25 giugno 2013

GoGo Team

Specifica tecnica

Informazioni sul documento

Nome Documento	Specifica tecnica
Versione	3.0
Stato	*Formale*
Uso	*Esterno*
Data Creazione	18 gennaio 2013
Data Ultima Modifica	25 giugno 2013
Redazione	Valentina Pasqualotto
Approvazione	Matteo Belletti
Verifica	Sara Lazzaretto
Lista distribuzione	GoGo Team
	Prof. Tullio Vardanega
	Prof. Riccardo Cardin
	Il proponente Zucchetti S.p.A.

Sommario

Il presente documento ha lo scopo di fornire una descrizione architetturale del sistema software **MyTalk**.
Partendo dai requisiti vengono descritti, attraverso l'utilizzo di un insieme di diagrammi UML, le attività, i componenti e le loro iterazioni. Verrà fornita una descrizione iniziale più generica per poi progredire sempre più nel dettaglio.

Registro delle modifiche

Versione	Autore	Data	Descrizione
3.0	Valentina Pasqualotto	2013-06-25	Approvazione. Approvazione del documento, cambio di stato in Formale ad uso esterno ed avanzamento di versione.
2.2	Valentina Pasqualotto	2013-06-25	Correzione. Correzioni dei contenuti ed ortografiche a seguito delle segnalazioni effettuate dal verificatore Sara Lazzaretto in data 2013-06-24. Revisione.
2.1	Valentina Pasqualotto	2013-06-24	Modifica. Modifica strutture package nei capitoli *Diagrammi dei package* e *Descrizione singoli componenti* a seguito delle segnalazioni effettuate in sede di RQ dal *Committente* prof. Prof. Tullio Vardanegain data 2013-06-23
2.0	Valentina Pasqualotto	2013-06-10	Approvazione. Approvazione del documento, cambio di stato in Formale ad uso esterno ed avanzamento di versione.
1.4	Francesco Zattarin	2013-05-30	Correzione. Correzioni dei contenuti ed ortografiche a seguito delle segnalazioni effettuate dal verificatore Davide Ceccon in data 2013-05-29. Revisione.
1.3	Elena Zerbato	2013-05-13	Aggiornamento. Aggiornamento della tabella in sezione *Tracciamento relazione componenti - requisiti*.

1.2	Elena Zerbato	2013-01-24	Modifica. Modifica strutture package nei capitoli *Diagrammi dei package*, *Descrizione singoli componenti* e dei diagrammi di sequenza in capitolo *Diagrammi di sequenza*.
1.1	Francesco Zattarin	2013-01-21	Modifica. Aggiornamento dell'introduzione e della descrizione generale a seguito delle segnalazioni effettuate in sede di RP dal *Committente* prof. Prof. Tullio Vardanegain data 2013-02-06
1.0	Valentina Pasqualotto	2013-01-29	Approvazione. Approvazione del documento, cambio di stato in Formalead uso esterno ed avanzamento di versione.
0.4	Davide Ceccon	2013-01-25	Correzione. Correzioni dei contenuti ed ortografiche a seguito delle segnalazioni effettuate dal verificatore Elena Zerbato 2013-01-24. Revisione.
0.3	Matteo Belletti	2013-01-24	Aggiunta contenuti. Diagrammi della classi, diagrammi delle attività e diagrammi di sequenza.
0.2	Davide Ceccon	2013-01-21	Aggiunta contenuti. Definite e specifiche tecnologiche e architetturali.
0.1	Davide Ceccon	2013-01-18	Prima stesura. Contenuti documento: scheletro di base dell'intero documento.

Tabella 1: Versionamento del documento

Storico

RR ->RP

Versione 1.0	Nominativo
Redazione	Matteo Belletti, Davide Ceccon
Verifica	Elena Zerbato
Approvazione	Valentina Pasqualotto

Tabella 2: Storico ruoli RR ->RP

RP ->RQ

Versione 2.0	Nominativo
Redazione	Elena Zerbato, Francesco Zattarin
Verifica	Davide Ceccon
Approvazione	Valentina Pasqualotto

Tabella 3: Storico ruoli RP ->RQ

RQ ->RA

Versione 3.0	Nominativo
Redazione	Valentina Pasqualotto
Verifica	Sara Lazzaretto
Approvazione	Matteo Belletti

Tabella 4: Storico ruoli RQ ->RA

Indice

1 Introduzione **10**
1.1 Scopo del documento . . . 10
1.2 Scopo del prodotto . . . 10
1.3 Glossario . . . 10
1.4 Riferimenti . . . 10
1.4.1 Normativi . . . 10
1.4.2 Informativi . . . 10

2 Definizione del prodotto **12**
2.1 Metodo e formalismo di specifica . . . 12
2.2 Architettura generale del sistema . . . 12
2.2.1 Componente View . . . 13
2.2.2 Componente Presenter . . . 13
2.2.3 Componente Model . . . 14

3 Diagrammi dei package **15**
3.1 Package mytalk.client.view . . . 16
3.2 Package mytalk.client.iView . . . 16
3.3 Package mytalk.client.presenter . . . 16
3.4 Package mytalk.client.iPresenter . . . 17
3.5 Package mytalk.client.model . . . 18
3.6 Package mytalk.server.presenter . . . 18
3.7 Package mytalk.server.model . . . 18

4 Descrizione singoli componenti **19**
4.1 Package mytalk.client.iView . . . 19
4.1.1 Package mytalk.client.iView.iAdministrator . . . 19
4.1.2 Package mytalk.client.iView.iUser . . . 20
4.2 Package mytalk.client.view . . . 24
4.2.1 Package mytalk.client.view.administrator . . . 24
4.2.2 Package mytalk.client.view.user . . . 25
4.3 Package mytalk.client.iPresenter . . . 29
4.3.1 Package mytalk.client.iPresenter.iAdministrator.iLogicAdmin . . . 29
4.3.2 Package mytalk.client.iPresenter.iAdministrator.iServerComAdmin 29
4.3.3 Package mytalk.client.iPresenter.iUser.iLogicUser . . . 30
4.3.4 Package mytalk.client.iPresenter.iUser.iCommunication . . . 31
4.3.5 Package mytalk.client.iPresenter.iUser.iServerComUser . . . 32
4.4 Package mytalk.client.presenter . . . 33
4.4.1 Package mytalk.client.presenter.administrator.logicAdmin . . . 34
4.4.2 Package mytalk.client.presenter.administrator.logicAdmin.common 36
4.4.3 Package mytalk.client.presenter.administrator.serverComAdmin 36
4.4.4 Package mytalk.client.presenter.user.serverComUser . . . 36
4.4.5 Package mytalk.client.presenter.user.logicUser . . . 37
4.4.6 Package mytalk.client.presenter.user.logicUser.common . . . 39
4.4.7 Package mytalk.client.presenter.user.communication . . . 39
4.4.8 com.webrtc . . . 40
4.4.9 com.WebSocket . . . 40

4.5 Package mytalk.server.presenter . . . 42
4.5.1 Package mytalk.server.presenter . . . 42
4.5.2 Package mytalk.server.presenter.user.logicUser . . . 43
4.5.3 Package mytalk.server.presenter.administrator.logicAdmin . . . 44
4.6 Package mytalk.client.model . . . 45
4.6.1 Package mytalk.client.model.localDataAdmin . . . 45
4.6.2 Package mytalk.client.model.localDataUser . . . 46
4.7 Package mytalk.server.model.dao . . . 46

5 Design pattern **48**
5.1 Adapter . . . 48
5.2 Data Access Object . . . 48
5.3 MVP . . . 50

6 Diagrammi di attività **52**
6.1 Diagrammi di attività utente . . . 52
6.1.1 Login . . . 52
6.1.2 Registrazione . . . 53
6.1.3 Modifica dati . . . 54
6.1.4 Ricevimento chiamata . . . 56
6.1.5 Chiama inserendo dato . . . 58
6.1.6 Chiama selezionando da lista . . . 60
6.1.7 Inizializza chiamata . . . 61
6.2 Diagrammi di attività amministratore . . . 62
6.2.1 Login . . . 62
6.2.2 Visualizzazione statistiche amministratore . . . 63

7 Diagrammi di sequenza **64**
7.1 Verifica dell'esistenza dello username . . . 64

8 Stima di fattibilità **65**

9 Tracciamento relazione componenti - requisiti **66**
9.1 Tracciamento package - componenti . . . 66
9.2 Tracciamento componenti - requisiti . . . 67
9.3 Tracciamento requisiti - componenti . . . 70

10 Interfacce Utenti **73**
10.1 GUI 1: Pagina accesso utente base . . . 73
10.1.1 GUI 1.1: Login . . . 73
10.1.2 GUI 1.2: Registrazione . . . 74
10.2 GUI 2: Pagina principale utente base . . . 74
10.2.1 GUI 2.1: Comunicazione . . . 74
10.2.2 GUI 2.1.1: Effettua chiamata . . . 75
10.2.3 GUI 2.1.1.1 Chiamata per IP . . . 75
10.2.4 GUI 2.1.1.2 Chiamata per Username . . . 76
10.2.5 GUI 2.1.1.3 Chiamata per scelta utente da lista . . . 76
10.2.6 GUI 2.1.2 Chiamata in entrata . . . 76
10.2.7 GUI 2.1.3 Informazioni chiamata . . . 76

10.2.8 GUI 2.1.4 Chiamata terminata . . . 77
10.2.9 GUI 2.2: Dati utente . . . 77
10.2.10 GUI 2.2.1: Visualizza dati . . . 77
10.2.11 GUI 2.2.2: Modifica dati . . . 78
10.3 GUI 3: Pagina accesso amministratore . . . 78
10.4 GUI 4: Pagina principale amministratore . . . 79
10.4.1 GUI 4.1 Filtro . . . 79
10.4.2 GUI 4.1.1 Filtro per giorno . . . 80
10.4.3 GUI 4.1.2 Filtro per giudizio . . . 80
10.4.4 GUI 4.1.3 Filtro per utente . . . 80
10.4.5 GUI 4.1.3.1 Filtro per utente da IP . . . 80
10.4.6 GUI 4.1.3.2 Filtro per utente da username . . . 80
10.4.7 GUI 4.1.3.3 Filtro per utente da lista . . . 82

Appendici **83**

A WebRTC **83**
A.1 Descrizione generale . . . 83
A.1.1 Ottenere dati multimediali locali . . . 84
A.1.2 Stabilire la Peer Connection . . . 84
A.1.3 Scambiare dati multimediali . . . 85
A.1.4 Chiudere la connessione . . . 85
A.1.5 Esempio di connessione . . . 85
A.2 L'hole punching . . . 86

B Tecnologie utilizzate **88**
B.1 MySQL . . . 88
B.2 JDBC . . . 88
B.3 JavaScript . . . 88
B.4 WebSocket . . . 88
B.5 HTML5 . . . 89
B.6 CSS3 . . . 89
B.7 AJAX . . . 89
B.8 Java 7 . . . 89
B.9 GWT (Google Web Toolkit) . . . 89

Elenco delle tabelle

1 Versionamento del documento . . . 3
2 Storico ruoli RR ->RP . . . 4
3 Storico ruoli RP ->RQ . . . 4
4 Storico ruoli RQ ->RA . . . 4
5 Tracciamento package - componenti . . . 66
6 Tracciamento componenti - requisiti . . . 69
7 Tracciamento requisiti - componenti . . . 72

Elenco delle figure

1 Schema generale client-server. . . . 12
2 Schema generale del design pattern utilizzato e della sua ripartizione tra client$_{|g|}$ e server$_{|g|}$. . . . 13
3 Diagramma dei package del prodotto MyTalk. . . . 15
4 Diagramma delle classi che illustra la View amministratore. . . . 19
5 Diagramma delle classi che illustra la View utente per le operazioni di login e gestione dati. . . . 20
6 Diagramma delle classi che illustra la View utente per la comunicazione. 21
7 Diagramma delle classi che illustra il Presenter per l'operazione di comunicazione. . . . 30
8 Diagramma delle classi che illustra il Presenter lato Client dell'administrator. . . . 33
9 Diagramma delle classi che illustra il Presenter lato Client dello user. . 34
10 Diagramma delle classi che illustra il Presenter lato server$_{|g|}$ . . . 42
11 Diagramma delle classi che illustra il Model lato Client. . . . 46
12 Diagramma delle classi che illustra il Model lato Client. . . . 46
13 Diagramma delle classi del design pattern Adapter in formato UML$_{|g|}$. . 48
14 Diagramma delle classi del design pattern DAO in formato UML$_{|g|}$. . . 48
15 Diagramma di sequenza del design pattern DAO in formato UML$_{|g|}$. . . 49
16 Diagramma del design pattern MVP in formato UML$_{|g|}$. . . . 50
17 Diagramma di sequenza del design pattern MVP in formato UML$_{|g|}$. . . 51
18 DA 1 - La procedura di Login. . . . 52
19 DA 2 - La procedura di Registrazione. . . . 53
20 DA 3 - La procedura di modifica dei dati dell'utente. . . . 54
21 DA 4 - La procedura di ricezione di una chiamata. . . . 56
22 DA 5 - La procedura di chiamata inserendo un dato conosciuto (username o indirizzo IP). . . . 59
23 DA 6 - La procedura di effettuazione di una chiamata dalla lista. . . . 60
24 DA 6.1 - La procedura di inizializzazione di una chiamata. . . . 61
25 DA 7 - La procedura di inizializzazione di una chiamata. . . . 62
26 DA 8 - La procedura di visualizzazione delle statistiche. . . . 63
27 DS 1 - La procedura di verifica dell'esistenza di un utente di cui si conosce un dato. . . . 64
28 GUI 1.1 - Interfaccia di login. . . . 73
29 GUI 1.2 - Interfaccia di registrazione. . . . 74
30 GUI 2 - Pagina principale utente. . . . 75

31 GUI 2.1.1 - Pannello effettua chiamata. 75
32 GUI 2.1.1.1 - Pannello effettua chiamata attraverso IP$_{|g|}$. 75
33 GUI 2.1.1.2 - Pannello effettua chiamata attraverso username. 76
34 GUI 2.1.1.3 - Pannello effettua chiamata attraverso scelta da lista. . . . 76
35 GUI 2.1.2 - Pannello di chiamata in entrata. 76
36 GUI 2.1.3 - Pannello informazioni chiamata. 77
37 GUI 2.1.4 - Pannello chiamata terminata. 77
38 GUI 2.2 - Dati utente. 78
39 GUI 2.2.2 - Pannello modifica dati. 78
40 GUI 3 - Pagina accesso amministratore. 79
41 GUI 4 - Pagina principale amministratore. 79
42 GUI 4.1.1 - Pannello filtro per giorno. 80
43 GUI 4.1.2 - Pannello filtro per giudizio. 80
44 GUI 4.1.3 - Pannello filtro per utente. 81
45 GUI 4.1.3.1 - Pannello filtro per utente da IP. 81
46 GUI 4.1.3.2 - Pannello filtro per utente da username. 81
47 GUI 4.1.3.3 - Pannello filtro per utente da lista. 82
48 Il Triangolo WebRTC$_{|g|}$. 83
49 Instaurazione della sessione WebRTC$_{|g|}$. 84
50 Instaurazione della sessione WebRTC$_{|g|}$. 85

1 Introduzione

1.1 Scopo del documento

Questo documento ha lo scopo di definire l'architettura del prodotto **MyTalk**. Tale definizione inizia descrivendo il prodotto ad alto livello, dopo la quale segue un'analisi a basso livello, applicando quindi un approccio top-down$_{|g|}$.

1.2 Scopo del prodotto

Il prodotto **MyTalk** è volto ad offrire la possibilità agli utenti di comunicare tra loro, trasmettendo il segnale audio e video, attraverso il browser$_{|g|}$ mediante l'utilizzo di soli componenti standard, senza che sia necessario installare plugin$_{|g|}$ o programmi aggiuntivi (*es.* Skype). Attualmente, infatti, la comunicazione istantanea tra utenti avviene solo tramite componenti non presenti di default nei browser$_{|g|}$. Il software$_{|g|}$ dovrà risiedere in una singola pagina web$_{|g|}$ e dovrà essere basato sulla tecnologia WebRTC$_{|g|}$ (`http://www.webrtc.org`).

1.3 Glossario

Al fine di migliorare la comprensione al lettore ed evitare ambiguità rispetto ai termini tecnici utilizzati nel documento, viene allegato il file *Glossario_v3.0.pdf*, nel quale vengono descritti i termini contrassegnati dal simbolo $_{|g|}$ alla fine della parola. Per i termini composti da più parole, oltre al simbolo $_{|g|}$, è presente anche la sottolineatura.

1.4 Riferimenti

1.4.1 Normativi

- Capitolato d'appalto: **MyTalk**, *software*$_{|g|}$ *di comunicazione tra utenti senza requisiti di installazione*, rilasciato dal proponente *Zucchetti S.p.A.*, reperibile all'indirizzo `http://www.math.unipd.it/~tullio/IS-1/2012/Progetto/C1.pdf`.
- Analisi dei requisiti (allegato *AnalisiDeiRequisiti_v4.0.pdf*).
- Piano di qualifica (allegato *PianoDiQualifica_v3.0.pdf*).
- Piano di progetto (allegato *PianoDiProgetto_v4.0.pdf*).
- Norme di progetto (allegato *NormeDiProgetto_v4.0.pdf*).

1.4.2 Informativi

- Ingegneria del software - Ian Sommerville - 8^a 2007
- UML Distilled - Martin Fowler - 4^a 2010
- Design Patterns - Erich Gamma, Richard Helm, Ralph Johnson, John Vlissides - 1^a 2002

- MVP: Model-View-Presenter - Mike Potel - 1996 reperibile al sito http://www.wildcrest.com/Potel/Portfolio/mvp.pdf
- WebRTC - Alan B. Johnston, Daniel C. Burnett - 1^a 2012
- Slide del docente per l'anno accademico 2012/2013 reperibili al sito http://www.math.unipd.it/~tullio/IS-1/2012/

2 Definizione del prodotto

2.1 Metodo e formalismo di specifica

La specifica di questo documento è realizzata per fornire un'idea generale di quali dovranno essere le componenti di alto livello da definire dettagliatamente nella prossima fase di progettazione.
I contenuti della specifica sono esposti seguendo l'approccio top-down$_{|g|}$: dalla descrizione generale del sistema si scende sempre più in dettaglio passando alla descrizione delle singole componenti.
Infine vengono descritti i design pattern utilizzati e come essi sono stati applicati.
Si è scelto di utilizzare i diagrammi dei package$_{|g|}$, delle classi, di attività e di sequenza relativi allo standard UML$_{|g|}$ 2.0 specificati alla sezione 6 di Norme di Progetto (*NormeDiProgetto_v4.0.pdf*).

2.2 Architettura generale del sistema

Il sistema si basa su un sistema client-server dove il servizio, situato in gran parte sul client, utilizza il server per comunicare con gli altri utenti e per accedere alla base di dati.

Studiando le architetture più comuni per questo tipo di sistemi abbiamo scelto di

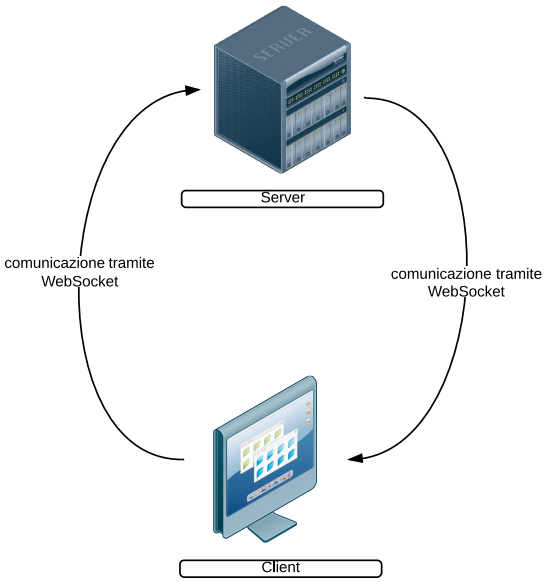

Figura 1: Schema generale client-server.

utilizzare il design pattern architetturale MVP (Model - View - Presenter) nella variante Passive View, per entrambe le tipologie di utenti che fruiranno del servizio.

L'architettura ad alto livello dell'applicazione è rappresentata dalla figura 2.
Viene ora fatta un'analisi delle funzionalità che ciascuna parte dovrà adempiere,

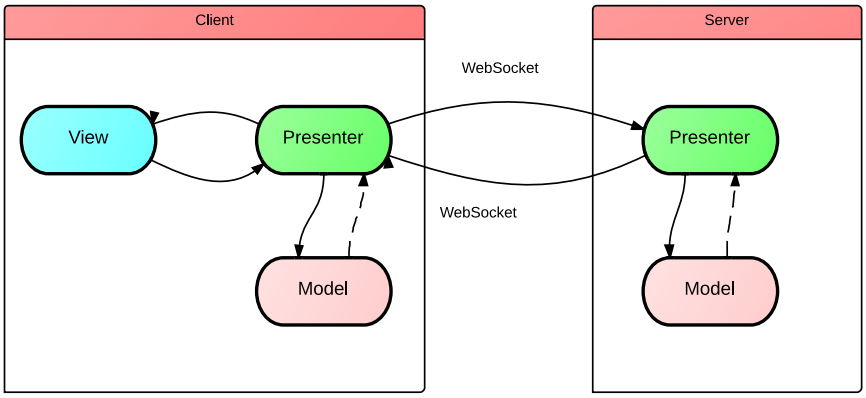

Figura 2: Schema generale del design pattern utilizzato e della sua ripartizione tra client$_{|g|}$ e server$_{|g|}$.

mentre la struttura di questo pattern viene analizzata nella sezione 5.3.

2.2.1 Componente View

La View del sistema è divisa in due parti:

- un'interfaccia utente: per la registrazione, l'autenticazione, la modifica dei dati dell'utente e la visualizzazione della comunicazione e delle relative statistiche;
- un'interfaccia amministratore: per l'autenticazione e per la visualizzazione delle statistiche sulle chiamate effettuate dagli utenti.

Entrambe le parti sono interfacce web$_{|g|}$ e si è scelto di utilizzare HTML5$_{|g|}$ per le componenti statiche e JavaScript$_{|g|}$ per quelle dinamiche. Il layout$_{|g|}$ delle interfacce viene gestito tramite CSS3$_{|g|}$.

2.2.2 Componente Presenter

Il Presenter, che rappresenta la Application Logic dell'applicazione, risiede per la maggior parte sul client$_{|g|}$ e in parte minore sul server$_{|g|}$.
Tale componente svolge quattro ruoli fondamentali:

- comprendere ed elaborare gli input dell'utente;
- instaurare e gestire la comunicazione;
- permettere la comunicazione tra client$_{|g|}$ e server$_{|g|}$;
- aggiornare la View con i dati ottenuti dal model.

Per poterlo fare è dotato di alcune caratteristiche:

- conosce le componenti Model e View. Fa da unico collegamento tra le parti permettendo il loro totale disaccoppiamento;
- è in grado di elaborare gli input della View in strutture logiche che gli permettano di utilizzarli successivamente.

Il Presenter del sistema è diviso in due parti: amministratore e utente.
Le principali funzionalità offerte dal Presenter lato amministratore sono:

- gestire gli eventi di accesso e di uscita dal servizio provenienti dalla componente View;
- gestire gli eventi generati dalla componente View, restituendo la modellazione dei dati reperiti dalla componente Model del server.

Le principali funzionalità offerte dal Presenter lato utente sono:

- gestire gli eventi di accesso e di uscita dal servizio provenienti dalla componente View;
- controllare i dati inseriti dall'utente in fase di registrazione per verificarne la correttezza;
- ricevere e gestire gli eventi riguardanti la visualizzazione e la modifica dei dati dell'utente;
- gestire gli eventi riguardanti la comunicazione, ricevendo i dati dalla componente Model e aggiornando la View.
- gestire la comunicazione peer-to-peer$_{|g|}$ grazie alle librerie WebRTC$_{|g|}$;
- contenere e gestire le informazioni riguardanti le sessioni, sia quella locale che quella remota. Ciò avviene grazie alle librerie WebRTC$_{|g|}$;
- gestire le informazioni riguardanti gli stream$_{|g|}$ audio e video che permettono agli utenti di comunicare; anche questo viene fatto grazie alle librerie WebRTC$_{|g|}$.

2.2.3 Componente Model

Il Model, che rappresenta la Business Logic dell'applicazione, risiede nel server$_{|g|}$ e in minima parte nel client$_{|g|}$.
La principale funzione svolta dal Model lato client$_{|g|}$ è:

- tener traccia dell'utente attualmente autenticato, se questo esiste;

Le principali funzioni svolte dal Model lato server$_{|g|}$ sono:

- ricevere, gestire e soddisfare le richieste che arrivano dal Presenter della componente amministratore e utente;
- interfacciarsi con il database$_{|g|}$.

3 Diagrammi dei package

I diagrammi dei package$_{|g|}$ descrivono le dipendenze che intercorrono tra i vari package$_{|g|}$ che compongono il sistema.
Una dipendenza tra due package$_{|g|}$ si ha quando una modifica di uno può causare una modifica nell'altro. Di seguito è riportato il diagramma dei package$_{|g|}$ dell'applicazione:

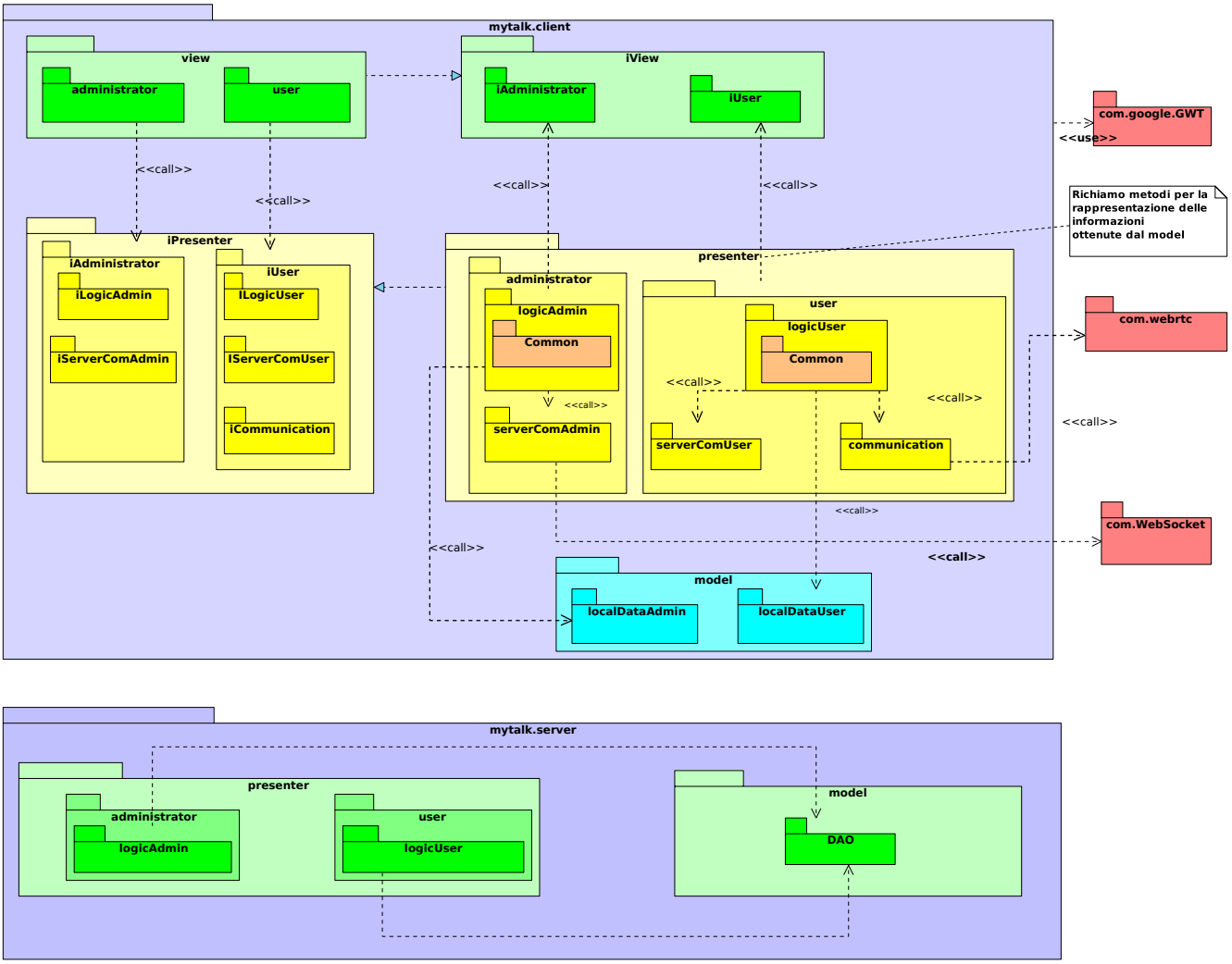

Figura 3: Diagramma dei package del prodotto MyTalk.

L'applicazione è costituita da due package$_{|g|}$ principali;

1. `mytalk.client`: le classi che compongono questo package$_{|g|}$ costituiscono la componente front-end$_{|g|}$ dell'applicazione.

2. `mytalk.server`: le classi che compongono questo package$_{|g|}$ costituiscono la componente back-end$_{|g|}$ dell'applicazione.

La struttura dei sotto-package$_{|g|}$ del package$_{|g|}$ `mytalk.client` ricalca quella del design pattern MVP.
I sotto-package$_{|g|}$ che compongono il package$_{|g|}$ `mytalk.client` sono:

- `mytalk.client.view`: è la componente View del design pattern MVP, implementa il package `mytalk.client.iView`;
- `mytalk.client.presenter`: è la componente Presenter del design pattern MVP, implementa il package `mytalk.client.iPresenter`;
- `mytalk.client.model`: ricalca la componente Model del design pattern MVP;
- `mytalk.client.iView`: contiene le interfacce implementate poi dalle classi che costituiscono la componente View del design pattern MVP, è implementato da `mytalk.client.view`;
- `mytalk.client.iPresenter`: contiene le interfacce implementate poi dalle classi che costituiscono la componente Presenter del design pattern MVP, è implementato da `mytalk.client.presenter`.

I sotto-package$_{|g|}$ che compongono il package$_{|g|}$ `mytalk.server` sono:

- `mytalk.server.presenter`;
- `mytalk.server.model`.

3.1 Package mytalk.client.view

Il package$_{|g|}$ `mytalk.client.view` contiene i seguenti sotto-package$_{|g|}$:

- `mytalk.client.view.administrator`: contiene gli elementi necessari a gestire l'interfaccia grafica e a generare gli eventi della parte grafica dell'amministratore.
- `mytalk.client.view.user`: contiene gli elementi necessari a gestire l'interfaccia grafica e a generare gli eventi della parte grafica dell'utente.

3.2 Package mytalk.client.iView

Il package$_{|g|}$ `mytalk.client.iView` contiene i seguenti sotto-package$_{|g|}$:

- `mytalk.client.iView.iAdministrator`: contiene le interfacce che verranno poi implementate dalle classi che gestiranno l'interfaccia grafica dell'amministratore;
- `mytalk.client.iView.iUser`: contiene le interfacce che verranno poi implementate dalle classi che gestiranno l'interfaccia grafica dell'utente.

3.3 Package mytalk.client.presenter

Il package$_{|g|}$ `mytalk.client.presenter` contiene tutte le classi del Presenter che risiedono nella parte client$_{|g|}$. Il package $_{|g|}$ contiene i seguenti sotto-package$_{|g|}$:

- `mytalk.client.presenter.administrator`:
  contiene tutti i package$_{|g|}$ e le classi che costituiscono la componente Presenter per l'amministratore, il package$_{|g|}$ è diviso nei seguenti sotto-package$_{|g|}$:

  - `mytalk.client.presenter.administrator.logicAdmin`:
    gestisce gli eventi che sono generati dal package$_{|g|}$
    `mytalk.client.view.administrator` e aggiorna la componente grafica dell'amministratore;
  - `mytalk.client.presenter.administrator.serverComAdmin`:
    contiene le componenti che servono ad interfacciarsi al server$_{|g|}$.
  - `mytalk.client.presenter.administrator.common`:
    contiene le componenti in comune tra i vari package$_{|g|}$ che compongono la componente presenter.administrator.

- `mytalk.client.presenter.user`:
  contiene tutti i package$_{|g|}$e le classi che costituiscono la componente Presenter per l'amministratore, il package$_{|g|}$ è diviso nei seguenti sotto-package$_{|g|}$:
  - `mytalk.client.presenter.user.logicUser`:
    gestisce gli eventi che sono generati dal package$_{|g|}$ `mytalk.view.user` e aggiorna la componente grafica dell'utente;
  - `mytalk.client.presenter.user.serverComUser`:
    contiene le componenti che servono ad interfacciarsi al server$_{|g|}$;
  - `mytalk.client.presenter.user.communication`:
    contiene le componenti che servono ad instaurare, gestire ed eventualmente chiudere la chiamata$_{|g|}$;
  - `mytalk.client.presenter.user.logicUser.common`:
    contiene le componenti in comune tra i vari package$_{|g|}$ che compongono la componente presenter.user.

Della componente Presenter fanno parte anche le librerie WebRTC$_{|g|}$ individuate dal package$_{|g|}$ `com.google.WebRTC` e le librerie WebSocket$_{|g|}$ individuate dal package$_{|g|}$ `com.WebSocket`.

3.4 Package mytalk.client.iPresenter

Il package$_{|g|}$ `mytalk.client.iPresenter` contiene tutte le interfacce del Presenter che risiedono nella parte client$_{|g|}$. Il package $_{|g|}$ contiene i seguenti sotto-package$_{|g|}$:

- `mytalk.client.iPresenter.iAdministrator`:
  contiene tutte le interfacce che costituiscono la componente Presenter per l'amministratore, il package$_{|g|}$ è diviso nei seguenti sotto-package$_{|g|}$:
  - `mytalk.client.iPresenter.iAdministrator.iLogicAdmin`:
    fornisce le interfacce che verranno poi implementate dalle classi contenute nel package$_{|g|}$
    `mytalk.client.presenter.administrator.logicAdmin`;
  - `mytalk.client.iPresenter.iAdministrator.iServerComAdmin`:
    fornisce le interfacce che verranno poi implementate dalle classi contenute nel package$_{|g|}$
    `mytalk.client.presenter.administrator.serverComAdmin`.

- `mytalk.client.iPresenter.iUser`:
  contiene tutte le interfacce che costituiscono la componente Presenter per l'amministratore, il package$_{|g|}$ è diviso nei seguenti sotto-package$_{|g|}$:
  - `mytalk.client.iPresenter.iUser.iLogicUser`:
    fornisce le interfacce che saranno poi implementate dal package$_{|g|}$ `mytalk.client.presenter.user.logicUser`;
  - `mytalk.client.presenter.iUser.iServerComUser`:
    fornisce le interfacce che saranno poi implementate dal package$_{|g|}$ `mytalk.client.presenter.user.serverComUser`;
  - `mytalk.client.presenter.iUser.iCommunication`:
    fornisce le interfacce che saranno poi implementate dal package$_{|g|}$ `mytalk.client.presenter.iUser.iCommunication`;

3.5 Package mytalk.client.model

Il package$_{|g|}$ `mytalk.client.model` contiene tutte le classi del Model che risiedono nella parte client$_{|g|}$. Il package$_{|g|}$ contiene i seguenti sotto-package$_{|g|}$:

- `mytalk.client.model.localDataAdmin`:
  contiene le informazioni relative ad un eventuale amministratore autenticato al servizio;
- `mytalk.client.model.localDataUser`:
  contiene le informazioni relative ad un eventuale utente autenticato al servizio.

3.6 Package mytalk.server.presenter

Il package$_{|g|}$ contiene tutte le classi del Presenter che risiedono nella parte server$_{|g|}$. Il package$_{|g|}$ contiene i seguenti sotto-package$_{|g|}$:

- `mytalk.server.presenter.administrator.logicAdmin`:
  questo package$_{|g|}$ contiene tutte le classi dell'amministratore che costituiscono la parte Presenter del server$_{|g|}$;
- `mytalk.server.presenter.user.logicUser`:
  questo package$_{|g|}$ contiene tutte le classi dell'utente che costituiscono la parte Presenter del server$_{|g|}$;

3.7 Package mytalk.server.model

Il package$_{|g|}$ `mytalk.server.model` contiene i seguenti sotto-package$_{|g|}$:

- `mytalk.server.model.dao`: questo package$_{|g|}$contiene gli oggetti e le interfacce che servono a gestire il database$_{|g|}$.

4 Descrizione singoli componenti

4.1 Package mytalk.client.iView

4.1.1 Package mytalk.client.iView.iAdministrator

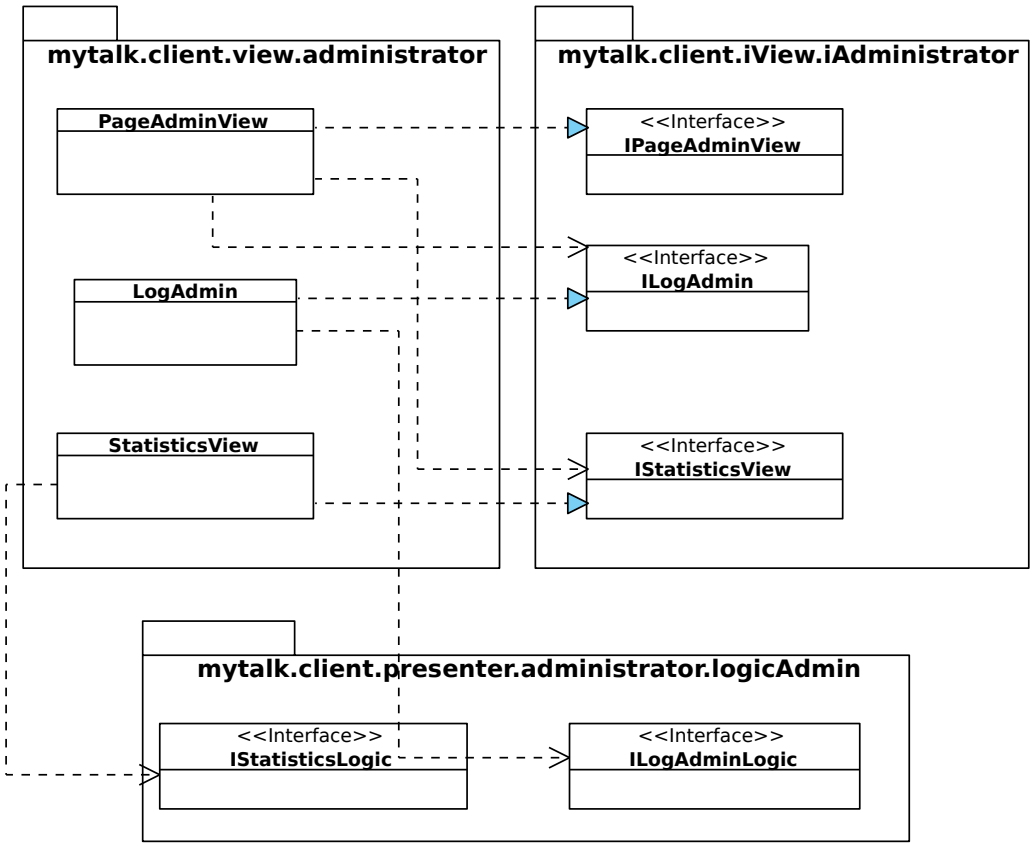

Figura 4: Diagramma delle classi che illustra la View amministratore.

IPageAdminView

Nome: `IPageAdminView`

Tipo: interface

Package: `mytalk.client.iView.iAdministrator`

Descrizione: interfaccia che permette di gestire gli oggetti che compongono la GUI$_{|g|}$ dell'amministratore.

ILogAdmin

Nome: `ILogAdmin`

Tipo: interface

Package: `mytalk.client.iView.iAdministrator`

Descrizione: interfaccia che permette di richiedere, da parte dell'utente amministratore, l'accesso al servizio o alla sua terminazione. Le classi che implementano questa interfaccia hanno il compito di raccogliere ed inoltrare al sottosistema Presenter i dati di accesso, nel caso di un login, e distruggere la sessione utente amministratore in caso di logout.

IStatisticView

Nome: `IStatisticView`

Tipo: interface

Package: `mytalk.client.iView.iAdministrator`

Descrizione: interfaccia che permette, agli oggetti che la implementano, di inviare messaggi al Presenter che permettono il recupero e il filtraggio delle statistiche.

4.1.2 Package mytalk.client.iView.iUser

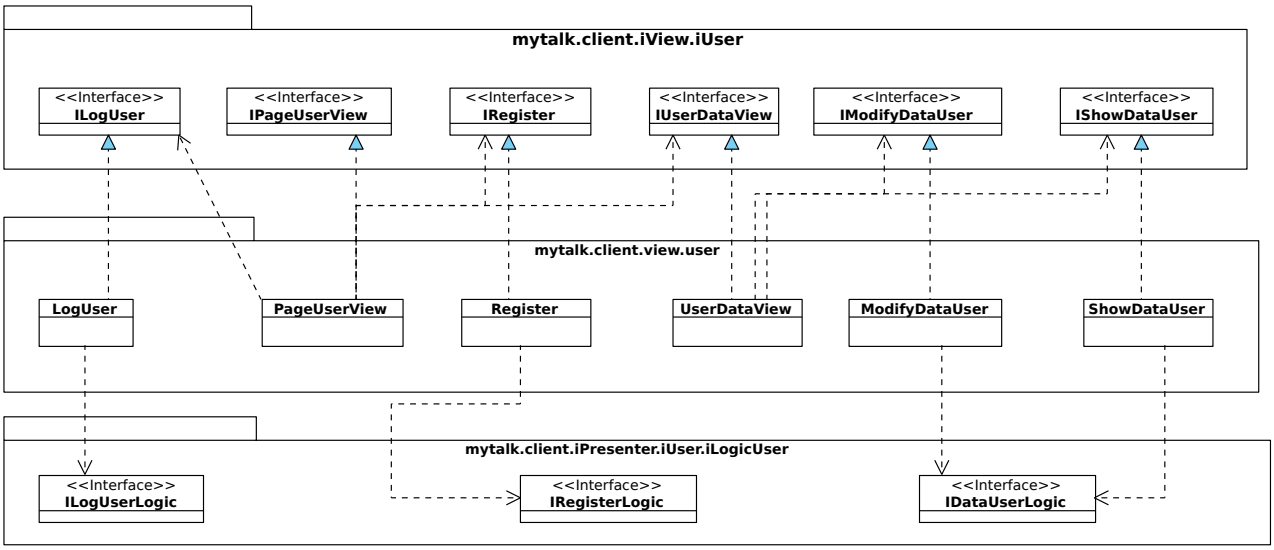

Figura 5: Diagramma delle classi che illustra la View utente per le operazioni di login e gestione dati.

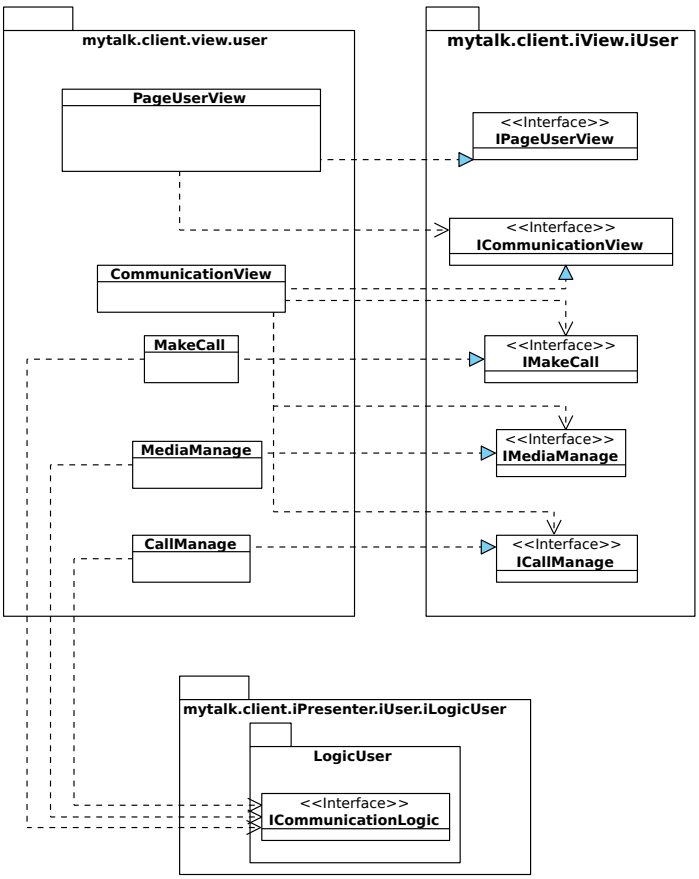

Figura 6: Diagramma delle classi che illustra la View utente per la comunicazione.

IPageUserView

Nome: `IPageUserView`

Tipo: interface

Package: `mytalk.client.iView.iUser`

Descrizione: interfaccia che permette di modificare la pagina dell'utente fruitore del servizio su richiesta del Presenter.

ILogUser

Nome: `ILogUser`

Tipo: interface

Package: `mytalk.client.iView.iUser`

Descrizione: interfaccia che permette di richiedere l'accesso o la terminazione dell'utente all'utilizzo del prodotto. Le classi che implementano questa interfaccia hanno il compito di raccogliere ed inoltrare al sottosistema Presenter i dati di accesso, nel caso di un login, e distruggere la sessione utente in caso di logout.

IRegister

Nome: `IRegister`

Tipo: interface

Package: `mytalk.client.iView.iUser`

Descrizione: interfaccia che rappresenta la richiesta di inserire i dati per la creazione di un nuovo utente. Le classi che implementano questa interfaccia hanno il compito di inviare i dati inseriti dall'utente al livello Presenter.

IUserDataView

Nome: `IUserDataView`

Tipo: interface

Package: `mytalk.client.iView.iUser`

Descrizione: interfaccia che permette all'utente di inviare al Presenter una richiesta per recuperare i propri dati e gestirli.

IShowDataUser

Nome: `IShowDataUser`

Tipo: interface

Package: `mytalk.client.iView.iUser`

Descrizione: interfaccia che rappresenta una richiesta di visualizzazione dei dati dell'utente selezionato. Le classi che implementano questa interfaccia hanno il compito di inviare al Presenter una richiesta di recupero dei dati dell'utente autenticato.

IModifyDataUser

Nome: `IModifyDataUser`

Tipo: interface

Package: `mytalk.client.iView.iUser`

Descrizione: interfaccia che permette di richiedere la modifica dei dati relativi all'utente selezionato. Le classi che implementano questa interfaccia hanno il compito di raccogliere i nuovi dati inseriti e di inviare una richiesta al Presenter che poi provvederà a modificarli.

ICommunicationView

Nome: `ICommunicationView`

Tipo: interface

Package: `mytalk.client.iView.iUser`

Descrizione: interfaccia che rappresenta la costruzione della $GUI_{|g|}$ per la gestione della comunicazione da parte dell'utente attuale.

ICallManage

Nome: `ICallManage`

Tipo: interface

Package: `mytalk.client.iView.iUser`

Descrizione: interfaccia che rappresenta la gestione delle comunicazioni in atto o in entrata. Le classi che implementano questa interfaccia hanno il compito di inviare segnali al Presenter, su richiesta dell'utente, che permettono di gestire la comunicazione.

IMediaManage

Nome: `IMediaManage`

Tipo: interface

Package: `mytalk.client.iView.iUser`

Descrizione: le classi che implementano questa interfaccia hanno il compito di emettere opportuni segnali al Presenter, per la gestione dei media.

IMakeCall

Nome: `IMakeCall`

Tipo: interface

Package: `mytalk.client.iView.iUser`

Descrizione: interfaccia che permette di richiedere l'effettuazione di una nuova comunicazione dato un riferimento all'utente destinatario. Le classi che implementano questa interfaccia hanno il compito di inviare i dati di comunicazione al livello Presenter e mostrare lo stato della chiamata all'utente.

4.2 Package mytalk.client.view

4.2.1 Package mytalk.client.view.administrator

PageAdminView

Nome: `PageAdminView`

Tipo: class

Package: `mytalk.client.view.administrator`

Descrizione: la classe ha il compito di gestire la creazione dei widget$_{|g|}$ che compongono l'interfaccia dell'amministratore su richiesta del Presenter.

Relazioni con altre componenti: la classe implementa l'interfaccia `mytalk.client.iView.iAdministrator.IPageAdminView`.
Richiama inoltre metodi delle classi `mytalk.client.view.administrator.LogAdminView`, `mytalk.client.view.administrator.StatisticView` e `mytalk.client.presenter.administrator.logicAdmin.UpdateViewLogic`.

LogAdmin

Nome: `LogAdmin`

Tipo: class

Package: `mytalk.client.view.administrator`

Descrizione: la classe ha il compito di creare i widget$_{|g|}$ che compongono la parte della pagina web$_{|g|}$ che permette all'amministratore di autenticarsi e di uscire dal servizio. La classe ha anche il compito di dialogare con il Presenter per verificare la correttezza dei dati inseriti e per dargli istruzione di distruggere i dati della sessione.

Relazioni con altre componenti: la classe implementa l'interfaccia `mytalk.client.iView.iAdministrator.ILogAdmin`
e fa parte per composizione della classe `mytalk.client.view.administrator.PageAdminView`.
Richiama inoltre metodi della classe `mytalk.client.presenter.administrator.logicAdmin.LogAdminLogic` per gestire gli eventi generati dall'utente.

StatisticView

Nome: `StatisticView`

Tipo: class

Package: `mytalk.client.view.administrator`

Descrizione:questa classe mette a disposizione dell'amministratore i widget$_{|g|}$ che gli permettono di visualizzare e filtrare, tramite vari parametri, le statistiche inviate dagli utenti che hanno effettuato una comunicazione e che sono memorizzate nel sistema, la classe dialoga con il Presenter per richiedere le statistiche.

Relazioni con altre componenti: la classe implementa l'interfaccia `mytalk.client.iView.iAdministrator.IStatisticView` e fa parte per composizione della classe `mytalk.client.view.administrator.PageAdminView`. Richiama inoltre metodi della classe `mytalk.client.presenter.administrator.logicAdmin.StatisticLogic` per gestire gli eventi generati dall'utente.

4.2.2 Package mytalk.client.view.user

PageUserView

Nome: `PageUserView`

Tipo: class

Package: `mytalk.client.view.user`

Descrizione: la classe ha il compito di gestire la creazione dei widget$_{|g|}$, su richiesta del Presenter, che compongono l'interfaccia dell'utente, con le informazioni che il Presenter provvederà a fornire.

Relazioni con altre componenti: la classe implementa l'interfaccia `mytalk.client.iView.iUser.IPageUserView`.
Richiama inoltre metodi delle classi `mytalk.client.view.user.LogUserView`, `mytalk.client.view.user.RegisterView`, `mytalk.client.view.user.UserDataView`, `mytalk.client.view.user.CommunicationView`, `mytalk.client.presenter.user.logicUser.UpdateViewLogic` e `mytalk.client.presenter.user.logicUser.serverComUser.WebSocketUser`.

LogUser

Nome: `LogUser`

Tipo: class

Package: `mytalk.client.view.user`

Descrizione: tale classe ha il compito di generare l'evento che permette al Presenter di gestire correttamente la fase di autenticazione al sistema, visualizzando eventualmente un output adeguato all'esito dell'operazione. La classe ha anche il compito di generare l'evento per l'uscita dal servizio.

Relazioni con altre componenti: la classe implementa l'interfaccia `mytalk.client.iView.iUser.ILogUser`.
Richiama inoltre metodi delle classi `mytalk.client.presenter.user.logicUser.LogUserLogic` e `mytalk.client.presenter.user.logicUser.UpdateViewLogic` per gestire gli eventi generati dall'utente.

Register

Nome: `Register`

Tipo: class

Package: `mytalk.client.view.user`

Descrizione: la classe `mytalk.client.view.user.PageUserView`, su richiesta del Presenter visualizza all'utente un form dove egli potrà inserire i propri dati per la registrazione. Tali dati vengono passati al livello Presenter dalla classe Register, il Presenter effettuerà la registrazione al servizio qualora i dati siano corretti.

Relazioni con altre componenti: la classe implementa l'interfaccia `mytalk.client.iView.iUser.IRegister`.
Richiama inoltre metodi delle classi `mytalk.client.presenter.user.logicUser.RegisterLogic` e `mytalk.client.presenter.user.logicUser.UpdateViewLogic` per gestire gli eventi generati dall'utente nella fase di registrazione.

UserDataView

Nome: `UserDataView`

Tipo: class

Package: `mytalk.client.view.user`

Descrizione:questa classe ha il compito di gestire la visualizzazione e di mettere a disposizione le opportune form che l'utente può utilizzare per modificare i dati.

Relazioni con altre componenti: la classe implementa l'interfaccia `mytalk.client.iView.iUser.IUserDataView` ed e' composta da due oggetti, uno di tipo
`mytalk.client.view.user.ShowDataUser` e uno di tipo
`mytalk.client.view.user.ModifyDataUser`.
Richiama inoltre metodi delle classi
`mytalk.client.presenter.user.logicUser.DataUserLogic` e `mytalk.client.presenter.user.logicUser.UpdateViewLogic` per gestire gli eventi generati dall'utente ed aggiornare l'interfaccia.

ShowDataUser

Nome: `ShowDataUser`

Tipo: class

Package: `mytalk.client.view.user`

Descrizione: tale classe ha la responsabilità di richiamare opportune funzioni del Presenter, che richiedono il reperimento delle informazioni personali dell'utente.

Relazioni con altre componenti: la classe implementa l'interfaccia `mytalk.client.iView.iUser.IShowDataUser`. Richiama inoltre metodi della classe `mytalk.client.presenter.user.logicUser.DataUserLogic` per gestire gli eventi generati dall'utente, nella gestione del proprio account.

ModifyDataUser

Nome: `ModifyDataUser`

Tipo: class

Package: `mytalk.client.view.user`

Descrizione: la classe ha il compito di permettere all'utente il cambiamento dei propri dati personali. La classe ha il compito di emettere eventi, su richiesta dell'utente, che richiedono al Presenter di modificare un certo dato.

Relazioni con altre componenti: la classe implementa l'interfaccia `mytalk.client.iView.iUser.IModifyDataUser`. Richiama inoltre metodi della classe `mytalk.client.presenter.user.logicUser.DataUserLogic` per gestire gli eventi generati dall'utente, nella gestione del proprio account.

CommunicationView

Nome: `CommunicationView`

Tipo: class

Package: `mytalk.client.view.user`

Descrizione: la classe crea e distribuisce i vari widget$_{|g|}$ nella pagina che consentono all'utente di comunicare, chiudere la comunicazione, visualizzare i media, inviare richieste di comunicazione ad altri utenti e chiudere le comunicazioni in atto.

Relazioni con altre componenti: della classe fanno parte per composizione le classi `mytalk.client.view.user.MediaManage`, `mytalk.client.view.user.MakeCall`, `mytalk.client.view.user.CallManage`.
Richiama inoltre metodi delle classi `mytalk.client.presenter.client.user.logicUser.CommunicationLogic`, `mytalk.client.view.user.MakeCall` e `mytalk.client.presenter.client.user.logicUser.UpdateViewLogic` per gestire l'interfaccia di comunicazione utente.

CallManage

Nome: `CallManage`

Tipo: class

Package: `mytalk.client.view.user`

Descrizione: tale classe permette all'utente di gestire chiamate in arrivo, permettendo all'utente di accettare o rifiutare la chiamata. Nel caso la chiamata sia accettata, la classe invierà opportuna comunicazione al livello Presenter il quale provvederà ad instaurare la chiamata e ad aggiornare la $GUI_{|g|}$.

Relazioni con altre componenti: la classe implementa l'interfaccia `mytalk.client.iView.iUser.ICallManage`.
Richiama inoltre metodi della classe `mytalk.client.presenter.user.logicUser.CommunicationLogic` per gestire gli eventi generati dall'utente, nella gestione e/o instaurazione della chiamata.

MediaManage

Nome: `MediaManage`

Tipo: class

Package: `mytalk.client.view.user`

Descrizione: la classe contiene i vari eventi generabili dall'utente che servono alla gestione dei media.

Relazioni con altre componenti: la classe implementa l'interfaccia `mytalk.client.iView.iUser.IMediaManage`.
Richiama inoltre metodi della classe `mytalk.client.presenter.user.logicUser.CommunicationLogic` per gestire gli eventi generati dall'utente, nella gestione e/o instaurazione della chiamata.

MakeCall

Nome: `MakeCall`

Tipo: class

Package: `mytalk.client.view.user`

Descrizione: la classe permette all'utente di chiamare altri utenti, richiamando opportuni metodi del Presenter con i dati inseriti dall'utente.

Relazioni con altre componenti: la classe implementa l'interfaccia `mytalk.client.iView.iUser.IMakeCall`.
Richiama inoltre metodi della classe `mytalk.client.presenter.user.logicUser.CommunicationLogic` per gestire gli eventi generati dall'utente, nella gestione e/o instaurazione della chiamata.

4.3 Package mytalk.client.iPresenter

4.3.1 Package mytalk.client.iPresenter.iAdministrator.iLogicAdmin

ILogAdminLogic

Nome: `ILogAdminLogic`

Tipo: interface

Package: `mytalk.client.iPresenter.iAdministrator.iLogicAdmin`

Descrizione: interfaccia che rappresenta l'accesso alle classi logiche che hanno il compito di gestire il login e logout dell'utente amministratore al servizio.

IStatisticLogic

Nome: `IStatisticLogic`

Tipo: interface

Package: `mytalk.client.iPresenter.iAdministrator.iLogicAdmin`

Descrizione: interfaccia che rappresenta l'accesso alle classi logiche che hanno il compito di recuperare ed elaborare i dati statistici delle chiamate effettuate.

IUpdateViewLogic

Nome: `IUpdateViewLogic`

Tipo: interface

Package: `mytalk.client.iPresenter.iAdministrator.iLogicAdmin`

Descrizione: interfaccia che fornisce le operazioni per aggiornare l'interfaccia grafica.

4.3.2 Package mytalk.client.iPresenter.iAdministrator.iServerComAdmin

IWebSocketAdmin

Nome: `IWebSocketAdmin`

Tipo: interface

Package: `mytalk.client.iPresenter.iAdministrator.iServerComAdmin`

Descrizione: interfaccia che rappresenta le classi logiche dell'amministratore che hanno il compito di formattare i dati delle altre classi per poi trasmetterli al server$_{|g|}$ tramite WebSocket$_{|g|}$.

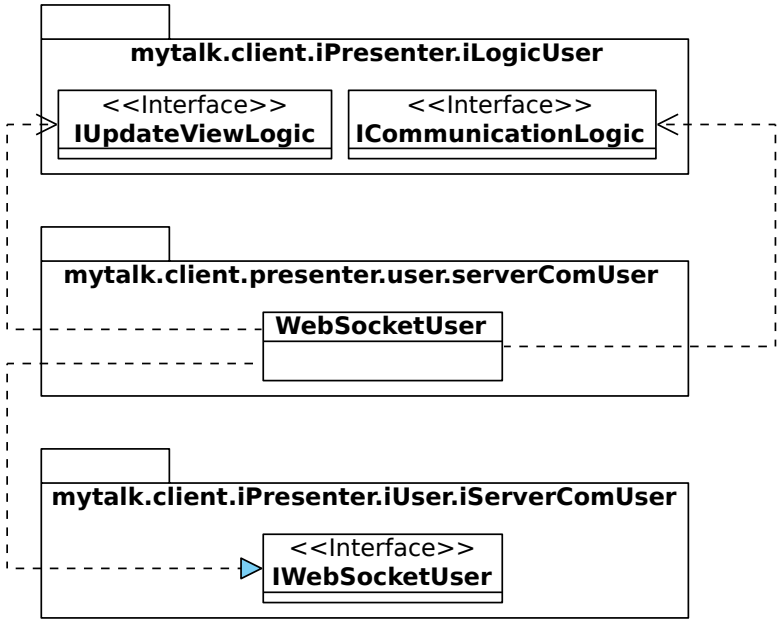

Figura 7: Diagramma delle classi che illustra il Presenter per l'operazione di comunicazione.

4.3.3 Package mytalk.client.iPresenter.iUser.iLogicUser

ILogUserLogic

Nome: `ILogUserLogic`

Tipo: interface

Package: `mytalk.client.iPresenter.iUser.iLogicUser`

Descrizione: interfaccia che rappresenta l'accesso alle classi logiche che hanno il compito di gestire il login e logout dell'utente al servizio.

IRegisterLogic

Nome: `IRegisterLogic`

Tipo: interface

Package: `mytalk.client.iPresenter.iUser.iLogicUser`

Descrizione: interfaccia che rappresenta l'accesso alle classi logiche che hanno il compito di gestire i dati di registrazione inseriti dall'utente nella View.

IDataUserLogic

Nome: `IDataUserLogic`

Tipo: interface

Package: `mytalk.client.iPresenter.iUser.iLogicUser`

Descrizione: interfaccia che rappresenta l'accesso agli oggetti che gestiscono il recupero e la modifica, verificandone la conformità, dei dati propri dell'utente e inoltra le richieste di comunicazione alla componente dedita alla comunicazione con il server$_{|g|}$.

IUpdateViewLogic

Nome: `IUpdateViewLogic`

Tipo: interface

Package: `mytalk.client.iPresenter.iUser.iLogicUser`

Descrizione: interfaccia che fornisce le operazioni per aggiornare l'interfaccia grafica.

ICommunicationLogic

Nome: `ICommunicationLogic`

Tipo: interface

Package: `mytalk.client.iPresenter.iUser.iLogicUser`

Descrizione: interfaccia che rappresenta l'accesso alle classi che hanno il compito di comporre gli oggetti che serviranno per instaurare una nuova comunicazione, gestire le comunicazioni in ingresso e l'accesso ai media locali e remoti.

4.3.4 Package mytalk.client.iPresenter.iUser.iCommunication

IPeerConnection

Nome: `IPeerConnection`

Tipo: interface

Package: `mytalk.client.iPresenter.iUser.iCommunication`

Descrizione: interfaccia che fornisce le operazioni per la gestione della connessione peer-to-peer.

IMediaStream

Nome: `IMediaStream`

Tipo: interface

Package: `mytalk.client.iPresenter.iUser.iCommunication`

Descrizione: interfaccia che fornisce le operazioni per gestire ed ottenere i media locali.

4.3.5 Package mytalk.client.iPresenter.iUser.iServerComUser

IWebSocketUser

Nome: `IWebSocketUser`

Tipo: interface

Package: `mytalk.client.iPresenter.iUser.iServerComUser`

Descrizione: interfaccia che rappresenta le classi logiche dell'utente che hanno il compito di formattare i dati delle altre classi per poi trasmetterli al server$_{|g|}$ tramite WebSocket$_{|g|}$.

4.4 Package mytalk.client.presenter

Di seguito è riportata la descrizione delle classi per quanto concerne la parte del Presenter che risiede sul client$_{|g|}$.

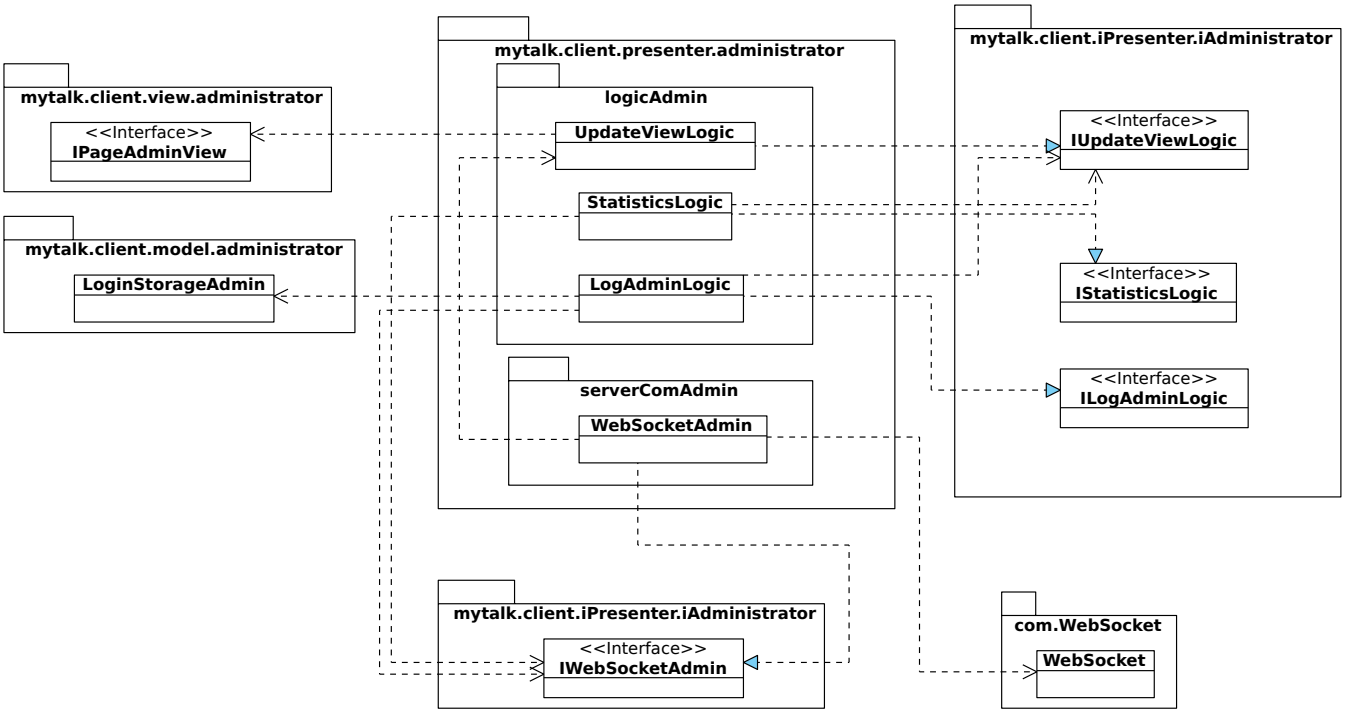

Figura 8: Diagramma delle classi che illustra il Presenter lato Client dell'administrator.

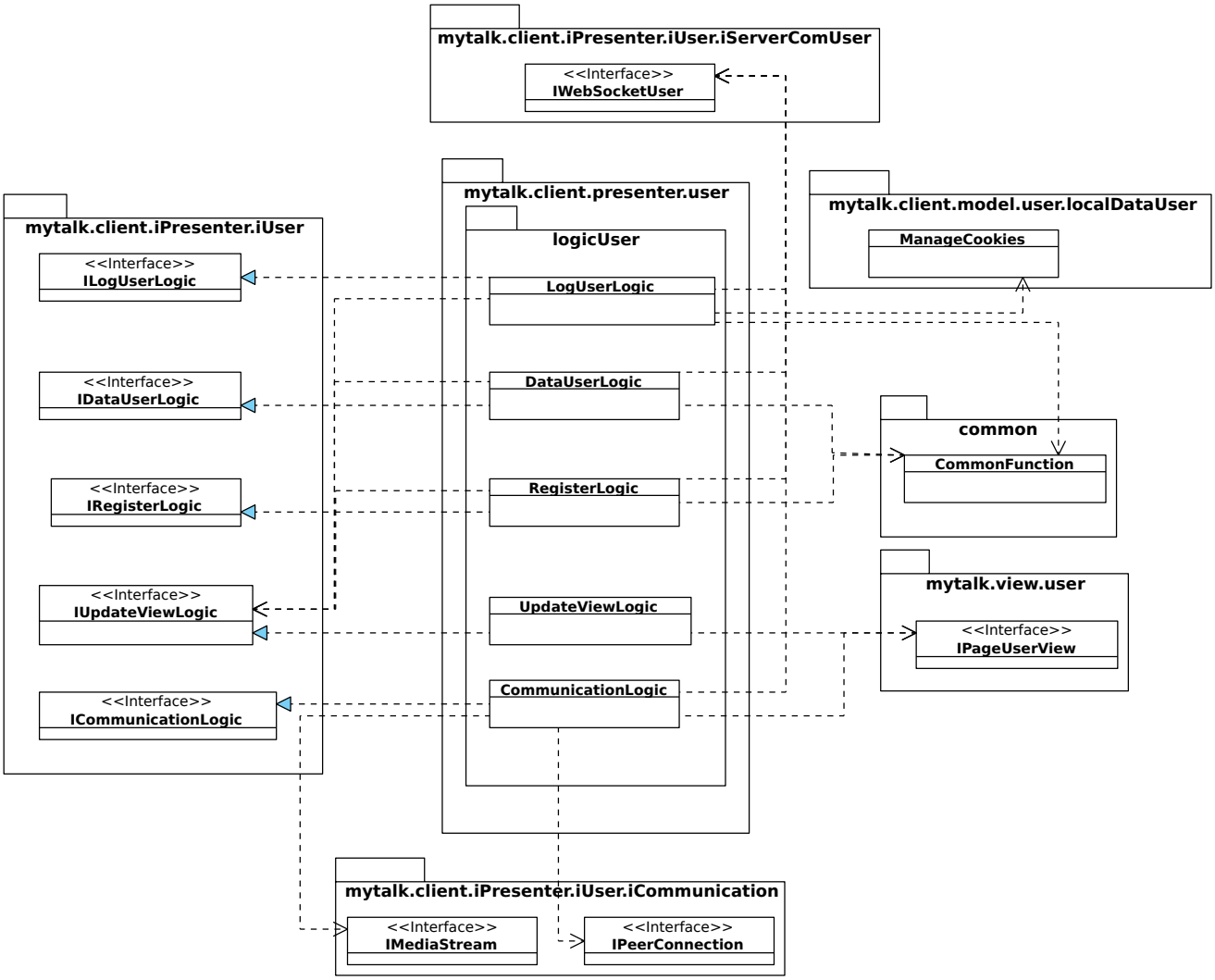

Figura 9: Diagramma delle classi che illustra il Presenter lato Client dello user.

4.4.1 Package mytalk.client.presenter.administrator.logicAdmin

LogAdminLogic

Nome: `LogAdminLogic`

Tipo: class

Package: `mytalk.client.presenter.administrator.LogicAdmin`

Descrizione: questa classe ha il compito di gestire gli eventi provenienti dalla componente View, di accesso ed uscita dal servizio, in particolare dalla classe `mytalk.client.view.administrator.LogAdmin`. La classe, dopo aver opportunamente gestito l'evento, ritorna un responso sull'esito dell'operazione alla View. Nel caso di login, il responso sarà o il reindirizzamento alla pagina principale (segno di autenticazione riuscita) o un opportuno messaggio che descrive la natura dell'errore avvenuto. Nel caso di logout, il responso sarà il reindirizzamento alla pagina di login.

Relazioni con altre componenti: implementa l'interfaccia `mytalk.client.iPresenter.iAdministrator.iLogicAdmin.ILogAdminLogic`, richiama metodi delle classi `mytalk.client.presenter.administrator.serverComAdmin.WebSocketAdmin` e `mytalk.client.presenter.user.logicUser.common.CommonFunctions` per inviare e ricevere messaggi dal server$_{|g|}$, comunica con la classe `mytalk.client.presenter.administrator.logicAdmin.UpdateViewLogic` per gestire l'aggiornamento della pagina e comunicare l'esito delle operazioni effettuate all'utente.

StatisticLogic

Nome: `StatisticLogic`

Tipo: class

Package: `mytalk.client.presenter.administrator.logicAdmin`

Descrizione: questa classe deve poter gestire gli eventi generati dalla View dell'amministratore, in particolare dalla classe `mytalk.client.view.administrator.StatisticView`, la quale è responsabile dell'invio degli eventi per la visualizzazione delle statistiche. La classe avrà il compito di reperire ed inviare alla View le statistiche raccolte dai vari client$_{|g|}$.

Relazioni con altre componenti: implementa l'interfaccia `mytalk.client.iPresenter.iAdministrator.iLogicAdmin.IStatisticsLogic`, richiama metodi delle classi `mytalk.client.presenter.administrator.logicAdmin.UpdateViewLogic` e `mytalk.client.presenter.administrator.logicAdmin.serverComAdmin.WebSocketAdmin` per inviare e ricevere messaggi dal server$_{|g|}$, comunica con la classe `mytalk.client.presenter.administrator.logicAdmin.UpdateViewLogic` per gestire l'aggiornamento della pagina e comunicare l'esito delle operazioni effettuate all'utente.

UpdateViewLogic

Nome: `UpdateViewLogic`

Tipo: class

Package: `mytalk.client.presenter.administrator.logicAdmin`

Descrizione: la classe avrà il compito di aggiornare l'interfaccia grafica in seguito ad eventi generati da altre classi della componente Presenter.

Relazioni con altre componenti: implementa l'interfaccia `mytalk.client.iPresenter.iAdministrator.iLogicAdmin.IUpdateViewLogic`, richiama metodi della classe `mytalk.client.view.administrator.logicAdmin.PageAdminView` per gestire gli aggiornamenti dell'interfaccia grafica.

4.4.2 Package mytalk.client.presenter.administrator.logicAdmin.common

CommonFunctions

Nome: `CommonFunctions`

Tipo: class

Package: `mytalk.client.presenter.administrator.logicAdmin.common`

Descrizione: classe che fornisce i metodi comuni a tutte le classi.

4.4.3 Package mytalk.client.presenter.administrator.serverComAdmin

WebSocketAdmin

Nome: `WebSocketAdmin`

Tipo: class

Package: `mytalk.client.presenter.administrator.serverComAdmin`

Descrizione: la classe permette di formattare i dati e le richieste degli amministratori per poi trasmetterli al server$_{|g|}$ tramite WebSocket$_{|g|}$.

Relazioni con altre componenti: la classe implementa l'interfaccia `mytalk.client.iPresenter.iAdministrator.iServerComAdmin.IWebSocketAdmin`, richiama metodi della classe `mytalkadmin.client.presenter.logicAdmin.UpdateViewLogic`.

4.4.4 Package mytalk.client.presenter.user.serverComUser

WebSocketUser

Nome: `WebSocketUser`

Tipo: class

Package: `mytalk.client.presenter.user.serverComUser`

Descrizione: la classe permette di formattare i dati e le richieste degli utenti per poi trasmetterli al server$_{|g|}$ tramite WebSocket$_{|g|}$.

Relazioni con altre componenti: la classe implementa l'interfaccia `IWebSocketUser`, richiama metodi delle classi `com.WebSocket.WebSocket`, `mytalk.client.presenter.user.logicUser.CommunicationLogic` e `mytalk.client.presenter.user.logicUser.UpdateViewLogic` per gestire gli aggiornamenti dell'interfaccia grafica.

4.4.5 Package mytalk.client.presenter.user.logicUser

LogUserLogic

Nome: `LogUserLogic`

Tipo: class

Package: `mytalk.client.presenter.user.logicUser`

Descrizione: questa classe ha il compito di gestire gli eventi provenienti dalla classe `mytalk.client.view.user.LogUser`. La classe ha quindi il compito di inoltrare la richiesta di login alla classe che ha il compito di comunicare con il server$_{|g|}$. La classe ha anche il compito di effettuare il logout dell'utente distruggendo i dati relativi alla sessione.

Relazioni con altre componenti: la classe implementa l'interfaccia `mytalk.client.iPresenter.iUser.iLogicUser.ILogUserLogic`, richiama metodi delle classi `mytalk.client.presenter.user.logicUser.common.CommonFunctions` e `mytalk.client.presenter.user.logicUser.serverComUser.WebSocketUser` per inviare e ricevere messaggi dal server$_{|g|}$, comunica con la classe `mytalk.client.presenter.user.logicUser.UpdateViewLogic` per gestire l'aggiornamento della pagina e comunicare l'esito delle operazioni effettuate all'utente.

RegisterLogic

Nome: `RegisterLogic`

Tipo: class

Package: `mytalk.client.presenter.user.logicUser`

Descrizione: la classe ha il compito di gestire gli eventi generati dalla classe `mytalk.client.view.Register`. La classe controlla i dati, dialogando con il server$_{|g|}$(tramite la classe `mytalk.client.presenter.user.serverComUser.WebSocketUser`) e se questi risultano essere corretti procede con la registrazione e, successivamente, aggiorna la View (tramite la classe `mytalk.client.presenter.user.logicUser.UpdateViewLogic`). Altrimenti, riporta all'utente un opportuno messaggio per informarlo dell'errore.

Relazioni con altre componenti: la classe implementa l'interfaccia `mytalk.client.iPresenter.iUser.iLogicUser.IRegisterLogic`, richiama metodi delle classi `mytalk.client.presenter.user.logicUser.common.CommonFunctions` e `mytalk.client.presenter.user.serverComUser.WebSocketUser` per inviare e ricevere messaggi dal server$_{|g|}$, comunica con la classe `mytalk.client.presenter.user.logicUser.UpdateViewLogic` per gestire l'aggiornamento della pagina e comunicare l'esito delle operazioni effettuate all'utente.

DataUserLogic

Nome: `DataUserLogic`

Tipo: class

Package: `mytalk.client.presenter.user.logicUser`

Descrizione: tale classe ha il compito di ricevere e gestire gli eventi riguardanti la visualizzazione e la modifica di informazioni personali dell'utente provenienti dall'interfaccia grafica. Dopo aver ricevuto e gestito gli eventi, tale classe ha inoltre il compito di aggiornare l'interfaccia grafica con eventuali errori riscontrati nel controllo , mediante l'interfaccia `mytalk.client.presenter.user.logicUser.UpdateViewLogic`.

Relazioni con altre componenti: la classe implementa l'interfaccia `mytalk.client.iPresenter.iUser.iLogicUser.IDataUserLogic`, richiama metodi della classe `mytalk.client.presenter.user.serverComUser.WebSocketUser` per inviare e ricevere messaggi dal server$_{|g|}$, comunica con la classe `mytalk.client.presenter.user.logicUser.UpdateViewLogic` per gestire l'aggiornamento della pagina e comunicare l'esito delle operazioni effettuate all'utente.

UpdateViewLogic

Nome: `UpdateViewLogic`

Tipo: class

Package: `mytalk.client.presenter.user.logicUser`

Descrizione: la classe avrà il compito di aggiornare l'interfaccia grafica in seguito ad eventi generati da altre classi della componente Presenter.

Relazioni con altre componenti: implementa l'interfaccia `mytalk.client.iPresenter.iUser.iLogicUser.IUpdateViewLogic`, richiama metodi delle classi `mytalk.client.view.user.PageUserView`, `mytalk.client.presenter.user.logicUser.common.CommonFunction` e `mytalk.client.presenter.user.serverComUser.WebSocketUser` per gestire gli aggiornamenti dell'interfaccia grafica.

CommunicationLogic

Nome: `CommunicationLogic`

Tipo: class

Package: `mytalk.client.presenter.user.logicUser`

Descrizione: tale classe ha il compito di gestire gli eventi provenienti dalla classe `mytalk.client.view.user.CommunicationView`. Gli eventi che gestisce questa classe sono quindi quelli che riguardano la comunicazione: effettuare e ricevere chiamate e gestione degli stream$_{|g|}$ audio e video. La classe, dopo aver ricevuto i dati dalla componente Model, aggiorna opportunamente la View visualizzando eventuali messaggi per l'utente ed aggiornando gli stream$_{|g|}$, il tutto tramite l'interfaccia `mytalk.client.view.user.IPageUserView`.

Relazioni con altre componenti: implementa l'interfaccia `mytalk.client.iPresenter.iUser.iLogicUser.ICommunicationLogic`, richiama metodi delle classi `mytalk.client.presenter.user.communication.PeerConnection`, `mytalk.client.presenter.user.communication.MediaStream` tramite le interfacce: `mytalk.client.iPresenter.iUser.iCommunicaton.IPeerConnection`, `mytalk.client.iPresenter.iUser.iCommunication.IMediaStream` per poter stabilire e gestire il canale di comunicazione. Richiama inoltre metodi della classe `mytalk.client.presenter.user.serverComUser.WebSocketUser` per inviare e ricevere messaggi dal server$_{|g|}$ e comunica con la classe `mytalk.client.presenter.user.logicUser.UpdateViewLogic` per gestire l'aggiornamento della pagina e comunicare l'esito delle operazioni effettuate all'utente.

4.4.6 Package mytalk.client.presenter.user.logicUser.common

CommonFunctions

Nome: `CommonFunctions`

Tipo: class

Package: `mytalk.client.presenter.user.logicUser.common`

Descrizione: classe che fornisce i metodi comuni a tutte le classi.

4.4.7 Package mytalk.client.presenter.user.communication

PeerConnection

Nome: `PeerConnection`

Tipo: class

Package: `mytalk.client.presenter.user.communication`

Descrizione: tale classe è la componente adapter del design pattern Adapter. Ha il compito di gestire la comunicazione peer-to-peer$_{|g|}$. Questo è reso possibile tramite la componente adaptee che è la classe WebRTCAdaptee che contiene le librerie WebRTC$_{|g|}$.

Relazioni con altre componenti: implementa l'interfaccia `mytalk.client.iPresenter.iUser.iCommunication.IPeerConnection`, richiama metodi della classe `com.webrtc.WebRTCAdaptee`.

MediaStream

Nome: `MediaStream`

Tipo: class

Package: `mytalk.client.presenter.user.communication`

Descrizione: tale classe è la componente adapter del design pattern Adapter. Ha il compito di gestire le informazioni riguardanti gli stream$_{|g|}$ audio e video che permettono agli utenti di comunicare. Questo è reso possibile tramite la componente adaptee che è la classe MediaStream delle librerie getUserMedia$_{|g|}$.

Relazioni con altre componenti: implementa l'interfaccia `mytalk.client.presenter.iUser.iCommunication.IMediaStream` e richiama metodi delle API `getUserMedia`.

4.4.8 com.webrtc

WebRTCAdaptee

Nome: `WebRTCAdaptee`

Tipo: class

Package: `com.webrtc`

Descrizione: tale classe è la componente adaptee del design pattern Adapter. Ha il compito di gestire la comunicazione peer-to-peer$_{|g|}$.

Relazioni con altre componenti: è utilizzata dalla classe `mytalk.client.presenter.user.communication.PeerConnection`.

4.4.9 com.WebSocket

WebSocket

Nome: `WebSocket`

Tipo: class

Package: `com.WebSocket`

Descrizione: tale classe è il webSocket$_{|g|}$ che ha il compito di trasmettere e di ricevere i dati al server$_{|g|}$.

MediaStream

Nome: `MediaStream`

Tipo: class

Package: `com.WebRTC`

Descrizione: tale classe è la componente adaptee del design pattern Adapter. Ha il compito di contenere e gestire le informazioni relative ai media di un peer$_{|g|}$ e fa parte delle API `getUserMedia`.

Relazioni con altre componenti: è utilizzata dalla classe `mytalk.client.presenter.user.communication.MediaStream`.

4.5 Package mytalk.server.presenter

Di seguito è riportata la descrizione delle classi per quanto concerne la parte del Presenter che risiede sul server$_{|g|}$.

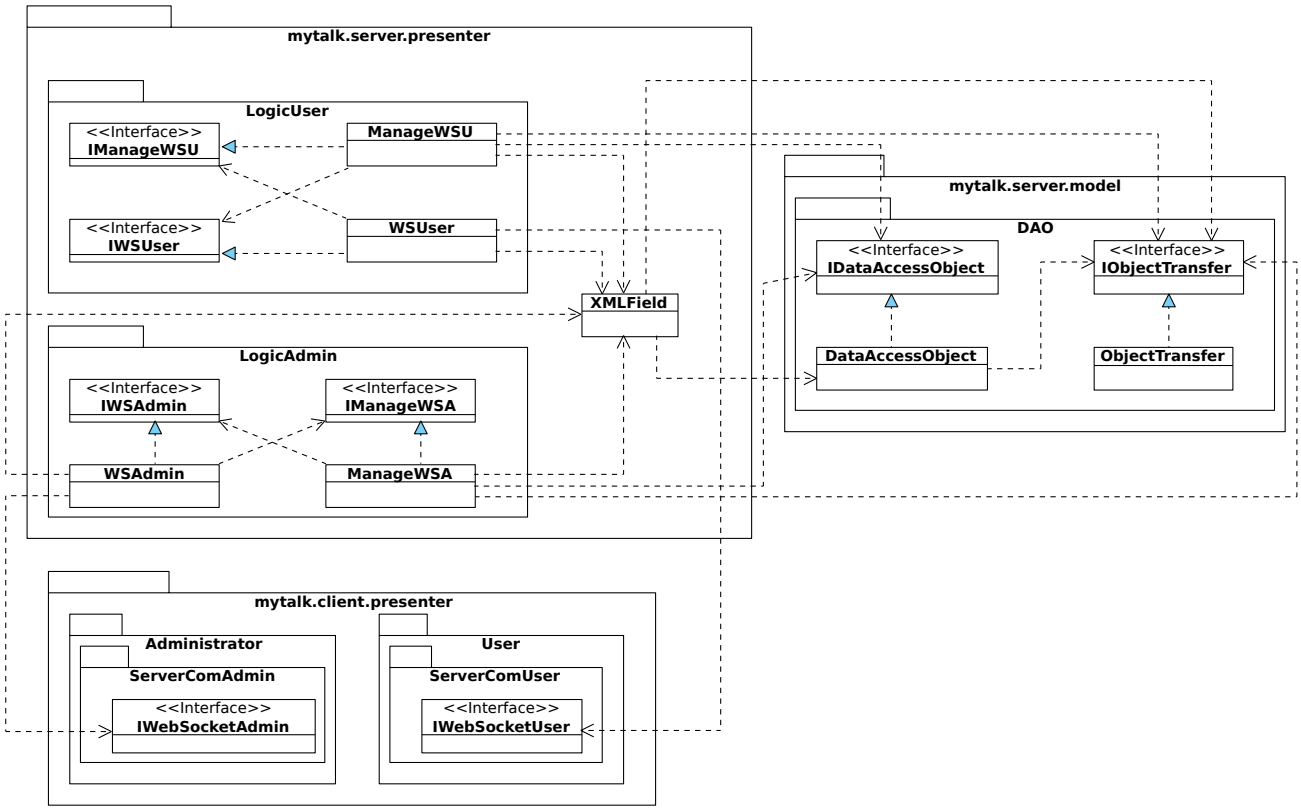

Figura 10: Diagramma delle classi che illustra il Presenter lato server$_{|g|}$.

4.5.1 Package mytalk.server.presenter

XMLField

Nome: `XMLField`

Tipo: class

Package: `mytalk.server.presenter`

Descrizione: la classe che raccoglie i metodi comuni dei package$_{|g|}$ `mytalk.server.presenter.user.logicUser` e `mytalk.server.presenter.administrator.logicAdmin` per gestione dei messaggi XML$_{|g|}$.

4.5.2 Package mytalk.server.presenter.user.logicUser

IManageWSU

Nome: `IManageWSU`

Tipo: interface

Package: `mytalk.server.presenter.user.logicUser`

Descrizione: interfaccia che rappresenta la classe contenitrice delle connessioni WebSocket$_{|g|}$ dal server$_{|g|}$ verso il client$_{|g|}$, per quanto riguarda gli utenti. Le classi che la implementano devono gestire l'inoltro delle richieste di comunicazione agli utenti destinatari ed eventualmente comunicare con il Model.

IWSUser

Nome: `IWSUser`

Tipo: interface

Package: `mytalk.server.presenter.user.logicUser`

Descrizione: interfaccia pubblica della classe che gestisce il collegamento, tramite WebSocket$_{|g|}$, dal server$_{|g|}$al client$_{|g|}$per l'utente.

ManageWSU

Nome: `ManageWSU`

Tipo: class

Package: `mytalk.server.presenter.user.logicUser`

Descrizione: la classe ha il compito di gestire le richieste di comunicazione tra il server$_{|g|}$e il client$_{|g|}$degli utenti.
La comunicazione tra questi due livelli deve avvenire tramite WebSocket$_{|g|}$. La classe dopo aver ricevuto un messaggio dalla classe che contiene il WebSocket$_{|g|}$, dovrà soddisfare la richiesta che è contenuta nel messaggio; eventualmente interrogando il Model o girando il messaggio verso un altro utente.

Relazioni con altre componenti: la classe implementa l'interfaccia `mytalk.server.presenter.user.logicUser.IManageWSU` e richiama metodi delle classi
`mytalk.server.model.dao.DataAccessObject` (tramite l'interfaccia `mytalk.server.model.dao.IDataAccessObject`),
`mytalk.server.model.dao.ObjectTranfer` (tramite l'interfaccia `mytalk.server.model.dao.IObjectTranfer`) e
`mytalk.server.presenter.XMLField` per interagire con la base di dati, dalla quale riceve, nel caso di restituzione di un risultato, oggetti di tipo `mytalk.server.model.dao.ObjectTransfer`.

WSUser

Nome: `WSUser`

Tipo: class

Package: `mytalk.server.presenter.user.logicUser`

Descrizione: la classe che gestisce il collegamento tramite WebSocket$_{|g|}$ all'utente. Tale classe, oltre a ricevere la richiesta dal livello Presenter del client$_{|g|}$ gira tale richiesta dopo averla tradotta in una opportuna chiamata di funzione alla sua classe contenitrice che è `mytalk.server.presenter.user.logicUser`.

Relazioni con altre componenti: la classe implementa l'interfaccia `mytalk.server.presenter.user.logicUser.IWSUser` e richiama metodi della classe `mytalk.server.presenter.user.logicUser.ManageWSU`.

4.5.3 Package mytalk.server.presenter.administrator.logicAdmin

IManageWSA

Nome: `IManageWSA`

Tipo: interface

Package: `mytalk.server.presenter.administrator.logicAdmin`

Descrizione: interfaccia che rappresenta la classe contenitrice delle connessioni WebSocket$_{|g|}$ dal server$_{|g|}$ verso il client$_{|g|}$, per quanto riguarda gli amministratori. Le classi che la implementano devono gestire l'inoltro delle richieste di interrogazione della base di dati da parte degli amministratori e le richieste di autenticazione.

IWSAdmin

Nome: `IWSAdmin`

Tipo: interface

Package: `mytalk.server.presenter.administrator.logicAdmin`

Descrizione: interfaccia pubblica della classe che gestisce il collegamento dal server$_{|g|}$verso il client$_{|g|}$, tramite WebSocket$_{|g|}$.

ManageWSA

Nome: `ManageWSA`

Tipo: class

Package: `mytalk.server.presenter.administrator.logicAdmin`

Descrizione: la classe ha il compito di ricevere, gestire e soddisfare le richieste che arrivano dal livello Presenter del client$_{|g|}$ della componente amministratore. Tale classe, per soddisfare le richieste, effettua delle interazioni con la classe che interfaccia il database$_{|g|}$.

Relazioni con altre componenti: la classe implementa l'interfaccia `mytalk.server.presenter.logicAdmin.IManageWSA` e richiama metodi della classe `mytalk.server.model.dao.DataAccessObject` (tramite l'interfaccia `mytalk.server.model.dao.IDataAccessObject`), `mytalk.server.model.dao.ObjectTranfer` (tramite l'interfaccia `mytalk.server.model.dao.IObjectTranfer`) e `mytalk.server.presenter.XMLField` per interagire con la base di dati, dalla quale riceve, nel caso di restituzione di un risultato, oggetti di tipo `mytalk.server.model.dao.ObjectTransfer`.

WSAdmin

Nome: `WSAdmin`

Tipo: class

Package: `mytalk.server.presenter.logicAdmin`

Descrizione: la classe che gestisce il collegamento tramite WebSocket$_{|g|}$ all'utente amministratore. Tale classe, oltre a ricevere, soddisfa la richiesta del livello Presenter che può riguardare operazioni di autenticazione o visualizzazione delle statistiche.

Relazioni con altre componenti: la classe implementa l'interfaccia `mytalk.server.presenter.administrator.logicAdmin.IWSAdmin` ed utilizza i metodi definiti nella classe `mytalk.server.presenter.administrator.logicAdmin.ManageWSA` tramite la sua interfaccia `mytalk.server.presenter.administrator.logicAdmin.IManageWSA`. I metodi della classe vengono richiamati da tutti gli oggetti del Presenter.

4.6 Package mytalk.client.model

Diseguito è riportata la descrizione delle componenti del Model che risiedono sul client$_{|g|}$.

4.6.1 Package mytalk.client.model.localDataAdmin

ManageCookies

Nome: `ManageCookies`

Tipo: class

Package: `mytalk.client.model.localDataAdmin`

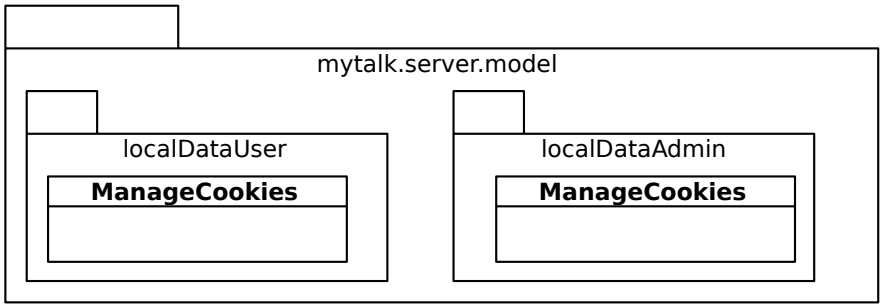

Figura 11: Diagramma delle classi che illustra il Model lato Client.

Descrizione: classe che rappresenta le informazioni di sessione salvate sul client$_{|g|}$, tali informazioni riguardano l'amministratore attualmente autenticato, se questo esiste.

4.6.2 Package mytalk.client.model.localDataUser

ManageCookies

Nome: `ManageCookies`

Tipo: class

Package: `mytalk.client.model.localDataUser`

Descrizione: classe che rappresenta le informazioni di sessione salvate sul client$_{|g|}$, tali informazioni riguardano l'utente attualmente autenticato, se questo esiste.

4.7 Package mytalk.server.model.dao

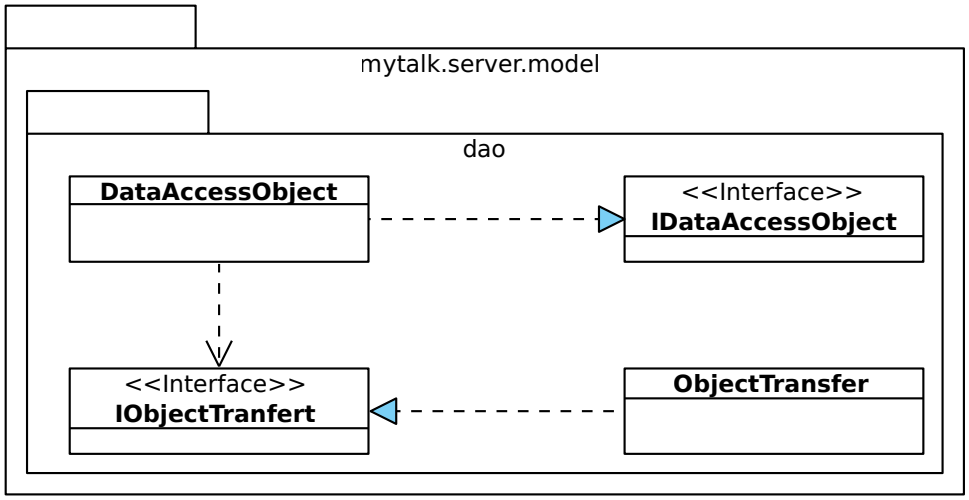

Figura 12: Diagramma delle classi che illustra il Model lato Client.

IDataAccessObject

Nome: `IDataAccessObject`

Tipo: interface

Package: `mytalk.server.model.dao`

Descrizione: interfaccia che rappresenta la connessione verso il database$_{|g|}$ MySQL per il recupero e l'inserimento delle informazioni degli utenti.

IObjectTransfer

Nome: `IObjectTransfer`

Tipo: interface

Package: `mytalk.server.model.dao`

Descrizione: interfaccia che fornisce le operazioni per la gestione delle informazioni reperite o inserite nel DBMS$_{|g|}$.

DataAccessObject

Nome: `DataAccessObject`

Tipo: class

Package: `mytalk.server.model.dao`

Descrizione: la classe ha il compito di interfacciare il server$_{|g|}$ con il database$_{|g|}$ MySQL, nel quale risiedono i dati relativi agli utenti, agli amministratori e alle statistiche. La classe fa parte del design pattern DAO (Data Access Object) e qualora debba restituire un risultato lo farà restituendo un oggetto di tipo `mytalk.server.model.dao.ObjectTransfer`, disaccoppiando in questo modo il resto dell'applicazione dalla base di dati sottostante.

Relazioni con altre componenti: la classe implementa l'interfaccia `mytalk.server.model.IDataAccessObject` ed utilizza i metodi definiti nella classe `mytalk.server.model.dao.ObjectTransfer` tramite la sua interfaccia `mytalk.server.model.dao.IObjectTransfer`.

ObjectTransfer

Nome: `ObjectTransfer`

Tipo: class

Package: `mytalk.server.model.dao`

Descrizione: questo oggetto rappresenta il tipo generico dell'oggetto che incapsulerà le informazioni restituite da una interrogazione al database.

Relazioni con altre componenti: la classe implementa l'interfaccia `mytalk.server.model.IObjectTransfer`.

5 Design pattern

Nell'architettura del sistema **MyTalk** sono stati impiegati dei design pattern che forniscono delle soluzioni note, corrette e ripetute di alcuni problemi affrontati.

5.1 Adapter

Descrizione: il Design Pattern Adapter permette a classi diverse di operare anche quando utilizzano interfacce incompatibili. Il suo fine è di evitare la scrittura di parti del sistema già implementate in librerie esterne permettendone l'interoperabilità.

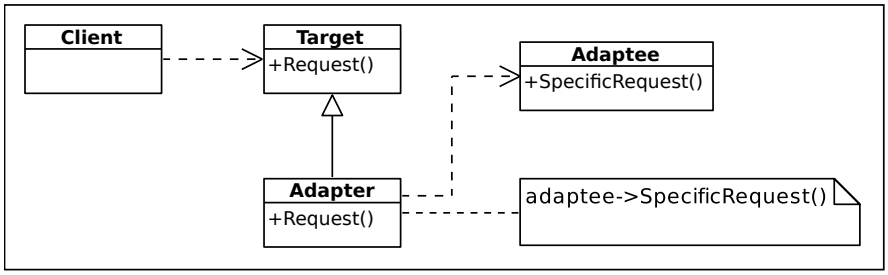

Figura 13: Diagramma delle classi del design pattern Adapter in formato $\text{UML}_{|g|}$.

Giustificazione: la scelta di questo Design Pattern dipende dal fatto che le librerie $\text{WebRTC}_{|g|}$ sono mutabili nel tempo in quanto non ancora standardizzate. L'adapter permetterà di modificare solo i metodi della classe senza invalidare l'interfaccia pubblica della classe adaptee.

Applicazione: il design pattern è utilizzato all'interno del `packagemytalk.client.presenter`.

5.2 Data Access Object

Descrizione: il DAO (Data Access Object) è un design pattern che riduce l'accoppiamento tra la business logic e la persistenza dei dati: la business logic necessita spesso di dati che sono memorizzati in un database. Il pattern permette all'applicazione di essere indipendente da eventuali cambiamenti nel database utilizzato.

Ogni classe della business logic offre i metodi set e get per operare sui campi dati

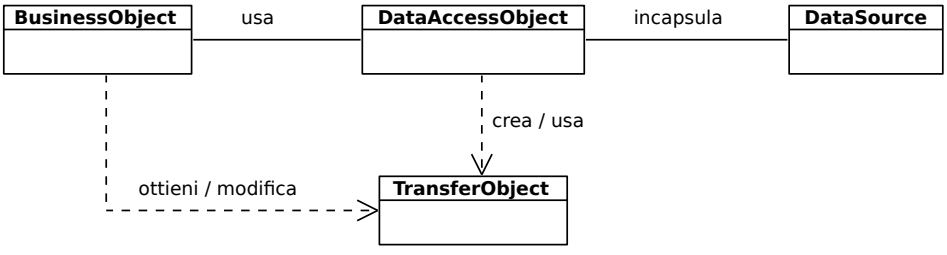

Figura 14: Diagramma delle classi del design pattern DAO in formato $\text{UML}_{|g|}$.

ai quali non è possibile avere l'accesso dall'esterno della classe; offre inoltre almeno i metodi:

- Leggi dati da database;
- Inserisci dati da database;
- Aggiorna dati da database;
- Cancella dati da database;

eventualmente ridefiniti per consentire la realizzazione di diverse politiche di accesso al database.
La figura 15 illustra il diagramma di sequenza che mostra l'interazione tra i partecipanti di questo pattern.

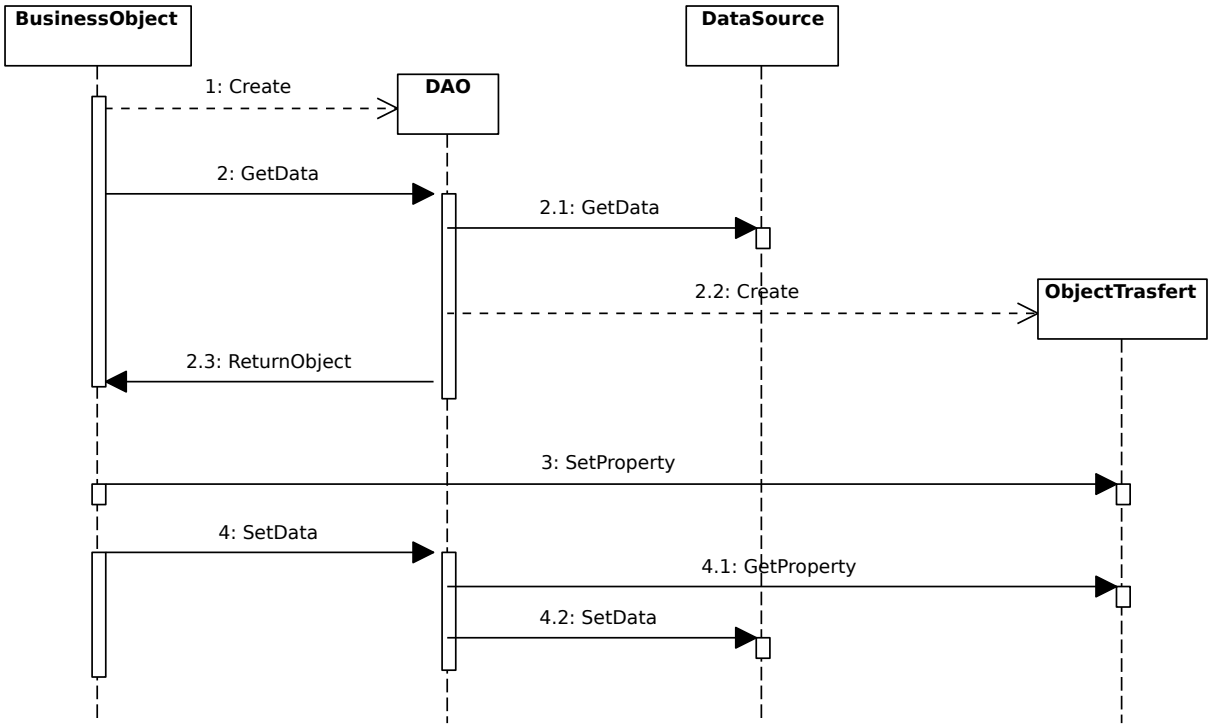

Figura 15: Diagramma di sequenza del design pattern DAO in formato $\text{UML}_{|g|}$.

- Business Object: rappresenta la classe che richiede l'accesso ai dati del database;
- Data Access Object: fornisce un'astrazione per l'implementazione dell'accesso ai dati del database sottostante per fornire al business object un accesso trasparente;
- Data Source: rappresenta l'implementazione della sorgente dei dati che può essere un database, un file system, una repository ecc;
- Transfer Object: rappresenta l'oggetto utilizzato per il trasporto dei dati che può essere utilizzato sia per la scrittura sul database che per il recupero dei dati.

Giustificazione:

Vantaggi:

- Permette la trasparenza: il business object può accedere al database senza conoscerne nello specifico l'implementazione;
- Semplifica la migrazione verso un'implementazione diversa del database, che coinvolge solo lo strato del DAO;
- Riduce la complessità del codice del business object aumentandone quindi la comprensibilità e la manutenibilità;
- Centralizza tutto l'accesso ai dati in un livello separato.

Svantaggi:

- Aggiunge un ulteriore livello: il DAO crea un livello dizionale tra il client e il database che deve essere progettato ed implementato. I benefici giustificano però ampiamente questo costo.

Applicazione: il design pattern è utilizzato completamente all'interno del package `mytalk.server.model.dao`.

5.3 MVP

Descrizione: il Model View Presenter (MVP) prevede di strutturare un'applicazione in tre componenti:

- Model: individua la rappresentazione dei dati dell'applicazione e delle politiche di accesso e modifica;
- View: si occupa di rappresentare l'interfaccia grafica con la quale l'utente interagirà con il Presenter;
- Presenter: identifica la parte logica del prodotto facendo da tramite tra le richieste inoltrate dalla View al Model ed aggiornando la View.

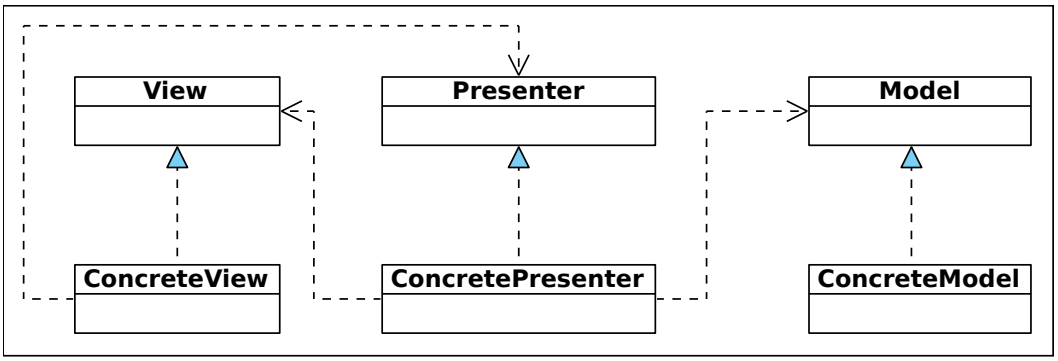

Figura 16: Diagramma del design pattern MVP in formato $\text{UML}_{|g|}$.

Il prodotto MyTalk è sviluppato secondo la variante del pattern denominata Passive View. Essa prevede che sia il Presenter ad aggiornare la view, comunicando tramite la

sua interfaccia, per riflettere i cambiamenti nel model; l'interazione con quest'ultimo è quindi gestita esclusivamente dal Presenter: la view non è consapevole dei cambiamenti del model.
La figura 17 illustra il diagramma di sequenza per il design pattern MVP.

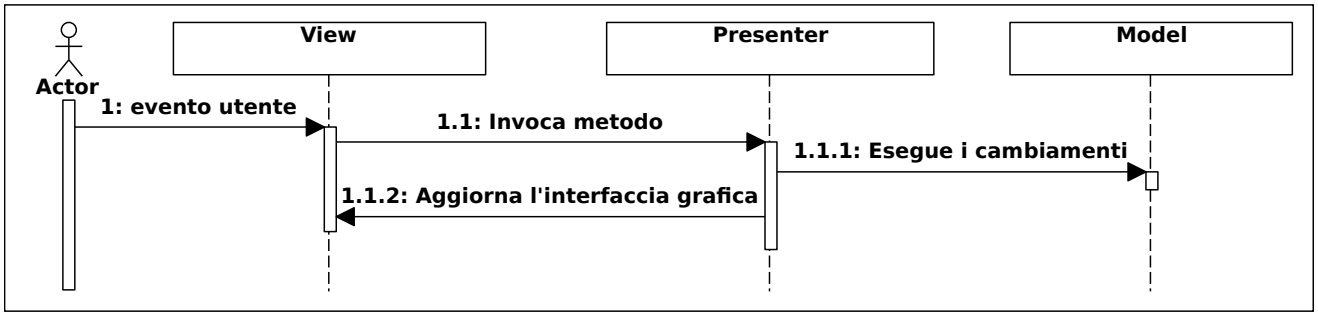

Figura 17: Diagramma di sequenza del design pattern MVP in formato UML$_{|g|}$.

Giustificazione: durante lo sviluppo di un'architettura complessa che richiede l'implementazione di un'interfaccia utente elaborata, si corre il rischio che nella GUI venga inserita della logica che dovrebbe risiedere in altri strati dell'applicazione. Ciò comporta i seguenti svantaggi:

- Il codice nello strato dell'interfaccia grafica è molto difficile da testare senza eseguire manualmente il programma o mantenere degli script che automatizzino l'interazione con la GUI$_{|g|}$;
- Duplicazione del codice che gestisce la logica di interfacce utente simili;
- Riduce la manutenibilità e la comprensibilità del codice.

Per evitare questi problemi viene adottato il design pattern MVP che permette di separare l'interfaccia grafica (View), l'application logic (Presenter) e la business logic (Model).
Questa scelta implementativa comporta inoltre una più agevole progettazione in quanto le comunicazioni tra le parti sono incentrate nella componente Presenter. Questo approccio porta ai seguenti vantaggi:

- Indipendenza tra le componenti Model e View;
- Separazione dei ruoli e delle relative interfacce;
- Semplice supporto per nuove funzionalità alla tipologia di utente;
- Ruolo cardine della componente Presenter che semplifica la progettazione e la manutenzione.

É stata scelta la variante Passive View in quanto migliora la testabilità, poiché la logica della GUI$_{|g|}$ può essere testata direttamente dal Presenter.

Applicazione: rappresenta la struttura portante dell'intera applicazione nonché la sua architettura generale come si può determinare alla sezione 2.2.

6 Diagrammi di attività

6.1 Diagrammi di attività utente

6.1.1 Login

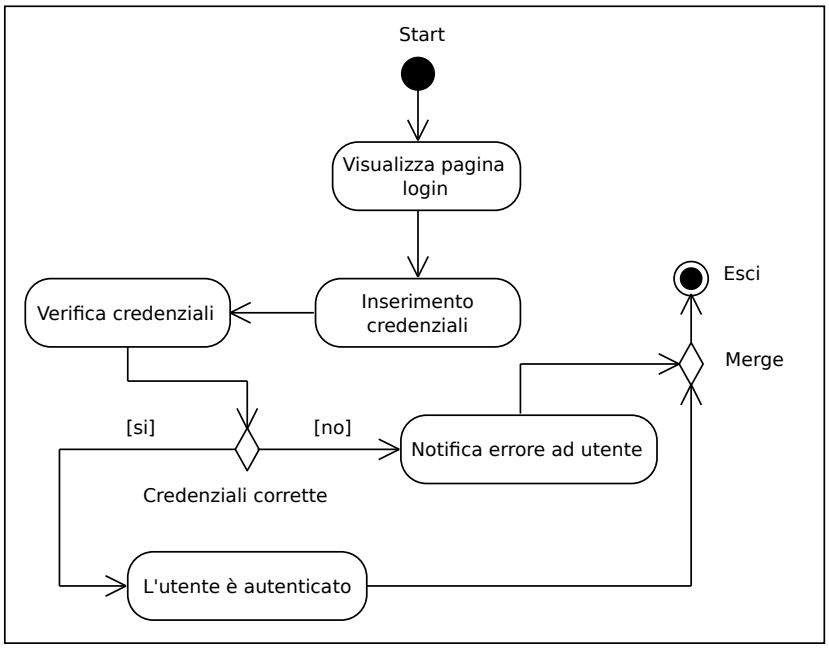

Figura 18: DA 1 - La procedura di Login.

Descrizione: l'utente visualizza la pagina di login ed inserisce le proprie credenziali. Le credenziali inserite vengono verificate. Se queste sono corrette l'utente diviene autenticato e la sequenza di attività termina, altrimenti gli viene notificato l'errore di autenticazione e la sequenza di attività termina.

6.1.2 Registrazione

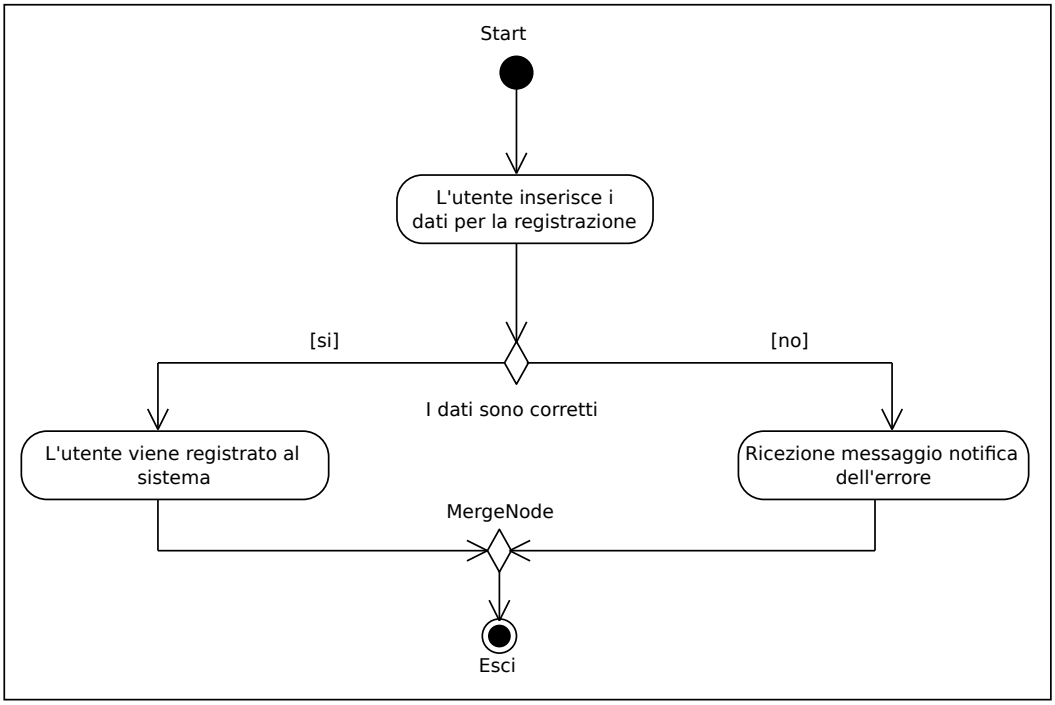

Figura 19: DA 2 - La procedura di Registrazione.

Descrizione: l'utente si vuole registrare al servizio, viene visualizzato il form per la registrazione e l'utente inserisce i propri dati. Se i dati sono corretti (soddisfano quindi le condizioni poste in fase di analisi dei requisiti), il sistema procede con la registrazione del nuovo utente e la sequenza di attività termina, altrimenti l'utente riceve un opportuno messaggio che descrive la natura dell'errore e la sequenza di attività termina.

6.1.3 Modifica dati

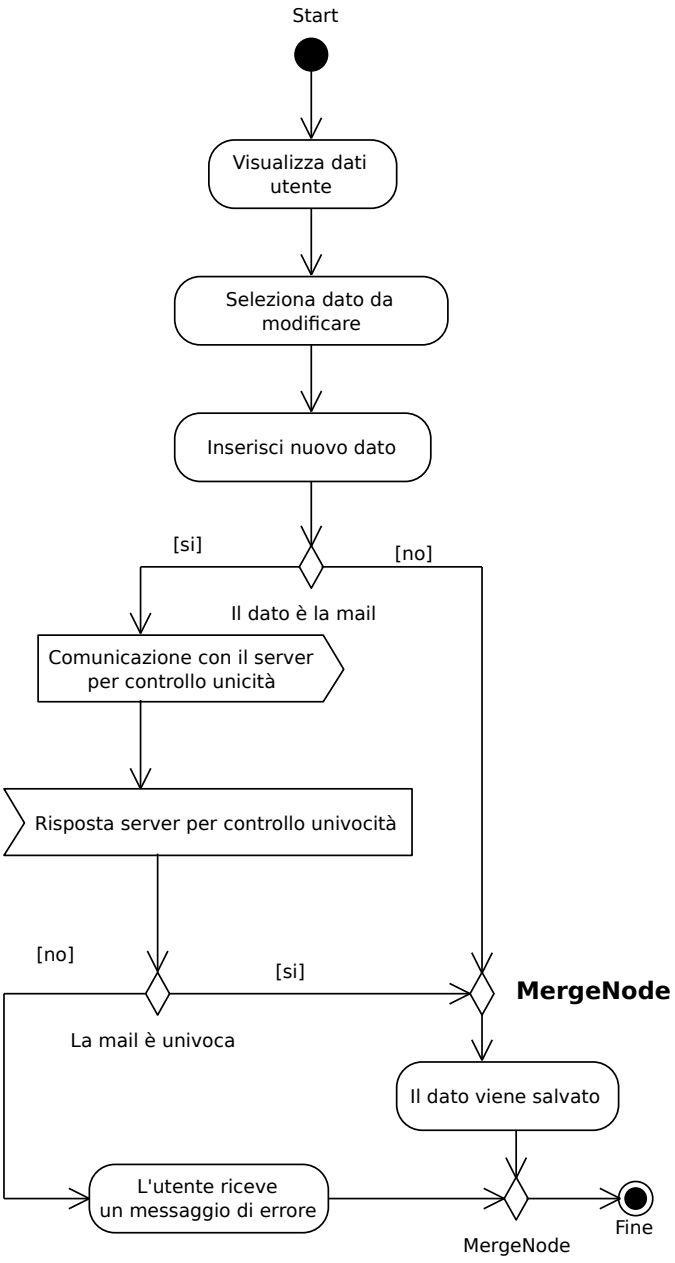

Figura 20: DA 3 - La procedura di modifica dei dati dell'utente.

Descrizione: l'utente visualizza i propri dati, seleziona il dato che vuole modificare e inserisce il nuovo dato.
Se il dato modificato non è la mail il sistema procede con la modifica e la sequenza di attività termina.
Se il dato modificato è la mail il sistema comunica con il server$_{|g|}$ per controllare la sua correttezza secondo le norme stabilite in analisi dei requisiti. Il sistema attende la risposta del server$_{|g|}$ in merito al controllo di unicità. Se la mail è univoca allora il sistema procede con la modifica e la sequenza di attività termina, altrimenti la

richiesta di modifica non viene eseguita e il sistema riporta un opportuno messaggio di errore all'utente e la sequenza di attività termina.

6.1.4 Ricevimento chiamata

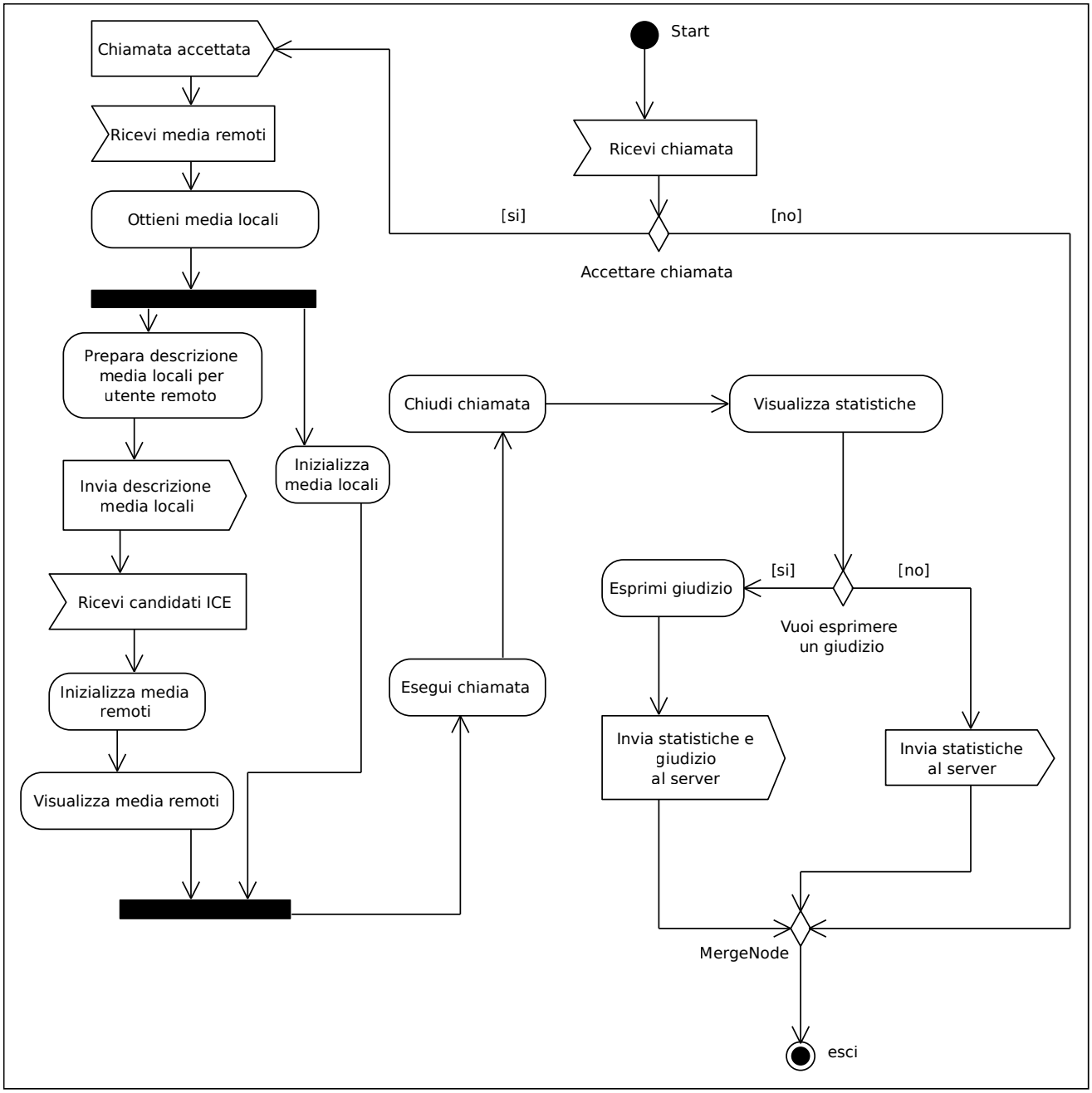

Figura 21: DA 4 - La procedura di ricezione di una chiamata.

Descrizione: l'utente riceve una notifica dal sistema di una chiamata in arrivo che può essere accettata o rifiutata. Nel caso la chiamata venga rifiutata, l'utente esce dalla fase di gestione della chiamata in arrivo e la sequenza di attività termina. Se, invece, l'utente accetta la chiamata avvengono le seguenti operazioni:

1. l'utente ha accettato la chiamata ed invia il segnale di accettazione al mittente;
2. l'utente riceve dal mittente la descrizione della sessione remota (media remoti);

3. il sistema richiede all'utente l'accesso ai media locali (gestito dal browser$_{|g|}$);
4. si eseguono in parallelo due sequenze di attività. La prima:
   (a) si prepara la descrizione della sessione locale da inviare all'utente remoto che poi provvederà a gestirla opportunamente;
   (b) si invia tale descrizione attraverso il server$_{|g|}$ all'utente remoto, senza attendere alcuna risposta;
   (c) si ricevono i candidati ICE;
   (d) si inizializzano e si visualizzano i media remoti all'interno della pagina web$_{|g|}$.

   La seconda:
   (a) si inizializzano e si visualizzano i media locali all'interno della pagina web$_{|g|}$.

Quando tutte queste attività sono terminate la chiamata è iniziata ed è possibile per gli utenti comunicare in tempo reale. Viene allora eseguita la chiamata, che continua fino a quando uno dei due utenti non decide di terminare la chiamata. Vengono quindi visualizzate dall'utente le statistiche relative alla chiamata appena conclusa. All'utente viene richiesto di esprimere un giudizio in merito alla qualità della chiamata. Se l'utente accetta, esprime il giudizio e vengono inviate al server$_{|g|}$ le statistiche e il giudizio e, senza attendere alcuna risposta, la sequenza di attività termina. Se invece rifiuta vengono inviate al server$_{|g|}$ le statistiche e, senza attendere alcuna risposta, la sequenza di attività termina.

6.1.5 Chiama inserendo dato

Descrizione: l'utente sceglie di chiamare un altro utente tramite lo username o l'indirizzo $IP_{|g|}$ e visualizza la pagina per l'inserimento di tale dato. L'utente inserisce lo username o l'indirizzo $IP_{|g|}$ dell'utente da chiamare. Il sistema invia il dato inserito al $server_{|g|}$ per avere la conferma dell'esistenza dell'utente ed attende la risposta. Se l'utente non esiste il sistema notifica l'esito negativo e la sequenza delle attività termina. Se l'utente esiste, gli viene inviata una richiesta di comunicazione e il sistema attende una risposta. Se la richiesta viene rifiutata l'utente che ha inviato la richiesta riceve debita comunicazione e la sequenza di attività termina. Se la richiesta viene accettata avviene la seguente sequenza di operazioni:

1. viene chiesto all'utente di accedere ai media locali (*es.* webcam e microfono);
2. la chiamata viene inizializzata, eseguita e successivamente terminata;
3. vengono mostrate all'utente le statistiche sulla chiamata appena terminata;
4. viene richiesto all'utente di esprimere un giudizio in merito alla qualità della chiamata:
   - se l'utente accetta, esprime il giudizio e vengono inviate al $server_{|g|}$ le statistiche e il giudizio e, senza attendere alcuna risposta, la sequenza di attività termina;
   - se invece rifiuta vengono inviate al $server_{|g|}$ le statistiche e, senza attendere alcuna risposta, la sequenza di attività termina.

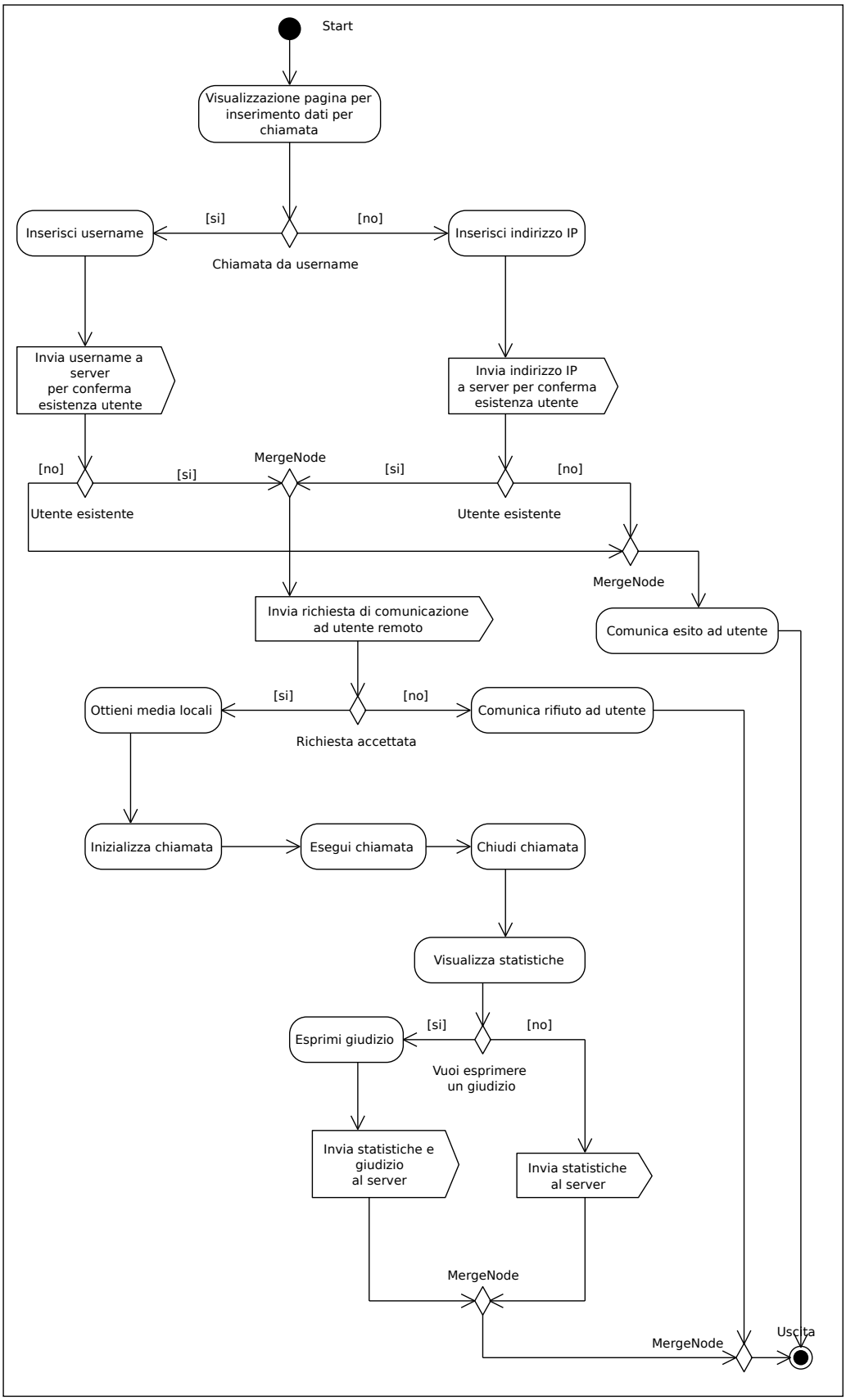

Figura 22: DA 5 - La procedura di chiamata inserendo un dato conosciuto (username o indirizzo IP).

6.1.6 Chiama selezionando da lista

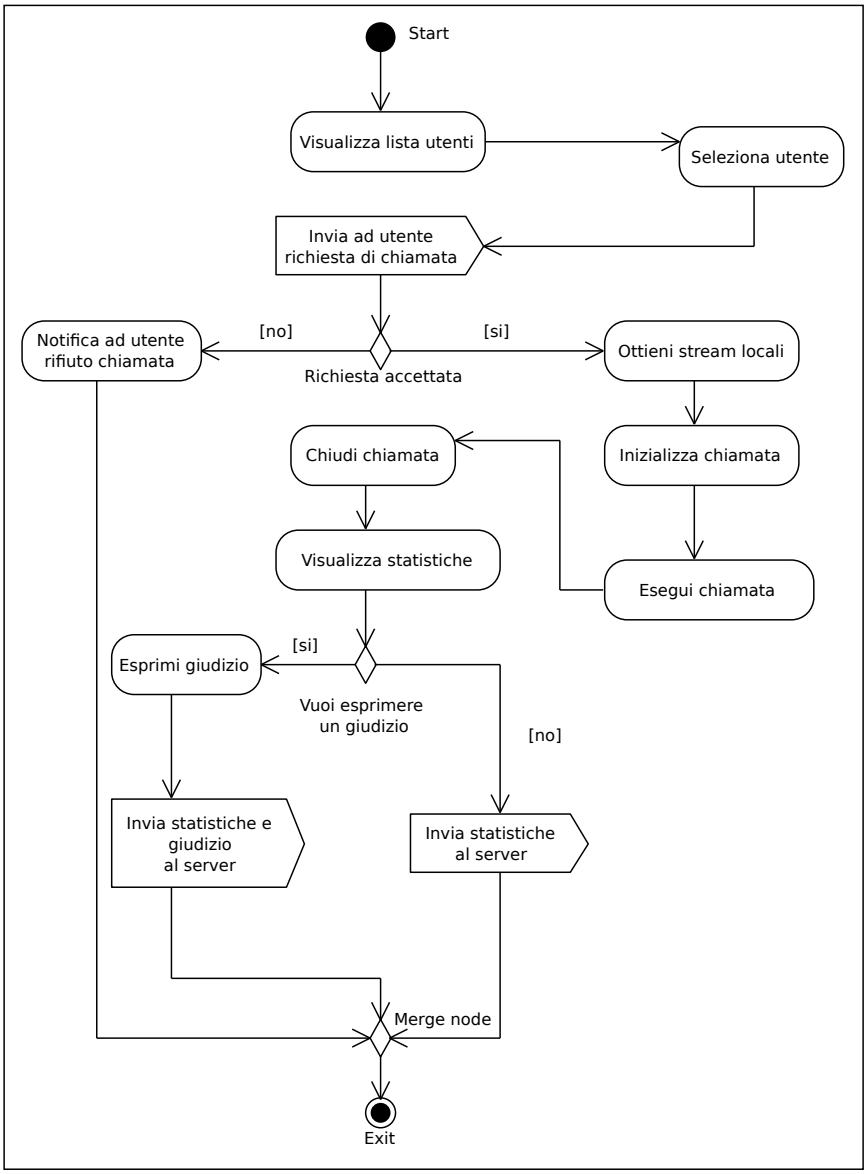

Figura 23: DA 6 - La procedura di effettuazione di una chiamata dalla lista.

Descrizione: l'utente visualizza la lista degli utenti registrati al server$_{|g|}$ e seleziona l'utente che desidera chiamare. Il sistema invia all'utente selezionato una richiesta di chiamata e attende la risposta. Se la chiamata viene rifiutata, viene notificato all'utente il rifiuto da parte del destinatario e la sequenza delle attività termina. Se la chiamata viene accettata avviene la seguente sequenza di operazioni:

1. si ottiene l'accesso ai media locali;
2. la chiamata viene inizializzata, eseguita ed infine chiusa;
3. vengono mostrate all'utente le statistiche sulla chiamata appena terminata;

4. viene richiesto all'utente di esprimere un giudizio in merito alla qualità della chiamata:
   - se l'utente accetta, esprime il giudizio e vengono inviate al server$_{|g|}$ le statistiche e il giudizio e, senza attendere alcuna risposta, la sequenza di attività termina;
   - se invece rifiuta vengono inviate al server$_{|g|}$ le statistiche e, senza attendere alcuna risposta, la sequenza di attività termina.

6.1.7 Inizializza chiamata

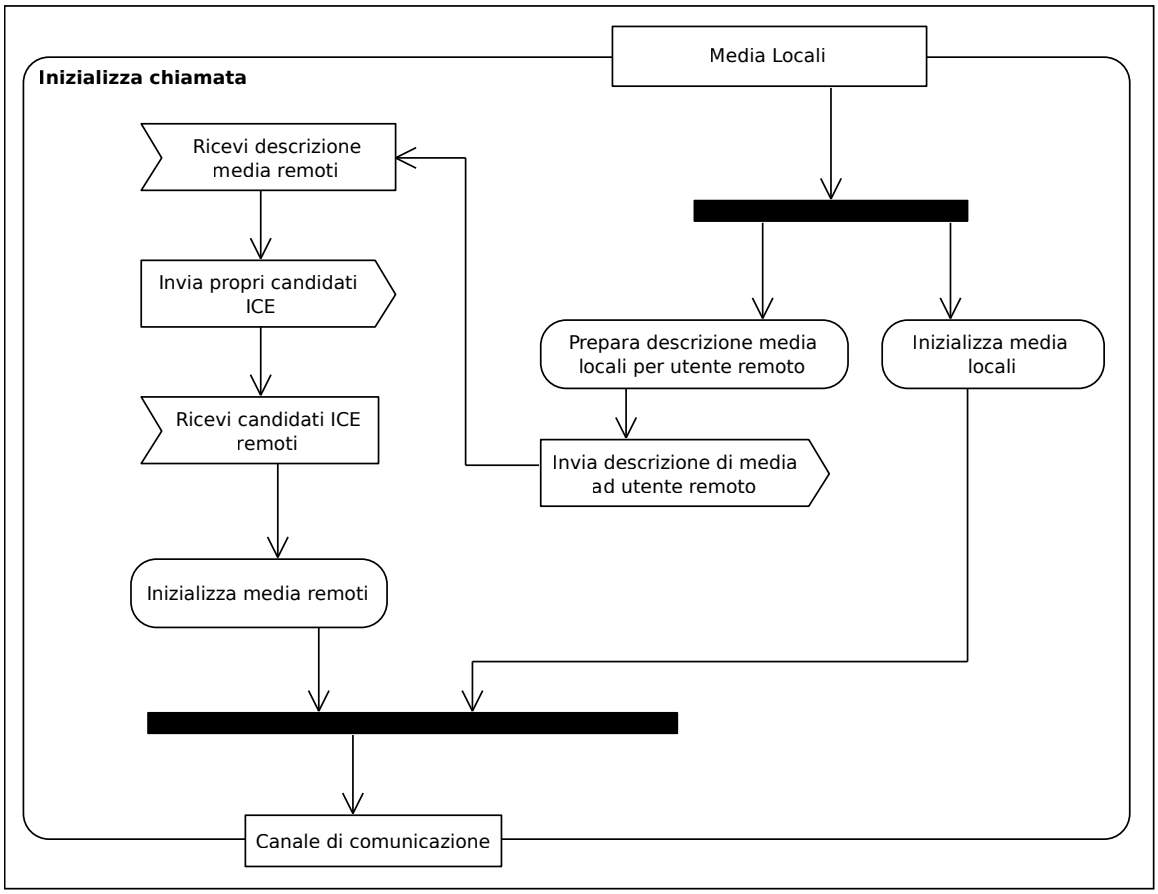

Figura 24: DA 6.1 - La procedura di inizializzazione di una chiamata.

Descrizione: una volta ottenuti i media locali si svolgono due sequenze di attività in parallelo. La prima:

1. si prepara la descrizione di media locali per l'utente remoto;
2. la descrizione preparata al punto 1 viene inviata all'utente remoto;
3. si attende come risposta dall'utente remoto la descrizione dei suoi media locali;
4. si inviano i propri candidati ICE;

5. si attende la ricezione dei candidati ICE remoti;
6. al ricevimento della risposta del punto 5, l'utente locale inizializza i media remoti nella propria pagina.

La seconda:

1. si inizializzano i media locali.

Una volta che le due sequenze sono state completate, si ottiene un canale di comunicazione tra i due utenti che vogliono comunicare.

6.2 Diagrammi di attività amministratore

6.2.1 Login

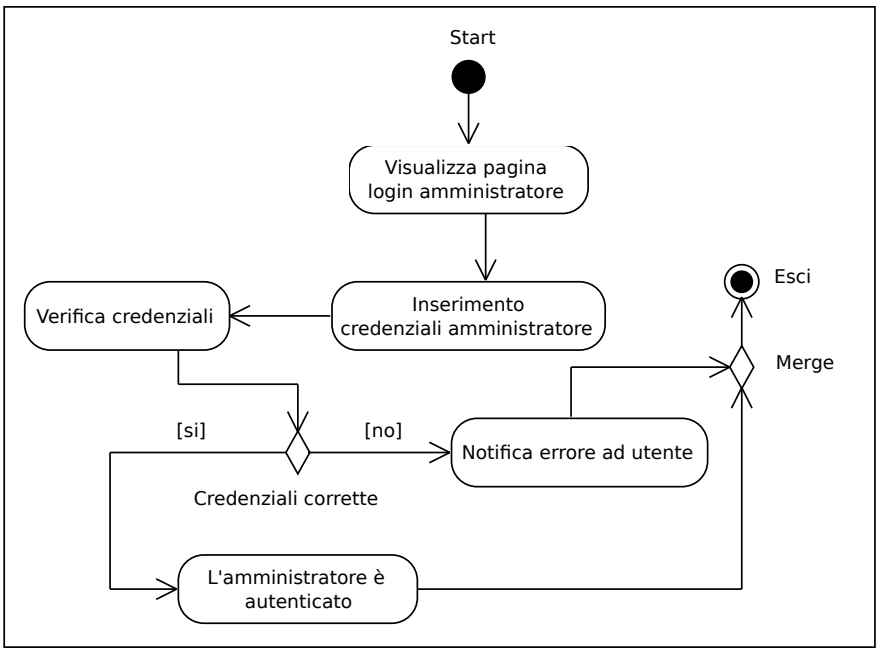

Figura 25: DA 7 - La procedura di inizializzazione di una chiamata.

Descrizione: l'utente amministratore visualizza la pagina di login ed inserisce le proprie credenziali. Le credenziali inserite vengono verificate. Se queste sono corrette l'utente amministratore diviene autenticato e la sequenza di attività termina, altrimenti gli viene notificato l'errore di autenticazione e la sequenza di attività termina.

6.2.2 Visualizzazione statistiche amministratore

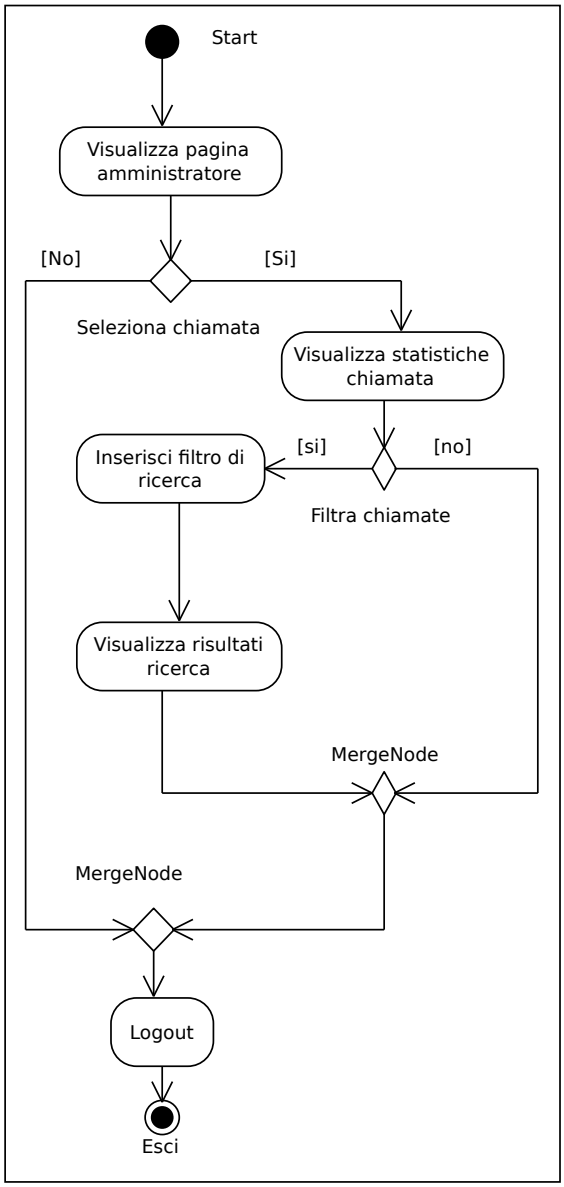

Figura 26: DA 8 - La procedura di visualizzazione delle statistiche.

Descrizione: l'utente amministratore visualizza la sua pagina principale nella quale visualizza le statistiche relative alle comunicazioni effettuate, tra gli utenti registrati, nell'ultima settimana. L'amministratore ha la possibilità di filtrare queste chiamate. Se decide di non filtrarle, effettua il logout e la sequenza di attività termina. Se decide di filtrarle può farlo in base alla data di effettuazione, al giudizio ottenuto o all'utente che vi ha partecipato. Dopo aver applicato il filtro il sistema permette all'amministratore di visualizzare i risultati della ricerca. Infine l'amministratore effettua il logout e la sequenza di attività termina.

7 Diagrammi di sequenza

Di seguito sono riportati i diagrammi di sequenza per quanto riguarda le operazioni principali del sistema.
Altri programmi di sequenza sono disponibili nel documento Definizione di Prodotto, sezione 7 (*DefinizioneDiProdotto_v1.0.pdf*).

7.1 Verifica dell'esistenza dello username

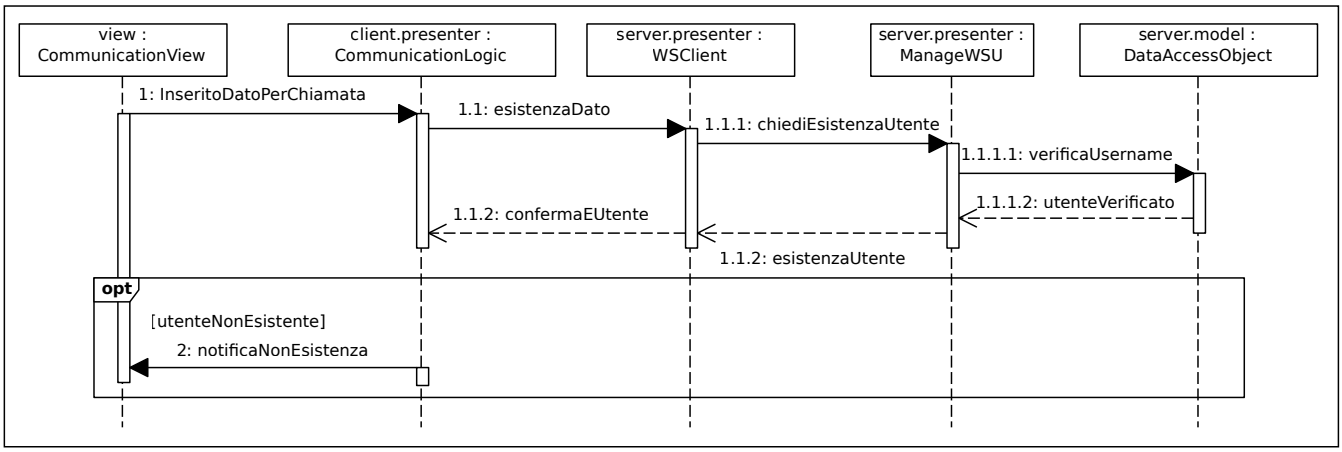

Figura 27: DS 1 - La procedura di verifica dell'esistenza di un utente di cui si conosce un dato.

Precondizione: l'utente vuole chiamare un altro utente inserendo un dato, username o indirizzo IP$_{|g|}$, che identifica univocamente l'utente da chiamare.
Descrizione: il diagramma mostra la procedura con la quale il sistema verifica se il dato inserito dall'utente è associabile univocamente ad un utente presente nel database$_{|g|}$. La classe `CommunicationView` genera l'evento associato all'inserimento di un dato che viene ricevuto dalla classe `CommunicationLogic`, la quale:

1. estrapola il dato inserito dall'utente sotto forma di testo;
2. invia il dato, tramite WebSocket$_{|g|}$, alla classe `WSUser` che risiede all'interno del server$_{|g|}$;
3. la classe `WSUser` invia il dato alla classe `DataAccessObject`, che gestisce il DBMS$_{|g|}$, la quale tramite una ricerca all'interno della base di dati, ritorna un valore che indica se il dato passato è associabile ad un utente univoco oppure no;
4. il risultato, identificabile con un booleano$_{|g|}$, viene restituito alla classe `CommunicationLogic`, la quale in questo modo ha scoperto se esiste un utente al quale corrisponde il dato inserito dall'utente. Se la risposta è negativa, la classe `CommunicationLogic` notifica la non esistenza dell'utente alla classe `CommunicationView`.

Postcondizione: l'utente sa se l'utente che vuole chiamare esiste oppure no. Il sistema è pronto a creare il gestore della connessione.

8 Stima di fattibilità

Da un'analisi delle caratteristiche che il prodotto dovrà avere, appare evidente come la scelta delle tecnologie utilizzate per il suo sviluppo sia adeguata a portarne a termine il compimento:

- l'interfaccia verrà realizzata con i linguaggi HTML5$_{|g|}$ e CSS3$_{|g|}$: ciò garantisce compatibilità con molti browser$_{|g|}$, in particolare quello di riferimento (Google Chrome$_{|g|}$), e una buona manutenibilità;
- il linguaggio Java$_{|g|}$ è idoneo ad implementare il server$_{|g|}$; l'accesso ai dati persistenti sarà inoltre semplificato dall'utilizzo del connettore JDBC;
- i dati persistenti vengono memorizzati tramite un database$_{|g|}$ MySQL, che garantisce semplicità di gestione ed estendibilità;
- la comunicazione full-duplex$_{|g|}$ sarà fornita dall'utilizzo della tecnologia WebSocket$_{|g|}$, che può essere utilizzata lato browser$_{|g|}$, lato server$_{|g|}$ o per qualsiasi applicazione client-server$_{|g|}$.
- il requisito che l'applicazione risieda su una singola pagina web$_{|g|}$ può essere soddisfatto utilizzando AJAX$_{|g|}$, il quale permette di aggiornare singole componenti della pagina senza doverla ricaricare completamente.
- il framework GWT$_{|g|}$ consente di sviluppare applicazioni web dinamiche in linguaggio Javascript programmando in linguaggio Java$_{|g|}$.

Pertanto, visto quanto descritto sopra, appare possibile realizzare il prodotto rimanendo all'interno dei vincoli temporali ed economici stabiliti.

9 Tracciamento relazione componenti - requisiti

9.1 Tracciamento package - componenti

Package	Componente
mytalk.client.view.administrator	VA1 - PageAdminView VA2 - LogAdmin VA3 - StatisticsView
mytalk.client.view.user	VU1 - PageUserView VU2 - LogUser VU3 - Register VU4 - UserDataView VU5 - ShowDataUser VU6 - ModifyDataUser VU7 - CommunicationView VU8 - CallManage VU9 - MediaManage VU10 - MakeCall
mytalk.client.presenter.administrator	PA1 - LogAdminLogic PA2 - StatisticsLogic PA3 - UpdateViewLogic PA4 - WebSocketAdmin PA5 - CommonFunctions
mytalk.client.model.administrator	MCA1 - ManageCookies
mytalk.presenter.user	PU1 - LogUserLogic PU2 - RegisterLogic PU3 - DataUserLogic PU4 - CommunicationLogic PU5 - UpdateViewLogic PU6 - PeerConnection PU7 - MediaStream PU8 - WebSocketUser PU9 - CommonFunctions
mytalk.client.model.user	MCU1 - ManageCookies
mytalk.server.model	MS1 - ManageWSU MS2 - WSClient MS3 - ManageWSA MS4 - WSAdmin MS5 - ManageDBMS

Tabella 5: Tracciamento package - componenti

9.2 Tracciamento componenti - requisiti

Componenti	Requisiti
VA1	FAOB0, FAOB1, FAOB2, FAOB3
VA2	FAOB1, FAOB1.1, FAOB1.2, FAOB3
VA3	FAOB2, FAOB2.1, FAOB2.1.1, FAOB2.1.1.1, FAOB2.1.2, FAOB2.1.2.1, FAOB2.1.3, FAOB2.1.3.1, FAOB2.1.3.1.1, FAOB2.1.3.2, FAOB2.1.3.2.1, FAOB2.2, FAOB2.2.1, FAOB2.2.2, FAOB2.2.3, FAOB2.2.4, FAOB2.2.5, FAOB2.2.6
VU1	FOB0, FOB1, FOB2, FOB3, FOB4
VU2	FOB1, FOB1.1, FOB1.2, FOB6
VU3	FOB2, FOB2.1, FOB2.1.1, FOB2.2, FOB2.3, FOB2.4, FOB2.5, FOB2.6
VU4	FOB3, FOB3.1, FOB3.1.1, FOB3.1.2, FOB3.1.3, FOB3.1.4, FOB3.1.5, FOB3.2, FOB3.2.1, FOB3.2.2, FOB3.2.2.1, FOB3.2.2.2, FOB3.2.3, FOB3.2.4, FOB3.2.5, FOB3.2.6
VU5	FOB3.1, FOB3.1.1, FOB3.1.2, FOB3.1.3, FOB3.1.4, FOB3.1.5
VU6	FOB3.2, FOB3.2.1, FOB3.2.2, FOB3.2.2.1, FOB3.2.2.2, FOB3.2.3, FOB3.2.4, FOB3.2.5, FOB3.2.6
VU7	FOB4
VU8	FOB4.1, FOB4.1.1, FOB4.1.1.1, FOB4.1.1.1.2, FOB4.1.1.1.3, FOB4.1.1.1.4, FOB4.1.1.1.4.1, FOB4.1.1.1.4.2, FOB4.1.1.1.4.3, FOB4.1.1.1.4.4, FOB4.1.2
VU9	FOB4.1.1.1.1, FOB4.1.1.1.1.1, FOB4.1.1.1.1.2
VU10	FOB4.2, FOB4.2.1, FOB4.2.1.1, FOB4.2.1.1.1, FOB4.2.1.2, FOB4.2.1.2.1, FOB4.2.1.3, FOB4.2.1.3.1,FOB4.2.1.3.2
PA1	FAOB0, FAOB1, FAOB1.1, FAOB1.2, FAOB3
PA2	FAOB0, FAOB2, FAOB2.1, FAOB2.1.1, FAOB2.1.1.1, FAOB2.1.2, FAOB2.1.2.1, FAOB2.1.3, FAOB2.1.3.1, FAOB2.1.3.1.1, FAOB2.1.3.2, FAOB2.1.3.2.1, FAOB2.2, FAOB2.2.1, FAOB2.2.2, FAOB2.2.3, FAOB2.2.4, FAOB2.2.5, FAOB2.2.6
PA3	FAOB0, FAOB2.2, FAOB2.2.1, FAOB2.2.2, FAOB2.2.3, FAOB2.2.4, FAOB2.2.5, FAOB2.2.6, FAOB2.2.7
PA4	FAOB1, FAOB2, FAOB2.1, FAOB2.1.1, FAOB2.1.1.1, FAOB2.1.2, FAOB2.1.2.1, FAOB2.1.3, FAOB2.1.3.1, FAOB2.1.3.1.1, FAOB2.2, FAOB2.2.1, FAOB2.2.2, FAOB2.2.3, FAOB2.2.4, FAOB2.2.5, FAOB2.2.6, FAOB3
PA5	Non tracciabile perché contiene metodi comuni a tutte le classi.
PU1	FOB1, FOB1.1, FOB1.2, FOB6

PU2	FOB2, FOB2.1, FOB2.1.1, FOB2.2, FOB2.2.1, FOB2.3, FOB2.4, FOB2.5, FOB2.6
PU3	FOB3, FOB3.1, FOB3.1.1, FOB3.1.2, FOB3.1.3, FOB3.1.4, FOB3.1.5, FOB3.2, FOB3.2.1, FOB3.2.2, FOB3.2.2.1, FOB3.2.2.2, FOB3.2.3, FOB3.2.4, FOB3.2.5, FOB3.2.6
PU4	FOB4, FOB4.1, FOB4.1.1, FOB4.1.1.1, FOB4.1.1.1.1, FOB4.1.1.1.1.1, FOB4.1.1.1.1.2, FOB4.1.1.1.2, FOB4.1.1.1.3, FOB4.1.1.1.4, FOB4.1.1.1.4.1, FOB4.1.1.1.4.2, FOB4.1.1.1.4.3, FOB4.1.1.1.4.4, FOB4.1.2, FOB4.2, FOB4.2.1, FOB4.2.1.1, FOB4.2.1.1.1, FOB4.2.1.2, FOB4.2.1.2.1, FOB4.2.1.3, FOB4.2.1.3.1, FOB4.2.1.3.2, FOB5
PU5	FOB3, FOB3.1, FOB3.1.1, FOB3.1.2, FOB3.1.3, FOB3.1.4, FOB3.1.5, FOB4, FOB4.1, FOB4.1.1.1.4, FOB4.1.1.1.4.1, FOB4.1.1.1.4.2, FOB4.1.1.1.4.2, FOB4.1.1.1.4.3, FOB4.1.1.1.4.4
PU6	FOB4, FOB4.1, FOB4.1.1, FOB4.1.1.1, FOB4.1.1.1.1, FOB4.1.1.1.1.1, FOB4.1.1.1.1.2, FOB4.1.1.1.4.1, FOB4.1.1.1.4.2, FOB4.1.1.1.4.3, FOB4.1.1.1.4.4, FOB4.1.2, FOB4.2, FOB4.2.1, FOB5
PU7	FOB4, FOB4.1, FOB4.1.1, FOB4.1.1.1, FOB4.1.1.1.1, FOB4.1.1.1.1.1, FOB4.1.1.1.1.2, FOB4.1.1.1.4.1, FOB4.1.1.1.4.2, FOB4.1.1.1.4.3, FOB4.1.1.1.4.4, FOB4.1.2, FOB4.2, FOB4.2.1, FOB5
PU8	FOB1, FOB2, FOB2.1, FOB3, FOB3.1, FOB3.1.1, FOB3.1.2, FOB3.1.3, FOB3.1.4, FOB3.1.5, FOB3.2, FOB3.2.1, FOB3.2.2, FOB3.2.2.1, FOB3.2.2.2, FOB3.2.3, FOB3.2.4, FOB3.2.5, FOB3.2.6, FOB4.2.1, FOB4.2.1.1, FOB4.2.1.1.1, FOB4.2.1.2, FOB4.2.1.2.1, FOB4.2.1.3, FOB4.2.1.3.1, FOB4.2.1.3.2, FOB5, FOB6
PU9	Non tracciabile perché contiene metodi comuni a tutte le classi.
MCU1	FOB1
MCA1	FAOB1
MS1	FOB4, FOB4.1, 4.1.1.1.2, FOB4.1.1.1.3, FOB4.2, FOB4.2.1, FOB5
MS2	FOB1, FOB2, FOB2.1, FOB3, FOB3.1, FOB3.1.1, FOB3.1.2, FOB3.1.3, FOB3.1.4, FOB3.1.5, FOB3.2, FOB3.2.1, FOB3.2.2, FOB3.2.2.1, FOB3.2.2.2, FOB3.2.3, FOB3.2.4, FOB3.2.5, FOB3.2.6, FOB4.2.1, FOB4.2.1.1, FOB4.2.1.1.1, FOB4.2.1.2, FOB4.2.1.2.1, FOB4.2.1.3, FOB4.2.1.3.1, FOB4.2.1.3.2, FOB5, FOB6
MS3	Non mappabile perché non ci sono requisiti associati.
MS4	FAOB1, FAOB2, FAOB2.1, FAOB2.1.1, FAOB2.1.1.1, FAOB2.1.2, FAOB2.1.2.1, FAOB2.1.3, FAOB2.1.3.1, FAOB2.1.3.1.1, FAOB2.2, FAOB2.2.1, FAOB2.2.2, FAOB2.2.3, FAOB2.2.4, FAOB2.2.5, FAOB2.2.6, FAOB3

MS5	FOB1, FOB2, FOB2.1, FOB3, FOB3.1, FOB3.1.1, FOB3.1.2, FOB3.1.3, FOB3.1.4, FOB3.1.5, FOB3.2, FOB3.2.1, FOB3.2.2, FOB3.2.2.1, FOB3.2.2.2, FOB3.2.3, FOB3.2.4, FOB3.2.5, FOB3.2.6, FOB4.1.1.1.3, FOB4.2.1, FOB4.2.1.1, FOB4.2.1.1.1, FOB4.2.1.2, FOB4.2.1.2.1, FOB4.2.1.3, FOB4.2.1.3.1, FOB4.2.1.3.2, FOB6, FAOB1, FAOB2, FAOB2.1, FAOB2.1.1, FAOB2.1.1.1, FAOB2.1.2, FAOB2.1.2.1, FAOB2.1.3, FAOB2.1.3.1, FAOB2.1.3.1.1, FAOB2.2, FAOB2.2.1, FAOB2.2.2, FAOB2.2.3, FAOB2.2.4, FAOB2.2.5, FAOB2.2.6, FAOB3

Tabella 6: Tracciamento componenti - requisiti

9.3 Tracciamento requisiti - componenti

Requisiti	Componenti
FOB0	VU1
FOB1	VU1, VU2, PU1, PU8, MS2, MS5, MCU1
FOB1.1	VU2, PU1
FOB1.2	VU2, PU1
FOB2	VU1, VU3, PU2, PU8, MS2, MS5
FOB2.1	VU3, PU2, PU8, MS2, MS5
FOB2.1.1	VU3, PU2
FOB2.2	VU3, PU2
FOB2.2.1	PU2
FOB2.3	VU3, PU2
FOB2.4	VU3, PU2
FOB2.5	VU3, PU2
FOB2.6	VU3, PU2
FOB3	VU1, VU4, PU3, PU8, MS2, MS5
FOB3.1	VU4, VU5, PU3, PU5, PU8, MS2, MS5
FOB3.1.1	VU4, VU5, PU3, PU5, PU8, MS2, MS5
FOB3.1.2	VU4, VU5, PU3, PU5, PU8, MS2, MS5
FOB3.1.3	VU4, VU5, PU3, PU5, PU8, MS2, MS5
FOB3.1.4	VU4, VU5, PU3, PU5, PU8, MS2, MS5
FOB3.1.5	VU4, VU5, PU3, PU5, PU8, MS2, MS5
FOB3.2	VU4, VU6, PU3, PU8, MS2, MS5
FOB3.2.1	VU4, VU6, PU3, PU8, MS2, MS5
FOB3.2.2	VU4, VU6, PU3, PU8, MS2, MS5
FOB3.2.2.1	VU4, VU6, PU3, PU8, MS2, MS5
FOB3.2.2.2	VU4, VU6, PU3, PU8, MS2, MS5
FOB3.2.3	VU4, VU6, PU3, PU8, MS2, MS5
FOB3.2.4	VU4, VU6, PU3, PU8, MS2, MS5
FOB3.2.5	VU4, VU6, PU3, PU8, MS2, MS5
FOB3.2.6	VU4, VU6, PU3, PU8, MS2, MS5
FOB4	VU1, VU7, VU8, PU4, PU5, PU6, PU7, MS1
FOB4.1	VU8, PU4, PU6, PU7, MS1

FOB4.1.1	VU8, PU4, PU6, PU7
FOB4.1.1.1	VU8, PU4, PU6, PU7, MS1
FOB4.1.1.1.1	VU9, PU6, PU7
FOB4.1.1.1.1.1	VU9, PU6, PU7
FOB4.1.1.1.1.2	VU9, PU6, PU7
FOB4.1.1.1.2	VU8, PU4
FOB4.1.1.1.3	VU8, PU4, MS1, MS5
FOB4.1.1.1.4	VU8, PU4, PU5
FOB4.1.1.1.4.1	VU8, PU4, PU5, PU6, PU7
FOB4.1.1.1.4.2	VU8, PU4, PU5, PU6, PU7
FOB4.1.1.1.4.3	VU8, PU4, PU5, PU6, PU7
FOB4.1.1.1.4.4	VU8, PU4, PU5, PU6, PU7
FOB4.1.2	VU8, PU4, PU6, PU7
FOB4.2	VU10, PU4, PU6, PU7, MS1
FOB4.2.1	VU10, PU4, PU6, PU7, PU8, MS1, MS2, MS5
FOB4.2.1.1	VU10, PU4, PU8, MS2, MS5
FOB4.2.1.1.1	VU10, PU4, PU8, MS2, MS5
FOB4.2.1.2	VU10, PU4, PU8, MS2, MS5
FOB4.2.1.2.1	VU10, PU4, PU8, MS2, MS5
FOB4.2.1.3	VU10, PU4, PU8, MS2, MS5
FOB4.2.1.3.1	VU10, PU4, PU8, MS2, MS5
FOB4.2.1.3.2	VU10, PU4, PU8, MS2, MS5
FOB5	PU4, PU5, PU6, PU7, PU8, MS1, MS2
FOB6	VU2, PU1, PU8, MS2, MS5
FAOB0	VA1, VA2, PA1, PA2, PA3, PA4, MS4, MS5
FAOB1	VA1, PA1, PA4, MS4, MS5, MCA1
FAOB1.1	VA2, PA1, PA4, MS4, MS5
FAOB1.2	VA2, PA1, PA4, MS4, MS5
FAOB2	VA1, VA3, PA2, PA4, MS4, MS5
FAOB2.1	VA3, PA2, PA4, MS4, MS5
FAOB2.1.1	VA3, PA2, PA4, MS4, MS5
FAOB2.1.1.1	VA3, PA2, PA4, MS4, MS5
FAOB2.1.2	VA3, PA2, PA4, MS4, MS5
FAOB2.1.2.1	VA3, PA2, PA4, MS4, MS5

FAOB2.1.3	VA3, PA2, PA4, MS4, MS5
FAOB2.1.3.1	VA3, PA2, PA4, MS4, MS5
FAOB2.1.3.1.1	VA3, PA2, PA4, MS4, MS5
FAOB2.1.3.2	VA3, PA2, PA4, MS4, MS5
FAOB2.1.3.2.1	VA3, PA2, PA4, MS4, MS5
FAOB2.2	VA3, PA2, PA3, PA4, MS4, MS5
FAOB2.2.1	VA3, PA2, PA3, PA4, MS4, MS5
FAOB2.2.2	VA3, PA2, PA3, PA4, MS4, MS5
FAOB2.2.3	VA3, PA2, PA3, PA4, MS4, MS5
FAOB2.2.4	VA3, PA2, PA3, PA4, MS4, MS5
FAOB2.2.5	VA3, PA2, PA3, PA4, MS4, MS5
FAOB2.2.6	VA3, PA2, PA3, PA4, MS4, MS5
FAOB3	VA1, VA2, PA1, PA3, PA4, MS4, MS5
VOB1	Vincolo soddisfatto in fase di Specifica Tecnica.
VOB2	Vincolo tecnologico implementato dal software.
VOB3	Vincolo tecnologico implementato dal software.
VOB4	Vincolo tecnologico implementato dal software.
VOB5	Vincolo tecnologico implementato dal software.
VOB6	Vincolo tecnologico implementato dal software.
VOB7	Vincolo tecnologico implementato dal software.
VOB8	Vincolo tecnologico implementato dal software.
VOB9	Vincolo soddisfatto in fase progettuale.
VOB10	Vincolo soddisfatto in fase di Specifica Tecnica.

Tabella 7: Tracciamento requisiti - componenti

10 Interfacce Utenti

Di seguito sono descritte le interfacce utente con le quali l'utente deve interagire per usufruire del prodotto MyTalk.
Le interfacce descritte in questa sezione sono esclusivamente a scopo illustrativo.

10.1 GUI 1: Pagina accesso utente base

La pagina accesso utente base è composta da due interfacce:

- **Login:** permette di effettuare l'autenticazione al sistema per accedere al sistema;
- **Registrazione:** permette di accedere alla form di registrazione nel sistema.

Per impostazione di default$_{|g|}$ la pagina iniziale al momento dell'accesso è aperta sull'interfaccia di login.

10.1.1 GUI 1.1: Login

Attraverso l'interfaccia di login l'utente può inserire le proprie credenziali (username e password) e accedere all'applicazione.

Figura 28: GUI 1.1 - Interfaccia di login.

10.1.2 GUI 1.2: Registrazione

Attraverso l'interfaccia di registrazione l'utente può inserire le proprie credenziali per ottenere la possibilità di accedere all'applicazione. La struttura della form di registrazione è descritta nella figura 10.1.2.

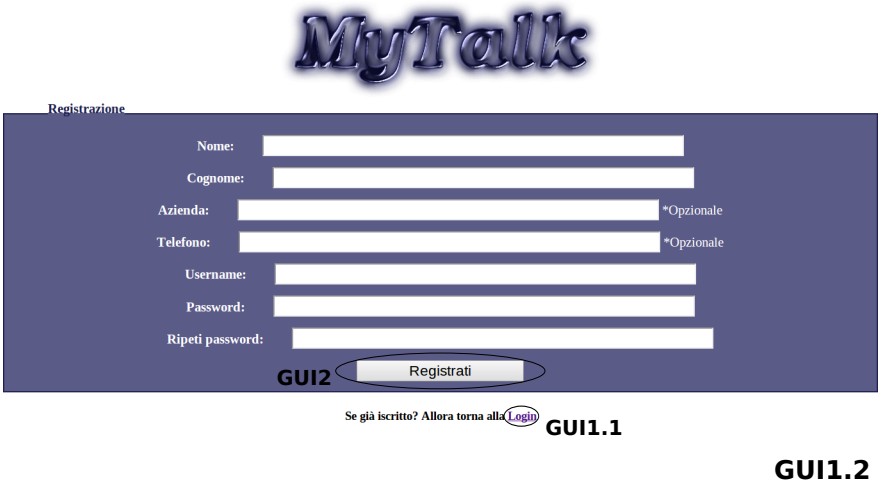

Figura 29: GUI 1.2 - Interfaccia di registrazione.

10.2 GUI 2: Pagina principale utente base

La pagina principale utente base presenta due parti fisse.
Nella prima vengono visualizzati i video in entrata e in uscita degli utenti in contatto durante una chiamata attiva. Se non ci sono comunicazioni attive, questa parte viene oscurata fino all'arrivo di una nuova trasmissione. Nella seconda, vengono visualizzati i dati principali dell'utente e viene offerta la possibilità di effettuare il logout e di scegliere tra una delle seguenti due interfacce:

- **Comunicazioni:** permette di gestire le comunicazioni;
- **Dati Utente:** permette di visualizzare e modificare le credenziali dell'utente.

10.2.1 GUI 2.1: Comunicazione

La seguente interfaccia è composta da vari pannelli che permettono di interagire con le comunicazioni in entrata ed uscita.

- **Effettua chiamata:** permette di decidere il metodo di selezione dell'utente da chiamare e successivamente di chiamare l'utente scelto.
- **Chiamata in arrivo:** questo pannello appare al momento dell'arrivo di una richiesta di comunicazione e permette di decidere se accettarla o rifiutarla.
- **Informazioni chiamata:** si presenta quando è presente una chiamata attiva e visualizza i dati della comunicazione e permette di terminare la conversazione.
- **Chiamata terminata:** si presenta al termine di tutte le comunicazione con lo scopo di valutare la qualità del servizio.

Figura 30: GUI 2 - Pagina principale utente.

10.2.2 GUI 2.1.1: Effettua chiamata

Il pannello si espande permettendo di scegliere:

- **Chiamata per IP**$_{|g|}$
- **Chiamata per username**
- **Chiamata per scelta utente da lista**

Una volta effettuata la scelta si espande e viene visualizzata la form richiesta dalla quale è possibile inviare una richiesta di comunicazione.
Il pannello non è accessibile durante una comunicazione in corso fino a dopo aver effettuato la valutazione.

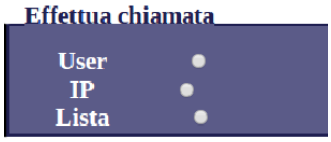

Figura 31: GUI 2.1.1 - Pannello effettua chiamata.

10.2.3 GUI 2.1.1.1 Chiamata per IP

Il pannello si espande e presenta una form per l'inserimento dell'indirizzo IP$_{|g|}$.

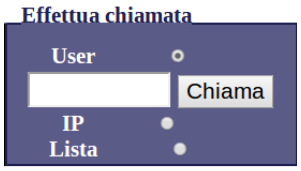

Figura 32: GUI 2.1.1.1 - Pannello effettua chiamata attraverso IP$_{|g|}$.

10.2.4 GUI 2.1.1.2 Chiamata per Username

Il pannello si espande e presenta una form per l'inserimento dello username.

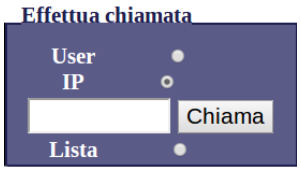

Figura 33: GUI 2.1.1.2 - Pannello effettua chiamata attraverso username.

10.2.5 GUI 2.1.1.3 Chiamata per scelta utente da lista

Il pannello si espande e permette di scegliere dalla lista di tutti gli utenti online l'utente da chiamare.

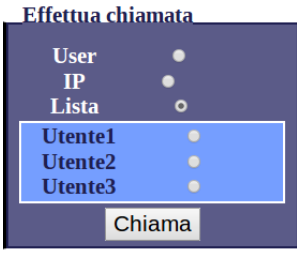

Figura 34: GUI 2.1.1.3 - Pannello effettua chiamata attraverso scelta da lista.

10.2.6 GUI 2.1.2 Chiamata in entrata

Il pannello appare all'arrivo di una richiesta di comunicazione in entrata e permette all'utente di scegliere se accettare o rifiutare di rispondere.

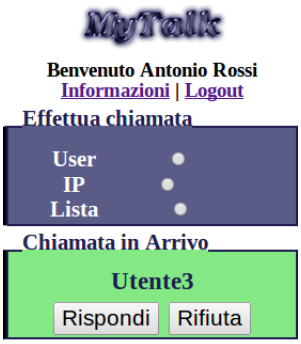

Figura 35: GUI 2.1.2 - Pannello di chiamata in entrata.

10.2.7 GUI 2.1.3 Informazioni chiamata

Il pannello informazioni chiamata visualizza durata, latenza, byte inviati e la velocità della comunicazione. Infine permette di terminare la conversazione.

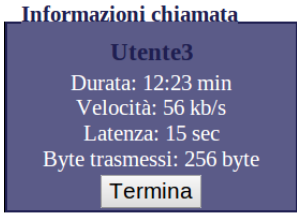

Figura 36: GUI 2.1.3 - Pannello informazioni chiamata.

10.2.8 GUI 2.1.4 Chiamata terminata

Il pannello si attiva al termine della comunicazione allo scopo di valutare la qualità della conversazione. La valutazione di qualità avviene su una scala da 1 a 5.

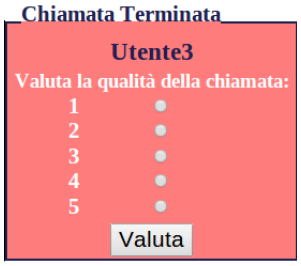

Figura 37: GUI 2.1.4 - Pannello chiamata terminata.

10.2.9 GUI 2.2: Dati utente

La seguente interfaccia visualizza le informazioni dell'utente e permette di modificarne i dati attraverso due pannelli:

- **Visualizza dati**
- **Modifica dati**

All'apertura di questa interfaccia si apre di default$_{|g|}$ il pannello visualizza dati.

10.2.10 GUI 2.2.1: Visualizza dati

Il pannello che visualizza i dati mostra le informazioni dell'utente e chiede all'utente se desidera modificarle.

Figura 38: GUI 2.2 - Dati utente.

10.2.11 GUI 2.2.2: Modifica dati

Il pannello che visualizza e modifica i dati dell'utente.

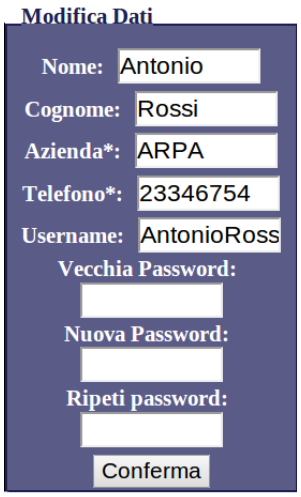

Figura 39: GUI 2.2.2 - Pannello modifica dati.

10.3 GUI 3: Pagina accesso amministratore

La pagina di accesso amministratore a differenza dell'interfaccia di accesso utente base presenta una sola interfaccia di login, solo un utente predefinito può accedere alla pagina principale amministratore.

Figura 40: GUI 3 - Pagina accesso amministratore.

10.4 GUI 4: Pagina principale amministratore

La pagina principale amministratore è divisa in due parti. Una è la lista delle chiamate con relative informazioni che l'amministratore vuole visualizzare, l'altra è la parte dalla quale l'amministratore può, tramite un pannello, filtrare le chiamate o effettuare il logout.
La lista contiene tutte le informazioni relative alle telefonate: mittente, destinatario, latenza, byte trasmessi, giudizio, velocità, durata e data.

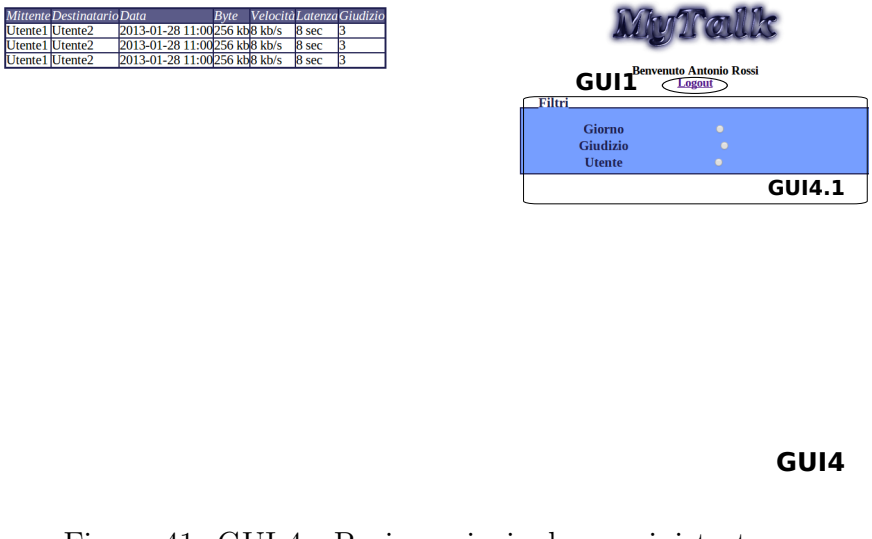

Figura 41: GUI 4 - Pagina principale amministratore.

10.4.1 GUI 4.1 Filtro

Il pannello filtro permette all'amministratore di selezionare in che modo effettuare la scelta del filtro. Ci possono essere tre possibili filtri:

- Per giorno
- Per giudizio
- Per utente

10.4.2 GUI 4.1.1 Filtro per giorno

Il pannello si espande e permette di selezionare il giorno da filtrare.

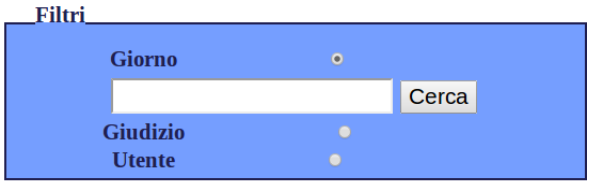

Figura 42: GUI 4.1.1 - Pannello filtro per giorno.

10.4.3 GUI 4.1.2 Filtro per giudizio

Il pannello si espande e permette di selezionare il giudizio da filtrare.

Figura 43: GUI 4.1.2 - Pannello filtro per giudizio.

10.4.4 GUI 4.1.3 Filtro per utente

Il pannello si espande e permette di selezionare in che modo selezionare l'utente da filtrare. Le possibilità sono tre:

- Scelta per $IP_{|g|}$
- Scelta per username
- Scelta da lista

10.4.5 GUI 4.1.3.1 Filtro per utente da IP

Il pannello si espande e presenta una form per l'inserimento dell'indirizzo $IP_{|g|}$.

10.4.6 GUI 4.1.3.2 Filtro per utente da username

Il pannello si espande e presenta una form per l'inserimento dello username.

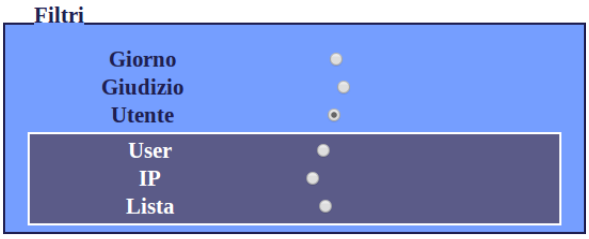

Figura 44: GUI 4.1.3 - Pannello filtro per utente.

Figura 45: GUI 4.1.3.1 - Pannello filtro per utente da IP.

Figura 46: GUI 4.1.3.2 - Pannello filtro per utente da username.

10.4.7 GUI 4.1.3.3 Filtro per utente da lista

Il pannello si espande permette di scegliere dalla lista di tutti gli utenti online l'utente da chiamare.

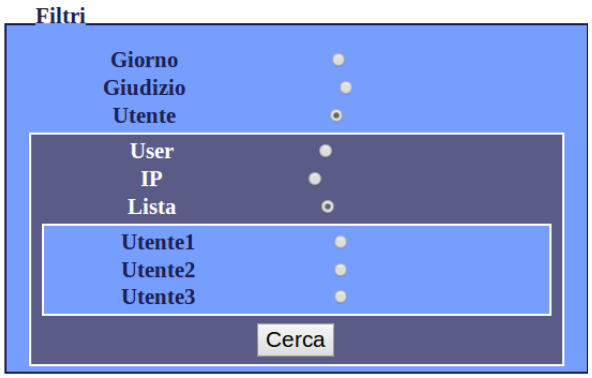

Figura 47: GUI 4.1.3.3 - Pannello filtro per utente da lista.

Appendici

A WebRTC

A.1 Descrizione generale

Le Web Real-Time Communications, d'ora in poi WebRTC$_{|g|}$, aggiungono nuove potenzialità al browser$_{|g|}$, permettendogli di interagire direttamente con altri browser$_{|g|}$ stabilendo una comunicazione real-time. Si può pensare di avere funzionalità simili a quelle di Skype^{TM}, ma senza installare software$_{|g|}$ o plugin$_{|g|}$. Queste funzionalità sono rese disponibili agli sviluppatori tramite tag standard HTML5$_{|g|}$ (in realtà non ancora approvato come standard anche se supportato da molti browser$_{|g|}$, lo diventerà da fine 2014) ed API$_{|g|}$ JavaScript$_{|g|}$.
Lo scenario più comune è quello in cui entrambi i browser$_{|g|}$ stanno eseguendo la stessa applicazione WebRTC$_{|g|}$, scaricata dalla medesima pagina web$_{|g|}$. Questo produce il cosiddetto "Triangolo WebRTC$_{|g|}$": il web server$_{|g|}$ si occupa della connessione con i due browser$_{|g|}$, detti anche *peer*$_{|g|}$ (lati del triangolo), tra i quali è instaurata una connessione (base del triangolo), detta *peer connection*, attraverso la quale passa il flusso di dati.

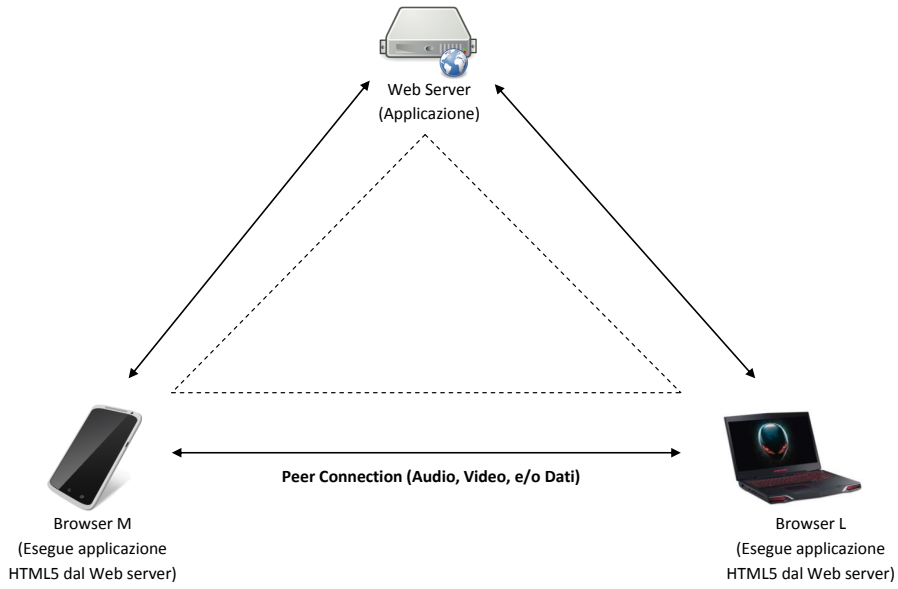

Figura 48: Il Triangolo WebRTC$_{|g|}$.

Le tre azioni principali richieste per instaurare una sessione WebRTC$_{|g|}$ sono:

1. ottenere i dati multimediali locali;
2. stabilire una connessione tra il browser$_{|g|}$ ed un altro peer$_{|g|}$ (solitamente un altro browser$_{|g|}$);

3. aggiungere i canali multimediale e di dati alla connessione.

Tali azioni sono illustrate nella figura A.1:

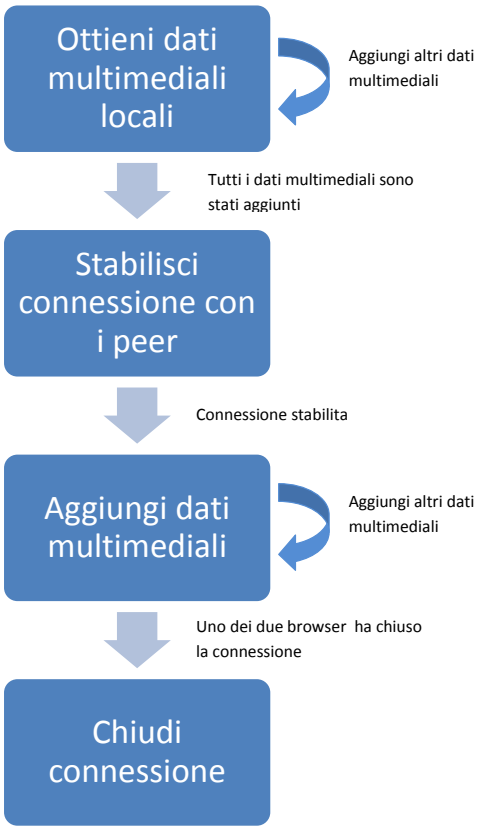

Figura 49: Instaurazione della sessione WebRTC$_{|g|}$.

Vengono ora descritti più in dettaglio i singoli passi, ai quali ne viene aggiunto un quarto che riguarda la chiusura della sessione.

A.1.1 Ottenere dati multimediali locali

Il metodo più comune per ottenere dati locali è quello di utilizzare il metodo `getUserMedia()`, che permette di ottenere un `LocalMediaStream`. I flussi multimediali possono poi essere messi insieme tramite l'API$_{|g|}$ `MediaStream`. Per ragioni di privacy, è richiesto il consenso dell'utente per accedere al suo microfono e alla sua webcam.

A.1.2 Stabilire la Peer Connection

Una Peer Connection è una connessione multimediale diretta tramite due web$_{|g|}$ browser$_{|g|}$ (peer$_{|g|}$), per la cui instaurazione viene utilizzata l'API$_{|g|}$ `RTCPeerConnection`.

Tale connessione permette il flusso audio/video tra i due peer$_{|g|}$. È necessaria una connessione per ogni coppia di peer$_{|g|}$. L'unico input che il costruttore di `RTCPeerConnection` richiede è un oggetto di configurazione che ICE$_{|g|}$ (Interactive Connectivity Establishment) userà per "fare buchi" (punch holes) attraverso i NAT$_{|g|}$ e i firewall$_{|g|}$ che separano i peer$_{|g|}$.

A.1.3 Scambiare dati multimediali

Ogni scambio di dati richiede una negoziazione (o rinegoziazione) tra i browser$_{|g|}$ su come i dati verranno rappresentati nel canale. A tale scopo quando viene effettuata una richiesta, in locale o in remoto, di aggiungere o rimuovere dati, il browser$_{|g|}$ genera un appropriato oggetto `SessionDescription` che rappresenta il flusso di dati attraverso la peer connection. Una volta che i browser$_{|g|}$ si sono scambiati questo oggetto, la sessione di scambio di dati può essere stabilita.

A.1.4 Chiudere la connessione

Uno qualsiasi dei browser$_{|g|}$ può chiudere la connessione. L'applicazione chiama il metodo `close()` sulla `RTCPeerConnection` per indicare che ha finito di usare la connessione, solitamente in seguito ad una richiesta dell'utente che ha chiuso il browser$_{|g|}$ o la scheda (tab) corrente.

A.1.5 Esempio di connessione

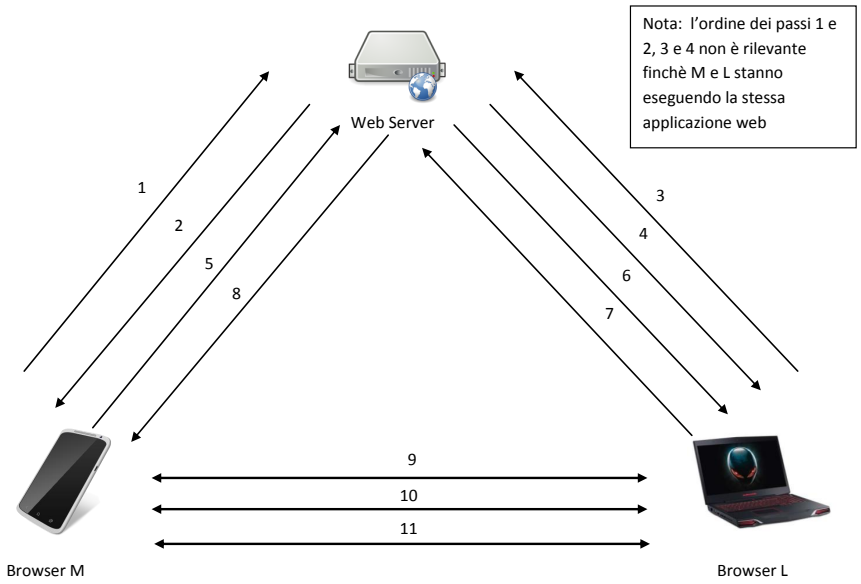

Figura 50: Instaurazione della sessione WebRTC$_{|g|}$.

Nella figura A.1.5 è illustrato un esempio di instaurazione della sessione WebRTC$_{|g|}$. Essa è composta dai seguenti passi:

1. il browser$_{|g|}$ M richiede la pagina web$_{|g|}$ dal web server$_{|g|}$;
2. il web server$_{|g|}$ fornisce la pagina web$_{|g|}$ a M con il codice JavaScript$_{|g|}$ relativo alle WebRTC$_{|g|}$;
3. il browser$_{|g|}$ L richiede la pagina web$_{|g|}$ dal web server$_{|g|}$;
4. il web server$_{|g|}$ fornisce la pagina web$_{|g|}$ a L con il codice JavaScript$_{|g|}$ relativo alle WebRTC$_{|g|}$;
5. M decide di comunicare con L, lo script JavaScript$_{|g|}$ eseguito su M permette la spedizione al web server$_{|g|}$ dell'oggetto `SessionDescription` di M;
6. il web server$_{|g|}$ invia l'oggetto `SessionDescription` di M al codice JavaScript$_{|g|}$ su L;
7. lo script JavaScript$_{|g|}$ eseguito su L permette la spedizione dell'oggetto `SessionDescription` (risposta) al web server$_{|g|}$;
8. il web server$_{|g|}$ invia l'oggetto `SessionDescription` di L ad M;
9. M ed L iniziano il processo di *hole punching* per determinare il modo migliore per comunicare;
10. al completamento dell'hole punching, M ed L negoziano una chiave per la sicurezza della trasmissione;
11. M ed L iniziano a scambiarsi dati, audio e video.

A.2 L'hole punching

La natura del NAT$_{|g|}$ rende difficoltosa la creazione di sessioni peer-to-peer$_{|g|}$, ma usando una tecnica chiamata "hole punching" essa può essere creata con successo in molti casi. L'hole punching è una tecnica usata nelle reti di computer per stabilire comunicazioni tra due parti, situate in sedi differenti, che sono entrambe dietro un NAT$_{|g|}$ e/o protette da firewall$_{|g|}$. Questa tecnica richiede quattro prerequisiti:

1. i due browser$_{|g|}$ che stanno provando a stabilire una connessione diretta devono, nello stesso tempo, inviare i pacchetti hole puching. Devono pertanto conoscere l'indirizzo al quale invieranno i pacchetti. Il requisito 1 è soddisfatto utilizzando il web server$_{|g|}$ per coordinare l'hole puching. Il web server$_{|g|}$ sa che una dovrà essere stabilita una sessione tra i browser$_{|g|}$, e assicura che essi inizino l'hole punching approssimativamente nello stesso tempo.
2. i due browser$_{|g|}$ hanno bisogno di conoscere più indirizzi IP$_{|g|}$ possibili che potranno essere usati per raggiungerli. Questi indirizzi sono spesso descritti come indirizzi privati se stanno all'interno di un NAT$_{|g|}$ o indirizzi pubblici se stanno al di fuori da un NAT$_{|g|}$. Il requisito 2 è soddisfatto utilizzando uno STUN$_{|g|}$ server$_{|g|}$. Ogni browser$_{|g|}$ chiede lo STUN$_{|g|}$ server$_{|g|}$ inviando un pacchetto STUN$_{|g|}$. Lo STUN$_{|g|}$ server$_{|g|}$ risponde indicando l'indirizzo IP$_{|g|}$ osservato nel pacchetto di test. Cioè, risponde con l'indirizzo mappato dal NAT$_{|g|}$. Questo indirizzo IP$_{|g|}$ ricavato dallo STUN$_{|g|}$ server$_{|g|}$ è condiviso con altri browser$_{|g|}$ e diventa un potenziale candidato di indirizzo. L'indirizzo può essere IPv4, IPv6 o una combinazione di entrambi.

3. come ultima possibilità, deve esistere un server$_{|g|}$ con un indirizzo IP$_{|g|}$ pubblico e non nascosto da un NAT$_{|g|}$ e quindi raggiungibile da entrambi i browser$_{|g|}$. Il requisito 3 è soddisfatto utilizzando un TURN$_{|g|}$ server$_{|g|}$, che fornisce ai browser$_{|g|}$ un indirizzo di riserva. Questo indirizzo viene poi aggiunto alla lista dei candidati.

4. devono essere usati flussi assimetrici. Ossia, UDP$_{|g|}$ deve operare in maniera simile a TCP$_{|g|}$. Il Requisito 4 è soddisfatto dal browser$_{|g|}$ inviando la trasmissione dalla stessa porta UDP$_{|g|}$ che il browser$_{|g|}$ sta usando per ascoltare la trasmissione in arrivo. Questo fa sì che le due sessioni monodirezionali RTP$_{|g|}$ inviate tramite UDP$_{|g|}$ appaiano al NAT$_{|g|}$ come una sessione RTP$_{|g|}$ bidirezionale.

B Tecnologie utilizzate

B.1 MySQL

MySQL, definito Oracle MySQL, è un Relational DataBase Management System (RDBMS), composto da un client$_{|g|}$ con interfaccia a riga di comando e un server$_{|g|}$, entrambi disponibili sia per sistemi GNU/Linux che per Windows. Verrà utilizzato un database$_{|g|}$ relazionale per la sua estendibilità, per la sua maggiore espressività rispetto ad un database$_{|g|}$ XML$_{|g|}$ e per la semplicità di accedervi, utilizzando il linguaggio Java$_{|g|}$, tramite JDBC. Nel progetto *MyTalk* è stato utilizzato per memorizzare i dati degli utenti, quelli degli amministratori e quelli relativi alle statistiche.

B.2 JDBC

JDBC (Java DataBase Connectivity) è un connettore per database$_{|g|}$ che consente l'accesso alle basi di dati da qualsiasi programma scritto con il linguaggio di programmazione Java$_{|g|}$, indipendentemente dal tipo di DBMS$_{|g|}$ utilizzato. È costituita da una API$_{|g|}$, raggruppata nel package$_{|g|}$ java.sql, che serve ai client$_{|g|}$ per connettersi a un database$_{|g|}$. Fornisce metodi per interrogare e modificare i dati. È orientata ai database$_{|g|}$ relazionali ed è Object Oriented. La piattaforma Java 2 Standard Edition contiene le API$_{|g|}$ JDBC, insieme all'implementazione di un bridge JDBC-ODBC$_{|g|}$, che permette di connettersi a database$_{|g|}$ relazionali che supportino ODBC$_{|g|}$. Verrà utilizzato per effettuare effettuare query$_{|g|}$ sul database$_{|g|}$; tale risultato verrà passato ad un oggetto che poi lo invierà, tramite web socket, al client$_{|g|}$ richiedente.
Verrà utilizzato con un design pattern DAO (Database Access Object).

B.3 JavaScript

JavaScript$_{|g|}$ è un linguaggio di scripting lato client$_{|g|}$ orientato agli oggetti, comunemente usato nei siti web$_{|g|}$, ed interpretato dai browser$_{|g|}$. Ciò permette di alleggerire il server$_{|g|}$ dal peso della computazione, che viene eseguita dal client$_{|g|}$. Questo è un aspetto molto importante per lo sviluppo del capitolato MyTalk, che richiede che il server$_{|g|}$ si occupi solamente della connessione tra i client$_{|g|}$. Avremo quindi un "thin server" e un "fat client". La caratteristica principale di JavaScript$_{|g|}$ è, appunto, quella di essere un linguaggio interpretato: il codice non viene compilato, ma interpretato, dal browser$_{|g|}$. Essendo molto popolare e ormai consolidato, JavaScript$_{|g|}$ può essere eseguito dalla maggior parte dei browser$_{|g|}$, sia desktop che mobile, grazie anche alla sua leggerezza. Uno degli svantaggi di questo linguaggio è che ogni operazione che richieda informazioni che devono essere recuperate da un database$_{|g|}$ deve passare attraverso un linguaggio che effettui esplicitamente la transazione, per poi restituire i risultati a JavaScript$_{|g|}$. Tale operazione richiede l'aggiornamento totale della pagina, ma grazie ad AJAX$_{|g|}$ è possibile superare questo limite.

B.4 WebSocket

WebSocket$_{|g|}$ è una tecnologia web$_{|g|}$ che fornisce canali di comunicazione full-duplex$_{|g|}$ attraverso una singola connessione TCP$_{|g|}$. WebSocket$_{|g|}$ può essere utilizzato da qualsiasi applicazione client-server$_{|g|}$, e permette ai browser$_{|g|}$ di comunicare tra loro in maniera asincrona senza l'intervento dell'utente. Le comunicazioni sono fatte

attraverso la porta TCP$_{|g|}$ 80, che è un vantaggio per quegli ambienti che bloccano porte non standard utilizzando dei firewall$_{|g|}$.

B.5 HTML5

HTML5$_{|g|}$ verrà utilizzato per definire la struttura della pagina web$_{|g|}$ che ospita il software MyTalk. Tale struttura sarà completamente separata dalla presentazione, che verrà realizzata tramite CSS3$_{|g|}$. HTML5$_{|g|}$ presenta, rispetto ad HTML$_{|g|}$ 4, diversi vantaggi per lo svolgimento del progetto:

- introduzione di elementi di controllo per i menu di navigazione (tag `<nav>`);
- introduzione di elementi specifici per l'inserimento di contenuti multimediali (tag `<video>` e `<audio>`).

B.6 CSS3

CSS$_{|g|}$ (Cascading Style Sheet) è un linguaggio utile a definire l'aspetto di pagine HTML$_{|g|}$, XHTML$_{|g|}$ e XML$_{|g|}$, che devono presentare un collegamento al loro foglio di stile nell'header (la parte del documento HTML$_{|g|}$ che introduce un gruppo di ausili introduttivi o di navigazione). Grazie ai CSS$_{|g|}$, è possibile una completa separazione tra la presentazione (cioè l'aspetto grafico delle pagine web$_{|g|}$) ed i contenuti delle pagine stesse. Ciò semplifica la comprensione, la manutenzione e la portabilità. Rispetto a CSS2, CSS3$_{|g|}$ introduce funzionalità grafiche più avanzate.

B.7 AJAX

È una tecnica per lo sviluppo di applicazioni web interattive. Grazie ad AJAX$_{|g|}$ può avvenire uno scambio di dati in background tra il browser$_{|g|}$ del client$_{|g|}$ e il server$_{|g|}$; ciò permette l'aggiornamento dinamico della pagina web$_{|g|}$ senza l'intervento esplicito dell'utente. Normalmente le funzioni richiamate sono scritte con il linguaggio JavaScript$_{|g|}$. AJAX$_{|g|}$ è una tecnica multi-piattaforma utilizzabile su molti sistemi operativi$_{|g|}$, architetture informatiche e sui principali browser$_{|g|}$ web$_{|g|}$.

B.8 Java 7

Linguaggio di programmazione ad oggetti sviluppato da Oracle. Esso è indipendentemente dalla piattaforma ma necessita l'installazione, sulla macchina che lo esegue, di un interprete detto Java Virtual Machine (JVM$_{|g|}$). Java$_{|g|}$ è particolarmente adatto per eseguire codice da sorgenti remote in modo sicuro. Verrà utilizzato, come da requisito del capitolato, per la programmazione del server$_{|g|}$ che si occuperà di gestire la connessione e la sessione di comunicazione tra i client$_{|g|}$.

B.9 GWT (Google Web Toolkit)

Framework$_{|g|}$ di sviluppo Java$_{|g|}$ con il quale è possibile realizzare applicazioni AJAX$_{|g|}$ complesse e aderenti agli standard web$_{|g|}$. Tale strumento permette di sviluppare mediante il linguaggio Java$_{|g|}$, ben conosciuto dal gruppo e ricco di strumenti di sviluppo e testing, e si occupa di tradurre il codice in JavaScript$_{|g|}$ e HTML$_{|g|}$, generando una soluzione AJAX$_{|g|}$. Ciò rende quindi possibile l'utilizzo dell'IDE$_{|g|}$ Eclipse

non solo per la parte server$_{|g|}$, ma anche per il front-end$_{|g|}$ dell'applicazione. GWT offre comunque la possibilità di scrivere codice JavaScript$_{|g|}$ nativo (JSNI, JavaScript Native Interface) per accedere alle funzionalità non esposte nell'API$_{|g|}$ GWT, come l'interazione con le librerie WebRTC$_{|g|}$ e con i WebSocket$_{|g|}$.