

# MyTalk

---

Software di comunicazione tra utenti senza  
requisiti di installazione



clockworkTeam7@gmail.com

---

Specifica Tecnica

v 2.0

---

## Informazioni sul documento

<b>Nome documento</b>	Specifica Tecnica
<b>Versione documento</b>	v 2.0
<b>Data creazione</b>	2013/01/14
<b>Data ultima modifica</b>	2013/03/02
<b>Uso documento</b>	Esterno
<b>Redazione</b>	Bain Giacomo Ceseracciu Marco Furlan Valentino La Bruna Agostino
<b>Verifica</b>	Zohouri Haghian Pardis
<b>Approvazione</b>	Gavagnin Jessica
<b>Lista distribuzione</b>	gruppo <i>Clockwork</i> Zucchetti SPA Prof. Tullio Vardanega

## Sommario

Questo documento vuol definire l'architettura generale che il prodotto dovrà avere.

## Diario delle modifiche

Autore	Modifica	Data	Versione
Gavagnin Jessica	Approvazione del documento	2013/03/02	v2.0
Zohouri Haghian Pardis	Verifica del documento	2013/03/01	v1.3
Furlan Valentino	Modifica del capitolo 9	2013/02/26	v 1.4
Ceseracciu Marco	Modifica del capitolo 6	2013/02/19	v 1.3
Bain Giacomo	Modifica del capitolo 5	2013/02/15	v 1.2
La Bruna Agostino	Modifica dei capitoli 7 e 8	2013/02/13	v 1.1
La Bruna Agostino	Verifica documento e approvazione	2013/01/29	v 1.0
Palmisano Maria Antonietta	Controllo dei tracciamenti	2013/01/29	v 0.16
Bain Giacomo	Controllo concettuale capitolo 9 e 10	2013/01/28	v 0.15
Palmisano Maria Antonietta	Controllo concettuale fino al capitolo 6	2013/01/28	v 0.14
Bain Giacomo	Controllo ortografico, strutturale e sintattico	2013/01/26	v 0.13
Furlan Valentino	Stesura capitolo 9	2013/01/25	v 0.12
Gavagnin Jessica	Stesura capitolo 10	2013/01/25	v 0.11
Ceseracciu Marco	Stesura capitolo 7	2013/01/24	v 0.10
Furlan Valentino	Stesura capitolo 8	2013/01/24	v 0.9
Ceseracciu Marco	Bozza capitolo 7	2013/01/23	v 0.8
Bain Giacomo	Stesura capitolo 2	2013/01/22	v 0.7
Gavagnin Jessica	Stesura capitolo Server	2013/01/21	v 0.6
Zohouri Haghian Pardis	Stesura capitolo Client	2013/01/21	v 0.5
Bain Giacomo	Stesura capitolo 4	2013/01/21	v 0.4
Gavagnin Jessica	Bozza capitolo 3	2013/01/17	v 0.3
Ceseracciu Marco	Bozza capitolo 2	2013/01/16	v 0.2
Zohouri Haghian Pardis	Creazione documento, stesura sezione Introduzione	2013/01/15	v 0.1

## Indice

<b>1</b>	<b>Introduzione</b>	<b>1</b>
1.1	Scopo del documento . . . . .	1
1.2	Scopo del prodotto . . . . .	1
1.3	Glossario . . . . .	1
1.4	Riferimenti . . . . .	1
1.4.1	Normativi . . . . .	1
1.4.2	Informativi . . . . .	1
<b>2</b>	<b>Architettura Generale</b>	<b>2</b>
2.1	Client . . . . .	2
2.2	Server . . . . .	3
2.3	Architettura dell'intero sistema . . . . .	3
<b>3</b>	<b>Design Pattern</b>	<b>6</b>
3.1	Multitier . . . . .	6
3.2	MVP . . . . .	7
3.3	MV* . . . . .	8
3.4	Observer . . . . .	8
<b>4</b>	<b>Strumenti Utilizzati</b>	<b>9</b>
4.1	Java . . . . .	9
4.2	HTML5 . . . . .	9
4.3	JavaScript . . . . .	9
4.4	SQLite . . . . .	9
4.5	Backbone.js . . . . .	10
<b>5</b>	<b>Client</b>	<b>11</b>
5.1	View . . . . .	11
5.1.1	mytalk.client.view.AuthenticationView . . . . .	12
5.1.2	mytalk.client.view.TutorialView . . . . .	12
5.1.3	mytalk.client.view.ContactsView . . . . .	12
5.1.4	mytalk.client.view.ContactView . . . . .	13
5.1.5	mytalk.client.view.FunctionsView . . . . .	13
5.1.6	mytalk.client.view.CallView . . . . .	13
5.1.7	mytalk.client.view.NotificationView . . . . .	14
5.1.8	mytalk.client.view.ChatView . . . . .	14
5.1.9	mytalk.client.view.RecordMessageView . . . . .	14
5.1.10	mytalk.client.view.StatisticsView . . . . .	15
5.1.11	mytalk.client.view.ModifyAccountDataView . . . . .	15
5.1.12	mytalk.client.view.SendFileView . . . . .	15
5.2	Communication . . . . .	16
5.2.1	mytalk.client.communication.AuthenticationCommunication . . . . .	16
5.2.2	mytalk.client.communication.ContactsCommunication . . . . .	16
5.2.3	mytalk.client.communication.CallCommunication . . . . .	16

5.2.4	mytalk.client.communication.NotificationCommunication	17
5.2.5	mytalk.client.communication.RecordMessageCommunication	17
5.2.6	mytalk.client.communication.TutorialCommunication . . .	17
5.2.7	mytalk.client.communication.SendFileCommunication . . .	17
5.3	Model . . . . .	18
5.3.1	mytalk.client.model.User . . . . .	18
5.3.2	mytalk.client.model.Contact . . . . .	18
5.3.3	mytalk.client.model.Statistics . . . . .	18
5.3.4	mytalk.client.model.TextMessage . . . . .	19
5.3.5	mytalk.client.model.Tutorial . . . . .	19
<b>6</b>	<b>Server</b>	<b>20</b>
6.1	Transfer Layer . . . . .	21
6.1.1	mytalk.server.transfer.CallTransfer . . . . .	21
6.1.2	mytalk.server.transfer.AuthenticationTransfer . . . . .	21
6.1.3	mytalk.server.transfer.TutorialTransfer . . . . .	21
6.1.4	mytalk.server.transfer.RecordMessageTransfer . . . . .	22
6.2	Manager Layer . . . . .	22
6.2.1	mytalk.server.usermanager.AuthenticationManager . . . . .	22
6.2.2	mytalk.server.usermanager.UserManager . . . . .	23
6.3	Data Layer . . . . .	23
6.3.1	mytalk.server.dao.UserDataDao . . . . .	23
6.3.2	mytalk.server.dao.UserDataDaoSQL . . . . .	24
6.3.3	mytalk.server.sqlite.UserData . . . . .	24
6.3.4	mytalk.server.shared.User . . . . .	24
6.3.5	mytalk.server.shared.UserList . . . . .	24
6.3.6	mytalk.server.dao.RecordMessageDataDao . . . . .	25
6.3.7	mytalk.server.dao.RecordMessageDataDaoSQL . . . . .	25
6.3.8	mytalk.server.sqlite.RecordMessageData . . . . .	25
6.3.9	mytalk.server.shared.RecordMessage . . . . .	25
6.3.10	mytalk.server.dao.TutorialDataDao . . . . .	26
6.3.11	mytalk.server.dao.TutorialDataDaoSQL . . . . .	26
6.3.12	mytalk.server.sqlite.TutorialData . . . . .	26
6.3.13	mytalk.server.shared.TutorialList . . . . .	26
<b>7</b>	<b>Tracciamento componenti-requisiti</b>	<b>27</b>
<b>8</b>	<b>Tracciamento requisiti-componenti</b>	<b>29</b>
<b>9</b>	<b>Diagramma delle attività</b>	<b>31</b>
9.1	Diagramma generale . . . . .	31
9.2	Sottoattività di Registrazione . . . . .	32
9.3	Sottoattività di Autenticazione . . . . .	33
9.4	Sottoattività di Modifica dati account . . . . .	34
9.5	Sottoattività di Chiamata IP . . . . .	35
9.6	Sottoattività di Servizi contatto . . . . .	36

9.7 Diagramma di Gestione delle notifiche . . . . .	37
<b>10 Prototipo interfaccia utente</b>	<b>38</b>

## Elenco delle figure

1	Architettura generale . . . . .	2
2	Architettura generale dell'intero sistema . . . . .	4
3	Rappresentazione Three Tier . . . . .	6
4	Rappresentazione MVP . . . . .	7
5	Client . . . . .	11
6	Server . . . . .	20
7	Diagramma della attività generale . . . . .	31
8	Diagramma delle attività di registrazione . . . . .	32
9	Diagramma delle attività di autenticazione . . . . .	33
10	Diagramma delle attività di modifica dei dati account . . . . .	34
11	Diagramma delle attività di una chiamata IP . . . . .	35
12	Diagramma delle attività dei servizi di un contatto . . . . .	36
13	Diagramma delle attività di gestione delle notifiche . . . . .	37
14	Pagina iniziale . . . . .	38
15	Pagina dopo login . . . . .	39
16	Schermata videochiamata . . . . .	40
17	Schermata chiamata audio . . . . .	41
18	Schermata comunicazione testuale . . . . .	42
19	Schermata Videoconferenza . . . . .	43

## Elenco delle tabelle

1	Tracciamento tra componenti e requisiti . . . . .	28
2	Tracciamento tra requisiti e componenti . . . . .	30



## 1 Introduzione

### 1.1 Scopo del documento

Il presente documento ha lo scopo di definire la progettazione ad alto livello che il prodotto dovrà avere. Verranno presentati i vari design pattern utilizzati nella creazione del prodotto, l'architettura generale secondo la quale saranno organizzate le varie componenti software e il tracciamento tra le componenti software ed i requisiti.

### 1.2 Scopo del prodotto

Il prodotto denominato **MyTalk** si propone di fornire un software per un sistema di comunicazione audio e video tra utenti. Lo scopo del progetto è poter comunicare con altri utenti tramite il browser, utilizzando solo componenti standard, senza dover installare plugin o programmi esterni. L'utilizzatore dovrà poter chiamare un altro utente, iniziare la comunicazione sia audio che video, svolgere la chiamata e terminare la chiamata ottenendo delle statistiche sull'attività.

### 1.3 Glossario

I termini tecnici o di uso non comune sono presenti nel documento allegato Glossario\_v2.0.pdf. Tali riferimenti vengono evidenziati tramite sottolineatura alla prima occorrenza del termine nel documento.

### 1.4 Riferimenti

#### 1.4.1 Normativi

- Norme di Progetto (allegato Norme\_di\_Progetto\_v3.0.pdf)

#### 1.4.2 Informativi

- Analisi dei Requisiti (allegato Analisi\_dei\_Requisiti\_v3.0.pdf)
- SWEBOK - Chapter 3: Software Design: <http://www.computer.org/protal/web/swebok/html/ch3>
- SWEBOK - Chapter 11: Software Quality: <http://www.computer.org/protal/web/swebok/html/ch11>
- Software Engineering - Chapter 11: Architectural design - Ian Sommerville - 8th ed. (2006)

## 2 Architettura Generale

L'applicativo è stato diviso in due sistemi, ovvero il client e il server, come indicato nella figura 1. L'indice di accoppiamento tra questi due sottosistemi sarà tenuto al minimo necessario.



**Figura 1:** Architettura generale

### 2.1 Client

Il client si occuperà di gestire l'interfaccia visibile all'utente le cui operazioni verranno poi inviate al server. Il lato client è stato suddiviso nel seguente modo:

- **CCLI1:** autenticazione
- **CCLI2:** gestione dati
- **CCLI3:** tutorial

- **CCLI4:** visualizzazione lista utenti
- **CCLI5:** gestione notifiche
- **CCLI6:** chiamata
- **CCLI7:** messaggio differito
- **CCLI8:** chat
- **CCLI9:** invio file

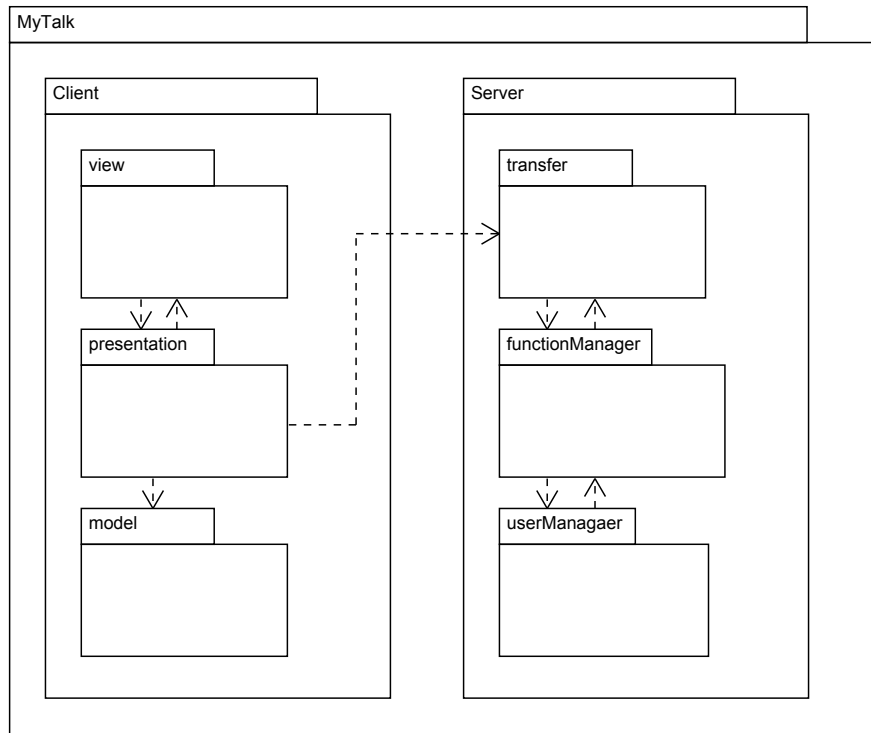
## 2.2 Server

Il server si occuperà di gestire le richieste fatte e di inviare una risposta al client. Il server è stato suddiviso in

- **CSER1:** autenticazione
- **CSER2:** gestione dati utenti
- **CSER3:** inizializzazione chiamata
- **CSER4:** salvataggio messaggio in differita
- **CSER5:** tutorial
- **CSER6:** notification

## 2.3 Architettura dell'intero sistema

Facendo riferimento alla figura 2 si può avere quindi una visione dell'architettura e delle relazioni tra gli strati:



**Figura 2:** Architettura generale dell'intero sistema

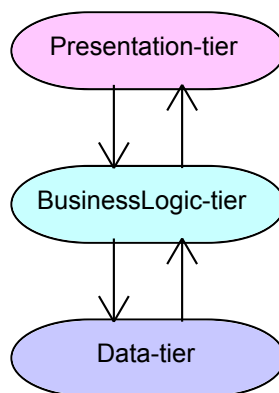
- **mytalk** incorpora l'intero sistema
- **mytalk.client** racchiude al suo interno il lato client di **MyTalk**
- **mytalk.client.view** racchiude al suo interno l'interfaccia grafica e la parte logica. Questo strato viene definito dal framework Backbone
- **mytalk.client.model** contiene dati locali
- **mytalk.client.communication** si occupa della gestione delle comunicazioni tra client e server
- **mytalk.server** racchiude al suo interno il lato server di **MyTalk**
- **mytalk.server.transfer** gestisce il collegamento tra server e client
- **mytalk.server.functionmanager** gestisce le operazioni di chiamata e dialoga con lo strato Data
- **mytalk.server.usermanager** gestisce le richieste da far arrivare alla base di dati e trasmette l'esito dei controlli al richiedente

- **mytalk.server.data** gestisce la base di dati, ovvero l'inserimento, la modifica e la rimozione dei dati e la loro permanenza

## 3 Design Pattern

Presentiamo qui di seguito i design pattern che andremo ad utilizzare per la rappresentazione dell'architettura del sistema.

### 3.1 Multitier



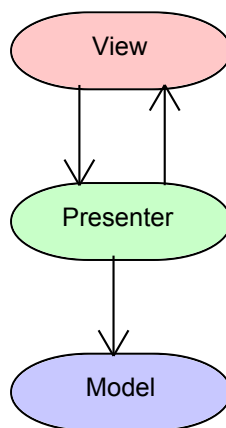
**Figura 3:** Rappresentazione Three Tier

Il design pattern di tipo Multitier verrà implementato nella forma Three Tier:

- **Descrizione:** tale design pattern permette una disgiunzione fra i vari gruppi di entità che cooperano per l'erogazione del servizio. Un primo livello si occuperà della comunicazione con il client, un secondo livello invece si occuperà di effettuare le operazioni logiche e di comunicare con il terzo strato che si occuperà di modificare la base di dati
- **Motivazione:** il beneficio principale di questo design pattern è la possibilità di aggiornare/cambiare un livello senza andare a modificare i livelli adiacenti. Nonostante i vari livelli si occupino di specifiche operazioni è presente una forte comunicazione rigidamente strutturata tra di loro, questo lo rende un ottimo modello per applicazioni client-server, poiché ogni livello non esiste come unità logica a se stante, ma si adegua allo specifico ambiente di rete in cui esegue
- **Contesto applicativo:** tale design pattern verrà utilizzato come struttura portante del server e sarà suddiviso come segue
  - Transfer Layer: offrirà un'interfaccia di comunicazione con il client e i livelli sottostanti del server

- Logic Layer: si occuperà di svolgere le funzionalità di comunicazione e di gestire la visualizzazione e la modifica delle informazioni presenti nello strato Data
- Data Layer: sarà un contenitore di informazioni riguardanti gli utenti e di messaggi registrati e collegamenti

### 3.2 MVP



**Figura 4:** Rappresentazione MVP

Il design pattern di tipo MVP verrà implementato per rappresentare l'architettura generale

- **Descrizione:** tale design pattern permette una completa disgiunzione tra le funzionalità che il prodotto deve offrire ed è strutturato in tre livelli
  - **Model:** rappresenta la parte in grado di recuperare le informazioni presenti nel sistema
  - **View:** rappresenta l'interfaccia grafica del sistema con la quale gli utenti possono interagire
  - **Presenter:** rappresenta la componente che riceve le richieste dalla view e le traduce in operazioni che agiranno sia sui dati che sui modelli; si occupa inoltre della comunicazione col server
- **Motivazione:** questo design pattern permette un buon livello di disaccoppiamento tra la vista ed il modello. L'architettura risulta facilmente manutenibile poiché le modifiche ad uno dei tre componenti non comporta modifiche ai restanti

### 3.3 MV\*

Il design pattern di tipo MV\* verrà implementato per rappresentare l'architettura del lato client

- **Descrizione:** tale design pattern deriva da MVC e MVP con la differenza che le funzioni di controller/presenter sono integrate nella view
  - **Model:** rappresenta le informazioni da cui necessita l'applicazione
  - **View:** contiene i metodi per visualizzare l'interfaccia grafica, composta dalle informazioni richieste, e per la creazione ed assegnazione di eventi
  - **Template:** sono frammenti di codice HTML che rappresentano il layout della view
- **Motivazione:** utilizzeremo questo pattern in quanto è quello utilizzato da Backbone.js che è il framework che andremo ad utilizzare per la realizzazione della parte client e perché permette lo sviluppo di applicazioni web dinamiche, strutturate e modulari

### 3.4 Observer

- **Descrizione:** tale design pattern si occupa di gestire i cambiamenti che avvengono in un oggetto riflettendone le conseguenze su tutti gli oggetti ad esso legato
- **Motivazione:** questo design pattern modifica delle componenti nel caso altre componenti ad esse legate vengano modificate
- **Contesto applicativo:** si rivela utile per quanto riguarda la manutenibilità della lista utenti connessi o meno o nel caso nuovi utenti si registrino



## 4 Strumenti Utilizzati

### 4.1 Java

L'utilizzo di Java è legato ai requisiti di capitolato; verrà utilizzato per l'avvio della comunicazione tra due utenti e per l'utilizzo del protocollo di comunicazione WebSocket. È stato deciso di utilizzare la versione 6.0

- **Vantaggi:**

- **Multipiattaforma:** grazie alla presenza della JVM si ha la sicurezza che il programma sarà eseguibile indipendentemente dal sistema operativo installato sulla macchina
- **Indipendenza delle risorse:** per lo stesso motivo sopra elencato l'utilizzo delle risorse fisiche sarà indipendente dal sistema operativo installato

- **Svantaggi:**

- Nessun svantaggio rilevato

### 4.2 HTML5

L'utilizzo di HTML5 è legato ai requisiti di capitolato e andrà a costituire insieme a CSS3 l'interfaccia web del prodotto

- **Vantaggi:**

- HTML5 supporta le ultime tecnologie riguardanti la creazione di applicazioni web
- Grafica più leggera e valida evitando l'utilizzo di tecnologia Flash

- **Svantaggi:**

- HTML5 e CSS3 non attualmente standard

### 4.3 JavaScript

Andremo ad utilizzare JavaScript per l'utilizzo di WebRTC e quindi per la comunicazione tra gli utenti

### 4.4 SQLite

Considerando la complessità relativamente limitata del database che occorrerà per la gestione degli utenti abbiamo deciso di utilizzare SQLite

- **Vantaggi:**

- Conoscenza del linguaggio SQL

- Gestibile attraverso una libreria di Java
- Non richiede installazione di un server
- Leggero e veloce

- **Svantaggi:**

- Alcune limitazioni rispetto alla versione MySql

## 4.5 Backbone.js

Per la gestione del lato client abbiamo deciso di utilizzare il framework Backbone.js, che risulta comodo ed efficiente per la gestione di applicazioni che utilizzino pesantemente il linguaggio JavaScript

- **Vantaggi:**

- Maggiore facilità nella gestione del lato client e nella sua programmazione
  - Backbone.js è un progetto open source
  - Dover seguire la struttura data dal framework

- **Svantaggi:**

- Non supporta l'architettura MVP

## 5 Client

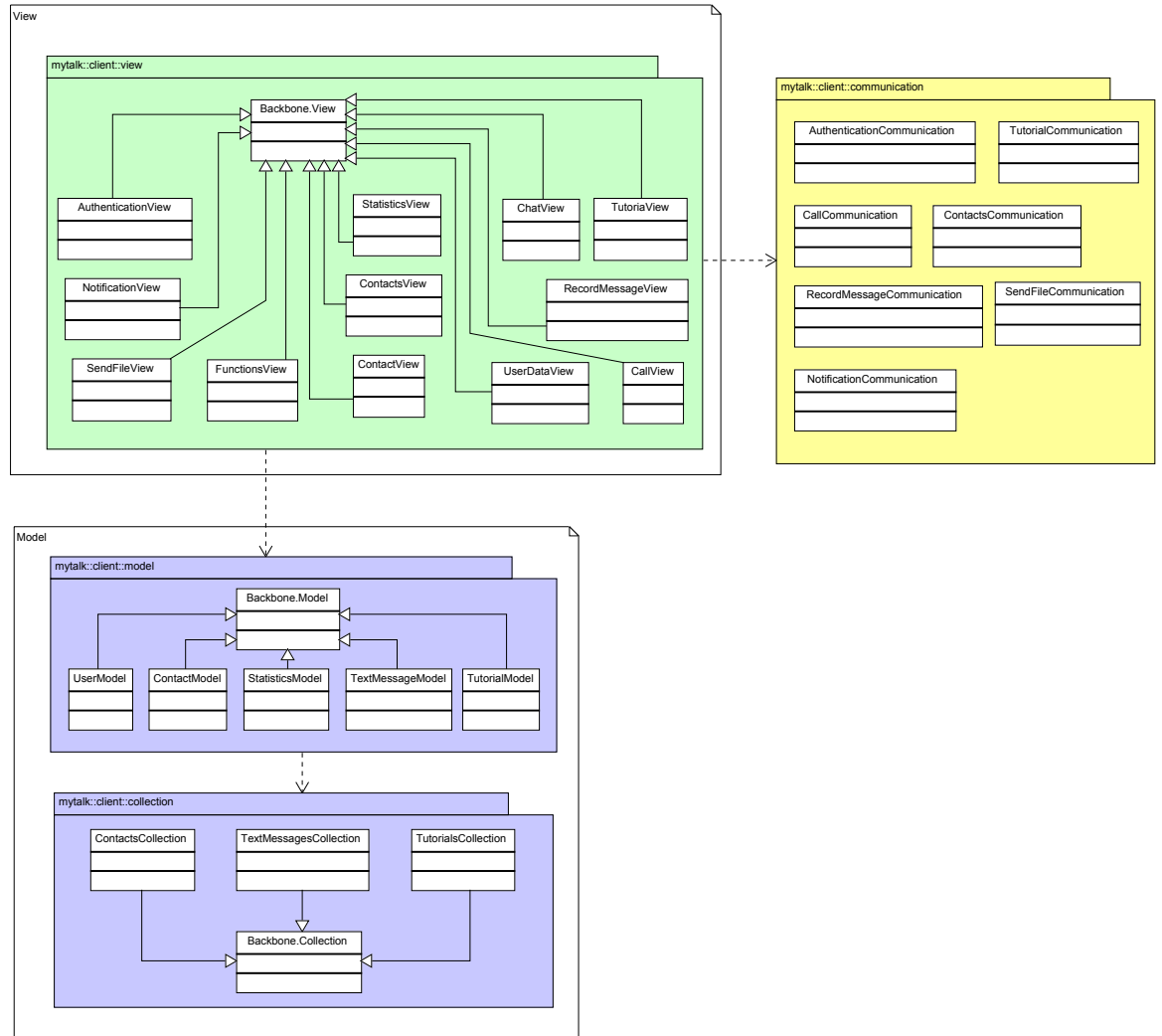


Figura 5: Client

### 5.1 View

- **Tipo, obiettivo e funzione del componente:** costituisce la parte del sistema che definisce ed implementa l'interfaccia web usufruibile dagli utenti mediante pagine web. Diversamente da uno schema di tipo MVC il package `view` contiene non solo le parti di interfaccia ma anche parti lo-

giche, come previsto dal framework Backbone.js, che prevede uno schema di tipo MV\*

- **Relazioni d'uso con altre componenti:** il componente è costituito dal package `view`, e andrà a comunicare con il package `mytalk.client.communication` che farà da tramite tra il client ed il server

#### 5.1.1 mytalk.client.view.AuthenticationView

- **Tipo, obiettivo e funzione del componente:** la classe `AuthenticationView` definisce la struttura, e la conseguente visualizzazione, della pagina web che consente di effettuare l'accesso al e la disconnessione dal sistema e la registrazione di un utente al server
- **Relazioni d'uso con altre componenti:** la classe `AuthenticationView` comunicherà con la classe `mytalk.client.communication.AuthenticationCommunication` per poter verificare se le credenziali inserite siano corrette o meno e quindi effettuare l'accesso al sistema o mostrare i relativi messaggi di errore, e per inviare i dati per effettuare la registrazione dell'utente al server tenendo in considerazione l'univocità dello username. Infine gestirà il passaggio delle credenziali volte a modificare i dati dell'utente autenticato
- **Attività svolte e dati trattati:** la classe fa in modo che vengano gestiti gli eventi per l'autenticazione dell'utente al server, per la registrazione e per la modifica dei dati tramite la classe `mytalk.client.communication.AuthenticationCommunication`

#### 5.1.2 mytalk.client.view.TutorialView

- **Tipo, obiettivo e funzione del componente:** la classe `TutorialView` definisce la struttura, e la conseguente visualizzazione, delle pagine web che consentono la visualizzazione dei video tutorial del prodotto **MyTalk**
- **Relazioni d'uso con altre componenti:** la classe `TutorialView` comunicherà con la classe `mytalk.client.communication.TutorialCommunication` al fine di visualizzare il video scelto dall'utente
- **Attività svolte e dati trattati:** la classe fa in modo che venga caricato, e successivamente visualizzato, il video tutorial selezionato

#### 5.1.3 mytalk.client.view.ContactsView

- **Tipo, obiettivo e funzione del componente:** la classe `ContactsView` definisce la struttura, e la conseguente visualizzazione, delle pagine web che consentono la visualizzazione della lista utenti iscritti al server. È composta da più `ContactView`
- **Relazioni d'uso con altre componenti:** la classe comunicherà con:

- `mytalk.client.communication.ContactsCommunication` che restituirà gli iscritti al server connessi in quel momento o meno

- **Attività svolte e dati trattati:** la classe fa in modo che venga visualizzata una lista di utenti registrati presso il server

#### 5.1.4 `mytalk.client.view.ContactView`

- **Tipo, obiettivo e funzione del componente:** la classe `ContactsView` definisce la struttura, e la conseguente visualizzazione, di un singolo utente iscritto al server; più `ContactView` insieme compongono `ContactsView`

- **Relazioni d'uso con altre componenti:** la classe comunicherà con:

- `mytalk.client.view.FunctionsView` poiché premendo sul singolo contatto verrà mostrata l'interfaccia per effettuare le operazioni di comunicazione

- **Attività svolte e dati trattati:** la classe rappresenta un singolo utente registrato sul server e fornisce all'utente le operazioni che, una volta che lo si ha selezionato, mostrano le funzionalità di comunicazione disponibili

#### 5.1.5 `mytalk.client.view.FunctionsView`

- **Tipo, obiettivo e funzione del componente:** la classe `FunctionsView` definisce la struttura, e la conseguente visualizzazione, delle pagine web che consentono la selezione delle operazioni da effettuare, quali inizializzazione di una chiamata, invio di messaggio di testo, invio del file o attivazione della funzionalità di registrazione chiamata

- **Relazioni d'uso con altre componenti:** la classe comunicherà con :

- `mytalk.client.communication.CallCommunication` per permettere la comunicazione audio e video tra due o più utenti
- `mytalk.client.communication.ChatCommunication` per permettere l'invio di messaggi testuali tra due utenti

e altre classi del package `communication` per altre funzionalità

- **Attività svolte e dati trattati:** la classe fa in modo che vengano visualizzate le componenti grafiche che permettono la scelta delle funzionalità di comunicazione disponibili

#### 5.1.6 `mytalk.client.view.CallView`

- **Tipo, obiettivo e funzione del componente:** la classe `CallView` definisce la struttura, e la conseguente visualizzazione, delle pagine web che consentono di effettuare chiamate

- **Relazioni d'uso con altre componenti:** la classe `CallView` comunicherà con la classe `mytalk.client.communication.CallCommunication` per permettere la gestione della comunicazione tra due o più utenti
- **Attività svolte e dati trattati:** la classe è costituita dalle componenti grafiche e i metodi che permettono la gestione delle funzionalità legate alla comunicazione audio/video

#### 5.1.7 `mytalk.client.view.NotificationView`

- **Tipo, obiettivo e funzione del componente:** la classe `NotificationView` definisce la struttura, e la conseguente visualizzazione, delle pagine web che consentono di visualizzare notifiche di chiamate in entrata
- **Relazioni d'uso con altre componenti:** la classe `NotificationView` verrà utilizzata dalla classe `mytalk.client.communication.NotificationCommunication` che invierà notifiche di richiesta di comunicazione
- **Attività svolte e dati trattati:** la classe è costituita dalle componenti grafiche per la visualizzazione delle notifiche e dai metodi per la gestione di quest'ultime

#### 5.1.8 `mytalk.client.view.ChatView`

- **Tipo, obiettivo e funzione del componente:** la classe `ChatView` definisce la struttura, e la conseguente visualizzazione, delle pagine web che consentono di effettuare comunicazioni testuali
- **Relazioni d'uso con altre componenti:** la classe `ChatView` comunicherà con la classe `mytalk.client.communication.ChatCommunication` per permettere la comunicazione testuale tra due utenti
- **Attività svolte e dati trattati:** la classe è costituita dalle componenti grafiche per la visualizzazione e dai metodi per la gestione della chat testuale

#### 5.1.9 `mytalk.client.view.RecordMessageView`

- **Tipo, obiettivo e funzione del componente:** la classe `RecordMessageView` definisce la struttura, e la conseguente visualizzazione, delle pagine web per effettuare registrazioni audio o audio/video da inviare ad un utente registrato
- **Relazioni d'uso con altre componenti:** la classe `RecordMessageView` comunicherà con la classe `mytalk.client.communication.RecordMessageCommunication` che con i suoi metodi permetterà la registrazione di un video da inviare ad un utente registrato

- **Attività svolte e dati trattati:** la classe definisce la struttura della pagina web che consente la registrazione di chiamate video da inviare verso utenti registrati

#### 5.1.10 mytalk.client.view.StatisticsView

- **Tipo, obiettivo e funzione del componente:** la classe `StatisticsView` definisce la struttura, e la conseguente visualizzazione, delle pagine web per la visualizzazione di statistiche durante lo svolgimento di una chiamata
- **Relazioni d'uso con altre componenti:** la classe `StatisticsView` comunicherà con la classe `mytalk.client.model.Statistics` che avrà lo scopo di visualizzare le statistiche della chiamata
- **Attività svolte e dati trattati:** la classe definisce la struttura della pagina web che visualizza i dati della chiamata che viene effettuata

#### 5.1.11 mytalk.client.view.ModifyAccountDataView

- **Tipo, obiettivo e funzione del componente:** la classe `ModifyAccountDataView` definisce la struttura, e la conseguente visualizzazione, delle pagine web per la visualizzazione e la modifica dei dati del proprio account
- **Relazioni d'uso con altre componenti:** la classe `ModifyAccountDataView` comunicherà con la classe `mytalk.client.model.Contact` per visualizzare i dati del proprio account e con la classe `mytalk.client.communication.AuthenticationCommunication` che avrà lo scopo di inviare al server le modifiche dei dati dell'utente. Tramite il design pattern `Observer` si aggiornerà il database locale
- **Attività svolte e dati trattati:** la classe definisce la struttura della pagina web per la modifica dei dati personali dell'utente

#### 5.1.12 mytalk.client.view.SendFileView

- **Tipo, obiettivo e funzione del componente:** la classe `SendFileView` definisce la struttura, e la conseguente visualizzazione, delle pagine web che consentono di selezionare un file salvato in locale per mandarlo all'utente autenticato selezionato
- **Relazioni d'uso con altre componenti:** la classe `SendFileView` comunicherà con la classe `mytalk.client.communication.SendFileCommunication` che si occuperà dell'invio dei file al server
- **Attività svolte e dati trattati:** la classe fa in modo che vengano visualizzati i componenti grafici necessari per le operazioni di selezione di un file salvato in locale e di scelta del destinatario

## 5.2 Communication

- **Tipo, obiettivo e funzione del componente:** definisce ed implementa la parte del sistema che si occupa della comunicazione tra client e server
- **Relazioni d'uso con altre componenti:** il componente è costituito dal package `communication`, riceverà richieste dal package `mytalk.client.view` ed andrà a comunicare con il package `mytalk.server.transfer` per ottenere da esso ed eventualmente restituire al package `mytalk.client.view` i dati richiesti presenti all'interno del server

### 5.2.1 `mytalk.client.communication.AuthenticationCommunication`

- **Tipo, obiettivo e funzione del componente:** la classe `AuthenticationCommunication` si occupa di inviare le credenziali inserite al server e inviare la risposta ricevuta a `mytalk.client.view.AuthenticationView`. Inoltre gestisce le operazioni di registrazione di un nuovo utente e di modifica dei dati, comunicando con `mytalk.client.view.ModifyAccountDataView`
- **Relazioni d'uso con altre componenti:** la classe riceverà i dati inseriti da `mytalk.client.view.AuthenticationView` e fornirà i metodi per stabilire una connessione col server dal quale riceverà una risposta che andrà trasmessa alla classe `mytalk.client.view.AuthenticationView`
- **Attività svolte e dati trattati:** la classe si occupa della comunicazione tra client e server per quanto riguarda qualsiasi controllo o modifica dei dati dell'utente

### 5.2.2 `mytalk.client.communication.ContactsCommunication`

- **Tipo, obiettivo e funzione del componente:** la classe `ContactsCommunication` si occupa di ricevere la lista degli utenti presenti nel server
- **Relazioni d'uso con altre componenti:** la classe `ContactsCommunication` fornirà i metodi per aggiornare i dati degli altri utenti registrati al server presenti nella classe `mytalk.client.model.Contact`
- **Attività svolte e dati trattati:** la classe si occupa di restituire la lista di tutti gli utenti iscritti al server

### 5.2.3 `mytalk.client.communication.CallCommunication`

- **Tipo, obiettivo e funzione del componente:** la classe `CallCommunication` si occupa di avviare la comunicazione tra utenti con il server
- **Relazioni d'uso con altre componenti:** la classe `CallCommunication` fornirà i metodi per l'avvio di una chiamata tra due peer
- **Attività svolte e dati trattati:** la classe si occupa di avviare una chiamata tra due diversi utenti



#### 5.2.4 mytalk.client.communication.NotificationCommunication

- **Tipo, obiettivo e funzione del componente:** la classe `NotificationCommunication` si occupa di comunicare con il server per quanto riguarda le notifiche
- **Relazioni d'uso con altre componenti:** la classe `NotificationCommunication` fornirà i metodi per segnalare la presenza di una chiamata in arrivo da parte di un altro utente, oppure la presenza di file o messaggi registrati all'interno del server, alla classe `mytalk.client.view.NotificationView`
- **Attività svolte e dati trattati:** la classe si occupa quindi di notificare l'utente di chiamate in arrivo o della presenza di file e/o o messaggi inviati da altri utenti in attesa

#### 5.2.5 mytalk.client.communication.RecordMessageCommunication

- **Tipo, obiettivo e funzione del componente:** la classe `RecordMessageCommunication` si occupa di inviare e ricevere i messaggi registrati
- **Relazioni d'uso con altre componenti:** la classe `RecordMessageCommunication` fornirà alla classe `mytalk.client.view.RecordMessageView` i metodi per l'invio e la ricezione di messaggi registrati
- **Attività svolte e dati trattati:** la classe si occupa di ricevere la richiesta di avvio registrazione e di mandare al server il compito di salvarla

#### 5.2.6 mytalk.client.communication.TutorialCommunication

- **Tipo, obiettivo e funzione del componente:** la classe `TutorialCommunication` si occupa di ricevere la locazione web dei videotutorial da caricare
- **Relazioni d'uso con altre componenti:** la classe `TutorialCommunication` fornirà i metodi a `mytalk.client.view.TutorialView` per la comunicazione col server a cui richiedere l'indirizzo del video tutorial richiesto
- **Attività svolte e dati trattati:** la classe fornirà quindi a `mytalk.client.view.TutorialView` il videotutorial richiesto

#### 5.2.7 mytalk.client.communication.SendFileCommunication

- **Tipo, obiettivo e funzione del componente:** la classe `SendFileCommunication` si occupa di gestire l'invio di file al server
- **Relazioni d'uso con altre componenti:** la classe `SendFileCommunication` fornisce i metodi alla classe `mytalk.client.view.SendFileView` per l'invio dei file al server
- **Attività svolte e dati trattati:** la classe si occupa quindi di effettuare il trasferimento tra client di un file

## 5.3 Model

- **Tipo, obiettivo e funzione del componente:** ha lo scopo di rappresentare localmente le informazioni presenti nel database del server al fine di popolare le viste in modo dinamico
- **Relazioni d'uso con altre componenti:** viene utilizzato dagli strati `view` e `communication` che invieranno le richieste di gestione dei dati

### 5.3.1 `mytalk.client.model.User`

- **Tipo, obiettivo e funzione del componente:** la classe `User` si occuperà di tenere aggiornate localmente le informazioni dell'account attuale
- **Relazioni d'uso con altre componenti:** la classe `User` riceverà da `mytalk.client.communication.AuthenticationCommunication` le informazioni riguardanti l'account dell'utente autenticato
- **Attività svolte e dati trattati:** la classe si occupa di tenere le informazioni del proprio account restituendo i dati all'utente

### 5.3.2 `mytalk.client.model.Contact`

- **Tipo, obiettivo e funzione del componente:** la classe `Contact` si occuperà di gestire la lista utenti
- **Relazioni d'uso con altre componenti:** la classe `Contact` riceverà da `mytalk.client.view.ContactsView` la lista degli utenti che sono presenti nella lista e il loro stato attuale una volta effettuato l'accesso
- **Attività svolte e dati trattati:** la classe si occupa di ricevere la lista utenti registrati, che include se siano connessi o meno, e l'`Observer` provvederà ad aggiornare la `view`

### 5.3.3 `mytalk.client.model.Statistics`

- **Tipo, obiettivo e funzione del componente:** la classe `Statistics` si occuperà di gestire le statistiche di chiamata
- **Relazioni d'uso con altre componenti:** la classe `Statistics` riceverà la richiesta da `mytalk.client.view.StatisticsView` per l'aggiornamento delle statistiche della chiamata
- **Attività svolte e dati trattati:** la classe si occupa di conservare le informazioni sulla chiamata

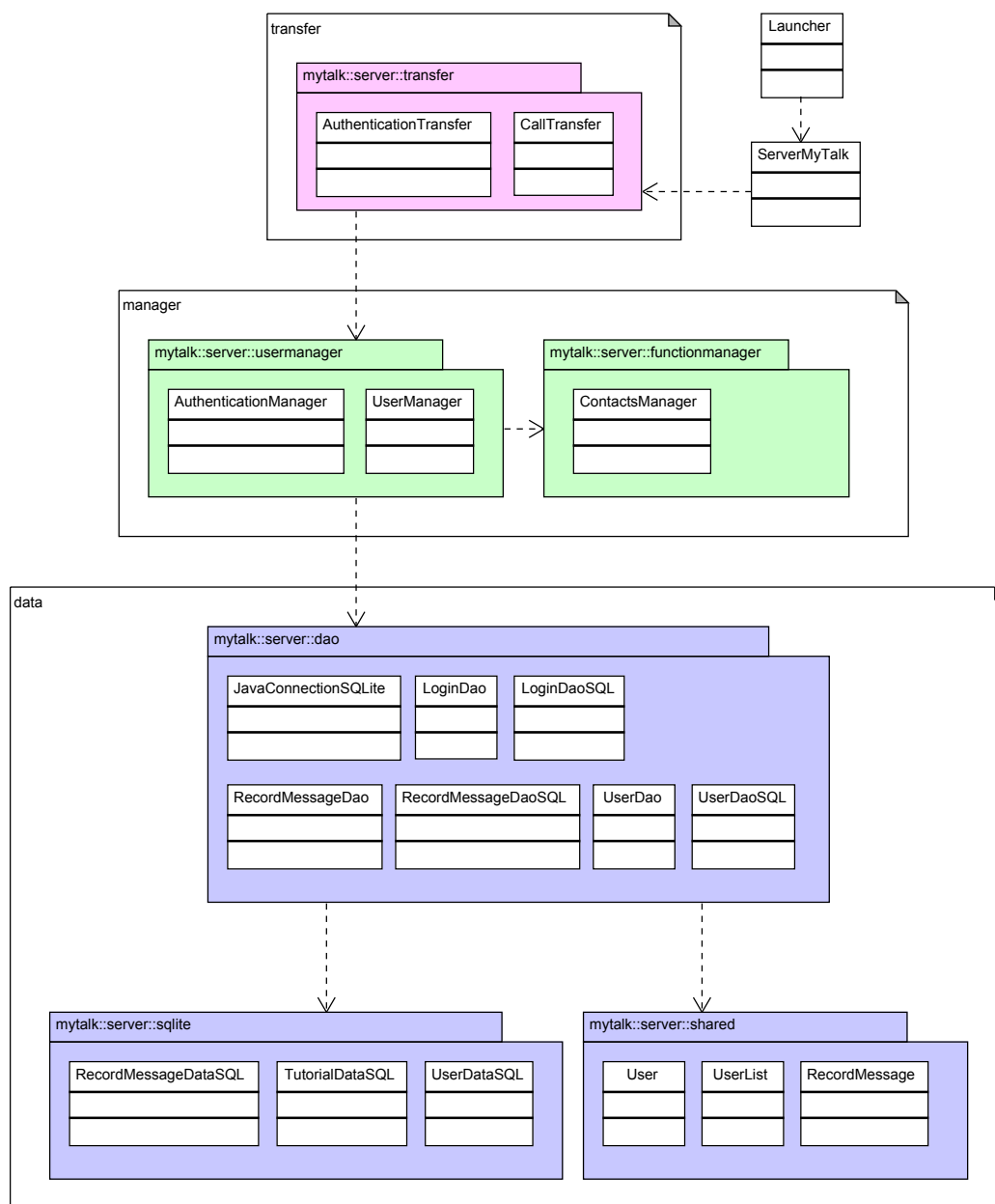
#### 5.3.4 mytalk.client.model.TextMessage

- **Tipo, obiettivo e funzione del componente:** la classe `TextMessage` si occupa della comunicazione testuale
- **Relazioni d'uso con altre componenti:** la classe `TextMessage` riceverà da `mytalk.client.communication.ChatCommunication` i messaggi di testo inviati da e verso altri utenti e li gestirà
- **Attività svolte e dati trattati:** la classe si occuperà di conservare per un tempo limitato le comunicazioni testuali tra utenti

#### 5.3.5 mytalk.client.model.Tutorial

- **Tipo, obiettivo e funzione del componente:** la classe `Tutorial` si occuperà di conservare i collegamenti ai tutorial
- **Relazioni d'uso con altre componenti:** la classe `Tutorial` riceverà richieste da parte di `mytalk.client.view.TutorialView` per il caricamento di un video tutorial riguardante le funzionalità del prodotto **MyTalk** restituendone l'indirizzo del video
- **Attività svolte e dati trattati:** la classe si occuperà di conservare i collegamenti da dove prelevare i video tutorial

## 6 Server



**Figura 6:** Server

## 6.1 Transfer Layer

- **Tipo, obiettivo e funzione del componente:** lo strato Transfer si occupa della comunicazione tra il client e il server. La struttura a classi non si addice a quella che è la struttura di un prodotto web-based, quindi le classi qui rappresentate vanno considerate come enti concettuali
- **Relazioni d'uso con altre componenti:** lo strato Transfer interagisce con il package `mytalk.server.usermanager` e con lo strato `mytalk.client.communication` allo scopo di interfacciare il server col client

### 6.1.1 `mytalk.server.transfer.CallTransfer`

- **Tipo, obiettivo e funzione del componente:** la classe `CallTransfer` si occupa di trasferire i pacchetti tra i client che desiderano iniziare una comunicazione al fine di creare il canale WebRTC necessario
- **Relazioni d'uso con altre componenti:** la classe `CallTransfer` viene contattata da e si rivolge a `mytalk.client.communication.CallCommunication`
- **Attività svolte e dati trattati:** la classe `CallTransfer` si occupa di gestire il trasferimento dei pacchetti necessari per l'avvio di una chiamata tramite Listener

### 6.1.2 `mytalk.server.transfer.AuthenticationTransfer`

- **Tipo, obiettivo e funzione del componente:** la classe `AuthenticationTransfer` si occupa del trasferimento delle richieste di autenticazione e registrazione al server
- **Relazioni d'uso con altre componenti:** la classe `AuthenticationTransfer` viene utilizzata da `mytalk.client.communication.AuthenticationCommunication`, `mytalk.server.usermanager.AuthenticationManager` e `mytalk.server.usermanager.UserManager`
- **Attività svolte e dati trattati:** la classe `AuthenticationTransfer` ha lo scopo di ricevere da `mytalk.client.communication.AuthenticationCommunication` richieste di autenticazione e manda i dati a `mytalk.server.usermanager.AuthenticationManager` che si occuperà di verificarli e dare una risposta che verrà quindi inviata al client. Inoltre gestirà il passaggio di richieste di modifica dei dati degli utenti comunicando con `mytalk.server.usermanager.UserManager` che contiene i metodi necessari per la registrazione degli utenti

### 6.1.3 `mytalk.server.transfer.TutorialTransfer`

- **Tipo, obiettivo e funzione del componente:** la classe `TutorialTransfer` classe si occupa del passaggio delle richieste di visualizzazione dei video tutorial dal client al server

- **Relazioni d'uso con altre componenti:** la classe `TutorialTransfer` viene chiamata da `mytalk.client.communication.TutorialCommunication` e preleva da `mytalk.server.shared.Tutorial`
- **Attività svolte e dati trattati:** la classe ha una funzione di collegamento, si occupa di restituire a `mytalk.client.communication.TutorialCommunication` il collegamento richiesto

#### 6.1.4 `mytalk.server.transfer.RecordMessageTransfer`

- **Tipo, obiettivo e funzione del componente:** la classe `RecordMessageTransfer` si occupa del trasferimento del messaggio registrato dal client al server e del seguente invio al destinatario
- **Relazioni d'uso con altre componenti:** la classe `RecordMessageTransfer` comunica con `mytalk.server.usermanager.UserManager` e `mytalk.client.communication.RecordMessageCommunication`
- **Attività svolte e dati trattati:** la classe si occupa quindi di ricevere dal client il messaggio registrato e l'utente a cui mandarlo e passa tali dati a `mytalk.server.usermanager.UserManager`. In seguito una volta ricevuto il video da `mytalk.server.usermanager.UserManager` lo manda al destinatario

### 6.2 Manager Layer

- **Tipo, obiettivo e funzione del componente:** lo strato Manager del server si occupa delle funzionalità logiche e di comunicazione con la base di dati
- **Relazioni d'uso con altre componenti:** lo strato Manager comunica con lo strato Transfer e invia allo strato Data richieste riguardanti i dati contenuti nella base di dati

#### 6.2.1 `mytalk.server.usermanager.AuthenticationManager`

- **Tipo, obiettivo e funzione del componente:** la classe `AuthenticationManager` si occupa di stabilire la riuscita o il fallimento di un tentativo di autenticazione e di gestire la registrazione o l'eventuale cancellazione di un utente
- **Relazioni d'uso con altre componenti:** la classe `AuthenticationManager` utilizza `mytalk.server.dao.UserDataDao` e comunica con `mytalk.server.transfer.AuthenticationTransfer`
- **Attività svolte e dati trattati:** la classe si occuperà quindi di mandare alla base di dati le credenziali inserite per verificare se un tentativo di autenticazione sia riuscito o meno, e restituire a `mytalk.server.transfer.AuthenticationTransfer`

l'opportuna risposta, e per effettuare il logout. Inoltre provvederà a comunicare alla base di dati un tentativo di registrazione di un nuovo utente, che avverrà solo se lo username scelto non sarà già presente nella base di dati

### 6.2.2 mytalk.server.usermanager.UserManager

- **Tipo, obiettivo e funzione del componente:** la classe `userManager` si occupa delle operazioni di aggiornamento delle informazioni sugli utenti, ovvero delle modifiche degli account, e della gestione lato server dei messaggi, sia di testo che audio/video
- **Relazioni d'uso con altre componenti:** la classe `userManager` viene utilizzata da `mytalk.server.transfer.AuthenticationTransfer` e agisce su `mytalk.server.dao.UserDataDao`. Inoltre interagisce con `mytalk.server.transfer.RecordMessageTransfer` e `mytalk.server.dao.RecordMessageDao`
- **Attività svolte e dati trattati:** la classe si occuperà di ricevere le istruzioni da `mytalk.server.transfer.AuthenticationTransfer`, controllare ed eventualmente modificare la base di dati; infine restituirà a `mytalk.server.transfer.AuthenticationTransfer` il successo o meno dell'eventuale cambiamento. Inoltre riceverà da `mytalk.server.transfer.RecordMessageTransfer` un messaggio registrato da mandare in differita e lo invierà alla base di dati per salvarlo. Quando il destinatario si connetterà dovrà prelevare dalla base di dati e inviarlo

### 6.3 Data Layer

- **Tipo, obiettivo e funzione del componente:** lo strato Data rappresenta la base di dati su cui vengono salvate le informazioni persistenti. Questo include i dati degli utenti registrati, i link ai tutorial video che saranno salvati su un sito esterno e i messaggi registrati
- **Relazioni d'uso con altre componenti:** lo strato Data verrà utilizzato da `mytalk.server.usermanager.UserManager` e `mytalk.server.usermanager.AuthenticationManager`

#### 6.3.1 mytalk.server.dao.UserDataDao

- **Tipo, obiettivo e funzione del componente:** la classe `UserDataDao` rappresenta l'interfaccia con cui comunicherà il package `mytalk.server.usermanager`
- **Relazioni d'uso con altre componenti:** la classe `UserDataDao` viene utilizzata da `mytalk.server.usermanager`; essendo un'interfaccia viene implementata da `UserDataDaoSQL` che agirà sulla base di dati
- **Attività svolte e dati trattati:** la classe fornisce i metodi per il salvataggio e la modifica dei dati presenti nella base di dati

### 6.3.2 mytalk.server.dao.UserDataDaoSQL

- **Tipo, obiettivo e funzione del componente:** la classe `UserDataDaoSQL` è l'implementazione di `UserDataDao` per la base di dati SQLite
- **Relazioni d'uso con altre componenti:** la classe `UserDataDaoSQL` implementa `UserDataDao` e agisce su `UserData`
- **Attività svolte e dati trattati:** la classe `UserDataDaoSQL` si occupa della lettura e scrittura dei dati nella base di dati

### 6.3.3 mytalk.server.sqlite.UserData

- **Tipo, obiettivo e funzione del componente:** la classe `UserData` è un semplice contenitore di informazioni
- **Relazioni d'uso con altre componenti:** la classe `UserData` viene utilizzata da `mytalk.server.dao.UserDataDaoSQL`; essendo un semplice contenitore di informazioni la classe non utilizza componenti
- **Attività svolte e dati trattati:** la classe preserva le informazioni riguardanti gli utenti, ovvero username, password, nome, cognome e indirizzo IP

### 6.3.4 mytalk.server.shared.User

- **Tipo, obiettivo e funzione del componente:** la classe `User` è la copia nel package `shared` delle informazioni su un utente contenute nel database
- **Relazioni d'uso con altre componenti:** la classe `User` viene scritta da `mytalk.server.dao.UserDataDaoSQL`; essendo un semplice contenitore di informazioni la classe non utilizza componenti
- **Attività svolte e dati trattati:** la classe preserva le informazioni riguardanti gli utenti, ovvero username, password, nome, cognome e indirizzo IP

### 6.3.5 mytalk.server.shared.UserList

- **Tipo, obiettivo e funzione del componente:** la classe `UserList` è un vector contenente la lista degli utenti registrati sul server e il loro IP
- **Relazioni d'uso con altre componenti:** la classe `UserList` viene scritta da `mytalk.server.dao.UserDataDaoSQL` e acceduta da `mytalk.server.usermanager.UserManager`
- **Attività svolte e dati trattati:** la classe preserva una lista con tutti gli utenti registrati, per evitare situazioni di incoerenza nei dati salvati nei singoli client; tale lista viene infatti mandata in broadcast ad ogni modifica



### 6.3.6 mytalk.server.dao.RecordMessageDataDao

- **Tipo, obiettivo e funzione del componente:** la classe `RecordMessageDataDao` rappresenta l'interfaccia con cui comunicherà `mytalk.server.usermanager.UserManager`
- **Relazioni d'uso con altre componenti:** la classe `RecordMessageDataDao` viene utilizzata dalla classe `mytalk.server.usermanager.UserManager`
- **Attività svolte e dati trattati:** la classe fornisce i metodi per il prelievo e la scrittura dei messaggi registrati

### 6.3.7 mytalk.server.dao.RecordMessageDataDaoSQL

- **Tipo, obiettivo e funzione del componente:** la classe `RecordMessageDataDaoSQL` è l'implementazione di `RecordMessageDataDao` per la base di dati SQLite
- **Relazioni d'uso con altre componenti:** la classe `RecordMessageDataDaoSQL` implementa `RecordMessageDataDao` e agisce su `RecordMessageData`
- **Attività svolte e dati trattati:** la classe `RecordMessageDataDaoSQL` si occupa del salvataggio e del prelievo dei messaggi registrati presenti nella base di dati

### 6.3.8 mytalk.server.sqlite.RecordMessageData

- **Tipo, obiettivo e funzione del componente:** la classe `RecordMessageData` è un semplice contenitore di informazioni
- **Relazioni d'uso con altre componenti:** la classe `RecordMessageData` viene utilizzata dalla classe `mytalk.server.dao.RecordMessageDaoSQL`; essendo un semplice contenitore di informazioni la classe non utilizza componenti
- **Attività svolte e dati trattati:** la classe preserva i messaggi registrati da mandare in differita

### 6.3.9 mytalk.server.shared.RecordMessage

- **Tipo, obiettivo e funzione del componente:** la classe `RecordMessageData` è un vector di messaggi registrati
- **Relazioni d'uso con altre componenti:** la classe `RecordMessage` viene utilizzata dalla classe `mytalk.server.dao.RecordMessageDaoSQL` che vi salva i messaggi registrati e da `mytalk.server.usermanager.UserManager` che preleva la lista di messaggi registrati
- **Attività svolte e dati trattati:** la classe contiene una lista con l'associazione tra un user e i messaggi a lui indirizzati

#### 6.3.10 mytalk.server.dao.TutorialDataDao

- **Tipo, obiettivo e funzione del componente:** la classe `TutorialDataDao` rappresenta l'interfaccia per agire sulla lista dei tutorial
- **Relazioni d'uso con altre componenti:** la classe `TutorialDataDao` viene implementata da `TutorialDataDaoSQL`
- **Attività svolte e dati trattati:** la classe `TutorialDataDao` è l'interfaccia che verrà implementata per agire sulla base di dati

#### 6.3.11 mytalk.server.dao.TutorialDataDaoSQL

- **Tipo, obiettivo e funzione del componente:** la classe `TutorialDataDaoSQL` è l'implementazione di `TutorialDataDao` per la base di dati SQLite
- **Relazioni d'uso con altre componenti:** la classe `TutorialDataDaoSQL` implementa `TutorialDataDao` e agisce su `TutorialData` e `TutorialList`
- **Attività svolte e dati trattati:** la classe `TutorialDataDaoSQL` si occupa della scrittura dei collegamenti ai tutorial video e della preparazione della lista di video tutorial da inserire in `mytalk.server.shared.TutorialList`

#### 6.3.12 mytalk.server.sqlite.TutorialData

- **Tipo, obiettivo e funzione del componente:** la classe `TutorialData` è un semplice contenitore di informazioni
- **Relazioni d'uso con altre componenti:** la classe `TutorialData` è utilizzata da `mytalk.server.dao.TutorialDaoSQL`; essendo un semplice contenitore di informazioni la classe non utilizza componenti
- **Attività svolte e dati trattati:** la classe `TutorialData` preserva i collegamenti ai tutorial video

#### 6.3.13 mytalk.server.shared.TutorialList

- **Tipo, obiettivo e funzione del componente:** la classe `TutorialList` è un contenitore di informazioni accessibile da `mytalk.server.transfer.TutorialTransfer`
- **Relazioni d'uso con altre componenti:** la classe `TutorialList` viene costruita da `mytalk.server.dao.TutorialDaoSQL` e acceduta da `mytalk.server.transfer.TutorialTransfer`; essendo un semplice contenitore di informazioni la classe non utilizza componenti
- **Attività svolte e dati trattati:** la classe `TutorialList` preserva la lista di collegamenti ai tutorial video

## 7 Tracciamento componenti-requisiti

Componente	Classi	Requisiti
CCLI1	mytalk.client.view.AuthenticationView	RUFO 2 RUFO 2.1 RUFO 2.2
CCLI2	mytalk.client.view.ModifyAccountDataView mytalk.client.view.AuthenticationView mytalk.client.view.ModifyAccountDataView  mytalk.client.model.User	RUFO 1 RUFO 1.1 RUFO 1.2 RUFO 1.3 RUFO 1.4 RUFO 1.5 RUFF 3 RUFF 3.1 RUFF 3.2 RUFF 3.3 RUFF 3.4 RUFF 3.5
CCLI3	mytalk.client.view.TutorialView  mytalk.client.model.Tutorial	RUFF 4 RUFF 4.1
CCLI4	mytalk.client.view.ContactsView  mytalk.client.model.Contacts	RUFO 5 RUFF 5.1 RUFO 6.3
CCLI5	mytalk.client.view.NotificationView mytalk.client.communication.NotificationCommunication	RUFF 7 RUFF 7.1 RUFF 7.2 RUFF 7.3 RUFF 7.4 RUFF 7.7 RUFF 7.8 RUFF 7.9
CCLI6	mytalk.client.view.CallView  mytalk.client.view.StatisticsView mytalk.client.communication.CallCommunication mytalk.client.view.StatisticsView mytalk.client.model.Statistics	RUFO 6 RUFO 6.1 RUFO 6.2 RUFO 6.4 RUFO 6.5 RUFF 6.6 RUFF 6.7 RUF 6.8 RUF 6.9 RUF 6.13 RUFO 6.15
CCLI7	mytalk.client.view.RecordMessageView	RUFF 6.10 RUFF 6.11

		RUFF 7.5 RUFF 7.6
CCLI8	mytalk.client.view.ChatView mytalk.client.communication.ChatCommunication mytalk.client.model.TextMessage	RUFF 6.12
CCLI9	mytalk.client.view.SendFileView mytalk.client.communication.SendFileCommunication	RUF 6.14 RUF 6.14.1
CSER1	mytalk.server.transfer.AuthenticationTransfer mytalk.server.usermanager.AuthenticationManager	RUFO2
CSER2	mytalk.server.transfer.AuthenticationTransfer mytalk.server.usermanager.UserManager mytalk.server.data.UserData	RUFO 1 RUFO 1.1 RUFF 3 RUFF 3.1 RUFF 3.2 RUFF 3.3 RUFO 5 RUFF 5.1
CSER3	mytalk.server.transfer.CallTransfer mytalk.server.transfer.CallTransfer	RUFO 6 RUFO 6.1 RUFO 6.2 RUFO 6.3 RUFO 6.4 RUFO 6.5 RUFF 6.6 RUFF 6.7
CSER4	mytalk.server.transfer.RecordMessageTransfer mytalk.server.usermanager.UserManager mytalk.server.data.RecordMessageData	RUFF 6.10 RUFF 6.11 RUFF 7.5 RUFF 7.6
CSER5	mytalk.server.transfer.TutorialTransfer mytalk.server.transfer.TutorialTransfer mytalk.server.data.TutorialData	RUFF 4 RUFF 4.1
CSER6	mytalk.server.transfer.CallTransfer mytalk.server.transfer.SendFileTransfer mytalk.server.transfer.RecordMessageTransfer	RUFF 7 RUFF 7.1 RUFF 7.2 RUFF 7.3 RUFF 7.4 RUFF 7.7 RUFF 7.8 RUFF 7.9

**Tabella 1:** Tracciamento tra componenti e requisiti

## 8 Tracciamento requisiti-componenti

Requisito	Componente Client	Componente Server
RUFO 1	CCLI2	CSER2
RUFO 1.1		-
RUFO 1.2		
RUFO 1.3		
RUFO 1.4		
RUFO 1.5		
RUFO 2	CCLI1	CSER1
RUFO 2.1		-
RUFO 2.2		
RUFF 3		CSER1
RUFF 3.1		
RUFF 3.2		
RUFF 3.3		
RUFF 3.4		-
RUFF 3.5		
RUFF 4	CCLI3	CSER5
RUFF 4.1		
RUFO 5	CCLI4	CSER2
RUFF 5.1		
RUFO 6	CCLI6	CSER3
RUFO 6.1		
RUFO 6.2		
RUFO 6.3		
RUFO 6.4		
RUFO 6.5		
RUFF 6.6		
RUFF 6.7		
RUFD 6.8		
RUFD 6.9		
RUFF 6.10		
RUFF 6.11		
RUFF 6.12	CCLI8	
RUFD 6.13	CCLI6	
RUFD 6.14	CCLI9	
RUFD 6.14.1		
RUFO 6.15	CCLI6	
RUFF 7	CCLI5	CSER6
RUFF 7.1		

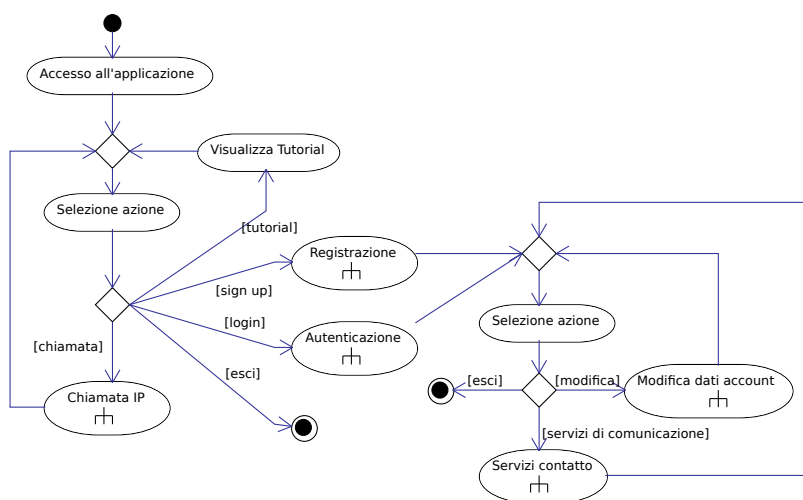
RUFF 7.2		-
RUFF 7.3		
RUFF 7.4	CCLI7	CSER4
RUFF 7.5		
RUFF 7.6	CCLI5	CSER6
RUFF 7.7		
RUFF 7.8		-
RUFF 7.9		
RUFO 8	CCLI1	CSER1

**Tabella 2:** Tracciamento tra requisiti e componenti

## 9 Diagramma delle attività

In questa sezione si illustreranno i diagrammi delle attività relativi alle possibili interazioni di un utente con il front-end del prodotto **MyTalk**.

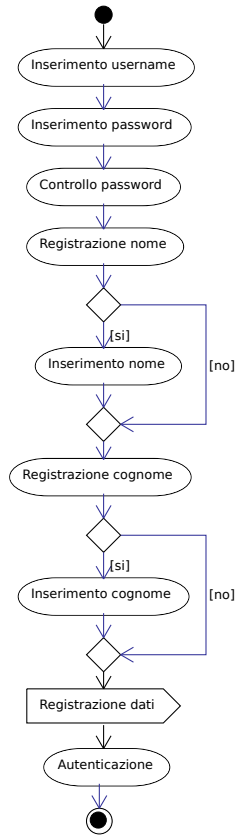
### 9.1 Diagramma generale



**Figura 7:** Diagramma della attività generale

Il diagramma rappresentato in figura 7 spiega le operazioni che un utente potrà svolgere per interagire con il sistema. Una volta aperta la pagina web del programma **MyTalk** l'utente potrà decidere di guardare i video tutorial riguardante il funzionamento del programma, contattare direttamente un utente se è a conoscenza del suo indirizzo IP, iscriversi al sito e, nel caso caso in cui la registrazione sia già stata effettuata, effettuare l'accesso con le proprie credenziali. Una volta effettuato l'accesso all'applicazione, l'utente avrà la possibilità di modificare le proprie credenziali e di selezionare uno dei servizi offerti per gli utenti autenticati.

## 9.2 Sottoattività di Registrazione

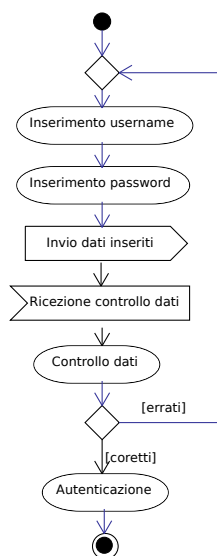


**Figura 8:** Diagramma delle attività di registrazione

Il diagramma rappresentato in figura 8 spiega le operazioni che un utente potrà svolgere una volta che quest'ultimo ha deciso di iscriversi al prodotto. Il modulo di registrazione prevede l'immissione obbligatoria di un'username e di una password, ripetuta due volte per controllare che la password inserita sia corretta. L'utente potrà inoltre, se lo desidera, inserire il proprio nome e cognome. Una volta inseriti tutti campi obbligatori più quelli opzionali, l'utente sarà registrato presso il server e autenticato automaticamente.



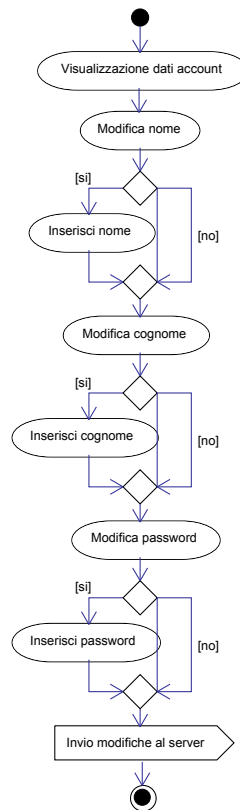
### 9.3 Sottoattività di Autenticazione



**Figura 9:** Diagramma delle attività di autenticazione

Il diagramma rappresentato in figura 9 spiega le operazioni che un utente potrà svolgere una volta che quest'ultimo ha deciso di autenticarsi. Il modulo di autenticazione prevede l'immissione obbligatoria di un'username e di una password. Una volta inseriti questi campi, sarà effettuato un controllo per vedere se l'utente risulta registrato presso il servizio e, in caso di esito positivo, l'utente sarà autenticato automaticamente, altrimenti dovrà inserire i dati nuovamente.

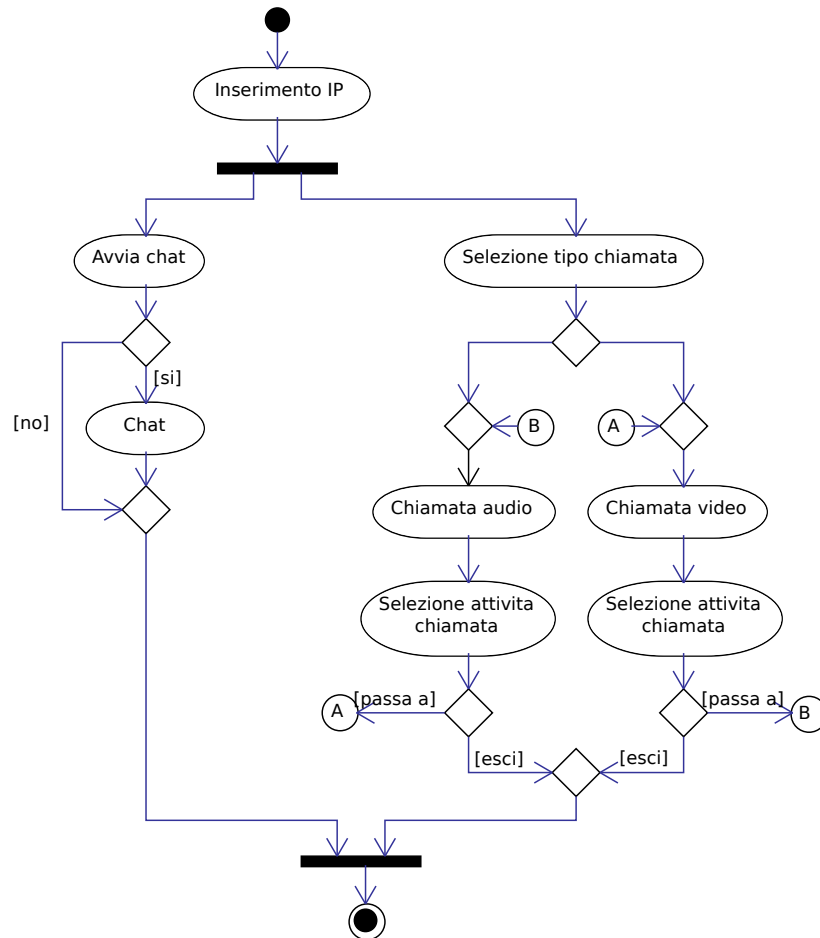
## 9.4 Sottoattività di Modifica dati account



**Figura 10:** Diagramma delle attività di modifica dei dati account

Il diagramma rappresentato in figura 10 spiega le operazioni che un utente potrà svolgere una volta che quest'ultimo ha deciso di modificare le proprie credenziali. Il modulo di modifica dati prevede la possibilità di inserire un nuovo nome, cognome o password. Una volta inseriti uno o più campi, saranno salvati i nuovi dati nel server.

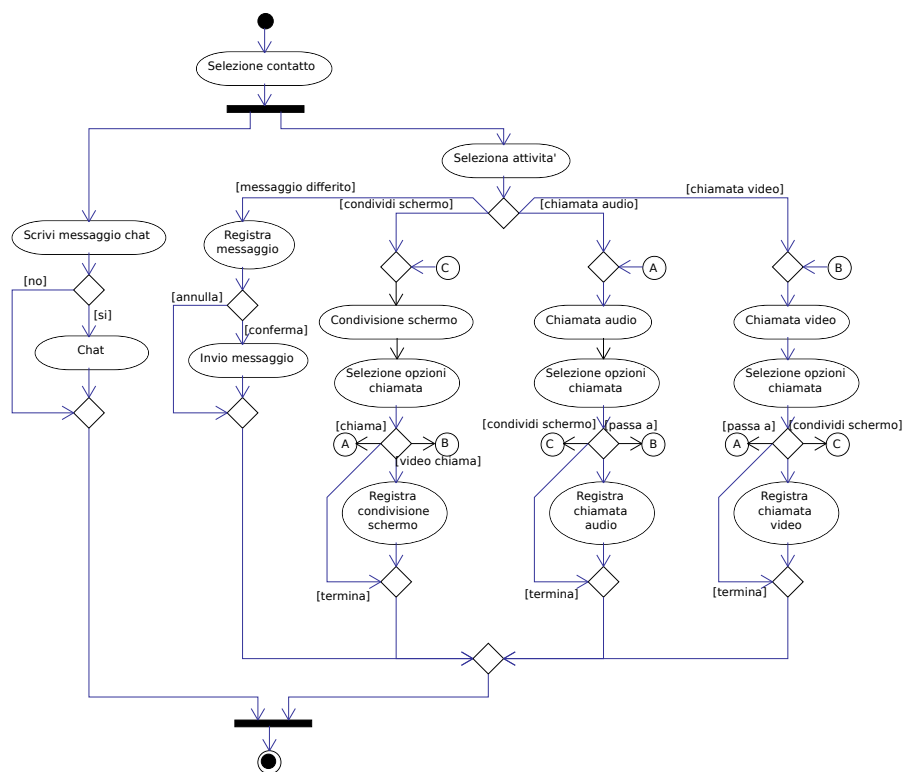
## 9.5 Sottoattività di Chiamata IP



**Figura 11:** Diagramma delle attività di una chiamata IP

Il diagramma rappresentato in figura 11 spiega le operazioni che un utente potrà svolgere una volta che quest'ultimo ha deciso di effettuare una chiamata IP. Una volta selezionato un indirizzo IP, l'utente avrà la possibilità di mandare messaggi di chat con la persona desiderata e contemporaneamente effettuare una chiamata audio o video. Durante la chiamata sarà possibile cambiare modalità, ovvero passare da chiamata audio a video e viceversa, o terminare la chiamata in corso.

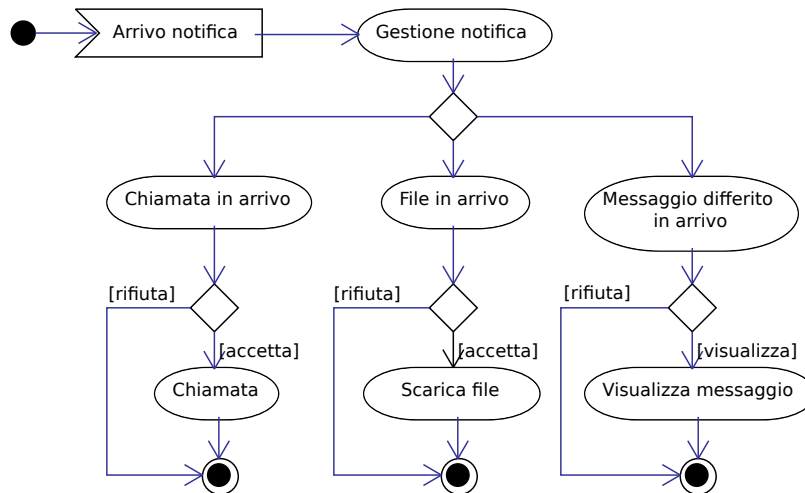
## 9.6 Sottoattività di Servizi contatto



**Figura 12:** Diagramma delle attività dei servizi di un contatto

Il diagramma rappresentato in figura 12 spiega le operazioni che un utente potrà svolgere una volta che quest'ultimo ha deciso di selezionare uno dei servizi offerti per gli utenti autenticati. Una volta selezionato un contatto dalla lista, l'utente avrà la possibilità di mandare messaggi di chat con la persona desiderata e contemporaneamente effettuare una chiamata audio o video, condividere lo schermo o mandare un messaggio differito. Come per la chiamata attraverso indirizzo IP, sarà possibile cambiare modalità o terminare la chiamata in corso, ed in più sarà possibile condividere lo schermo con la persona chiamata e registrare la chiamata. Qualora l'utente abbia scelto di condividere lo schermo e necessitasse anche di poter parlare con l'utente desiderato avrà la possibilità di attivare la chiamata audio o video mentre condivide lo schermo. Se l'utente infine decide di mandare un messaggio differito al contatto selezionato, avrà la possibilità di registrare un messaggio e, nel caso il messaggio vada bene, di inviarlo.

## 9.7 Diagramma di Gestione delle notifiche

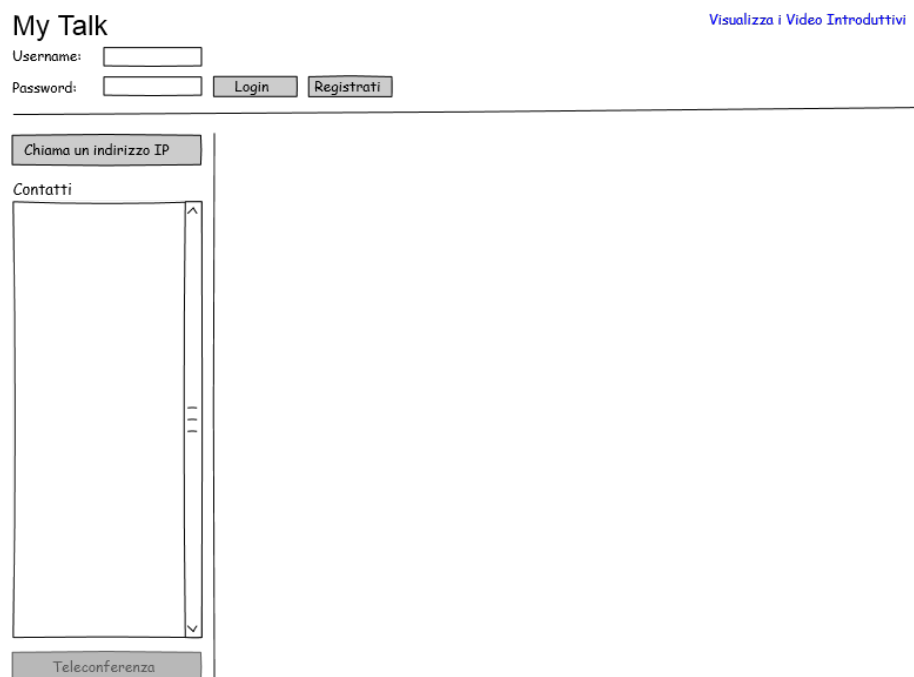


**Figura 13:** Diagramma delle attività di gestione delle notifiche

Il diagramma rappresentato in figura 13 spiega le operazioni che un utente potrà svolgere una volta che quest'ultimo riceve una notifica dal server. Se la notifica riguarda una chiamata in arrivo, l'utente potrà scegliere se accettare e avviare la chiamata o rifiutarla. Se la notifica riguarda un file in arrivo l'utente potrà scegliere se accettare e scaricare il file o rifiutarlo. Infine se la notifica riguarda un messaggio differito in arrivo, l'utente potrà scegliere se visualizzare il messaggio o di rifiutarlo.

## 10 Prototipo interfaccia utente

In questo capitolo andremo a descrivere come sarà l'applicazione a livello visivo e le operazioni che l'utente avrà modo di effettuare.



The screenshot shows the 'My Talk' web application interface. At the top left, the text 'My Talk' is displayed. To its right is a blue link that reads 'Visualizza i Video Introduttivi'. Below the title, there are two input fields: 'Username:' and 'Password:'. To the right of the 'Password:' field are two buttons: 'Login' and 'Registrati'. A horizontal line separates the login section from the main content area. On the left side of this area, there is a vertical sidebar. At the top of the sidebar is a button labeled 'Chiama un indirizzo IP'. Below it is a section titled 'Contatti' which contains a large, empty rectangular area with a vertical scrollbar on its right side. At the bottom of the sidebar is a button labeled 'Teleconferenza'.

**Figura 14:** Pagina iniziale

Come si può notare dalla la figura 14 una volta acceduti al sito le uniche operazioni che potremo fare saranno la possibilità di effettuare chiamate sapendo l'indirizzo IP del destinatario, guardare i video tutorial riguardanti il prodotto **MyTalk** e ricevere infine le chiamate da parte di terzi. Guardando la figura 15 le operazioni che possiamo quindi eseguire sono le stesse nel caso in cui non si è autenticati ma è possibile chiamare gli utenti presenti nella lista e eseguire videoconferenze con più utenti.

MyTalk

[Visualizza i Video Introduttivi](#)

Ciao, Clockwork Team!

[Modifica dati](#)[Logout](#)

Chiama un indirizzo IP

Contatti

Contatto 1  
Contatto 2  
Contatto 3  
Contatto 4  
Contatto 5  
Contatto 6  
Contatto 7  
Contatto 8  
Contatto 9  
Contatto 10

Teleconferenza

**Figura 15:** Pagina dopo login

La figura 16, la figura 17 e la figura 18 mostrano le varie schermate di comunicazioni possibili con i vari utenti. Presente anche la possibilità di registrare le comunicazioni in tempo reale, o inviare messaggi anche se l'utente non è connesso.

## MyTalk

[Visualizza i Video Introduttivi](#)

Ciao, Clockwork Team!

Modifica dati

Logout

Chiama un indirizzo IP

Contatti

Contatto 1
Contatto 2
Contatto 3
Contatto 4
Contatto 5
Contatto 6
Contatto 7
Contatto 8
Contatto 9
Contatto 10

Teleconferenza

Stai chiamando Contatto 3

Disattiva video

Condividi schermata

Termina

☐ Registra chiamata

Chat

Clockwork Team: posso chiamarti?
Contatto 3: sì, certo
Contatto 3: ma videochiamata o solo audio?
Clockwork Team: anche solo audio, tanto è una cosa veloce

Invia

Figura 16: Schermata videochiamata



## MyTalk

[Visualizza i Video Introduttivi](#)

Ciao, Clockwork Team!

[Modifica dati](#)[Logout](#)

Chiama un indirizzo IP

Contatti

Contatto 1

Contatto 2

Contatto 3

Contatto 4

Contatto 5

Contatto 6

Contatto 7

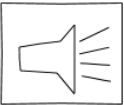
Contatto 8

Contatto 9

Contatto 10

Teleconferenza

Stai chiamando Contatto 3



Attiva video

Condividi schermata

Termina

☐ Registra chiamata

Chat

Clockwork Team: posso chiamarti?

Contatto 3: si, certo

Contatto 3: ma videochiamata o solo audio?

Clockwork Team: anche solo audio, tanto è una cosa veloce

Invia

Figura 17: Schermata chiamata audio

## MyTalk

[Visualizza i Video Introduttivi](#)

Ciao, Clockwork Team!

Modifica dati

Logout

Chiamata un indirizzo IP

Contatti

Contatto 1
Contatto 2
Contatto 3
Contatto 4
Contatto 5
Contatto 6
Contatto 7
Contatto 8
Contatto 9
Contatto 10

Teleconferenza

Contatto 3

Chiamata
Videochiamata
Condividi schermata
Invia un Videomessaggio

☐ Registra chiamata

Chat

Clockwork Team: posso chiamarti?
Contatto 3: sì, certo
Contatto 3: ma videochiamata o solo audio?
Clockwork Team: anche solo audio, tanto è una cosa veloce

Invia

**Figura 18:** Schermata comunicazione testuale

la figura 19 rappresenta la schermata di videoconferenza.

## MyTalk

[Visualizza i Video Introduttivi](#)

Ciao, Clockwork Team!

Modifica dati

Logout

Chiama un indirizzo IP

Contatti

Contatto 1  
Contatto 2  
Contatto 3  
Contatto 4  
Contatto 5  
Contatto 6  
Contatto 7  
Contatto 8  
Contatto 9  
Contatto 10

Teleconferenza

Disattiva video

Termina

☐ Registra chiamata

Chat

Clockwork Team: posso chiamarti?  
Contatto 3: si, certo  
Contatto 3: ma videochiamata o solo audio?  
Clockwork Team: anche solo audio, tanto è una cosa veloce

^  
|  
|  
|  
v

Teleconferenza

Invia

Figura 19: Schermata Videoconferenza

Specifica Tecnica v 2.0

43 di 43