member Function Documentation . . . 33
7.10 server.Main Class Reference . . . 34
7.10.1 Member Function Documentation . . . 34
7.10.2 Member Data Documentation . . . 35
7.11 server.RegRMImplementation Class Reference . . . 36
7.11.1 Detailed Description . . . 36
7.11.2 Constructor & Destructor Documentation . . . 36
7.11.3 Member Function Documentation . . . 37
7.11.4 Member Data Documentation . . . 37
7.12 server.RegRMInterface Class Reference . . . 38
7.12.1 Detailed Description . . . 38
7.12.2 Member Function Documentation . . . 38
7.13 server.User Class Reference . . . 39
7.13.1 Detailed Description . . . 40
7.13.2 Constructor & Destructor Documentation . . . 40
7.13.3 Member Function Documentation . . . 41
7.13.4 Member Data Documentation . . . 44
7.14 server.Users Class Reference . . . 46
7.14.1 Detailed Description . . . 47
7.14.2 Constructor & Destructor Documentation . . . 47
7.14.3 Member Function Documentation . . . 47
7.14.4 Member Data Documentation . . . 50

8 File Documentation **51**

8.1 resources/doxyhome.md File Reference 51

8.2 src/client/Main.java File Reference 51

8.3 src/server/Main.java File Reference 51

8.4 src/server/ACK.java File Reference 52

8.5 src/server/Challenge.java File Reference 52

8.6 src/server/ClientTasks.java File Reference 52

8.7 src/server/ConfParser.java File Reference 52

8.8 src/server/Friendships.java File Reference 52

8.9 src/server/RegRMImplementation.java File Reference 53

8.10 src/server/RegRMInterface.java File Reference 53

8.11 src/server/User.java File Reference 53

8.12 src/server/Users.java File Reference 53

Index **55**

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 5

server 5

3 Hierarchical Index

3.1 Class Hierarchy

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

server.ACK 6

server.ClientMSG 14

client.ClientMSG 17

server.ConfParser 23

server.Friendships 29

client.Main 33

server.Main 34
RemoteServer

server.RegRMImplementation 36
Runnable

server.Challenge 11

server.ClientTasks 18

server.DBsWriter 27
Remote

server.RegRMInterface 38

server.RegRMImplementation 36
Serializable

server.User 39

server.Users 46

4 Class Index

4.1 Class List

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

server.ACK 6

server.Challenge 11

server.ClientMSG 14

client.ClientMSG 17

server.ClientTasks 18

server.ConfParser 23

server.DBsWriter 27

server.Friendships 29

client.Main 33

server.Main 34

server.RegRMImplementation 36

server.RegRMInterface 38

server.User 39

server.Users 46

5 File Index

5.1 File List

Here is a list of all files with brief descriptions:

src/client/Main.java 51

src/server/ACK.java 52

src/server/Challenge.java 52

src/server/ClientTasks.java 52

src/server/ConfParser.java 52

src/server/Friendships.java 52

src/server/Main.java 51

src/server/RegRMImplementation.java 53

src/server/RegRMInterface.java 53

src/server/User.java 53

src/server/Users.java 53

6 Namespace Documentation

6.1 Package client

Classes

- enum ClientMSG
- class Main

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 server.ACK Enum Reference

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 da inviare al client a termine delle operazioni richieste

Author

Luca Canessa (Mat. 516639)

Version

1.6

Since

1.0

7.1.2 Member Data Documentation

7.1.2.1 Accepted

`server.ACK.Accepted`

sfida accettata

7.1.2.2 AlreadyExisting

`server.ACK.AlreadyExisting`

7.1.2.3 AlreadyFriends

`server.ACK.AlreadyFriends`

User(s) già amici.

7.1.2.4 AlreadyRegistered

`server.ACK.AlreadyRegistered`

User già registrato.

7.1.2.5 EmptyField

`server.ACK.EmptyField`

campi utente vuoti

7.1.2.6 FriendAdded

`server.ACK.FriendAdded`

User registrato come amico.

7.1.2.7 FriendNotFound

`server.ACK.FriendNotFound`

User amico non trovato.

7.1.2.8 FriendRemoved

`server.ACK.FriendRemoved`

User amico rimosso dalla lista degli amici.

7.1.2.9 INCHALLENGE

`server.ACK.INCHALLENGE`

stato dell'User in sfida

7.1.2.10 LoggedIn

`server.ACK.LoggedIn`

User loggato.

7.1.2.11 LoggedOut

`server.ACK.LoggedOut`

User uscito.

7.1.2.12 NotLogged

`server.ACK.NotLogged`

< User non loggato

7.1.2.13 OFFLINE

`server.ACK.OFFLINE`

stato dell'User offline

7.1.2.14 OK

`server.ACK.OK`

azione eseguita

7.1.2.15 ONLINE

`server.ACK.ONLINE`

stato dell'User online

7.1.2.16 OperationUnknown

`server.ACK.OperationUnknown`

operazione richiesta non riconosciuta

7.1.2.17 PasswordUnmatch

`server.ACK.PasswordUnmatch`

password non corrisponde a quella del User

7.1.2.18 RegistrationError

`server.ACK.RegistrationError`

problema generale durante la registrazione dell'User

7.1.2.19 Rejected

`server.ACK.Rejected`

sfida rifiutata

7.1.2.20 UserAdded

`server.ACK.UserAdded`

User inserito nel database.

7.1.2.21 UserAlreadyLoggedIn

`server.ACK.UserAlreadyLoggedIn`

User già entrato.

7.1.2.22 UserDeleted

`server.ACK.UserDeleted`

User deregistrato.

7.1.2.23 UserFound

`server.ACK.UserFound`

User trovato nel database.

7.1.2.24 UserNotDeleted

`server.ACK.UserNotDeleted`

User non deregistrato.

7.1.2.25 UserNotFound

`server.ACK.UserNotFound`

User non trovato nel database.

7.1.2.26 UserRegistered

`server.ACK.UserRegistered`

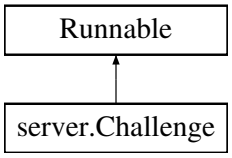
User registrato.

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

- src/server/ACK.java

7.2 server.Challenge Class Reference

Inheritance 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

7.2.1 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

7.2.2 Constructor & Destructor Documentation

7.2.2.1 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

7.2.3 Member Function Documentation

7.2.3.1 recv()

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

Rimane in attesa sulla socket per ricevere i dati dal client

Returns

il buffer (String) contenente i dati

7.2.3.2 run()

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

Metodo che mette in esecuzione il thread che gesitsce il gioco di un client. Invia una parola italaian 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 toale della partita un altro valore. Tali valori di partita sono presi dal file di configurazione. Quando termina aggiorna i punti della partita del User e alza un interrupt per il thread principale che lo avvisa della terminazione del gioco.

7.2.3.3 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

7.2.4 Member Data Documentation

7.2.4.1 __inBuff

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

legge sulla socket

7.2.4.2 __outBuff

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

stampa sulla socket

7.2.4.3 __udb

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

database degli utenti

7.2.4.4 __user

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

User che sta giocando.

7.2.4.5 __usrT

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

thread principale del client

7.2.4.6 __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

7.3 server.ClientMSG Enum Reference

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

7.3.1 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

7.3.2 Member Data Documentation

7.3.2.1 ACCEPTEDCH

`server.ClientMSG.ACCEPTEDCH`

accettazione della sfida arrivata

7.3.2.2 ADDFRIEND

`server.ClientMSG.ADDFRIEND`

richiesta di aggiunta di amico alla lista

7.3.2.3 GETFRIENDS

`server.ClientMSG.GETFRIENDS`

richiesta lista amicizie

7.3.2.4 GETNFRIENDS

`server.ClientMSG.GETNFRIENDS`

richiesto numero di amici

7.3.2.5 GETPOINTS

`server.ClientMSG.GETPOINTS`

richiesta punti totalizzati

7.3.2.6 GETRANK

`server.ClientMSG.GETRANK`

richiesta della classifica

7.3.2.7 LOGIN

`server.ClientMSG.LOGIN`

richiesta di login

7.3.2.8 LOGOUT

`server.ClientMSG.LOGOUT`

richiesta di logout

7.3.2.9 RMUSER

`server.ClientMSG.RMUSER`

richiesta rimozione dell'User

7.3.2.10 STARTCH

`server.ClientMSG.STARTCH`

richiesta di inizio di una sfida

7.3.2.11 UPDATEINFO

`server.ClientMSG.UPDATEINFO`

richiesta di aggiornamento dei dati

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

- src/server/ClientTasks.java

7.4 client.ClientMSG Enum Reference

Public Attributes

- LOGIN
- LOGOUT
- ADDFRIEND
- GETFRIENDS
- GETNFRIENDS
- SENDCH
- ACCEPTEDCH
- UPDATEINFO
- GETPOINTS

7.4.1 Member Data Documentation

7.4.1.1 ACCEPTEDCH

```
client.ClientMSG.ACCEPTEDCH
```

7.4.1.2 ADDFRIEND

```
client.ClientMSG.ADDFRIEND
```

7.4.1.3 GETFRIENDS

```
client.ClientMSG.GETFRIENDS
```

7.4.1.4 GETNFRIENDS

```
client.ClientMSG.GETNFRIENDS
```

7.4.1.5 GETPOINTS

```
client.ClientMSG.GETPOINTS
```

7.4.1.6 LOGIN

```
client.ClientMSG.LOGIN
```

7.4.1.7 LOGOUT

`client.ClientMSG.LOGOUT`

7.4.1.8 SENDCH

`client.ClientMSG.SENDCH`

7.4.1.9 UPDATEINFO

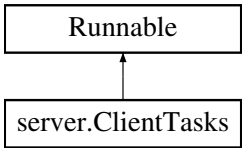
`client.ClientMSG.UPDATEINFO`

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

- src/client/Main.java

7.5 server.ClientTasks Class Reference

Inheritance 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 ()
- void Game (String nickname, String rival)

Private Attributes

- 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

7.5.1 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

1.5

Since

1.0

7.5.2 Constructor & Destructor Documentation

7.5.2.1 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

7.5.3 Member Function Documentation

7.5.3.1 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

7.5.3.2 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 nuovamente lo stato dell'User in ONLINE e aggiornando i punti guadagnati all'User e decretare il vincitore della sfida

Parameters

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

7.5.3.3 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

7.5.3.4 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

7.5.3.5 run()

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

Metodo che mette in esecuzione il 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

7.5.3.6 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

7.5.3.7 sendRequest()

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

TODO

Returns

7.5.4 Member Data Documentation

7.5.4.1 __cSocket

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

socket di comunicazione con il client

7.5.4.2 __fdb

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

database delle amicizie

7.5.4.3 __inBuff

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

buffer su cui scrive il client per comunicare

7.5.4.4 __me

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

User che si collega.

7.5.4.5 __outBuff

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

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

7.5.4.6 __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

7.6 server.ConfParser Class Reference

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 ()

Private Attributes

- JSONObject __conf = null
  configurazione completa del server

7.6.1 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

7.6.2 Constructor & Destructor Documentation

7.6.2.1 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

Parameters

confPath	path del file di configurazione

7.6.3 Member Function Documentation

7.6.3.1 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

7.6.3.2 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

7.6.3.3 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

7.6.3.4 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

7.6.3.5 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

7.6.3.6 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

7.6.3.7 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'

7.6.3.8 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

7.6.3.9 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

7.6.3.10 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

7.6.3.11 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

7.6.3.12 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

7.6.4 Member Data Documentation

7.6.4.1 __conf

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

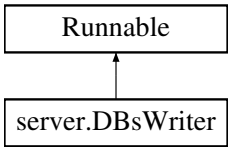
configurazione completa del server

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

- src/server/ConfParser.java

7.7 server.DBsWriter Class Reference

Inheritance diagram for server.DBsWriter:



Public Member Functions

- void run ()

Package Functions

- DBsWriter (Users UDB, Friendships FDB)

Private Attributes

- Users __udb = null
- Friendships __fdb = null

7.7.1 Constructor & Destructor Documentation

7.7.1.1 DBsWriter()

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

7.7.2 Member Function Documentation

7.7.2.1 run()

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

7.7.3 Member Data Documentation

7.7.3.1 __fdb

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

7.7.3.2 __udb

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

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

- src/server/Main.java

7.8 server.Friendships Class Reference

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

7.8.1 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

7.8.2 Constructor & Destructor Documentation

7.8.2.1 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.

7.8.3 Member Function Documentation

7.8.3.1 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

7.8.3.2 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)

7.8.3.3 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

7.8.3.4 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

7.8.3.5 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

7.8.3.6 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

7.8.3.7 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

7.8.3.8 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

7.8.3.9 writeONfile()

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

Metodo utilizzato per scrivere il db delle amicizie su file

7.8.4 Member Data Documentation

7.8.4.1 __dbFriend

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

struttra dati del db in memoria

7.8.4.2 __friends

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

singleton

7.8.4.3 __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

7.9 client.Main Class Reference

Static Public Member Functions

- static void main (String[ ] args)

7.9.1 Member Function Documentation

7.9.1.1 main()

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

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

- src/client/Main.java

7.10 server.Main Class Reference

Static Public Member Functions

- static void main (String[ ] args)
- static void setWords (String username) throws IOException, ParseException
- static String getWinner (User U1, User U2)

Static Public Attributes

- static ConfParser parser = null

Static Private Member Functions

- static ArrayList< String > getDictionary (String filepath)

Static Private Attributes

- static Users UDB = null
- static Friendships FDB = null
- static ExecutorService Tpool = null
- static ServerSocket LISTENER = null
- static ArrayList< String > Dictionary = null
- static Thread updater = null

7.10.1 Member Function Documentation

7.10.1.1 getDictionary()

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

7.10.1.2 getWinner()

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

7.10.1.3 main()

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

7.10.1.4 setWords()

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

7.10.2 Member Data Documentation

7.10.2.1 Dictionary

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

7.10.2.2 FDB

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

7.10.2.3 LISTENER

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

7.10.2.4 parser

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

7.10.2.5 Tpool

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

7.10.2.6 UDB

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

7.10.2.7 updater

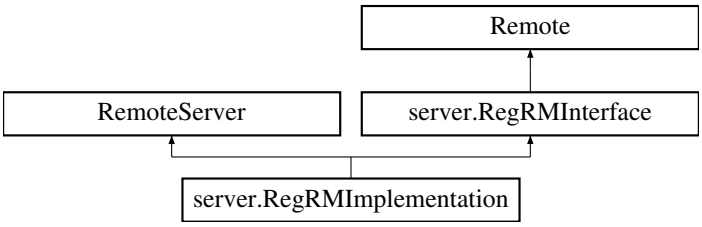
```
Thread server.Main.updater = null  [static], [private]
```

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

- src/server/Main.java

7.11 server.RegRMImplementation Class Reference

Inheritance 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

7.11.1 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

7.11.2 Constructor & Destructor Documentation

7.11.2.1 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

7.11.3 Member Function Documentation

7.11.3.1 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

//TODO

Exceptions

RemoteException	

7.11.4 Member Data Documentation

7.11.4.1 __fdb

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

database delle amicizie

7.11.4.2 __udb

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

database degli utenti

7.11.4.3 serialVersionUID

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

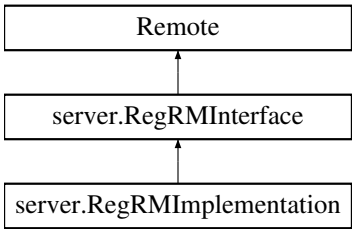
variabile necessaria per la serializzazione dell'oggetto

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

- src/server/RegRMImplementation.java

7.12 server.RegRMInterface Class Reference

Inheritance diagram for server.RegRMInterface:



Public Member Functions

- ACK RegUser (User u) throws RemoteException

7.12.1 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

7.12.2 Member Function Documentation

7.12.2.1 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

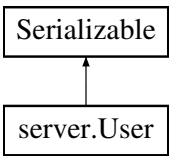
RemoteException	

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

- src/server/RegRMInterface.java

7.13 server.User Class Reference

Inheritance 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

7.13.1 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

7.13.2 Constructor & Destructor Documentation

7.13.2.1 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

7.13.2.2 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

7.13.3 Member Function Documentation

7.13.3.1 getChalScore()

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

Restituisce il valore del punteggio guadagnato durante l'ultima partita

Returns

punteggio della partita

7.13.3.2 getID()

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

Restituisce il nickname dell'User

Returns

il codice univoco (nickname) come String

7.13.3.3 getName()

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

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

Returns

il nome dell'utente

7.13.3.4 getPassword()

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

Restituisce la password dell'utente

Returns

la password dell'utente

7.13.3.5 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

7.13.3.6 getSurname()

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

Restituisce il cognome dell'utente se impostato, null altrimenti

Returns

il cognome dell'utente

7.13.3.7 getTotScore()

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

Restituisce il valore del punteggio attuale totale dell'utente

Returns

punteggio totale

7.13.3.8 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

7.13.3.9 resetName()

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

Imposta il nome dell'utente

Parameters

Name	nome dell'utente

7.13.3.10 resetPassword()

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

Reimposta password

Parameters

Password	nuova password

7.13.3.11 resetSurname()

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

Imposta il cognome dell'utente

Parameters

Surname	cognome dell'utente

7.13.3.12 setCScore()

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

Imposta il valore dei punti guadagnati nell'ultima partita

Parameters

Score	valore ottenuto dall'ultima partita

7.13.3.13 setInChallenge()

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

Imposta lo stato dell'utente come INCHALLENGE

7.13.3.14 setOffline()

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

Imposta lo stato dell'utente come OFFLINE

7.13.3.15 setOnline()

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

Imposta lo stato dell'utente come ONLINE

7.13.3.16 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

7.13.4 Member Data Documentation

7.13.4.1 __chScore

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

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

7.13.4.2 __ID

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

nickname dell'utente - ID del database

7.13.4.3 __name

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

nome dell'utente

7.13.4.4 __password

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

password dell'utente

7.13.4.5 __status

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

stato sul server dell'utente

7.13.4.6 __surname

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

cognome dell'utente

7.13.4.7 __tScore

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

numero di punti totalizzati dall'utente

7.13.4.8 serialVersionUID

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

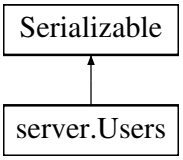
variabile necessaria per rendere l'oggetto serializzabile

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

- src/server/User.java

7.14 server.Users Class Reference

Inheritance 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 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

7.14.1 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

7.14.2 Constructor & Destructor Documentation

7.14.2.1 Users()

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

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

7.14.3 Member Function Documentation

7.14.3.1 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

7.14.3.2 getNusers()

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

Restituisce il numero di oggetti User all'interno del database

Returns

il numero degli utenti iscritti

7.14.3.3 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

7.14.3.4 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

7.14.3.5 init()

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

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

Returns

oggetto che rappresenta il database

7.14.3.6 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

7.14.3.7 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

7.14.3.8 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

7.14.3.9 writeONfile()

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

Scrive sul file il db degli utenti

7.14.4 Member Data Documentation

7.14.4.1 __dbFile

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

scrittore del db sul file '__PATHDBU'

7.14.4.2 __dbUser

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

il db reale

7.14.4.3 __parser

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

file parser

7.14.4.4 __PATHDBU

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

path al file del db utent

7.14.4.5 __size

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

numero di utenti iscritti

7.14.4.6 __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/Main.java File Reference

Classes

- enum client.ClientMSG
- class client.Main

Packages

- package client

8.3 src/server/Main.java File Reference

Classes

- class server.DBsWriter
- class server.Main

Packages

- package server

8.4 src/server/ACK.java File Reference

Classes

- enum server.ACK

Packages

- package server

8.5 src/server/Challenge.java File Reference

Classes

- class server.Challenge

Packages

- package server

8.6 src/server/ClientTasks.java File Reference

Classes

- enum server.ClientMSG
- class server.ClientTasks

Packages

- package server

8.7 src/server/ConfParser.java File Reference

Classes

- class server.ConfParser

Packages

- package server

8.8 src/server/Friendships.java File Reference

Classes

- class server.Friendships

Packages

- package server

8.9 src/server/RegRMImplementation.java File Reference

Classes

- class server.RegRMImplementation

Packages

- package server

8.10 src/server/RegRMInterface.java File Reference

Classes

- class server.RegRMInterface

Packages

- package server

8.11 src/server/User.java File Reference

Classes

- class server.User

Packages

- package server

8.12 src/server/Users.java File Reference

Classes

- class server.Users

Packages

- package server

Index

__ID
    server::User, 45
__PATHDBF
    server::Friendships, 33
__PATHDBU
    server::Users, 50
__cSocket
    server::ClientTasks, 22
__chScore
    server::User, 44
__conf
    server::ConfParser, 27
__dbFile
    server::Users, 50
__dbFriend
    server::Friendships, 33
__dbUser
    server::Users, 50
__fdb
    server::ClientTasks, 22
    server::DBsWriter, 28
    server::RegRMImplementation, 37
__friends
    server::Friendships, 33
__inBuff
    server::Challenge, 13
    server::ClientTasks, 22
__me
    server::ClientTasks, 22
__name
    server::User, 45
__outBuff
    server::Challenge, 13
    server::ClientTasks, 22
__parser
    server::Users, 50
__password
    server::User, 45
__size
    server::Users, 50
__status
    server::User, 45
__surname
    server::User, 45
__tScore
    server::User, 45
__udb
    server::Challenge, 13
    server::ClientTasks, 22
    server::DBsWriter, 28
    server::RegRMImplementation, 37
__user
    server::Challenge, 14
__users
    server::Users, 51
__usrT
    server::Challenge, 14
__words
    server::Challenge, 14

ACCEPTEDCH
    client::ClientMSG, 17
    server::ClientMSG, 15
ADDFRIEND
    client::ClientMSG, 17
    server::ClientMSG, 15
Accepted
    server::ACK, 7
addFriend
    server::Friendships, 30
AlreadyExisting
    server::ACK, 7
AlreadyFriends
    server::ACK, 7
AlreadyRegistered
    server::ACK, 7

Challenge
    server::Challenge, 12
checkFStatus
    server::ClientTasks, 20
client, 5
client.ClientMSG, 17
client.Main, 33
client::ClientMSG
    ACCEPTEDCH, 17
    ADDFRIEND, 17
    GETFRIENDS, 17
    GETNFRIENDS, 17
    GETPOINTS, 17
    LOGIN, 17
    LOGOUT, 17
    SENDCH, 18
    UPDATEINFO, 18
client::Main
    main, 33
ClientTasks
    server::ClientTasks, 19
ConfParser
    server::ConfParser, 24

DBsWriter
    server::DBsWriter, 28
deleteUser
    server::Users, 47
Dictionary
    server::Main, 35

EmptyField
    server::ACK, 8

FDB

server::Main, 35
FriendAdded
server::ACK, 8
FriendNotFound
server::ACK, 8
FriendRemoved
server::ACK, 8
Friendships
server::Friendships, 30

GETFRIENDS
client::ClientMSG, 17
server::ClientMSG, 15
GETNFRIENDS
client::ClientMSG, 17
server::ClientMSG, 15
GETPOINTS
client::ClientMSG, 17
server::ClientMSG, 15
GETRANK
server::ClientMSG, 16
Game
server::ClientTasks, 20
getChalScore
server::User, 41
getCorrectPoints
server::ConfParser, 24
getDictionary
server::Main, 34
getDictionaryPath
server::ConfParser, 24
getExtraPoints
server::ConfParser, 24
getFriendPath
server::ConfParser, 25
getFriends
server::Friendships, 30
getID
server::User, 41
getListnerPort
server::ConfParser, 25
getNFriends
server::Friendships, 30
getNWords
server::ConfParser, 25
getName
server::User, 41
getNusers
server::Users, 48
getPassword
server::User, 42
getRMIPort
server::ConfParser, 25
getRanking
server::Users, 48
getStatus
server::User, 42
getSurname
server::User, 42
getTimeUpdater
server::ConfParser, 26
getTimeoutGame
server::ConfParser, 26
getTimeoutReply
server::ConfParser, 26
getTotScore
server::User, 42
getUser
server::Users, 48
getUsersPath
server::ConfParser, 26
getWinner
server::Main, 34
getWrongPoints
server::ConfParser, 27

INCHALLENGE
server::ACK, 8
init
server::Friendships, 31
server::Users, 48
insertUser
server::Users, 49

LISTENER
server::Main, 35
LOGIN
client::ClientMSG, 17
server::ClientMSG, 16
LOGOUT
client::ClientMSG, 17
server::ClientMSG, 16
LoggedIn
server::ACK, 8
LoggedOut
server::ACK, 8

main
client::Main, 33
server::Main, 34

newUser
server::Friendships, 31
NotLogged
server::ACK, 9

OFFLINE
server::ACK, 9
ONLINE
server::ACK, 9
OK
server::ACK, 9
OperationUnknown
server::ACK, 9

parser
server::Main, 35
PasswordUnmatch
server::ACK, 9

personalRank
server::ClientTasks, 20
pswIsEqual
server::User, 42

RMUSER
server::ClientMSG, 16
recv
server::Challenge, 12
server::ClientTasks, 20
RegRMImplementation
server::RegRMImplementation, 36
RegUser
server::RegRMImplementation, 37
server::RegRMInterface, 38
RegistrationError
server::ACK, 9
Rejected
server::ACK, 10
removeFriend
server::Friendships, 31
removeUser
server::Friendships, 32
resetName
server::User, 43
resetPassword
server::User, 43
resetSurname
server::User, 43
resources/doxyhome.md, 51
run
server::Challenge, 13
server::ClientTasks, 21
server::DBsWriter, 28

SENDCH
client::ClientMSG, 18
STARTCH
server::ClientMSG, 16
searchFriend
server::Friendships, 32
searchUser
server::Users, 49
send
server::Challenge, 13
server::ClientTasks, 21
sendRequest
server::ClientTasks, 22
serialVersionUID
server::RegRMImplementation, 37
server::User, 45
server, 5
server.ACK, 6
server.Challenge, 11
server.ClientMSG, 14
server.ClientTasks, 18
server.ConfParser, 23
server.DBsWriter, 27
server.Friendships, 29
server.Main, 34
server.RegRMImplementation, 36
server.RegRMInterface, 38
server.User, 39
server.Users, 46
server::ACK
Accepted, 7
AlreadyExisting, 7
AlreadyFriends, 7
AlreadyRegistered, 7
EmptyField, 8
FriendAdded, 8
FriendNotFound, 8
FriendRemoved, 8
INCHALLENGE, 8
LoggedIn, 8
LoggedOut, 8
NotLogged, 9
OFFLINE, 9
ONLINE, 9
OK, 9
OperationUnknown, 9
PasswordUnmatch, 9
RegistrationError, 9
Rejected, 10
UserAdded, 10
UserAlreadyLoggedIn, 10
UserDeleted, 10
UserFound, 10
UserNotDeleted, 10
UserNotFound, 10
UserRegistered, 11
server::Challenge
__inBuff, 13
__outBuff, 13
__udb, 13
__user, 14
__usrT, 14
__words, 14
Challenge, 12
recv, 12
run, 13
send, 13
server::ClientMSG
ACCEPTEDCH, 15
ADDFRIEND, 15
GETFRIENDS, 15
GETNFRIENDS, 15
GETPOINTS, 15
GETRANK, 16
LOGIN, 16
LOGOUT, 16
RMUSER, 16
STARTCH, 16
UPDATEINFO, 16
server::ClientTasks
__cSocket, 22
__fdb, 22

__inBuff, 22
__me, 22
__outBuff, 22
__udb, 22
checkFStatus, 20
ClientTasks, 19
Game, 20
personalRank, 20
recv, 20
run, 21
send, 21
sendRequest, 22
server::ConfParser
__conf, 27
ConfParser, 24
getCorrectPoints, 24
getDictionaryPath, 24
getExtraPoints, 24
getFriendPath, 25
getListnerPort, 25
getNWords, 25
getRMIPort, 25
getTimeUpdater, 26
getTimeoutGame, 26
getTimeoutReply, 26
getUsersPath, 26
getWrongPoints, 27
server::DBsWriter
__fdb, 28
__udb, 28
DBsWriter, 28
run, 28
server::Friendships
__PATHDBF, 33
__dbFriend, 33
__friends, 33
addFriend, 30
Friendships, 30
getFriends, 30
getNFriends, 30
init, 31
newUser, 31
removeFriend, 31
removeUser, 32
searchFriend, 32
writeONfile, 32
server::Main
Dictionary, 35
FDB, 35
getDictionary, 34
getWinner, 34
LISTENER, 35
main, 34
parser, 35
setWords, 34
Tpool, 35
UDB, 35
updater, 35
server::RegRMImplementation
__fdb, 37
__udb, 37
RegRMImplementation, 36
RegUser, 37
serialVersionUID, 37
server::RegRMInterface
RegUser, 38
server::User
__ID, 45
__chScore, 44
__name, 45
__password, 45
__status, 45
__surname, 45
__tScore, 45
getChalScore, 41
getID, 41
getName, 41
getPassword, 42
getStatus, 42
getSurname, 42
getTotScore, 42
pswIsEqual, 42
resetName, 43
resetPassword, 43
resetSurname, 43
serialVersionUID, 45
setCScore, 43
setInChallenge, 44
setOffline, 44
setOnline, 44
setTScore, 44
User, 40, 41
server::Users
__PATHDBU, 50
__dbFile, 50
__dbUser, 50
__parser, 50
__size, 50
__users, 51
deleteUser, 47
getNusers, 48
getRanking, 48
getUser, 48
init, 48
insertUser, 49
searchUser, 49
updateUser, 49
Users, 47
writeONfile, 50
setCScore
server::User, 43
setInChallenge
server::User, 44
setOffline
server::User, 44
setOnline

server::User, 44
setTScore
server::User, 44
setWords
server::Main, 34
src/client/Main.java, 51
src/server/ACK.java, 52
src/server/Challenge.java, 52
src/server/ClientTasks.java, 52
src/server/ConfParser.java, 52
src/server/Friendships.java, 52
src/server/Main.java, 51
src/server/RegRMImplementation.java, 53
src/server/RegRMInterface.java, 53
src/server/User.java, 53
src/server/Users.java, 53

Tpool
server::Main, 35

UDB
server::Main, 35
UPDATEINFO
client::ClientMSG, 18
server::ClientMSG, 16
updateUser
server::Users, 49
updater
server::Main, 35
User
server::User, 40, 41
UserAdded
server::ACK, 10
UserAlreadyLoggedIn
server::ACK, 10
UserDeleted
server::ACK, 10
UserFound
server::ACK, 10
UserNotDeleted
server::ACK, 10
UserNotFound
server::ACK, 10
UserRegistered
server::ACK, 11
Users
server::Users, 47

writeONfile
server::Friendships, 32
server::Users, 50