responsabile@hourglabs.org

Specifica Tecnica

Versione 3.0

Informazioni documento

Nome	Specifica_Tecnica_v3.0.pdf
Versione	3.0
Data creazione	2013-01-21
Data ultima modifica	2013-09-16
Stato del Documento	Formale ad uso esterno
Redazione	Thomas Rossetto Gioele Lorenzo Cresce Sasa Ilievski Paolo Bustreo Riccardo Cesarotto Giovanni Morlin Umberto Martinati
Verifica	Giovanni Morlin, Paolo Bustreo
Approvazione	Sasa Ilievski
Distribuzione	**hourglass** Prof. Tullio Vardanega Prof. Riccardo Cardin

Registro delle modifiche

Data	Versione	Ruolo	Descrizione	Autore
2013-09-16	3.0	Responsabile	Approvazione versione finale documento	Thomas Rossetto
2013-09-13	2.3	Verificatore	Verifica del documento	Paolo Bustreo
2013-09-12	2.2	Progettista	Aggiunta introduzione tecnologie, riscrittura pattern MVP e modifica diagrammi delle classi errati.	Gioele Lorenzo Cresce
2013-09-04	2.1	Progettista	Aggiunta package Display	Riccardo Cesarotto
2013-05-12	2.0	Responsabile	Approvazione documento	Sasa Ilievski
2013-05-12	1.9	Verificatore	Verifica documento	Giovanni Morlin, Paolo Bustreo
2013-05-11	1.8.1	Progettista	Stesura classe ContactProfile (presenter e view), stesura subpackage view.contactprofile	Umberto Martinati
2013-05-10	1.8	Progettista	Inizio stesura componenti per requisiti opzionali: ContactList e Message (presenter e view)	Riccardo Cesarotto
2013-03-14	1.7	Responsabile	Validazione documento	Riccardo Cesarotto

Data	Versione	Ruolo	Descrizione	Autore
2013-03-13	1.6	Verificatore	Verifica documento	Umberto Martinati, Sasa Sasa Ilievski
2013-03-12	1.5	Progettista	Stesura parte Server e tracciamento	Gioele Lorenzo Cresce
2013-03-11	1.4	Progettista	Completamento stesura client: package view. Stesura package event, shared, rpcservice, wrapper	Paolo Bustreo
2013-03-10	1.3	Progettista	Completamento stesura client: package presenter. Inizio stesura package view	Giovanni Morlin
2013-03-07	1.2	Progettista	Inizio stesura client: package appcontroller, connection, presenter (e subpackage)	Gioele Lorenzo Cresce
2013-02-20	1.1	-	Inizio riprogettazione totale dell'architettura (requisiti obbligatori) in seguito alle correzioni ricevute nella revisione RP	-
2013-01-29	1.0	Responsabile	Approvazione documento	Giovanni Morlin
2013-01-29	0.10	Progettista	Correzione errori emersi dalla verifica	Sasa Ilievski
2013-01-28	0.9	Verificatore	Verifica documento	Riccardo Cesarotto
2013-01-27	0.8	Progettista	Ordinamento sezioni esistenti e stesura sezioni rimanenti	Paolo Bustreo
2013-01-26	0.7	Progettista	Aggiunta sezione *7 - Tracciamento della relazione componenti - requisiti*	Thomas Rossetto
2013-01-25	0.6	Progettista	Aggiunta la sottosezione *3.2 - Presenter*	Thomas Rossetto
2013-01-23	0.5	Progettista	Aggiunta la sottosezione *3.1 - Package View*	Paolo Bustreo
2013-01-22	0.4	Progettista	Aggiunta dei diagrammi delle attività	Sasa Ilievski
2013-01-20	0.3	Progettista	Aggiunta la sottosezione *3.3 - Package Model*	Gioele Lorenzo Cresce
2013-01-19	0.2	Progettista	Stesura preliminare delle sezioni *2 - Definizione del Prodotto* e *4 - Design Pattern*	Gioele Lorenzo Cresce
2013-01-18	0.1	Responsabile	Creazione del documento e stesura preliminare dello scheletro dello stesso	Thomas Rossetto

Sommario

Scopo del presente documento è di illustrare a quali risultati il gruppo ***hourglass*** *sia pervenuto durante gli approcci preliminari della Progettazione. Sono emerse da tali attività una iniziale visione del prodotto e una sua possibile architettura ad alto livello, che vengono nel presente documento condivise.*

Indice

1 Introduzione

1.1 Scopo del documento

Lo scopo della specifica tecnica è quello di mostrare le scelte progettuali che il gruppo **hourglass** ha deciso di seguire nella realizzazione del prodotto **MyTalk**. Verranno di seguito presentati la struttura dei package e le principali classi che li compongono, dando di queste ultime una descrizione accurata e prestando particolare attenzione alla loro funzione ed alle dipendenze e relazioni con altre classi. Verranno descritti i pattern utilizzati, presentati alcuni diagrammi di attività per presentare il flusso di funzionamento del sistema. Segue poi una stima sulla fattibilità e sul bisogno di risorse, in funzione dell'architettura riportata. Infine, verrà presentato il tracciamento di Componenti-Requisiti e viceversa.

1.2 Scopo del prodotto

Lo scopo del prodotto sarà quello di offrire una piattaforma di interazione in tempo reale di tipo audio-video tra Utenti, usufruibile con la sola necessità di un browser e di una connessione Internet e senza necessità di plugin o software aggiuntivi.

1.3 Glossario

Per migliorare la comprensione dei documenti ed evitare ogni ambiguità riguardante il linguaggio e i termini utilizzati, si allega il glossario nel file *Glossario_ v3.0.pdf*, al cui interno sarà possibile trovare una descrizione dei suddetti. Ogni ricorrenza di un termine da glossario sarà segnalata con l'utilizzo di *sottolineatura*.

1.4 Riferimenti

1.4.1 Normativi

- Capitolato d'appalto: **MyTalk** rilasciato dal proponente **Zucchetti SpA**, reperibile all'indirizzo: http://www.math.unipd.it/ tullio/IS-1/2012/Progetto/C1.pdf
- Norme di progetto (allegato nel file*Norme_ di_ Progetto_ v4.0.pdf*)
- Analisi dei Requisiti (allegato nel file *Analisi_ dei_ Requisiti_ v4.0.pdf*)
- Standard UML 2.0: http://www.omg.org/spec/UML/2.0/

1.4.2 Informativi

- Glossario (allegato nel file *Glossario_ v2.0.pdf*
- Google Web Kit code.google.com/webtoolkit/

2 Introduzione Tecnologie

In questa sezione vengono presentate le principali tecnologie utilizzate nell'architettura. Nello specifico vengono introdotte le seguenti tecnologie:

- Framework GWT
- WebSocket
- Servlet

2.1 Framework GWT

Google Web Toolkit (*GWT*) è un framework *open-source* sviluppato da Google che permette lo sviluppo e il mantenimento di applicativi web front-end *Javascript* scrivendoli in linguaggio *Java*.
Il framwork GWT offre molte features molto utili per lo sviluppo di un applicativo. Nella seguente architettura viene utilizzato il meccanismo di Event Handling. Tale meccanismo utilizza eventi i quali vengono intercettati da un'istanza della classe `HandlerManager` fornita da GWT. A vengono aggiunti gli eventi di cui deve gestire la logica di risposta tramite il metodo `addHandler`. Ogni volta che un evento viene lanciato se tale evento è stato aggiunto al `HandlerManager` verrà automaticamente eseguita la logica definita nell'Handler.
Nell'architettura tutti gli eventi definiti si trovano nel pacakge `it.hourglass.myTalk.client.event`.
Un altro meccanismo, simile a quello appena definito, è quello di History. Gli History token sono degli eventi predefiniti i quali vengono identificati da una stringa che permette al sistema di identificali. Per utilizzare tale meccanismo la classe nella quale si vuole utilizzare deve implementare l'interfaccia: `ValueChangeHandler<String>`. Tale interfaccia espone il metodo `onValueChange(ValueChangeEvent<String> event)` nella quale verrà definita la logica che andrà eseguita in risposta ad ogni History Token definito. Una volta che verrà lanciato un nuovo History Token (tramite il metodo `History.newItem(String token)`) la classe che implementa l'interfaccia `ValueChangeHandler` intercetterà il token e eseguirà la logica definita per quel determinato token.

2.2 WebSocket

WebSocket è una tecnologia web che fornisce canali di comunicazione full-duplex attraverso una singola connessione. Le comunicazioni sono fatte attraverso la porta TCP 80, che è un vantaggio per quegli ambienti che bloccano porte non standard utilizzando dei firewall. Nella nostra architettura questa tecnologia è utilizzata per collegarsi al server *WebSockt* e scambiare con lo stesso i messaggi *JSON* necessari per la registrazione nel server, lo scambio di notifiche e per l'inizializzazione della chiamata tramite la tecnologia *WebRTC*.

2.3 WebRTC

WebRTC è una tecnologia open source nata il 1º giugno 2011 che consente ai browser di effettuare in tempo reale la videochat. È basata su HTML5 e JavaScript. La sua inclusione

nel World Wide Web Consortium (W3C) standard è supportato da Google, Mozilla e Opera. WebRTC utilizza per l'audio i codec iSAC (per connessioni veloci) eiLBC (se la connessione è più lenta) e il codec VP8 per il video e si sta lavorando a migrare il plugin di Google Talk video chat per il quadro webRTC. Questa tecnologia sta alla base della funzionalità della chiamata e dello scambio di messaggi tramite chat.

3 Definizione del prodotto

3.1 Metodo e formalismo di specifica

La specifica del presente documento è realizzata per fornire un'idea generale dei componenti e della organizzazione dei questi all'interno del progetto, che dovranno poi essere descritte in dettaglio nella Progettazione di Dettaglio. Questo non significa che tutte le unità software qui descritte saranno sviluppate, ma sarà responsabilità dei Progettisti decidere se realizzarle o meno. Questa libertà è limitata dall'obbligo di non snaturare lo schema generale del prodotto qui successivamente descritto.

3.1.1 Diagrammi UML

Nella seguente documentazione verrà fatto uso dei diagrammi UML, secondo i seguenti criteri:

- Per rappresentare l'architettura generale del progetto sono stati utilizzati dei diagrammi dei package che rappresentano le parti più significative. I package relativi all'architettura software avranno uno sfondo colorato, mentre i relativi sotto-package interni ad essi avranno uno sfondo bianco.
- Per rappresentare le singole parti si utilizzeranno i diagrammi dei package e i diagrammi delle classi, così da poter presentare una descrizione di maggior dettaglio.

3.2 Presentazione dell'architettura generale del sistema e identificazione dei componenti architetturali di alto livello

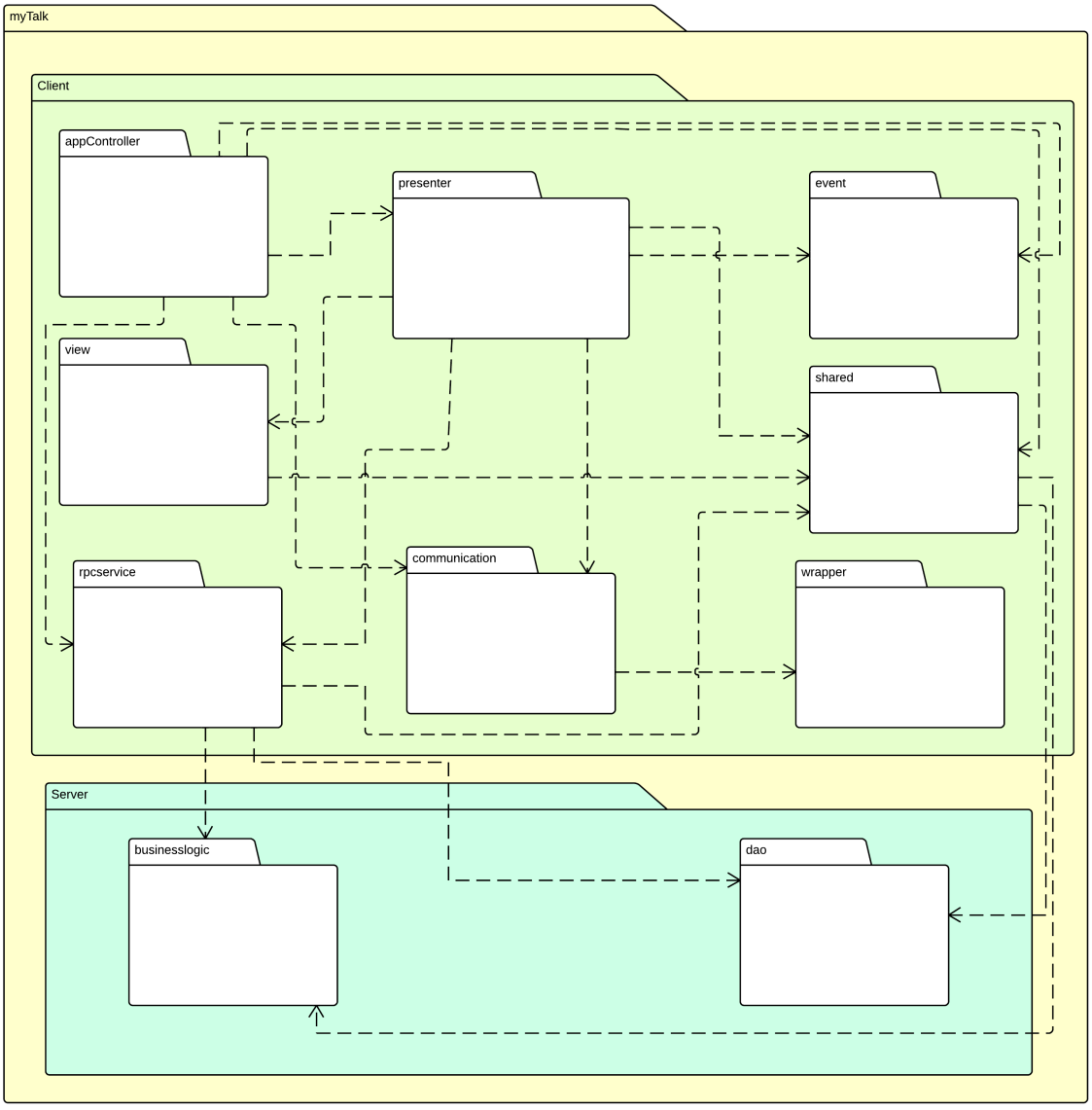

Figura 1: Diagramma dei package generale

La logica generale del progetto è descritta dal seguente diagramma UML :

- **it.hourglass.myTalk**
  Contiene l'intero sistema MyTalk.
- **it.hourglass.myTalk.client**
  Questo package contiene le classi contenenti i metodi e i campi dati necessari alla creazione della classe `AppController` per l'avvio del client.
- **it.hourglass.myTalk.client.appcontroller**
  Questo package contiene la definizione della classe `AppController` la quale si occupa del controllo delle richieste fatte al sistema.
- **it.hourglass.myTalk.client.communication**
  Questo package contiene le classi necessarie per la connessione al server WebSocket del programma MyTalk. Inoltre il package contiene anche i metodi per l'avvio della chiamata.
- **it.hourglass.myTalk.client.event**
  Questo package contiene la definizione e gli handler dei vari eventi che si possono scatenare.
- **it.hourglass.myTalk.Client.presenter**
  Questo package contiene tutta la application logic dell'applicazione software.
- **it.hourglass.myTalk.Client.presenter.display**
  Questo package contiene la definizione di tutte le interfacce Display che espongono i metodi che un presenter può richiedere alla propria view.
- **it.hourglass.myTalk.Client.presenter.profilemanagment**
  Questo subpackage contiene le classi contenenti l'application logic per la gestione del proprio profilo.
- **it.hourglass.myTalk.client.rpcservice**
  Questo package contiene le classi necessarie all'applicativo per richiedere una chiamata RPC (offerte dal framework GWT).
- **it.hourglass.myTalk.shared**
  Questo package contiene gli la definizione degli oggetti comuni a Client e Server.Inoltre contiene la definizione delle classi necessarie al controllo del profilo locale.
- **it.hourglass.myTalk.client.view**
  Questo package contiene tutti i componenti dell'interfaccia utente dell'applicazione ed ha lo scopo di gestire il layout di tali elementi.
- **it.hourglass.myTalk.client.view.contactprofile**
  Questo subpackage del package `view` riunisce i componenti della pagina di visione del profilo altrui.

- **it.hourglass.myTalk.client.view.profilemanagment**
  Questo subpackage del package `view` riunisce i componenti della pagina di gestione del proprio profilo.
- **it.hourglass.myTalk.client.wrappers**
  Questo package contiene la definizione delle classi wrapper delle varie funzioni di webRTC.
- **it.hourglass.myTalk.server.dao**
  Questo package contiene le classi che si occupano di permettere all'applicativo di interagire con il database sottostante.
- **it.hourglass.myTalk.server.bussineslogic**
  Questo package contiene le classi che si occupano di permettere all'applicativo di interagire con il database sottostante.

4 Descrizione dei singoli componenti

In questa sezione si descriverà in dettaglio ogni singolo componente. Si presterà particolare attenzione a funzionalità offerte, relazioni con altri componenti e dipendenze. Per evitare ridondanza, d' ora in poi classi appartenenti a package trattati in quel momento non saranno presentati indicando il percorso completo del nome, ma solo il nome del package a cui esse appartengono. Verrà altresì indicato il nome completo nel momento in cui si indicheranno componenti esterne al package considerato.

4.1 Client

4.2 Package it.hourglass.myTalk.client

- **Tipo, obiettivo, funzione del componente:** Il package `client` contiene tutti i sottopackage e le classi che compongono la parte client del sistema che non richiede accesso al database. Contiene inoltre le classi necessarie alla creazione della connessione diretta con un altro utente per l'instanziazione di una chiamata. Il package conterrà dunque l'application logic, cioè tutta la logica dell'applicativo che non richiede una elaborazione dei dati, le viste che permettono all'utente l'interazione con il sistema, la logica delle stesse e le componenti che consentono il collegamento con il server WebSocket.

- **Relazioni d'uso con altre componenti:** Questo package utilizzerà i metodi esposti dall'interfaccia `it.hourglass.myTalk.client.rpcservice.RPCService` per il collegamento con il database e il package `it.hourglass.myTalk.communication` per la connessione diretta con un altro utente per effettuare una chiamata. Il package `client` utilizzerà inoltre alcune classi fornite dal framework GWT. L'utilizzo di tali classi sarà descritto nelle descrizioni delle componenti che seguono.

- **Attività svolte e dati trattati:** Il package `client` si occupa di dare la possibilità all'utente di interagire con il sistema.Le classi del package si occuperanno dunque di inizializzare il programma, la gestione di instaziazione di una chiamata (o la ricezione di una chiamata), l'autenticazione e registrazione di un utente nel sistema e la visualizzazione o modifica del profilo di un utente. Le classi che permettono tali operazioni saranno descritte nelle descrizioni delle componenti che seguono.

4.2.1 Package it.hourglass.myTalk.client.appcontroller.AppController

`Classe client.appcontroller.AppController`

- **Tipo, obiettivo, funzione del componente:** La classe AppController si occupa di ricevere le richieste inviate dal utente e di modificare lo stato del sistema di conseguenza.

- **Relazioni d'uso con altre componenti:** La classe fa largo uso delle funzionalità offerte da GWT.

Per il corretto funzionamento della classe sono necessarie le classi fornite da GWT `com.google.gwt.event.shared.HandlerManager` e `com.google.gwt.user.client.History`. L'HandlerManager ha la funzionalità di un event Bus. La classe si occuperà di aggiungere gli eventi che il sistema deve riconoscere, (tali eventi sono definiti nel package `it.hourglass.myTalk.client.event`), all'event bus e sarà lo stesso event bus a far eseguire al sistema le operazioni necessarie per soddisfare la richiesta.
Tali operazioni sono gestite utilizzando il meccanismo History Token sempre fornito da GWT tramite la classe `com.google.gwt.user.client.History`.
Ogni volta che al sistema viene richiesta una operazione tramite l'event Bus sarà generato un token. Tale token sarà utilizzato dall'AppController per capire l'operazione da effettuare e modificare lo stato del sistema di conseguenza. Più precisamente si occuperà di richiedere lo switch view o recuperare il profilo in seguito all'autenticazione.
Per ogni evento implementato dall'EventHandler (cioè tutti gli eventi di `it.hourglass.myTalk.client.event`) sarà implementato un metodo che si occupa di eseguire l'azione a lui associata. La classe utilizzerà la classe : `it.hourglass.myTalk.client.presenter.BasePresenter` per richiedere lo switch view, la classe : `it.hourglass.myTalk.client.communication.WSConnectionn` per la creazione della connessione con il server WebSocket.

- **Attività svolte e dati trattati:** La classe si occupa di ricevere le richieste dalle classi presenter (contenute nel package `it.hourglass.myTalk.client.presenter` e di modificare lo stato del sistema di conseguenza. Si occuperà inoltre della creazione della connessione con il server WebSocket e del recupero del profilo in seguito alla richiesta di autenticazione.

4.2.2 Package it.hourglass.myTalk.client.communication

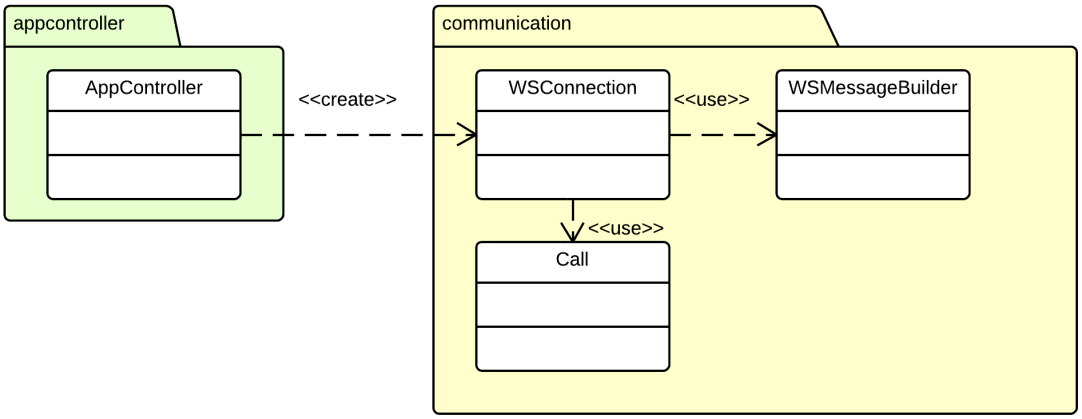

Figura 2: Package it.hourglass.myTalk.client.communication

`Classe client.communication.WSConnection`

- **Tipo, obiettivo, funzione del componente:** Questa classe si occupa principalmente della connessione, dell'invio delle richieste al server WebSocket e della ricezione delle risposte dello stesso. Tutte le informazioni scambiate saranno in formato JSON.
- **Relazioni d'uso con altre componenti:** Una istanza di questa classe è creata dalla classe : `it.hourglass.myTalk.client.appcontroller.AppController`. La classe si occupa di creare un istanza di `it.hourglass.myTalk.client.communication.Call` in seguito alla richiesta o ricezione di una chiamata. Inoltre si occupa di smistare le richieste in arrivo e di delegarle al chiamante, ovvero `it.hourglass.myTalk.client.appcontroller.AppController`, oppure all'istanza creata di Call.
- **Attività svolte e dati trattati:** La classe WSConnection si occupa inizialmente di instaurare una connessione con il server WebSocket: Da quel momento in poi sarà in grado di ricevere i vari messaggi provenienti da esso e fare le elaborazioni conseguenti. Le elaborazioni più importanti che dovrà fare internamente sono quelle legate alla registrazione nel server con ID univoco e la gestione delle fasi iniziali della chiamata. Per quanto riguarda l'invio delle informazioni, la classe espone un metodo per l'invio di stringhe, che verranno formattate in formato JSON prima dell'invio utilizzando i metodi di `it.hourglass.myTalk.client.communication.WSMessageBuilder`.

`Classe client.connection.Call`

- **Tipo, obiettivo, funzione del componente:** Questa classe si occupa di gestire una chiamata del sistema MyTalk. La classe avrà il compito di instaurare un collegamento fra due utenti e di gestire gli stream audio / video / dati.
- **Relazioni d'uso con altre componenti:** Una istanza di questa classe è creata dalla classe :
  `it.hourglass.myTalk.client.communication.WSConnectionn`.
  La classe fa uso dei metodi definiti in :
  `it.hourglass.myTalk.client.communication.WSMessageBuilder` per la creazione dei JSON da inviare al server WebSocket tramite la classe
  `it.hourglass.myTalk.client.communication.WSConnectionn`.
- **Attività svolte e dati trattati:** La classe si occupa di tutta la gestione lato client della chiamata per il sistema MyTalk. La parte iniziale della chiamata, ovvero lo scambio delle informazioni preliminari tra due utenti, avviene tramite l'elaborazione dei messaggi ricevuti e poi smistati a Call dalla classe `it.hourglass.myTalk.client.communication.WSConnectionn`. Una volta avviata la chiamata, la classe Call avrà il solo compito di monitorare le statistiche della chiamata e di interromperla nel caso arrivi il comando dedicato. Per tutte le operazioni legate alle librerie webRTC, Call fa largo uso delle classi presente nel package `it.hourglass.myTalk.client.wrapper`: In esso sono contenute tutte le classi che fanno da wrapper ai metodi delle librerie di webRTC.

`Classe client.communication.WSMessageBuilder`

- **Tipo, obiettivo, funzione del componente:** Questa classe si occupa di formattare le varie informazioni in formato JSON.
- **Relazioni d'uso con altre componenti:** I metodi di questa classe sono utilizzati dalla classi:
  `it.hourglass.myTalk.client.communication.WSConnectionn` e
  `it.hourglass.myTalk.client.communication.Call`.
- **Attività svolte e dati trattati:** La classe è composta da metodi statici che verranno utilizzati per formattare JSON strutturalmente corretti e processabili dal server in base alle informazioni date in ingresso. I principali JSON sono:
  - Registrazione
  - Chiamata
  - Risposta chiamata
  - Declinazione chiamata
  - Fine chiamata
  - Scambio di informazioni RTC

- Ping
- Notifica
- Lista contatti

4.2.3 Package it.hourglass.myTalk.client.view

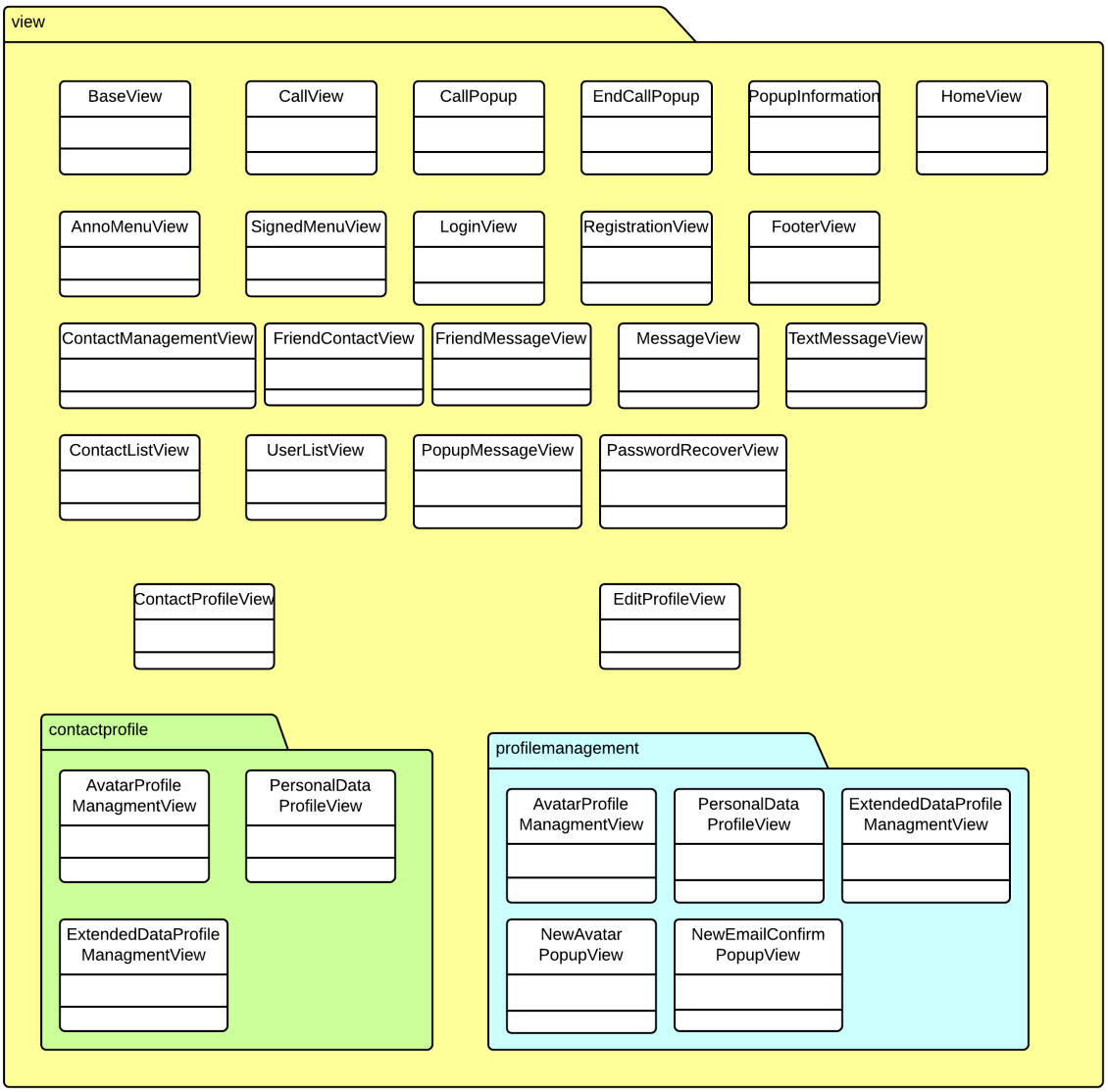

Figura 3: Package it.hourglass.myTalk.view

- **Tipo, obiettivo, funzione del componente:** Questo package costituisce la parte del sistema che si occupa della grafica. Le varie classi vengono viste come widget i quali vengono inseriti su una base che è rappresentata dalla classe BaseView. Tutte le classi saranno costituite da vari componenti grafici resi disponibili dalle librerie grafiche del framework GWT.

- **Relazioni d'uso con altre componenti:** Ogni classe contenuta nel package utilizza le interfacce Display presenti nel package `it.hourglass.myTalk.client.presenter.display`. Le istanze di tali classi vengono create dalla classe `it.hourglass.myTalk.client.presenter.BasePresenter` in seguito ad una richiesta della classe `it.hourglass.myTalk.client.appcontroller.AppController`.
- **Attività svolte e dati trattati:** Specificati nelle componenti che seguono.

`Classe client.view.BaseView`

- **Tipo, obiettivo, funzione del componente:** Questa classe si occupa di fornire una base alla view del programma. Su di essa verranno aggiunte istanze delle altre classi contenute in `it.hourglass.myTalk.client.view` le quali permetteranno l'interazione dell'utente con il sistema.
  Tali classi vengono viste come dei widget da aggiungere alla classe `BaseView`. La classe conterrà tre pannelli sui quali verranno inseriti i vari widget che compongono la view richiesta dall'utente.
- **Relazioni d'uso con altre componenti:** Questa classe non utilizzerà nessuna classe. Su di essa invece andranno ad aggiungersi i vari widget (che sono rappresentati dalle altre classi del package).
- **Attività svolte e dati trattati:** Questa classe fungerà da base per l'inserimento dei vari widget per comporre la view del sistema. La classe non conterrà alcuna logica ma avrà solo un compito di presentazione all'utente.

`Classe client.view.AnnoMenuView`

- **Tipo, obiettivo, funzione del componente:** Questa classe rappresenta il widget che funge da Menù per un utente non autenticato. Tramite questo widget sarà possibile richiedere il cambiamento del corpo principale della `BaseView`.
- **Relazioni d'uso con altre componenti:** Un istanza di questa classe verrà aggiunta dentro la classe BaseView.
  Questa classe implementa i metodi presenti nell'interfaccia `it.hourglass.myTalk.client.presenter.display.AnnoMenuDisplay`. Tali metodi rappresentano le richieste che il `it.hourglass.myTalk.client.presenter.AnnoMenuPresenter` può richiedere alla classe `AnnoMenuView`.
- **Attività svolte e dati trattati:** La classe non conterrà alcuna logica ma avrà solo un compito di presentazione all'utente. Essa contiene solo le componenti grafiche che permettono all'utente non autenticato di navigare nel sistema.

`Classe client.view.SignedMenuView`

- **Tipo, obiettivo, funzione del componente:** Questa classe rappresenta il widget che funge da Menù per un utente autenticato. Tramite questo widget sarà possibile richiedere il cambiamento del corpo principale della BaseView.
- **Relazioni d'uso con altre componenti:** Un istanza di questa classe verrà aggiunta dentro la classe BaseView.
  Questa classe implementa i metodi presenti nell'interfaccia `it.hourglass.myTalk.client.presenter.display.SignedMenuDisplay`. Tali metodi rappresentano le richieste che il `it.hourglass.myTalk.client.presenter.AnnoMenuPresenter` può richiedere alla classe `SignedMenuView`.
- **Attività svolte e dati trattati:** La classe non conterrà alcuna logica ma avrà solo un compito di presentazione all'utente. Essa contiene solo le componenti grafiche che permettono all'utente autenticato di navigare nel sistema.

`Classe client.view.HomeView`

- **Tipo, obiettivo, funzione del componente:** Questa classe rappresenta il widget che funge da homepage del client.
- **Relazioni d'uso con altre componenti:** Un istanza di questa classe verrà aggiunta dentro la classe BaseView.
  Questa classe implementa i metodi presenti nell'interfaccia `it.hourglass.myTalk.client.presenter.display.HomeViewDisplay`. Tali metodi rappresentano le richieste che il `it.hourglass.myTalk.client.presenter.HomeView` può richiedere alla classe `HomeView`.
- **Attività svolte e dati trattati:** La view ha solo funzionalità di presentazione all'utente. Questa view non conterrà nessuna componente grafica utile a richiedere una azione specifica al sistema ma fungerà solo da pagina iniziale del programma.

`Classe client.view.CallView`

- **Tipo, obiettivo, funzione del componente:** Questa classe rappresenta il widget per la gestione della chiamata. Tale widget permetterà all'utente di richiedere la creazione di una nuova chiamata (con la possibilità di scegliere il tipo di flusso di dati richiesto dalla chiamata (Audio, Video, Dati)), di ricevere una chiamata e di visualizzare la statistiche di una chiamata.
- **Relazioni d'uso con altre componenti:** Un istanza di questa classe verrà aggiunta dentro la classe BaseView.
  Questa classe implementa i metodi presenti nell'interfaccia `it.hourglass.myTalk.client.presenter.display.CallDisplay`. Tali metodi rappresentano le richieste che il `it.hourglass.myTalk.client.presenter.CallPresenter`

può richiedere alla classe `CallView`. La richiesta di inizializzazione della classe `CallView` viene effettuata in risposta ad un evento da `it.hourglass.myTalk.client.appcontroller.AppController`.

- **Attività svolte e dati trattati:** La view ha solo funzionalità di presentazione all'utente. La classe conterrà le componenti grafiche necessarie alla richiesta di una chiamata (con la selezione del tipo di stream da inviare), di visualizzare il video e infine di visualizzare le statistiche della chiamata.

`Classe client.view.LoginView`

- **Tipo, obiettivo, funzione del componente:** Questa classe rappresenta il widget che permette ad un utente di richiedere il login nel sistema.
- **Relazioni d'uso con altre componenti:** Un istanza di questa classe verrà aggiunta dentro la classe BaseView.
  Questa classe implementa i metodi presenti nell'interfaccia `it.hourglass.myTalk.client.presenter.display.LoginDisplay`. Tali metodi rappresentano le richieste che il `it.hourglass.myTalk.client.presenter.LoginPresenter` può richiedere alla classe `LoginView`.
- **Attività svolte e dati trattati:** La view ha solo funzionalità di presentazione all'utente. La classe conterrà tutti gli elementi grafici necessari alla richiesta di login da parte di un utente.

`Classe client.view.RegistrationView`

- **Tipo, obiettivo, funzione del componente:** Questa classe rappresenta il widget che permette ad un utente di registrarsi nel sistema.
- **Relazioni d'uso con altre componenti:** Un istanza di questa classe verrà aggiunta dentro la classe BaseView.
  Questa classe implementa i metodi presenti nell'interfaccia `it.hourglass.myTalk.client.presenter.display.RegistrationDisplay`. Tali metodi rappresentano le richieste che il `it.hourglass.myTalk.client.presenter.RegistrationPresenter` può richiedere alla classe `RegistrationView`.
- **Attività svolte e dati trattati:** La view ha solo funzionalità di presentazione all'utente. La classe conterrà tutti gli elementi grafici necessari alla richiesta di registrazione da parte di un utente nel sistema richiedendo tutte le informazioni personali necessarie.

`Classe client.view.PasswordRecoverView`

- **Tipo, obiettivo, funzione del componente:** Questa classe rappresenta il widget che permette di richiedere il recupero della password in caso l'utente non la ricordi.

- **Relazioni d'uso con altre componenti:** Un istanza di questa classe verrà aggiunta dentro la classe BaseView.
  Questa classe implementa i metodi presenti nell'interfaccia `it.hourglass.myTalk.client.presenter.display.PasswordRecoverDisplay`. Tali metodi rappresentano le richieste che il `it.hourglass.myTalk.client.presenter.PasswordRecoverPresenter` può richiedere alla classe `PasswordRecoverView`.

- **Attività svolte e dati trattati:** La view ha solo funzionalità di presentazione all'utente. La classe conterrà tutti gli elementi grafici necessari a richiedere la password dimenticata.

`Classe client.view.FooterView`

- **Tipo, obiettivo, funzione del componente:** Questa classe rappresenta il widget che funge da footer alla view del sistema contenente l'identificativo unico di un utente

- **Relazioni d'uso con altre componenti:** Un istanza di questa classe verrà aggiunta dentro la classe BaseView.
  Questa classe implementa i metodi presenti nell'interfaccia `it.hourglass.myTalk.client.presenter.display.FooterDisplay`. Tali metodi rappresentano le richieste che il `it.hourglass.myTalk.client.presenter.FooterPresenter` può richiedere alla classe `FooterView`.

- **Attività svolte e dati trattati:** La view ha solo funzionalità di presentazione all'utente. La classe conterrà tutti gli elementi grafici necessari alla visualizzazione dell'identificativo di un utente.

`Classe client.view.EditProfileView`

- **Tipo, obiettivo, funzione del componente:** Questa classe rappresenta il widget che permette ad un utente di visualizzare il proprio profilo e di richiedere una modifica allo stesso. La view rappresentata da questa classe è costituita dai sottowidget presenti nel package :
  `it.hourglass.myTalk.client.view.it.hourglass.myTalk.client.view.ProfileManagment`.

- **Relazioni d'uso con altre componenti:** Un istanza di questa classe verrà aggiunta dentro la classe BaseView.
  Questa classe implementa i metodi presenti nell'interfaccia `it.hourglass.myTalk.client.presenter.display.EditProfileDisplay`. Tali metodi rappresentano le richieste che il `it.hourglass.myTalk.client.presenter.EditProfilePresenter` può richiedere alla classe `EditProfileView`. I sottowidget che compongono la view sono rappresentati dalle classi del sottopackage `it.hourglass.myTalk.client.view.ProfileManagment`.

- **Attività svolte e dati trattati:** La view ha solo funzionalità di presentazione all'utente. La classe conterrà servirà da base dove aggiungere i sottowidget della classi del sottopackage `profilemanagment`

`Classe client.view.ContactProfileView`

- **Tipo, obiettivo, funzione del componente:** Questa classe rappresenta il widget che permette ad un utente di visualizzare il profilo di un amico. La view rappresentata da questa classe è costituita dai sottowidget presenti nel package : `it.hourglass.myTalk.client.view.it.hourglass.myTalk.client.view.ContactProfile`.

- **Relazioni d'uso con altre componenti:** Un istanza di questa classe verrà aggiunta dentro la classe BaseView.
  Questa classe implementa i metodi presenti nell'interfaccia `it.hourglass.myTalk.client.presenter.display.ContactProfileDisplay`. Tali metodi rappresentano le richieste che il `it.hourglass.myTalk.client.presenter.ContactProfilePresenter` può richiedere alla classe `ContactProfileView`. I sottowidget che compongono la view sono rappresentati dalle classi del sottopackage `it.hourglass.myTalk.client.view.ContactProfile`.

- **Attività svolte e dati trattati:** La view ha solo funzionalità di presentazione all'utente. La classe conterrà servirà da base dove aggiungere i sottowidget della classi del sottopackage `contactprofile`

`Classe client.view.ContactListView`

- **Tipo, obiettivo, funzione del componente:** Questa classe rappresenta il widget che permette la visualizzazione della lista amici dell'utente. Ogni amico presente nella lista sarà visualizzato tramite un sottowidget rappresentato da una istanza della classe `UserListView`. Sarà inoltre possibile richiedere la pagina di gestione delle amicizie.

- **Relazioni d'uso con altre componenti:** Un istanza di questa classe verrà aggiunta dentro la classe BaseView.
  Questa classe implementa i metodi presenti nell'interfaccia `it.hourglass.myTalk.client.presenter.display.ContactListDisplay`. Tali metodi rappresentano le richieste che il `it.hourglass.myTalk.client.presenter.ContactListPresenter` può richiedere alla classe `ContactListView`.

- **Attività svolte e dati trattati:** La view ha solo funzionalità di presentazione all'utente. La classe conterrà per ogni amico un sottowidget associato. La view conterrà inoltre un bottone per richiedere la pagina di gestione delle amicizie.

`Classe client.view.UserListView`

- **Tipo, obiettivo, funzione del componente:** Questa classe rappresenta il sottowidget inserito nella classe `ContactListView` per rappresentare un amico nella lista amici. Questo sottowidget permetterà all'utente di richiedere un widget popup per inviare un messaggio(rappresentato dalla classe `PopupMessageView`), di chiamare l'utente o di richiedere la visualizzazione del profilo di tale utente.
- **Relazioni d'uso con altre componenti:** Un istanza di questa classe verrà aggiunta dentro la classe ContactListView per ogni amico dell'utente autenticato
  Questa classe implementa i metodi presenti nell'interfaccia `it.hourglass.myTalk.client.presenter.display.UserListDisplay`. Tali metodi rappresentano le richieste che il `it.hourglass.myTalk.client.presenter.UserListPresenter` può richiedere alla classe `UserListView`.
- **Attività svolte e dati trattati:** La view ha solo funzionalità di presentazione all'utente. La classe conterrà tutti gli elementi grafici per richiedere il popup per inviare un messaggio ad un utente (rappresentato dalla classe `PopupMessageView`) e per richiedere una chiamata rapida o il profilo dell'amico.

`Classe client.view.ContactManagementView`

- **Tipo, obiettivo, funzione del componente:** Questa classe rappresenta il widget che permette all'utente di gestire le proprie amicizie. Deve essere possibile aggiungere e eliminare un amicizia e visualizzare le richieste di amicizia. Per ogni richiesta di amicizia si utilizzerà un sottowidget `FriendMessageView`, mentre per ogni amico presente nella lista amici si utilizzerà un sottowidget `FriendContactView`. Utilizzando questi sottowidget aggiungerò gli elementi grafici per gestire le richieste di amicizia e gestire le amicizie.
- **Relazioni d'uso con altre componenti:** Un istanza di questa classe verrà aggiunta dentro la classe BaseView.
  Questa classe implementa i metodi presenti nell'interfaccia `it.hourglass.myTalk.client.presenter.display.ContactManagementDisplay`. Tali metodi rappresentano le richieste che il `it.hourglass.myTalk.client.presenter.ContactManagementPresenter` può richiedere alla classe `ContactManagementView`. Verranno inoltre utilizzate le classi : `FriendMessageView` e `FriendContactView` come sottowidget da aggiungere alla view ContactManagementView.
- **Attività svolte e dati trattati:** La view ha solo funzionalità di presentazione all'utente. La classe conterrà tutti gli elementi grafici necessari a gestire le richieste di amicizia e le amicizie (dando la possibilità ad un utente di richiedere o eliminare un amicizia).

`Classe client.view.FriendMessageView`

- **Tipo, obiettivo, funzione del componente:** Questa classe rappresenta il sotto-widget che permette all'utente di visualizzare, accettare o rifiutare una richiesta di amicizia.
- **Relazioni d'uso con altre componenti:** Un istanza di questa classe verrà aggiunta dentro la classe ContactManagementView.
  Questa classe implementa i metodi presenti nell'interfaccia `it.hourglass.myTalk.client.presenter.display.FriendMessageDisplay`. Tali metodi rappresentano le richieste che il `it.hourglass.myTalk.client.presenter.FriendMessagePresenter` può richiedere alla classe `FriendMessageView`.
- **Attività svolte e dati trattati:** La view ha solo funzionalità di presentazione all'utente. Questa classe conterrà tutti gli elementi grafici necessari a visualizzare, accettare o rifiutare un'amicizia.

`Classe client.view.FriendContactView`

- **Tipo, obiettivo, funzione del componente:** Questa classe rappresenta il sotto-widget che permette all'utente di eliminare un'amicizia già confermata.
- **Relazioni d'uso con altre componenti:** Un istanza di questa classe verrà aggiunta dentro la classe ContactManagementView.
  Questa classe implementa i metodi presenti nell'interfaccia `it.hourglass.myTalk.client.presenter.display.FriendContactDisplay`. Tali metodi rappresentano le richieste che il `it.hourglass.myTalk.client.presenter.FriendContactPresenter` può richiedere alla classe `FriendContactView`.
- **Attività svolte e dati trattati:** La view ha solo funzionalità di presentazione all'utente. Questa classe conterrà tutti gli elementi grafici necessari ad eliminare un'amicizia già confermata.

`Classe client.view.CallPopup`

- **Tipo, obiettivo, funzione del componente:** Questa classe rappresenta il popup che appare all'utente in caso di richiesta di chiamata. Deve essere possibile accettare o rifiutare una chiamata.
- **Relazioni d'uso con altre componenti:** Un istanza di questa classe verrà creata dalla classe `it.hourglass.myTalk.client.communication.WSConnectionn` in seguito alla ricezione di una richiesta di chiamata. Questa classe implementa i metodi presenti nell'interfaccia `it.hourglass.myTalk.client.presenter.display.CallPopupDisplay`. Tali metodi rappresentano le richieste che il `it.hourglass.myTalk.client.presenter.CallPopupPresenter` può richiedere alla classe `CallPopup`.

- **Attività svolte e dati trattati:** La view ha solo funzionalità di presentazione all'utente. Questa classe conterrà tutti gli elementi grafici necessari accettare o rifiutare una chiamata.

`Classe client.view.EndCallPopup`

- **Tipo, obiettivo, funzione del componente:** Questa classe rappresenta il popup che deve apparire quando una chiamata si conclude. In questo popup si visualizzeranno le statistiche finali di una chiamata.
- **Relazioni d'uso con altre componenti:** Un istanza di questa classe verrà creata dalla classe `it.hourglass.myTalk.client.communication.CallPresenter` in seguito alla chiusura di una chiamata. Questa classe implementa i metodi presenti nell'interfaccia `it.hourglass.myTalk.client.presenter.display.EndCallPopupDisplay`. Tali metodi rappresentano le richieste che il `it.hourglass.myTalk.client.presenter.EndCallPopupPresenter` può richiedere alla classe `EndCallPopup`.
- **Attività svolte e dati trattati:** La view ha solo funzionalità di presentazione all'utente. Questa classe conterrà tutti gli elementi grafici necessari a visualizzare le statistiche di fine chiamata.

`Classe client.view.PopupMessageView`

- **Tipo, obiettivo, funzione del componente:** Questa classe rappresenta il popup permette all'utente di inviare un messaggio a un amico.
- **Relazioni d'uso con altre componenti:** Questa classe non utilizza alcuna classe. Un istanza di questa classe verrà creata dalle classi presenti nel package `it.hourglass.myTalk.client.presenter.UserListPresenter`.
- **Attività svolte e dati trattati:** La view ha solo funzionalità di presentazione all'utente.

`Classe client.view.PopupInformation`

- **Tipo, obiettivo, funzione del componente:** Questa classe rappresenta il popup che mostra una qualsiasi notifica del sistema.
- **Relazioni d'uso con altre componenti:** Questa classe non utilizza alcuna classe. Un istanza di questa classe verrà creata dalle classi presenti nel package `it.hourglass.myTalk.client.presenter` e dalla classe `it.hourglass.myTalk.client.communication.WSConnectionn`.
- **Attività svolte e dati trattati:** La view ha solo funzionalità di presentazione all'utente.

4.2.4 Package it.hourglass.myTalk.client.view.contactprofile

Questo sottopackage di `it.hourglass.myTalk.client.view` contiene i sottowidget che verranno inseriti nella view `ContactProfileView`. Si è scelto di creare questo sottopackage con le relative classi per limitare la complessità totale della view `ContactProfileView`.

`Classe client.view.ContactProfile.AvatarContactProfileView`

- **Tipo, obiettivo, funzione del componente:** Questa classe rappresenta il widget che permette all'utente di visualizzare l'avatar del profilo di un utente amico.
- **Relazioni d'uso con altre componenti:** Un istanza di questa classe verrà aggiunta dentro la classe ContactProfileView.
- **Attività svolte e dati trattati:** La view ha solo funzionalità di presentazione all'utente. Questa classe conterrà tutti gli elementi grafici necessari alla visualizzazione dell'avatar di un amico.

`Classe client.view.ContactProfile.PersonalDataContactProfileView`

- **Tipo, obiettivo, funzione del componente:** Questa classe rappresenta il widget che permette all'utente di visualizzare i dati personali di un utente amico (quali nome ,cognome, data di nascita, indirizzo e-mail e sesso).
- **Relazioni d'uso con altre componenti:** Un istanza di questa classe verrà aggiunta dentro la classe ContactProfileView.
- **Attività svolte e dati trattati:** La view ha solo funzionalità di presentazione all'utente.

`Classe client.view.ContactProfile.ExtendedDataContactProfileView`

- **Tipo, obiettivo, funzione del componente:** Questa classe rappresenta il widget che permette all'utente di visualizzare le informazioni estese di un utente amico (quali indirizzo e-mail alternativo, interessi, ispirazioni e la descrizione dell'utente).
- **Relazioni d'uso con altre componenti:** Un istanza di questa classe verrà aggiunta dentro la classe ContactProfileView.
- **Attività svolte e dati trattati:** La view ha solo funzionalità di presentazione all'utente.

4.2.5 Package it.hourglass.myTalk.client.view.profilemanagment

Questo sottopackage di `it.hourglass.myTalk.client.view` contiene i sottowidget che verranno inseriti nella view `EditProfileView`. Si è scelto di creare questo sottopackage con le relative classi per limitare la complessità totale della view `EditProfileView`.

`Classe client.view.profileManagment.AvatarProfileManagementView`

- **Tipo, obiettivo, funzione del componente:** Questa classe rappresenta il widget che permette all'utente di visualizzare l'avatar del profilo del proprio profilo. Deve essere inoltre possibile richiedere la modifica dell'avatar stesso.
- **Relazioni d'uso con altre componenti:** Un istanza di questa classe verrà aggiunta dentro la classe EditProfileView.
  Questa classe implementa i metodi presenti nell'interfaccia `it.hourglass.myTalk.client.presenter.display.AvatarProfileManagementDisplay`. Tali metodi rappresentano le richieste che il `it.hourglass.myTalk.client.presenter.profilemanagment.AvatarProfileManagementPresenter` può richiedere alla classe `AvatarContactProfileView`.
- **Attività svolte e dati trattati:** La view ha solo funzionalità di presentazione all'utente. Questa classe conterrà tutti gli elementi grafici necessari alla visualizzazione del proprio avatar e per la richiesta di cambiare l'avatar.

`Classe client.view.profileManagment.PersonalDataProfileManagementView`

- **Tipo, obiettivo, funzione del componente:** Questa classe rappresenta il widget che permette all'utente di visualizzare propri i dati personali e di richiedere la modifica degli stessi.
- **Relazioni d'uso con altre componenti:** Un istanza di questa classe verrà aggiunta dentro la classe EditProfileView.
  Questa classe implementa i metodi presenti nell'interfaccia `it.hourglass.myTalk.client.presenter.display.PersonalDataProfileManagementDisplay`. Tali metodi rappresentano le richieste che il `it.hourglass.myTalk.client.presenter.profilemanagment.PersonalDataProfileManagementPresenter` può richiedere alla classe `PersonalDataProfileManagementView`.
- **Attività svolte e dati trattati:** La view ha solo funzionalità di presentazione all'utente. Questa classe conterrà tutti gli elementi grafici necessari alla visualizzazione e la modifica dei propri dati personali.

`Classe client.view.profileManagment.ExtendedDataProfileManagementView`

- **Tipo, obiettivo, funzione del componente:** Questa classe rappresenta il widget che permette all'utente di visualizzare propri i dati estesi e di richiedere la modifica degli stessi.
- **Relazioni d'uso con altre componenti:** Un istanza di questa classe verrà aggiunta dentro la classe EditProfileView.
  Questa classe implementa i metodi presenti nell'interfaccia `it.hourglass.myTalk.client.presenter.display.ExtendedDataProfileManagementDisplay`. Tali metodi rappresentano le richieste che il `it.hourglass.myTalk.client.presenter.profilemanagment.ExtendedDataProfileManagementPresenter` può richiedere alla classe `ExtendedDataProfileManagmentView`.
- **Attività svolte e dati trattati:** La view ha solo funzionalità di presentazione all'utente. Questa classe conterrà tutti gli elementi grafici necessari alla visualizzazione e la modifica dei propri dati personali.

`Classe client.view.profileManagment.NewAvatarPopupView`

- **Tipo, obiettivo, funzione del componente:** Questa classe rappresenta popup che apparirà all'utente quando richiede il cambio di avatar del proprio profilo.
- **Relazioni d'uso con altre componenti:** Un istanza di questa classe verrà creata da :
  `it.hourglass.myTalk.client.presenter.profilemanagment.AvatarProfileManagmentPresenter` in risposta alla richiesta di cambio avatar. Questa classe implementa i metodi presenti nell'interfaccia `it.hourglass.myTalk.client.presenter.display.NewAvatarPopupDisplay`. Tali metodi rappresentano le richieste che il `it.hourglass.myTalk.client.presenter.profilemanagment.NewAvatarPopupPresenter` può richiedere alla classe `NewAvatarPopupView`.
- **Attività svolte e dati trattati:** La view ha solo funzionalità di presentazione all'utente. Questa classe conterrà tutti gli elementi grafici necessari alla modifica dell'avatar del proprio profilo.

`Classe client.view.profileManagment.NewEmailConfirmPopupView`

- **Tipo, obiettivo, funzione del componente:** Questa classe rappresenta il popup che apparirà all'utente in risposta alla richiesta di cambio e-mail.
- **Relazioni d'uso con altre componenti:** Un istanza di questa classe verrà creata da :

`it.hourglass.myTalk.client.presenter.profilemanagment.PersonalDataProfilePresenter` in risposta alla richiesta di cambio e-mail. Questa classe implementa i metodi presenti nell'interfaccia `it.hourglass.myTalk.client.presenter.display.NewEmailConfirmPopupDisplay`. Tali metodi rappresentano le richieste che il

`it.hourglass.myTalk.client.presenter.profilemanagment.NewEmailConfirmPopupPresenter` può richiedere alla classe `NewEmailConfirmPopupView`.

- **Attività svolte e dati trattati:** La view ha solo funzionalità di presentazione all'utente. Questa classe conterrà tutti gli elementi grafici necessari alla conferma del cambio dell'e-mail.

4.2.6 Package it.hourglass.myTalk.client.presenter

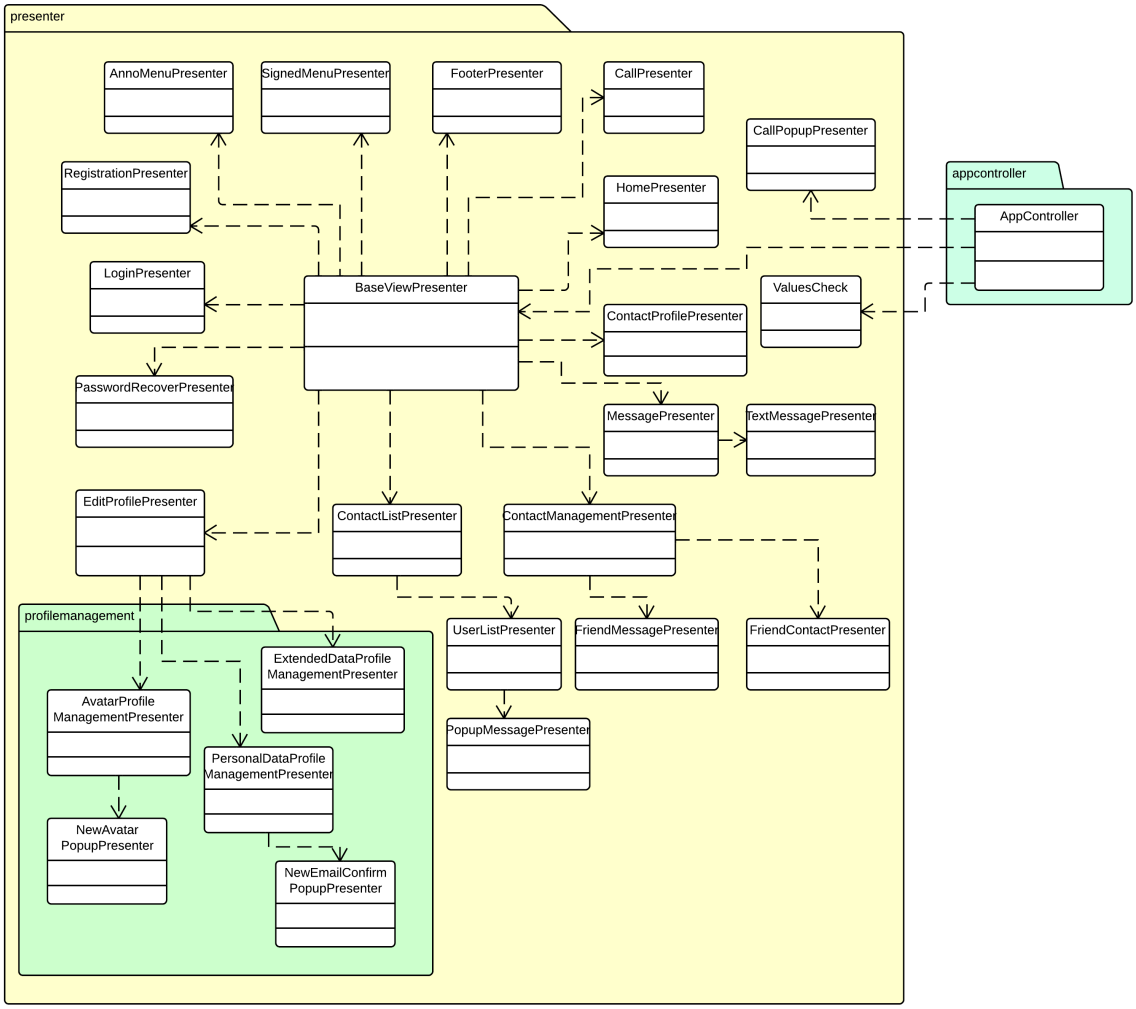

Figura 4: Package it.hourglass.myTalk.client.Presenter

- **Tipo, obiettivo, funzione del componente :** questo package costituisce l'insieme delle classi che si occupano di definire il comportamento di ogni pagina del sistema. Ogni classe conterrà un'istanza della classe `it.hourglass.client.myTalk.presenter.display.***Display` (ove *** identifica l'interfaccia Display corrispettiva al presenter ***) la quale rappresenta la view a cui la classe si riferisce. Questa interfaccia contiene tutti i metodi che il presenter può richiedere alla view.
- **Relazioni d'uso di altre componenti :** All'inizializzazione del sistema viene creata un'istanza della classe

`it.hourglass.myTalk.client.presenter.BaseViewPresenter`. Tale presenter si occupa di fungere da presenter base al sistema. In questo presenter saranno istanziati gli altri presenter specifici di ogni view.
Le classi contenute in questo package utilizzano le classi contenute nei package: `it.hourglass.myTalk.client` e `it.hourglass.myTalk.server` . Per ogni presenter sarà specificato quale classe utilizzerà in seguito.

- **Attività svolte e dati trattati:** specificati nelle componenti che seguono.

`Interfaccia client.presenter.WidgetPresenter`

- **Tipo, obiettivo, funzione del componente:** Questa interfaccia rappresenta l'interfaccia comune a tutti i presenter (tranne al presenter `BaseViewPresenter`.
- **Relazioni d'uso con altre componenti:** nessuna.
- **Attività svolte e dati trattati:** Fornisce un'interfaccia comune a tutti i presenter.

`Classe client.presenter.BaseViewPresenter`

- **Tipo, obiettivo, funzione del componente:** Questa classe rappresenta il presenter base del sistema. Al suo interno saranno presenti quattro presenter che conterranno il presenter che corrisponde al il menu,il corpo,il footer e alla lista utenti. Inoltre questa classe si occuperà di controllare se è presente una sessione attiva. In caso positivo si occuperà di visualizzare automaticamente la view di un utente autenticato.
- **Relazioni d'uso con altre componenti:** Questa classe è utilizzata dalla classe `it.hourglass.myTalk.client.appcontroller.AppController` per richiedere lo switch view. Quando è richiesto uno switch view BaseViewPresenter si occuperà di eseguire il cambio di presenter corrente in seguito alla richiesta. La classe utilizza tutte le classi presenti nel package
`it.hourglass.myTalk.client.presenter`. Per il controllo della sessione attiva la classe utilizza i metodi esposti dall'interfaccia
`it.hourglass.myTalk.client.rpcservice.RPCService` Questa classe utilizza l'interfaccia `BaseDisplay` implementata alla classe `view.BaseView`.
- **Attività svolte e dati trattati:** Questa classe è creata all'inizializzazione del sistema. Si occupa di eseguire lo swich view del sistema. Inoltre il controllo di sessione avviene in questa classe.

`Classe client.presenter.AnnoMenuPresenter`

- **Tipo, obiettivo, funzione del componente:** Questa classe rappresenta il presenter associato al menù di un utente non autenticato.

- **Relazioni d'uso con altre componenti:** Questa classe utilizza l'interfaccia `AnnoMenuDisplay` implementata alla classe `view.AnnoMenuView`. Questa classe utilizza alcuni eventi contenuti nel package `it.hourglass.myTalk.client.event`. La classe viene istanziata dalla classe `it.hourglass.myTalk.client.presenter.BaseViewPresenter`
- **Attività svolte e dati trattati:** Questa classe si occuperà di scatenare gli eventi necessari allo switch view per le pagine a cui un utente non autenticato può accedere.

`Classe client.presenter.SignedMenuPresenter`

- **Tipo, obiettivo, funzione del componente:** Questa classe rappresenta il presenter associato al menù di un utente autenticato.
- **Relazioni d'uso con altre componenti:** Questa classe utilizza l'interfaccia `SignedMenuDisplay` implementata alla classe `view.SignedMenuView`. Questa classe utilizza alcuni eventi contenuti nel package `it.hourglass.myTalk.client.event`. La classe viene istanziata dalla classe `it.hourglass.myTalk.client.presenter.BaseViewPresenter`.
- **Attività svolte e dati trattati:** Questa classe si occuperà di scatenare gli eventi necessari allo switch view per le pagine a cui un utente autenticato può accedere.

`Classe client.presenter.HomePresenter`

- **Tipo, obiettivo, funzione del componente:** Questa classe rappresenta il presenter della classe HomeView.
- **Relazioni d'uso con altre componenti:** Questa classe utilizza l'interfaccia `HomeDisplay` implementata alla classe `view.HomeView`.
  La classe viene istanziata dalla classe `it.hourglass.myTalk.client.presenter.BaseViewPresenter`.
- **Attività svolte e dati trattati:** Questa classe non gestisce alcuna logica specifica.

`client.Presenter.CallPresenter`

- **Tipo, obiettivo, funzione del componente :** Questa classe rappresenta il presenter associato alla view di chiamata. In particolare questo presenter si occupa di gestire tutti gli aspetti della funzione di chiamata.
- **Relazioni con altre componenti:** Questa classe utilizza la classe `it.hourglass.myTalk.client.communication.WSConnectionn` per la creazione della chiamata.
  La classe viene istanziata dalla classe `it.hourglass.myTalk.client.presenter.BaseViewPresenter`.
- **Attività svolte e dati trattati:** Questa classe si occupa di gestire le chiamate in ingresso ed in uscita attraverso le librerie webRTC tramite l'utilizzo della classe `it.hourglass.myTalk.client.communication.WSConnectionn`.

`Classe client.presenter.LoginPresenter`

- **Tipo, obiettivo, funzione del componente:** Il compito di questa classe è di gestire la logica necessaria a richiedere l'autenticazione nel sistema da parte di un utente.
- **Relazioni d'uso con altre componenti:** Questa classe utilizza l'interfaccia `LoginDisplay` implementata alla classe `view.LoginView` che rappresenta le azioni che il presenter può richiedere alla view. La classe viene istanziata dalla classe `it.hourglass.myTalk.client.presenter.BaseViewPresenter`. Per il controllo di autenticazione la classe utilizza i metodi esposti dall'interfaccia `it.hourglass.myTalk.client.rpcservice.RPCService` . In caso di autenticazione riuscita la classe scatenerà un evento definito nel package `it.hourglass.myTalk.client.event`.
- **Attività svolte e dati trattati:** La classe si occupa di effettuare il controllo di autenticazione di un utente. In caso positivo scatenerà un evento per richiedere lo switch della view per un utente autenticato e il recupero del profilo. In caso negativo si occuperà di segnalarlo (tramite la view associata).

`Classe client.presenter.RegistretionPresenter`

- **Tipo, obiettivo, funzione del componente:** Questa classe si occupa di gestire la logica necessaria alla registrazione di un utente nel sistema. La classe si occuperà anche di controllare la validità dei dati prima di inserirli nel database.
- **Relazioni d'uso con altre componenti:** Questa classe usa l'interfaccia `RegistrationDisplay` implementata alla classe `view.RegistationView` che rappresenta le azioni che il presenter può richiedere alla view. Per il controllo dei dati si utilizzerà la classe `it.hourglass.myTalk.client.presenter.CheckValues`. Per la registrazione la classe utilizza i metodi esposti dall'interfaccia `it.hourglass.myTalk.client.rpcservice.RPCService` La classe viene istanziata dalla classe `it.hourglass.myTalk.client.presenter.BaseViewPresenter`.
- **Attività svolte e dati trattati:** La classe si occupa di gestire la logica necessaria alla registrazione nel sistema. Si occupa del controllo dei dati e in caso negativo lo segnalerà all'utente attraverso la view altrimenti si occuperà di registrare il nuovo utente nel database.

`Classe client.presenter.PasswordRecoverPresenter`

- **Tipo, obiettivo, funzione del componente:** Questa classe si occupa di gestire la logica necessaria al recupero della password di un utente che non la ricorda.
- **Relazioni d'uso con altre componenti:** Questa classe usa l'interfaccia `PasswordRecoverDisplay` implementata alla classe `view.PasswordRecoverView` che rappresenta le azioni che il presenter può richiedere alla view. Per il recupero della password la classe utilizza i metodi esposti dall'interfaccia `it.hourglass.myTalk.client.rpcservice.RPCService`.

(NOTA: per l'invio dell'email contenente il codice di verifica si utilizzerà la classe `it.hourglass.myTalk.client.presenter.client.presenter.it.hourglass.myTalk.server.model.businesslogic.EmailSender`. La classe viene istanziata dalla classe `it.hourglass.myTalk.client.presenter.BaseViewPresenter`.

- **Attività svolte e dati trattati:** La classe si occupa di gestire la logica necessaria al recupero della password di un utente attraverso l'invio di un codice di verifica. La classe si occuperà di controllare il codice e in caso positivo permetter di modificare la password.

`Classe client.presenter.FooterPresenter`

- **Tipo, obiettivo, funzione del componente:** Questa classe si occupa di gestire la logica per la visualizzazione del identificativo unico dell'utente nel footer.
- **Relazioni d'uso con altre componenti:** Questa classe usa l'interfaccia `FooterDisplay` implementata alla classe `view.FooterView` che rappresenta le azioni che il presenter può richiedere alla view. La classe utilizza la classe `it.hourglass.myTalk.client.shared.Profile` per recuperare l'identificativo dell'utente. La classe viene istanziata dalla classe `it.hourglass.myTalk.client.presenter.BaseViewPresenter`.
- **Attivita svolte e dati trattati:** La classe si occupa di gestire la visualizzazione dell'identificativo unico e di aggiornare la view quand'esso viene cambiato.

`Classe client.presenter.EditProfilePresenter`

- **Tipo, obiettivo, funzione del componente:** Per diminuire la complessità del presenter che si occupa di gestire la visualizzazione e modifica del profilo si è deciso di progettare delle classi più piccole che si occupano del controllo di: visualizzazione e modifica dell'avatar, dei dati personali e dei dati estesi. Queste classi sono contenute nel sottopackage: `it.hourglass.myTalk.client.presenter.profilemanagment`. Questa classe si occuperà dell'inizializzazione delle classi di questo sottopackage.
- **Relazioni d'uso con altre componenti:** Questa classe usa l'interfaccia `EditProfileDisplay` implementata alla classe `view.EditProfileView` che rappresenta le azioni che il presenter può richiedere alla view. Per il recupero del profilo la classe utilizza la classe : `it.hourglass.myTalk.client.Profile`. La classe viene istanziata dalla classe `it.hourglass.myTalk.client.presenter.BaseViewPresenter`.
- **Attivita svolte e dati trattati:** La classe si occuperà della sola inizializzazione delle classi del sottopackage `ProfileManagment`.

`Classe client.presenter.ContactProfilePresenter`

- **Tipo, obiettivo, funzione del componente:** Questa classe si occupa di gestire la logica di recupero del profilo di un amico e la visualizzazione dei dati dello stesso.
- **Relazioni d'uso con altre componenti:** Questa classe usa l'interfaccia `ContactProgileDisplay` implementata alla classe `view.ContactProfileView` che rappresenta le azioni che il presenter può richiedere alla view. Per il recupero del profilo dell'amico la classe utilizza i metodi esposti dall'interfaccia `it.hourglass.myTalk.client.rpcservice.RPCService` . La classe viene istanziata dalla classe `it.hourglass.myTalk.client.presenter.BaseViewPresenter`.
- **Attivita svolte e dati trattati:** La classe si occupa di gestire la logica del recpuero del profilo di un amico. Una volta che viene recuperato sarà sempre la stessa classe a inizializzare i campi dati la view.

`Classe client.presenter.ContactListPresenter`

- **Tipo, obiettivo, funzione del componente:** Questa classe si occupa di gestire la logica per il controllo della lista amici. Per ogni amico recuperato dalla lista amici del profilo verrà inizializzata una istanza di `UserListPresenter`.
- **Relazioni d'uso con altre componenti:** Questa classe usa l'interfaccia `ContactListDisplay` implementata alla classe `view.ContactListView` che rappresenta le azioni che il presenter può richiedere alla view. La classe utilizza la classe `it.hourglass.myTalk.client.Profile` per accedere alla lista amici, inoltre utilizzerà la classe `it.hourglass.myTalk.client.UserListPresenter` per il controllo delle azioni su un particolare amico.
  La classe viene istanziata dalla classe `it.hourglass.myTalk.client.presenter.BaseViewPresenter`.
- **Attivita svolte e dati trattati:** La classe si occuperà di gestire la logica di controllo della lista amici. Dovrà automatizzare la creazione della stessa e rendere possibile all'utente l'accesso alla pagina di gestione delle amicizie (ContactManagment).

`Classe client.presenter.UserListPresenter`

- **Tipo, obiettivo, funzione del componente:** Questa classe si occupa di gestire la logica riguardo le operazioni rapide da poter eseguire su un utente nella lista amici (quali richiedere chiamata rapida, richiedere la visione profilo e richiesta di di mandare un messaggio).
- **Relazioni d'uso con altre componenti:** Questa classe usa l'interfaccia `UserListDisplay` implementata alla classe `view.User` che rappresenta le azioni che il presenter può richiedere alla view.
  La classe viene istanziata dalla classe `it.hourglass.myTalk.client.presenter.`

`ContactListPresenter`.
La classe fa uso della classe `it.hourglass.myTalk.client.PopupMessagePresenter` per richiedere il popup per l'invio di un messaggio.

- **Attivita svolte e dati trattati:** La classe si occupa della gestione della logica per richiedere una chiamata rapida, richiedere la visualizzazione del profilo o la richiesta del popup per mandare un messaggio verso un utente specifico. Per ogni amico nella lista amici sarà presente una istanza di questa classe.

`Classe client.presenter.ContactManagmentePresenter`

- **Tipo, obiettivo, funzione del componente:** Questa classe si occupa di gestire la logica della pagina di gestione delle amicizie. Questa classe si occupa di visualizzare le amicizie (con la possibilità di eliminarne) e di visualizzare le richieste di amicizia.
- **Relazioni d'uso con altre componenti:** Questa classe usa l'interfaccia `ContactManagementDisplay` implementata alla classe `view.ContactMangmenteView` che rappresenta le azioni che il presenter può richiedere alla view. Per ogni amico presente nella lista la classe inizializza una istanza della classe `it.hourglass.myTalk.client.FriendContactPresenter`, mentre per ogni messaggio di amicizia la classe inizializza un'istanza della classe `it.hourglass.myTalk.client.FriendMessagePresenter`. La classe viene istanziata dalla classe `it.hourglass.myTalk.client.presenter.BaseViewPresenter`.
- **Attivita svolte e dati trattati:** La classe si occupa di gestire la logica della pagina di gestione di amicizie di un utente. La classe automatizza la creazione delle istanze di `FriendContactPresenter` per ogni amico presente nella lista utenti e `FriendMessagePresenter` per ogni messaggio di amicizia ricevuto.

`Classe client.presenter.FriendMessagePresenter`

- **Tipo, obiettivo, funzione del componente:** Questa classe si occupa di gestire la logica di risposta a una richiesta di amicizia nella pagina di gestione amicizie.
- **Relazioni d'uso con altre componenti:** Questa classe usa l'interfaccia `FriednMessageDisplay` implementata alla classe `view.FriendMessageView` che rappresenta le azioni che il presenter può richiedere alla view. La classe viene istanziata dalla classe `it.hourglass.myTalk.client.presenter.ContactManagmentePresenter`. Per la risposta al messaggio la classe utilizza i metodi esposti dall'interfaccia `it.hourglass.myTalk.client.rpcservice.RPCService`
- **Attivita svolte e dati trattati:** La classe si occupa di gestire la logica di risposta a una richiesta di amicizia, occupandosi di notificare la risposta al database. Per ogni amico nella lista utente `ContactManagmentePresenter` si occupa di creare un'istanza di questa classe.

`Classe client.presenter.FriendContactPresenter`

- **Tipo, obiettivo, funzione del componente:** La classe si occupa di gestire la logica per eliminare un'amicizia di un utente.
- **Relazioni d'uso con altre componenti:** Questa classe usa l'interfaccia `FriendContactDisplay` implementata alla classe `view.FriendContactView` che rappresenta le azioni che il presenter può richiedere alla view. Per eliminare un'amicizia la classe utilizza i metodi esposti dall'interfaccia `it.hourglass.myTalk.client.rpcservice.RPCService`.
- **Attivita svolte e dati trattati:** La classe si occupa di gestire la logica per eliminare un'amicizia dalla lista amici di un utente. Per ogni amico nella lista utente `ContactManagmentePresenter` si occupa di creare un'istanza di questa classe.

`Classe client.presenter.CallPopupPresenter`

- **Tipo, obiettivo, funzione del componente:** Questa classe si occupa di gestire la logica del popup di ricezione di una chiamata.
- **Relazioni d'uso con altre componenti:** Questa classe usa l'interfaccia `CallPopupDisplay` implementata alla classe `view.CallPopup` che rappresenta le azioni che il presenter può richiedere alla view. La classe viene istanziata dalla classe `it.hourglass.myTalk.client.appcontroller.AppController`.
- **Attivita svolte e dati trattati:** La classe si occupa di gestire la logica del popup di ricezione di una chiamata. Esso si occuperà di accettare, rifiutare o ignorare una chiamata.

`Classe client.presenter.PopupMessagePresenter`

- **Tipo, obiettivo, funzione del componente:** Questa classe si occupa di gestire la logica del popup per l'invio di un messaggio ad un utente.
- **Relazioni d'uso con altre componenti:** Questa classe usa l'interfaccia `PopupMessageDisplay` implementata alla classe `view.PopupMessageView` che rappresenta le azioni che il presenter può richiedere alla view. Per inviare un messaggio la classe utilizza i metodi esposti dall'interfaccia `it.hourglass.myTalk.client.rpcservice.RPCService`
- **Attivita svolte e dati trattati:** La classe si occupa di gestire la logica per l'invio di un messaggio ad un utente. La classe si occuperà di inviare il messaggio al database.

4.2.7 Package it.hourglass.myTalk.client.presenter.profilemanagment

Questo sottopackage di `it.hourglass.myTalk.presneter` contiene i sottowidget che verranno inseriti nella view `EditProfilePresenter`. Si è scelto di creare questo sottopackage con le relative classi per limitare la complessità totale della view `EditProfilePresenter`.

`Classe client.presenter.profilemanagment.AvatarProfileManagementPresenter`

- **Tipo, obiettivo, funzione del componente:** Questa classe si occupa di gestire la logica per la richiesta di modifica dell'avatar di un utente
- **Relazioni d'uso con altre componenti:** Questa classe usa l'interfaccia `AvatarProfileManagementDi` implementata alla classe `view.profilemanagment.AvatarProfileManagementView` che rappresenta le azioni che il presenter può richiedere alla view. La classe, quando viene richiesta la modifica dell'avatar, utilizza la la classe `it.hourglass.myTalk.client.presenter.profilemanagment.NewAvatarPopupPresenter`. La classe viene istanziata dalla classe `it.hourglass.myTalk.client.presenter.EditContactPresenter`.
- **Attivita svolte e dati trattati:** La classe si occupa di gestire la logica per il cambio di avatar di un utente. Se viene richiesto la classe si occupa di inizializzare un popup `NewAvatarPopupPresenter`.

`Classe client.presenter.profilemanagment.NewAvatarPopupPresenter`

- **Tipo, obiettivo, funzione del componente:** Questa classe si occupa di gestire la logica del popup utilizzato per gestire la logica di cambio dell'avatar di un utente.
- **Relazioni d'uso con altre componenti:** Questa classe usa l'interfaccia `NewAvatarPopupDisplay` implementata alla classe `view.profilemanagment.NewAvatarPopupView` che rappresenta le azioni che il presenter può richiedere alla view. Per la richiesta di cambio di avatar la classe utilizza i metodi esposti dall'interfaccia `it.hourglass.myTalk.client.rpcservice.RPCService`. La classe viene istanziata dalla classe `it.hourglass.myTalk.client.presenter.profilemanagment.AvatarContactProfilePresenter`.
- **Attivita svolte e dati trattati:** Questo classe gestisce la logica del popup per la richiesta di cambio avatar di un utente. In caso di richiesta sarà questa classe a salvare la modifica nel database.

`Classe client.presenter.profilemanagment.PersonalDataProfilePresenter`

- **Tipo, obiettivo, funzione del componente:** Questa classe si occupa di gestire la logica per le modifiche ai dati personali di un utente.

- **Relazioni d'uso con altre componenti:** Questa classe usa l'interfaccia `PersonalDataProfileDispla` implementata alla classe `view.profilemanagment.PersonalDataProfileView` che rappresenta le azioni che il presenter può richiedere alla view. Per la modifica dei dati personali la classe utilizza i metodi esposti dall'interfaccia `it.hourglass.myTalk.client.rpcservice.RPCService` (NOTA per inviare la richiesta di conferma per la modifica dell'email si utilizza la classe `it.hourglass.myTalk.server.model.businesslogic.EmailSender.`). La classe viene istanziata dalla classe `it.hourglass.myTalk.client.presenter.ProfileManagmentPresenter`.
- **Attivita svolte e dati trattati:** La classe si occupa di gestire la logica per la richiesta di modifica dei dati personali di un utente. In caso di richiesta di modifica sarà questa classe a salvare le modifiche nel database. In caso di richiesta di modifica dell'indirizzo email di un utente verrà utilizzata la classe `NewEmailConfirmPopupPresenter`.

`Classe client.presenter.profilemanagment.NewEmailConfirmPopupPresenter`

- **Tipo, obiettivo, funzione del componente:** Questa classe si occupa di gestire la logica per il popup utilizzato per il controllo del codice di conferma per la richiesta di cambio email di un utente.
- **Relazioni d'uso con altre componenti:** Questa classe usa l'interfaccia `NewEmailConfirmPopupDispl` implementata alla classe `view.profilemanagment.NewEmailConfirmPopupView` che rappresenta le azioni che il presenter può richiedere alla view. Per il controllo del codice di conferma e per la modifica dell'email la classe utilizza i metodi esposti dall'interfaccia `it.hourglass.myTalk.client.rpcservice.RPCService` La classe viene istanziata dalla classe `it.hourglass.myTalk.client.presenter.ProfileManagmentPresenter`.
- **Attivita svolte e dati trattati:** La classe si occupa di gestire la logica per il controllo del codice di conferma per la richiesta di cambio email. Se il codice controllato viene confermato sarà questa classe a salvare la modifica dell'email nel database.

`Classe client.presenter.profilemanagment.ExtendedDataProfileManagementPresenter`

- **Tipo, obiettivo, funzione del componente:** Questa classe si occupa di gestire la logica per le modifiche ai dati estesi di un utente.
- **Relazioni d'uso con altre componenti:** Questa classe usa l'interfaccia `ExtendedDataProfileManage` implementata alla classe `view.profilemanagment.ExtendedDataProfileManagementView` che rappresenta le azioni che il presenter può richiedere alla view. Per la modifica dei dati la classe utilizza i metodi esposti dall'interfaccia `it.hourglass.myTalk.client.rpcservice.RPCService` La classe viene istanziata dalla classe `it.hourglass.myTalk.client.presenter.ProfileManagmentPresenter`.

- **Attivita svolte e dati trattati:** La classe si occupa di gestire la logica per la richiesta di modifica dei dati estesi di un utente. In caso di richiesta di modifica sarà questa classe a salvare le modifiche nel database.

4.2.8 Package it.hourglass.myTalk.client.presenter.display

- **Tipo, obiettivo, funzione del componente:** Questo package contiene le definizioni delle interfacce `Display` le quali espongono i metodi che un presenter può richiedere alla sua corrispettiva view.
- **Relazioni d'uso con altre componenti:** Le interfacce presenti in questo package sono utilizzate dalle classe presenti nel package `it.hourglass.myTalk.client.presenter`.
- **Attività svolte e dati trattati:** Queste interfacce espongono i metodi che un presenter può richiedere alla sua corrispettiva view.

`Interfaccia ***Display` *Per rendere più leggibile il documento si è deciso di non descrivere tutte le interfacce* `Display` *presenti nell'architettura. Ogni presenter avrà un'interfaccia* `Display` *associata.*

- **Tipo, obiettivo, funzione del componente:** Questa interfaccia espone i metodi che il presenter `it.hourglass.myTalk.client.presenter.***Presenter` può richiedere alla corrispettiva view.
- **Relazioni d'uso con altre componenti:** Questa interfaccia viene implementata da: `it.hourglass.myTalk.client.view.***View` e che il presenter `it.hourglass.myTalk.client.presenter.***Presenter` può richiedere alla view.
- **Attività svolte e dati trattati:** Espone i metodi che il presenter `it.hourglass.myTalk.client.presenter.***Presneter` può richiedere alla sua corrispettiva view.

4.2.9 Package it.hourglass.myTalk.client.event

- **Tipo, obiettivo, funzione del componente:** Questo package contiene le definizioni degli eventi che il sistema deve poter riconoscere. Ogni evento è composto da una classe e da una relativa interfaccia.
  Qui viene rappresentato lo schema generale di un evento. Gli eventi specifici saranno presentati nel documento *Definizione di Prodotto_ v1.0.pdf*
- **Relazioni d'uso con altre componenti:** Ogni evento viene gestito dall'eventHandler presente in `it.hourglass.myTalk.client.appcontroller.AppController`.
- **Attività svolte e dati trattati:** L'interfaccia si occupa di indicare qual'è l'azione che il sistema deve compiere relativa a tale evento mentre la classe è utilizzata come marcatore dell'evento (cioè l'interfaccia associa a ogni classe un TYPE specifico ad uno specifico evento).

`Interfaccia client.event.***EventHandler`

- **Tipo, obiettivo, funzione del componente:** Questa interfaccia espone l'operazione associata ad un particolare evento.
- **Relazioni d'uso con altre componenti:** Questa interfaccia viene implementata da: `it.hourglass.myTalk.client.appcontroller.AppController`.
- **Attività svolte e dati trattati:** Espone all'AppController l'operazione associata ad un particolare evento.

`Classe client.event.***Event`

- **Tipo, obiettivo, funzione del componente:** Questa classe ha la funzione di fungere da marcatore per un evento.
- **Relazioni d'uso con altre componenti:** Viene utilizzata da: `it.hourglass.myTalk.client.appcontroller.AppController` per marcare un evento.
- **Attività svolte e dati trattati:** Alla classe viene associato un TYPE attraverso il quale l'EventHandelr presente in `it.hourglass.myTalk.client.appcontroller.AppController` saprà riconoscere il tipo di evento.

4.2.10 Package it.hourglass.myTalk.client.shared

- **Tipo, obiettivo, funzione del componente:** Questo package contiene la definizione degli oggetti comuni tra il client e il server.
- **Relazioni d'uso con altre componenti:** Questo package contiene la classe `Profile` la quale è utilizzata da tutte le classi che devono recuperare informazioni sul profilo dell'utente. Il package sarà utilizzato inoltre dalle classi del package `it.hourglass.myTalk.server.model` per il collegamento con il database (sia per l'invio dei dati che per la richiesta).
- **Attività svolte e dati trattati:** Specificati nelle componenti che seguono.

`Classe client.shared.Profile`

- **Tipo, obiettivo, funzione del componente:** Questa classe è implementa il design patter *Singleton*. Questa classe è utilizzata per poter accedere ad una unica istanza di `User` e, una volta che l'utente si sia autenticato, saranno salvati in questa la lista messaggi e la lista amici. Ad essa saranno anche demandate alcune chiamate asincrone al server, al fine di permettere la persistenza di alcuni cambiamenti effettuati alle liste di amicizia o di messaggi, o ancora ai dati personali dell'utente.
- **Relazioni d'uso con altre componenti:** La classe non utilizza nessuna classe mentre è utilizzata da tutte le classi che necessitano di recuperare dati dal profilo di un utente. Viene dunque utilizzata dalle classi presenti nel package `client.presenter`. Nello specifico viene utilizzato dalle classi:
  `client.appcontroller.AppController` la quale si occupa di inizializzarlo e di aggiornarlo in caso di autenticazione e de-autenticazione,
  `client.presenter.EditContactPresenter` per recuperare le informazioni del proprio profilo,
  `client.presenter.ContactListPresenter` per recuperare la lista amici,
  `client.presenter.ContactManagementPresenter` per recuperare la lista amici e i messaggi di richiesta di amicizia e infine dalla classe
  `client.presenter.MessagePresenter` per il recupero dei messaggi.
- **Attività svolte e dati trattati:** La classe implementa il design pattern *Singleton*. Ha la funzione di raggruppare tutte le informazioni del profilo di un utente quali: dati del profilo (User), lista amici e la lista messaggi. Utilizzando il design pattern Singleton si può assicurare un accesso unico e ovunque raggiungibile in qualunque punto del programma.

`Classe client.shared.User`

- **Tipo, obiettivo, funzione del componente:** Questa classe rappresenta l'oggetto che contiene tutte le informazioni personali (avatar, dati personali e dati estesi) di un utente.

- **Relazioni d'uso con altre componenti:** L'oggetto `User` è utilizzato in qualsiasi sezione richieda l'utilizzo anche parziale delle informazioni legate ad un utente (sia in lettura che scrittura). Nella fattispecie, si fa riferimento a questo tipo di oggetto in multiple classi dei package `client.view`, `client.presenter`, `server.model.businesslogic`, `server.model.dao`.
- **Attivita svolte e dati trattati:** La classe rappresenta l'oggetto contenente tutte le informazioni personali di un utente. Questa classe viene usata per visualizzare e modificare il proprio profilo e per visualizzare il profilo di un altro utente.

`Classe client.shared.Friendships`

- **Tipo, obiettivo, funzione del componente:** Questo oggetto rappresenta la richiesta di un'amicizia da inviare al database.
- **Relazioni d'uso con altre componenti:** Una istanza di questa classe viene creata e spedita al database dalla classe `client.presenter.ContactManagementPresenter` quando si richiede un'amicizia ad un utente.
- **Attivita svolte e dati trattati:** La classe è utilizzata per rappresentare una richiesta di amicizia da parte di un utente. Nel momento in cui la richiesta relativa viene accettata, il record generato nel *database* su questo oggetto rappresenterà il rapporto di amicizia stesso.

`Classe client.shared.SignIn`

- **Tipo, obiettivo, funzione del componente:** Oggetto scambiato tra *database* e client nel momento dell'autenticazione. Esso contiene tutti dati rilevanti legati ad un utente.
- **Relazioni d'uso con altre componenti:** Questa classe viene utilizzata dalla classe `it.hourglass.myTalk.client.appcontroller` in seguito all'autenticazione di un utente.
- **Attività svolte e dati trattati:** Viene utilizzato a seguito delle operazioni di autenticazione, al fine di inizializzare la classe `Profile` con i valori corretti e relativi all'account dell'utente. E' composto da un oggetto `User`, una lista amicizie e una lista di oggetti `Message`.

4.2.11 Package it.hourglass.myTalk.client.rpcservice

- **Tipo, obiettivo, funzione del componente:** Questo package contiene la definizione dei metodi RPC (offerti dal framework GWT). Le chiamate RPC (Remote Procedure Call) sono chiamate asincrone utilizzate dal client per aver accesso al server.
- **Relazioni d'uso con altre componenti:** Questo package contiene la definizione di tutti i metodi RPC. Questi metodi saranno poi implementati in `it.hourglass.myTalk.client.communication.it.hourglass.myTalk.server.rpcservice.RPCServiceImpl`
- **Attività svolte e dati trattati:** Specificati nelle componenti che seguono.

`Interfaccia client.rpcservice.RPCService`

- **Tipo, obiettivo, funzione del componente:** Questa interfaccia definisce le chiamate RPC che il sistema può richiedere.
- **Relazioni d'uso con altre componenti:** Questi metodi saranno poi implementati in `it.hourglass.myTalk.client.communication.it.hourglass.myTalk.server.rpcservice.RPCServiceImpl`.
- **Attività svolte e dati trattati:** Espone i metodi RPC che il sistema può richiedere che andranno successivamente implementati da `RPCServiceImpl`.

`Interfaccia client.rpcservice.RPCServiceAsync`

- **Tipo, obiettivo, funzione del componente:** Questa interfaccia definisce il tipo di `AsyncCallback` che è associata ad una *RPC*. Ogni *RPC* per funzionare deve avere una `AsyncCallback` associata che definisce il tipo di oggetto che verrà ritornato in seguito alla chiamata.
- **Relazioni d'uso con altre componenti:** La `AsyncCallback` verrà definita per ogni metodo nella classe che ne fa uso.
- **Attivita svolte e dati trattati:** Espone il tipo di `AsyncCallback` richiesto dalla RPC.

4.2.12 Package it.hourglass.myTalk.client.wrappers

- **Tipo, obiettivo, funzione del componente:** Questo package contiene la definizione delle classi wrapper delle varie funzioni di webRTC.
- **Relazioni d'uso con altre componenti:** Viene utilizzato dalle classi che utilizzano le librerie RTC.
- **Attività svolte e dati trattati:** Specificati nelle componenti che seguono.

`Class it.hourglass.myTalk.client.wrappers.ConsoleLog`

- **Tipo, obiettivo, funzione del componente:** Questa classe è il wrap dell'oggetto console Javascript ,e delle sue funzioni.
- **Relazioni d'uso con altre componenti:** Questa classe verrà utilizzata dalle componenti che vorranno visualizzare una stringa sulla console del browser.
- **Attivita svolte e dati trattati:** Permette di stampare stringhe nella console del browser.

`Class it.hourglass.myTalk.client.wrappers.MediaStream`

- **Tipo, obiettivo, funzione del componente:** Questa classe è il wrap per rappresentare lo stream in Javascript .
- **Relazioni d'uso con altre componenti:** Questa classe verrà utilizzata dalle componenti che vorranno utilizzare lo stream audio/video, come la classe `it.hourglass.myTalk.client.communication.Call` o le classi wrapper .
- **Attivita svolte e dati trattati:** Specificata nei componenti che seguono.

`Class it.hourglass.myTalk.client.wrappers.GetUserMediaUtils`

- **Tipo, obiettivo, funzione del componente:** Questa classe è il wrap delle funzioni di webRTC preposte alla cattura dello stream audio/video della periferica di acquisizione dell'utente .
- **Relazioni d'uso con altre componenti:** Questa classe verrà utilizzata dalle componenti che vorranno avere accesso allo stream dell'utente, in particolare la classe `it.hourglass.myTalk.client.communication.Call` .
- **Attivita svolte e dati trattati:** Viene chiesto all'utente l'autorizzazione per avere accesso allo stream. Una volta concessa, viene restituito un riferimento allo stream di tipo MediaStream.

`Class it.hourglass.myTalk.client.wrappers.PeerConnectionWrapper`

- **Tipo, obiettivo, funzione del componente:** Questa classe è il wrap delle funzioni di webRTC preposte alla gestione della connessione tra utenti, lo scambio dei loro stream multimediali e la gestione dello stream di dati, utilizzato nel nostro caso per la chat .
- **Relazioni d'uso con altre componenti:** Questa classe verrà utilizzata dalle componenti che vorranno instaurare una connessione con un altro utente , in particolare la classe `it.hourglass.myTalk.client.communication.Call` .
- **Attivita svolte e dati trattati:** Gestisce tutte le possibili situazioni in una chiamata. Un elenco delle principali funzioni in seguito:
  - Inizio chiamata
  - Fine chiamata
  - Scambio di informazioni
  - Canale di chat

`Class it.hourglass.myTalk.client.wrappers.RTCConfiguration`

- **Tipo, obiettivo, funzione del componente:** Questa classe è il wrap delle funzioni di webRTC preposte alla gestione della connessione con il server ICE.
- **Relazioni d'uso con altre componenti:** Questa classe verrà utilizzata dalle componenti che vorranno instaurare una connessione al server ICE , in particolare la classe `it.hourglass.myTalk.client.communication.Call` .
- **Attivita svolte e dati trattati:** Si occupa di instaurare una connessione al server ICE, ritornando la risposta di quest'ultimo.

`Interfaccia it.hourglass.myTalk.client.wrappers.PeerConnectionCallbacks`

- **Tipo, obiettivo, funzione del componente:** Questa interfaccia definisce le funzioni che dovranno essere implementate per effettuare una chiamata.
- **Relazioni d'uso con altre componenti:** Questi metodi saranno poi implementati in `it.hourglass.myTalk.model.client.communication.Call`.
- **Attivita svolte e dati trattati:** Espone i metodi che verranno utilizzati per la chiamata ed andranno andranno successivamente implementati da `Call`.

`Class it.hourglass.myTalk.client.wrappers.RTCSessionDescription`

- **Tipo, obiettivo, funzione del componente:** Questa classe è il wrap del valore dell'SDP di ciascun client.
- **Relazioni d'uso con altre componenti:** Questa classe verrà utilizzata dalle componenti che vorranno trattare l'SDP, come la classe `it.hourglass.myTalk.client.communication.Call` e le classi appartenenti a wrappers .
- **Attivita svolte e dati trattati:** Si occupa di rappresentare l'SDP dei vai user.

`Class it.hourglass.myTalk.client.wrappers.mozRTCSessionDescription`

- **Tipo, obiettivo, funzione del componente:** Questa classe è il wrap del valore dell'SDP di ciascun client. Utilizzato su browser Firefox.
- **Relazioni d'uso con altre componenti:** Questa classe verrà utilizzata dalle componenti che vorranno trattare l'SDP, come la classe `it.hourglass.myTalk.client.communication.Call` e le classi appartenenti a wrappers .
- **Attivita svolte e dati trattati:** Si occupa di rappresentare l'SDP dei vai user.

`Interfaccia it.hourglass.myTalk.client.wrappers.SDPCreateOfferCallback`

- **Tipo, obiettivo, funzione del componente:** Questa classe è wrap delle funzioni Javascript utilizzate per la creazione delle offerte in base al tipo di stream scelto, da mandare poi all'utente ricevente della telefonata.
- **Relazioni d'uso con altre componenti:** Questa classe viene utilizzata dalla classe `it.hourglass.myTalk.model.client.Call`.
- **Attivita svolte e dati trattati:** Viene creato il pacchetto di offerta o di risposta in base agli stream scelti. `Call`.

4.3 Server

4.4 Package it.hourglass.myTalk.server

- **Tipo, obiettivo, funzione del componente:** Il package `Server` contiene tutti i sottopackage e le classi che compongono la parte server del sistema. Esso sarà quindi suddiviso in due principali aree di funzionalità: il prelievo dei dati, la loro elaborazione e persistenza (Model) da una parte, la logica dello svolgimento, del controllo e dell'esecuzione delle chiamate dall'altra (WSServer).
- **Relazioni d'uso con altre componenti:** I componenti del package sono pensati per avere comportamenti e spazio d'esistenza indipendenti dal resto dell'applicativo. Tuttavia, esistono alcune dipendenze con classi presenti nel macro-package client (ad esempio per quanto concerne i tipi di dati trattati, le cui definizioni delle classi sono mantenute nel package `it.hourglass.myTalk.client.shared` che verranno di volta in volta specificate nei casi particolari più avanti.
- **Attività svolte e dati trattati:** Si faccia riferimento a descrizioni più dettagliate nelle seguenti sezioni.

4.4.1 Package it.hourglass.myTalk.server.model

- **Tipo, obiettivo, funzione del componente:** Package contenente le componenti destinate al prelievo, elaborazione, persistenza dei dati utilizzati dall'applicativo.
- **Relazioni d'uso con altre componenti:** Si veda in dettaglio nelle sezioni successive.
- **Attività svolte e dati trattati:** Si faccia riferimento a descrizioni più dettagliate nelle sezioni successive.

4.4.2 Package it.hourglass.myTalk.server.model.businesslogic

- **Tipo, obiettivo, funzione del componente:** Contiene classi demandate all'elaborazione delle tipologie di dati inviate dal Client e da questo richieste. Ciò prevede funzionalità per la gestione di chiamate remote in grado di eseguire comandi attingendo dalla base di dati, di criptazione delle informazioni, di comunicazione tra server e singolo Utente per mezzo di automatizzazione di invio di email.
- **Relazioni d'uso con altre componenti:** Utilizza i metodi esposti dall'interfaccia `server.model.dao.DAO` per il prelievo effettivo dei dati dalla banca di dati, nonché la loro persistenza. Fornisce l'implementazione di metodi remoti specificati nel package `it.hourglass.myTalk.client.rpcservice`. Utilizza tipi di oggetti comuni al *Client*, le cui definizioni possono essere trovate in `it.hourglass.myTalk.client.shared`
- **Attività svolte e dati trattati:** Specificati in dettaglio nelle componenti che seguono.

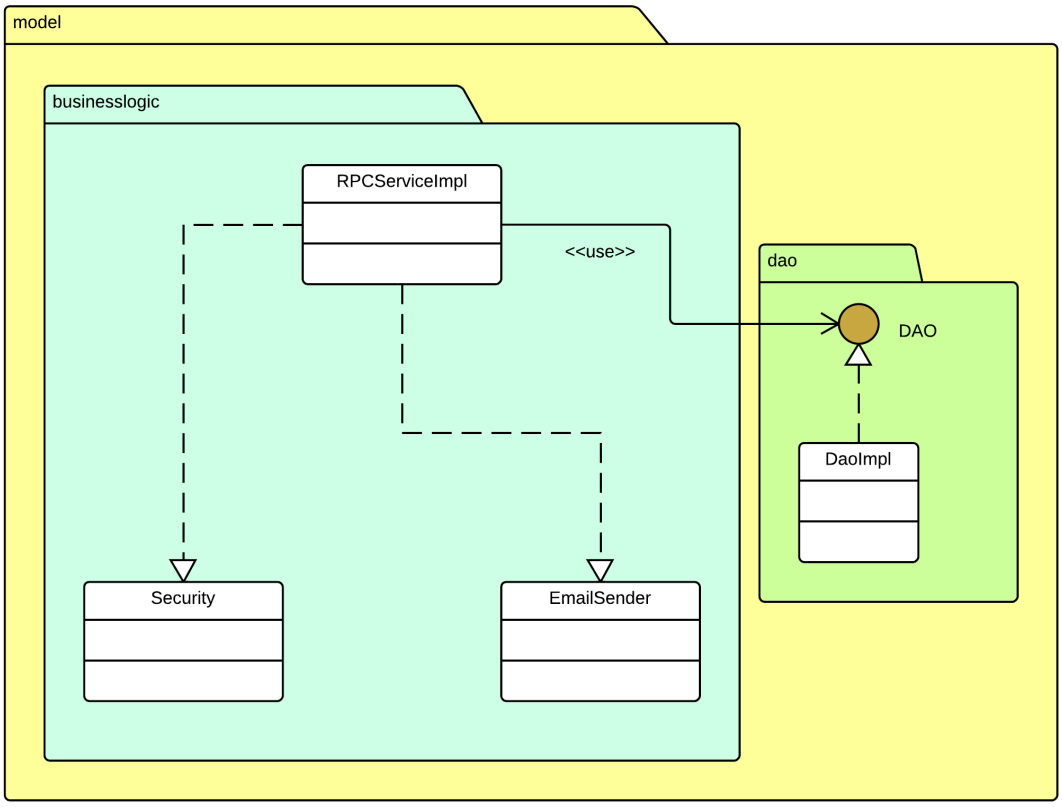

Figura 5: Package it.hourglass.myTalk.server.model

`Classe server.model.businesslogic.RPCServiceImpl`

- **Tipo, obiettivo, funzione del componente:** Costituisce una *servlet*, rendendo possibile l'effettuazione di chiamate remote asincrone da parte del *Client*. Utilizzando classi di supporto contenute nel medesimo package, effettua le elaborazioni richieste sui dati prima di restituirli o persisterli nel database.
- **Relazioni d'uso con altre componenti:** Utilizza metodi della componente `Security` per la criptazione/decriptazione di dati sensibili, della componente `EmailSender` per l'invio di email all'Utente, necessarie all'utilizzo di alcune funzionalità, di `server.model.dao.DAO` per prelievo e salvataggio di dati dal *database*. I suoi metodi sono poi implementazioni delle definizioni presenti nell'interfaccia `it.hourglass.myTalk.client.rpcservice.RPCService`.
- **Attività svolte e dati trattati:** Offre metodi utilizzabili tramite chiamate remote asincrone finalizzate all'elaborazione degli input inviati dalle componenti *client-side* dell'applicativo.

`Classe server.model.businesslogic.EmailSender`

- **Tipo, obiettivo, funzione del componente:** Consente l'elaborazione e l'invio di messaggi di posta rivolti all'Utente finalizzati prevalentemente alla verifica di codici d'autorizzazione.
- **Relazioni d'uso con altre componenti:** E' utilizzata da `RPCServiceImpl`.
- **Attività svolte e dati trattati:** Colleziona i dati necessari alla compilazione della mail desiderata e ne consente l'invio.

`Classe server.model.businesslogic.Security`

- **Tipo, obiettivo, funzione del componente:** Svolge operazioni legate alla sicurezza dei dati, nella fattispecie conversione di dati sensibili in controparti criptate e viceversa.
- **Relazioni d'uso con altre componenti:** E' utilizzata da `RPCServiceImpl`.
- **Attività svolte e dati trattati:** Riceve in input stringhe, ne effettua criptazioni e decriptazioni, controlla corrispondenze.

4.4.3 Package it.hourglass.myTalk.server.model.dao

- **Tipo, obiettivo, funzione del componente:** Questo package contiene le classi che implementano il design pattern DAO. Le classi conterranno tutti i metodi utili per accedere al database, così come eventuali file di configurazione o di supporto alle operazioni eseguite.
- **Relazioni d'uso con altre componenti:** Le classi definite in questo package sono utilizzate da tutte le classi che necessitano dell'accesso al database.
- **Attività svolte e dati trattati:** Specificati nelle componenti che seguono.

`Interfaccia server.model.dao.DAO`

- **Tipo, obiettivo, funzione del componente:** Questa interfaccia espone i metodi invocabili dal sistema per aver accesso al database. Vengono descritte delle operazioni fondamentali utili a portare a termine i compiti delle funzionalità dell'applicativo e che rimarranno teoricamente immutabili indipendentemente dal tipo di strumenti utilizzati per l'accesso al *database*. Starà poi alle controparti implementative definire gli effettivi metodi o strumenti che andranno a realizzare le operazioni.
- **Relazioni d'uso con altre componenti:** Questa classe verrà implementata dalla classe `DAOImpl`.
- **Attivita svolte e dati trattati:** Questa interfaccia ha il solo compito di esporre i metodi indicanti le operazioni che si richiede possano essere eseguite sul *database*.

`Classe server.model.dao.DAOImpl`

- **Tipo, obiettivo, funzione del componente:** Questa classe implementa i metodi esposti dall'interfaccia `DAO`. Essa si occupa di definire la logica dei metodi per accedere al *database*. E' indifferente che tipo di strumenti verranno effettivamente utilizzati per consentire le operazioni di accesso al *database*, purché i metodi dell'implementazione accettino i parametri e ritornino i tipi d'oggetto indicati dall'interfaccia generale.
  Questa classe rappresenta l'implementazione del design pattern DAO. Inoltre la classe implementerà anche il design pattern Singleton per assicurare che gli accessi al database da un unico punto d'accesso
- **Relazioni d'uso con altre componenti:** La classe è indirettamente utilizzata da tutte le classi che richiedono accesso al database, ma ciascun accesso ai dati richiesto da queste è regolato dalla componente `server.model.businesslogic.RPCServiceImpl`.
- **Attività svolte e dati trattati:** La classe è utilizzata per fornire un'implementazione effettiva di tutti i possibili metodi per l'accesso al database. La classe si occuperà dunque del collegamento col database, di effettuare le query necessarie e ritornare i valori alle classi che lo hanno richiesto. I tipi di dato che essa si troverà a manipolare e a ritornare al client sono prevalentemente di tipo `client.shared.Friendships`, `client.shared.Message`, `client.shared.User`.

4.4.4 Package it.hourglass.myTalk.server.WSServer

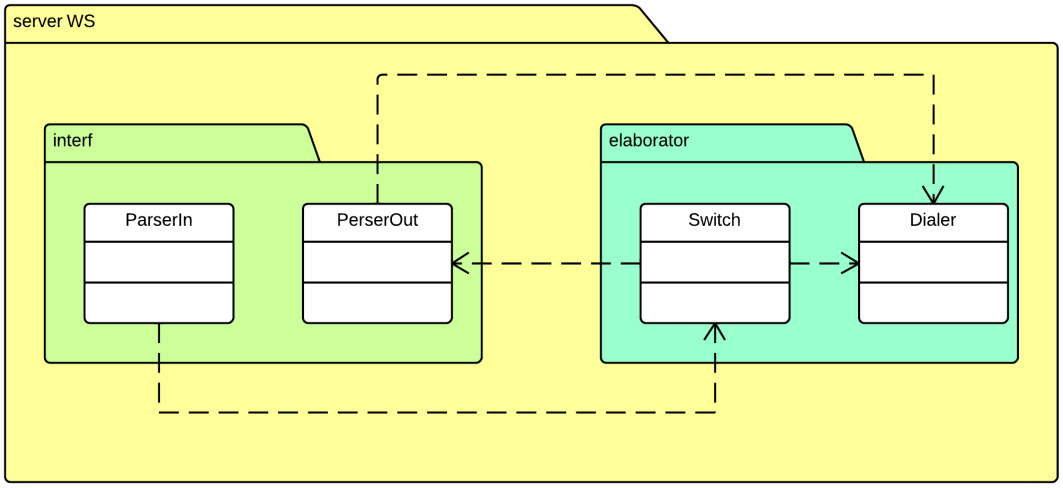

Figura 6: Package it.hourglass.myTalk.server.WSServer

- **Tipo, obiettivo, funzione del componente :** il package `Server.WSServer` ha il compito di elaborare e smistare i JSON provenienti dai vari client.
- **Relazioni con altre componenti :** Questo package non ha alcuna relazione con i package esterni. Qualsiasi interazione avviene attraverso lo scambio di messaggi di tipo JSON.
- **Attività svolte e dati trattati :** Questo package si occupa di gestire i vari JSON in entrata ed in uscita . I JSON in entrata possono scatenare varie elaborazioni che verranno discusse in seguito.

`Class Server.WSServer.interf.parseIn`

- **Tipo, obiettivo, funzione del componente :** la classe ha lo scopo di aprire un WebSocket e di interpretare correttamente i JSON in ingresso e successivamente inoltrare le informazioni alla classe `Server.WSServer.elaboration.Switch` per l'elaborazione.
- **Relazioni con altre componenti :** Questa classe crea un istanza della classe `Server.WSServer.elaboration.Switch`. Non ha altre relazioni con classi esterne.
- **Attività svolte e dati trattati :** Si occupa principalmente di aprire il WebSocket per ricevere le informazioni dal client, e di gestirne i vari eventi. Le informazioni in arrivo vengono elaborate e mandate alla classe di elaborazione.

`Class Server.WSServer.interf.parseOut`

- **Tipo, obiettivo, funzione del componente :** la classe ha lo scopo di comporre dei JSON che andranno poi spedite al client.
- **Relazioni con altre componenti :** Questa classe viene utilizzata dalla classe `Server.WSServer.elaboration.Dialer` e dal package `Server.WSServer.exception`.
- **Attività svolte e dati trattati :** Si compone di metodi statici che inviano o messaggi interpretabili come JSON o come messaggi informativi al client.

`Class Server.WSServer.elaborator.Switch`

- **Tipo, obiettivo, funzione del componente :** la classe ha lo scopo di avviare la corretta elaborazione in base al tipo di JSON arrivato utilizzando un metodo opportuno di `Server.WSServer.elaboration.Dialer`.
- **Relazioni con altre componenti :** Questa classe crea un istanza di `Server.WSServer.elaboration.Dialered` è istanziata da `Server.WSServer.interf.ParseIn`.
- **Attività svolte e dati trattati :** La classe ha il compito di richiamare un metodo di `Server.WSServer.elaboration.Dialer` in base al tipo di JSON arrivato in ingresso.

`Class Server.WSServer.elaborator.Dialer`

- **Tipo, obiettivo, funzione del componente :** la classe ha lo scopo fate l'elaborazione richiesta, restituendo in output un JSON da inviare al client.
- **Relazioni con altre componenti :** Questa classe è istanziata dalla classe `Server.WSServer.elaboration.Dialer`
- **Attività svolte e dati trattati :** La classe contiene un registro dei client connessi e per ogni client ne conserva la lista dei contatti, in modo da notificare i cambiamenti di stato di quest'ultimi e di fare le varie elaborazioni.

`Package Server.WSServer.exception`

- **Tipo, obiettivo, funzione del componente :** Il package contiene i vari tipi di eccezioni del server.
- **Relazioni con altre componenti :** Questa package è utilizzato da tutti i sotto-package di `Server.WSServer` .
- **Attività svolte e dati trattati :** Contiene le definizioni delle varie eccezioni.

5 Design Pattern

In questa sezione verranno esposti *design pattern* utilizzati nella progettazione.

5.1 MVP

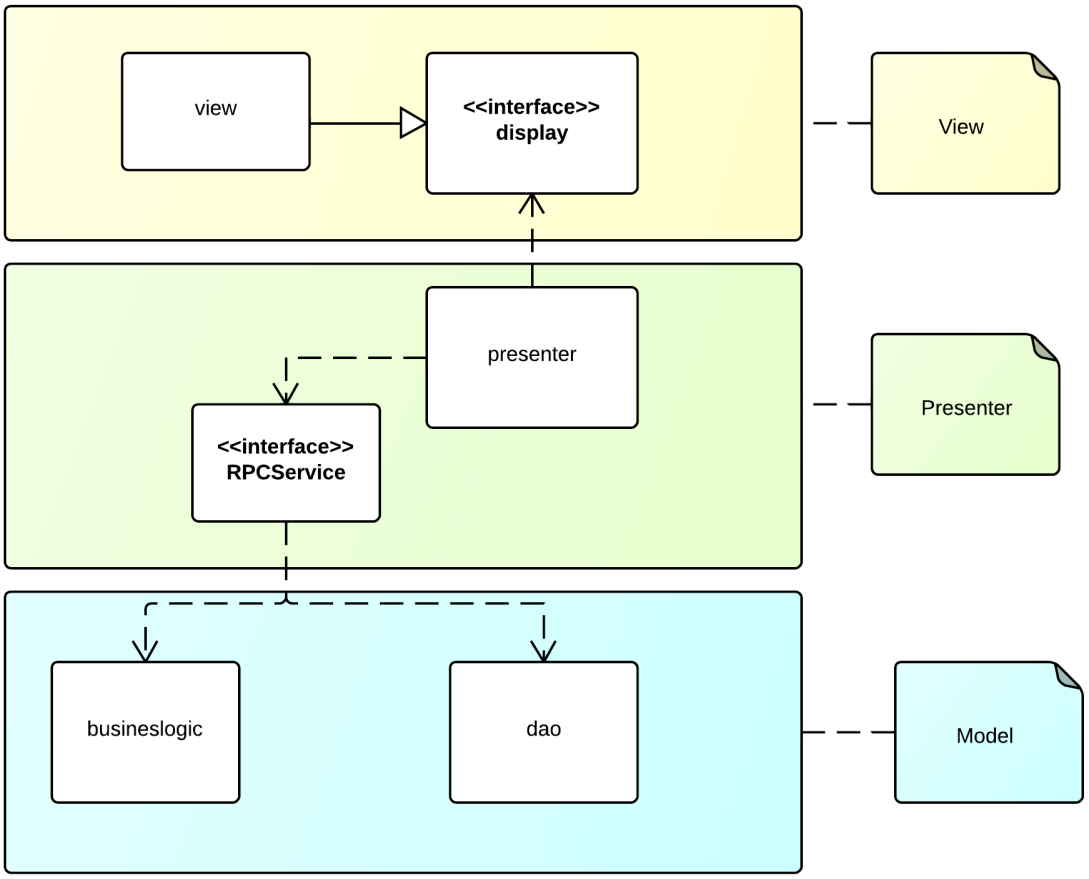

Figura 7: Design Pattern MVPC

Descrizione : questo pattern architetturale permette la divisione tra le funzionalità della vista e del Modello dell'applicazione.
Il pattern architetturale consigliato utilizzando il framework GWT è il pattern MVP. Nella progettazione dell'architettura si è seguita l'applicazione consigliata di tale Pattern fornita dal team di GWT[1].

[1]Fonte: http://www.gwtproject.org/articles/mvp-architecture.html

- Model: Rappresenta il modulo che contiene tutta la business logic del sistema. Il model è costituito da tutto il contenuto del pacakge `it.hourglass.myTalk.server` (che si occupa del recupero e memorizzazione dei dati nel database)e dal packageit.hourglass.myTalk.client.shared (che contiene gli oggetti comuni a client e server).
  Il model è collegato con i presenter tramite i metodi definiti nell'interfaccia `it.hourglass.myTalk.client,rpcservice.RPCService`. Questi metodi permettono ai presenter il recupero dei dati dal model.
- View: Rappresenta il modulo che contiene tutta l'interfaccia grafica del sistema. Si occupa di visualizzare le informazioni all'utente (nelle pagine in cui è richiesto) e permette l'interazione con l'utente. La view è costituita dalle classi contenute nel package: `it.hourglass.myTalk.client.view`. La view ha solo compito di presentazione cioè non si occupa di gestire la logica di controllo dell'informazione o dell'elaborazione dei dati. Si noti che ogni view è legata ad uno specifico presenter tramite l'implementazione di una particolare interfaccia `Display` contenuta nel package
  `it.hourglass.myTalk.client.presenter.display`.
- Presenter: Rappresenta il modulo che si occupa di ricevere le richieste di un utente tramite l'interazione con una view e di farle eseguire (elaborando i dati recuperati). Così facendo, si limita il compito della view alla sola visualizzazione, eliminando dunque qualsiasi logica di elaborazione.
  Il presenter è costituito dalle classi contenute nel package:
  `it.hourglass.myTalk.client.presenter`. Ogni presenter è legato ad una particolare view (si esegue cioè il *bind*) tramite l'utilizzo di una particolare interfaccia `Display`. Ogni view implementa una particolare interfaccia la quale viene utilizzata da una specifica classe presenter. Questa interfaccia espone tutti i metodi che il presenter può richiedere alla view gestendo quindi logica (la quale è definita nel presenter) di ogni possibile operazione che l'utente può richiedere limitando quindi la view a solo il compito di presentazione.
  Il collegamento con il model invece avviene tramite i metodi definiti nell'interfaccia
  `it.hourglass.myTalk.client,rpcservice.RPCService` attraverso i quali il presenter richiede e recupera le informazioni necessarie dal model.

Applicazione : L'intera architettura è strutturata secondo tale design pattern.
L'input dell'utente, catturato dalla view, viene elaborato dal presenter. Ogni richiesta fatta dal sistema viene eseguita attraverso un presenter. In questo modo si ha una completa separazione tra la Business Logic e la view.
Si noti che come consigliato dagli sviluppatori di GWT è stata progettata la classe AppController la quale si occupa di gestire l'Application Logic che non fa parte delle caratteristiche dei presenter. In particolare l'AppController si occupa di gestire lo *switch-view* e il recupero e il salvataggio dei dati del server. Si noti però che l'App-

Controller tuttavia esegue azioni solamente in risposta ad una richiesta effettuata da un presenter (tramite il meccanismo degli eventi fornito da GWT).

Motivazione : L'applicazione di tale pattern è necessario per ottenere un sistema con minimo grado di accoppiamento. Inoltre implementando correttamente il pattern si ottiene un sistema facilmente mantenibile ed estendibile in quanto se è necessario cambiare un elemento del sistema (per esempio la GUI, il sistema di memorizzazione dati ecc.) non è necessario cambiare tutto il sistema ma basterà modificare la sezione specifica del corrispettivo modulo.

5.2 Singleton

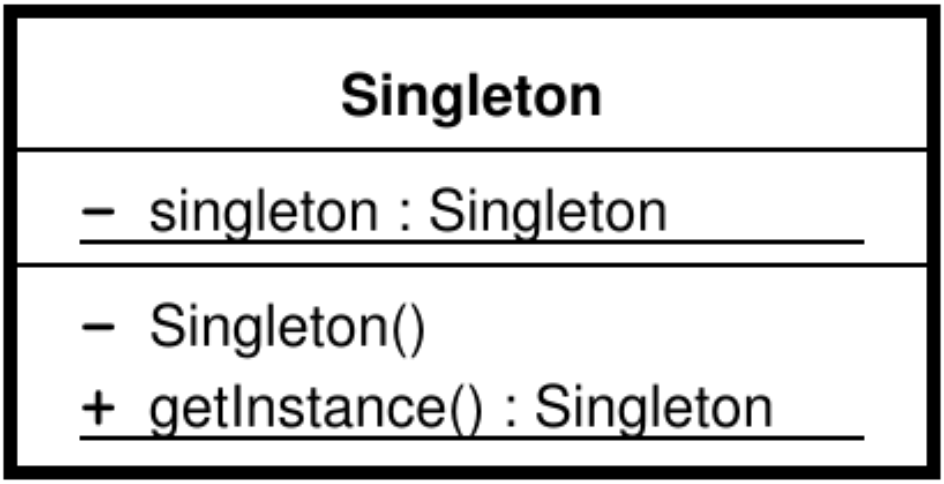

Figura 8: Design Pattern - DAO

Descrizione: questo design pattern creazionale assicura che venga creata una sola istanza di una determinata classe fornendo così un punto di accesso globale a essa. Per implementare questo pattern si deve creare una apposita classe che definisca un costruttore, unico, privato che assicuri l'impossibilità di istanziazione diretta dalle classe e un metodo statico che ritorni l'istanza della classe.

Motivazione : questo pattern è essenziale per la gestione del profilo locale del client. Utilizzando il pattern infatti mi assicuro che il profilo locale venga creato una sola volta e ottengo un metodo di recupero dell'istanza del profilo accessibile in qualunque punto del sistema.

Applicazione : tale design pattern sarà implementato dalla classe Profile contenuta nel package:
`it.hourglass.myTalk.client.profile`. Verrà dunque implementato un metodo per il recupero del profilo.
Il pattern viene inoltre applicato nella classe `it.hourlgass.myTalk.server.dao.DAOImpl` per assicurare che gli accessi al database da parte delle servlet siano sempre possibile da un unico punto d'accesso.

5.3 DAO

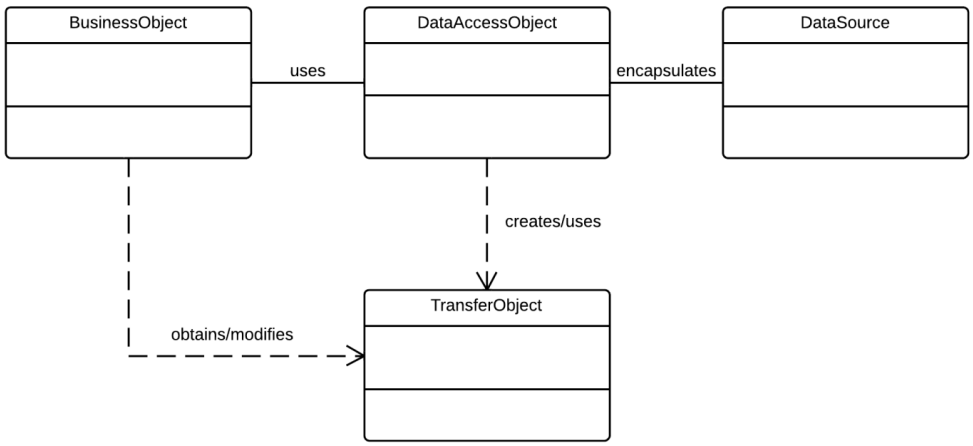

Figura 9: Design Pattern - DAO

Descrizione : questo design pattern consente di ottenere un'interfaccia di accesso ad un database mettendo a disposizione operazioni specifiche e nascondendo i dettagli implementativi del database.

Motivazione : questo design pattern è utilizzato per rendere i dettagli implementativi della base di dati indipendenti dall'architettura. In questo modo eventuali modifiche strutturali alla base di dati non implicheranno modifiche anche alla logica dell'applicazione, assicurando una maggiore manutenibilità ed estendibilità del codice.

Applicazione : tale design pattern sarà implementato dalle classi contenute nel package `it.hourglass.myTalk.server.dao`.

6 Diagrammi delle attività

I diagrammi delle attività vengono utilizzati per descrivere la logica procedurale di una particolare attività. Nel seguente capitolo sono usati per rappresentare le operazioni principali che il sistema deve permettere di svolgere. Per crearli sono stati seguiti i casi d'uso descritti sul documento *Analisi dei Requisiti_v2.0*, mostrando il flusso logico di alcune attività. Di seguito saranno descritte le seguenti attività: Registrazione, Autenticazione, Attuazione e Ricezione di una chiamata. Per una maggior comprensibilità su chi esegue una determinata attività, i diagrammi saranno partizionati.

6.1 Flusso generale

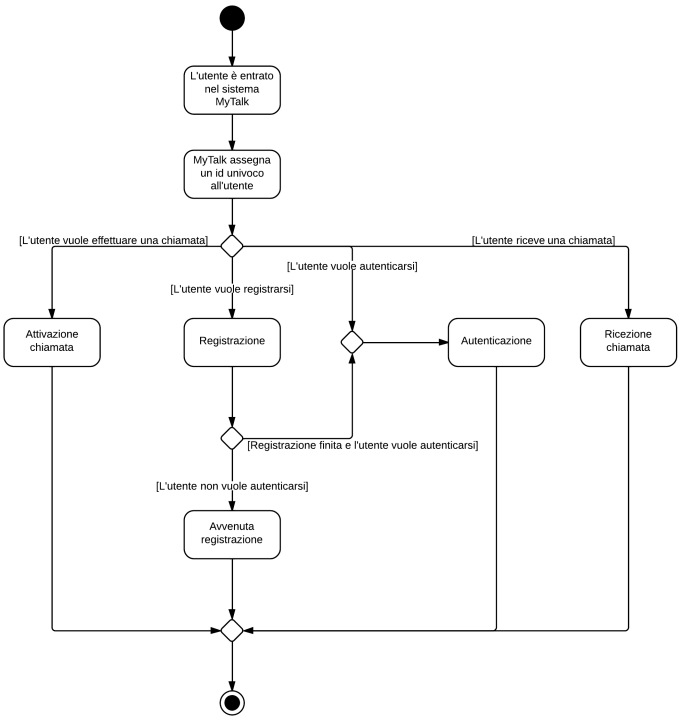

Figura 10: Flusso generale delle attività dall'entrata del sistema fino alla sua uscita.

Descrizione : Il diagramma rappresentato elenca le possibili attività che un Utente può effettuare all'interno del sistema **MyTalk**. All'Utente che accede al sistema **MyTalk** gli viene, prima di tutto, assegnato un id univoco dal sistema. Successivamente se l'Utente non ha ancora un profilo personale si può registrare e, se vuole, autenticarsi. L'Utente che si autentica interagisce col sistema come Utente Autenticato oppure come Amministratore. Un Utente può comunque effettuare una chiamata verso gli altri utenti del sistema, oppure riceverla, senza bisogno di essere registrato. Una volta fatta una di queste operazioni l'utente potrà scegliere se rimanere nel sistema oppure uscirvi.

6.2 Registrazione

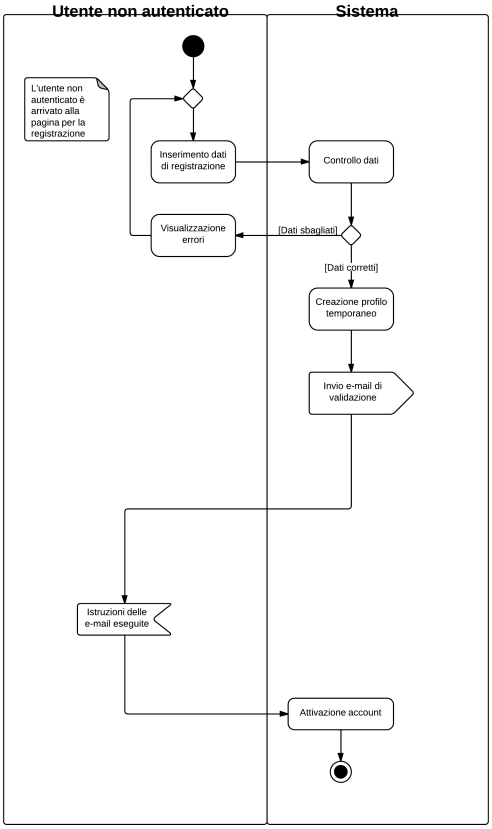

Figura 11: Diagramma delle attività di registrazione nel sistema MyTalk.

Descrizione: Il diagramma rappresentato elenca i passi dell'attività di registrazione nel sistema **MyTalk**. Quando l'Utente accede alla pagina di registrazione gli verrà chiesto di inserire i propri dati personali: Username, password e-mail, nome, cognome, sesso e data di nascita da associare al nuovo account. Il sistema controllerà la correttezza dei dati sollevando eventuali errori che avvisando l'Utente del problema richiedendogli un nuovo tentativo di registrazione. Se i dati inseriti sono corretti il sistema crea un profilo temporaneo e invierà all'Utente una e-mail alla e-mail inserita. Se vengono eseguite le istruzioni specificate al suo interno si procederà all'attivazione dell'account. L'Utente avrà così un proprio profilo per potersi autenticare come Utente Autenticato.

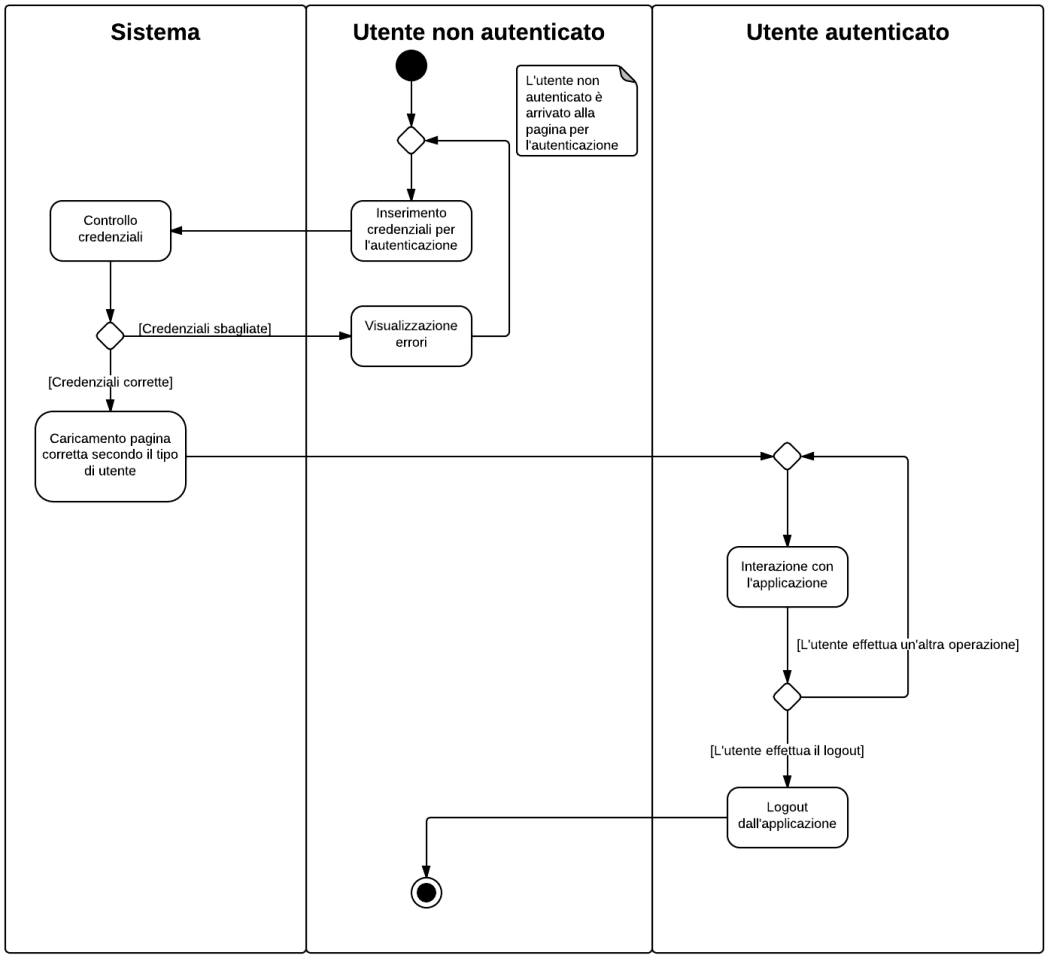

Figura 12: Diagramma delle attività di autenticazione nel sistema MyTalk.

6.3 Autenticazione

Descrizione : Il diagramma rappresentato elenca i passi dell'attività di autenticazione nel sistema **MyTalk**. Quando l'Utente accede alla pagina di autenticazione dovrà inserire il proprio username e la password oppure prelevare le proprie credenziali da account esistenti di altre piattaforme (Facebook, Google+). Se i dati inseriti sono errati allora il sistema avvisa l'Utente dell'errore permettendogli il reinserimento di username e password altrimenti se l'autenticazione è andata a buon fine **MyTalk** fa accedere l'Utente (a seconda del suo ruolo) come Utente Autenticato o come Amministratore . Quando l'Utente in questione finisce di svolgere le proprie attività all'interno del sistema potrà effettuare il logout.

6.4 Attivazione di una chiamata

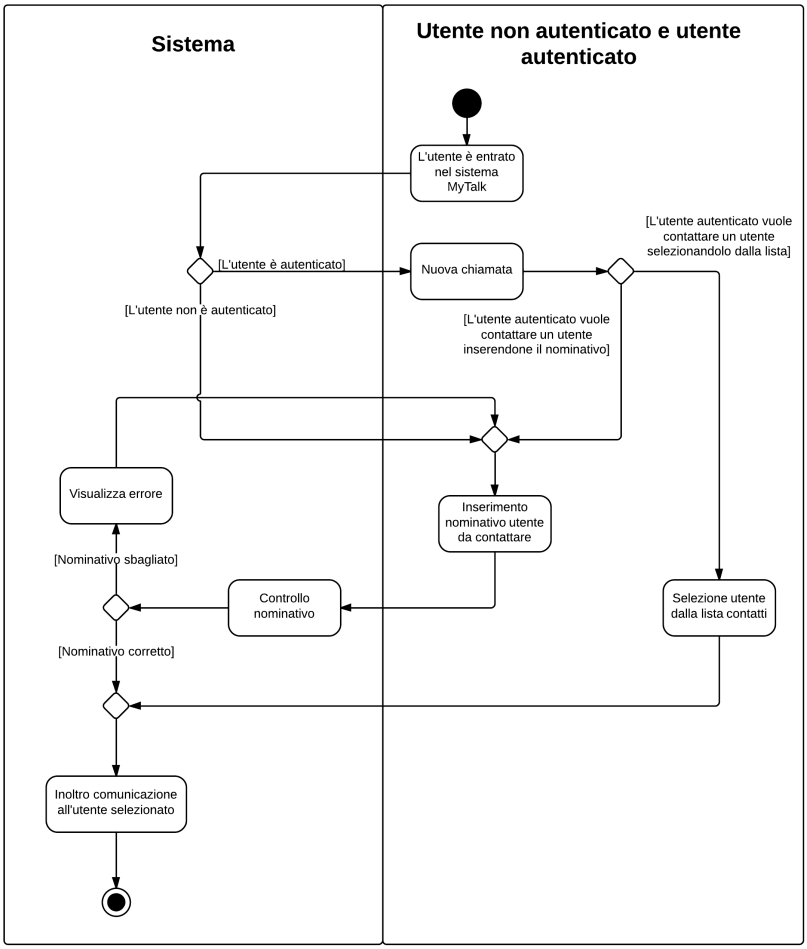

Figura 13: Diagramma delle attività di attivazione di una chiamata nel sistema MyTalk.

Descrizione : Il diagramma rappresentato descrive l'attività di attivazione di una chiamata da parte di un Utente nel sistema **MyTalk**. L'Utente può effettuare una chiamata sia se è un Utente Base o un Utente Autenticato. Un utente Base potrà inserire l'identificativo di un Utente e, se tale identificativo esiste, potrà chiamarlo. Se l'identificativo non esiste il Sistema lo notificherà all'Utente il quale potrà riprovare ad inserire un nuovo identificativo. Un Utente Autenticato può inoltre contattare un Utente dalla sua lista contatti.

6.5 Ricezione di una chiamata

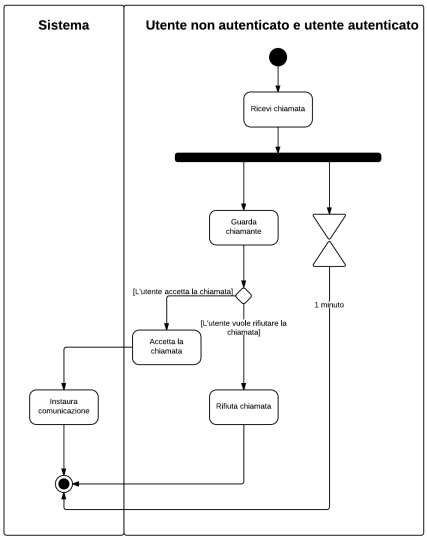

Figura 14: Diagramma delle attività di ricezione di una chiamata nel sistema MyTalk.

Descrizione : Il diagramma rappresentato descrive l'attività di ricezione di una chiamata da parte di un Utente nel sistema **MyTalk**. Quando un Utente, sia Base che Autenticato, riceve una chiamata partirà automaticamente un timeout d'attesa per la risposta. Nel caso che l'Utente chiamato rifiuti la chiamata allora l'attività di chiamata finisce altrimenti se dopo 1 minuto l'Utente non risponde la chiamata verrà ritenuta ignorata dal sistema chiudendo il tentativo di comunicazione. Invece se l'Utente risponde il sistema instaura la connessione per la chiamata tra il chiamante e il chiamato.

7 Stime di fattibilità e di bisogno di risorse

Analizzando l'architettura progettata è stato rilevato che le tecnologie ricercate per la realizzazione sono adeguate e ricoprono tutte le necessità progettuali, ma data l' esperienza esigua che i membri del team dispongono, non si esclude che ci possano essere dei cambiamenti. La tecnologia più solida (che probabilmente rimarrà come scelta definitiva) è *Google Web Toolkit*, che ci permette di realizzare i pattern adottati e le altre scelte architetturali previste.
Poiché tutti gli strumenti da utilizzare nello sviluppo sono gratuiti, il bisogno di risorse non si dimostra essere particolarmente problematico. In aggiunta al sopra citato *GWT*, è nelle intenzioni sfruttare la piattaforma di *cloud-computing Google App Engine* per ospitare l'applicativo. Verrà più approfonditamente esplorata questa possibilità nelle fasi successive (in caso d'impossibilità si esploreranno altre alternative).
Per quanto riguarda la redazione dei diagrammi delle classi, è stato sfruttato lo strumento *Lucidchart*, già utilizzato con successo nelle fasi di analisi e progettazione per la creazione di diagrammi *UML*.

8 Tracciamento della relazione componenti - requisiti - classi

Al fine di facilitare il tracciamento (oltre che rendere gestibile, scorrevole, consultabile e quindi effettivamente utile la lettura delle tabelle riassuntive da esso ricavate), si è deciso di effettuare una preliminare suddivisione delle varie classi che compongono l'applicativo, in modo da delineare delle macro-componenti.
Tali macro-componenti avranno lo scopo di individuare in modo univoco gruppi di classi con funzionalità, finalità e impieghi simili, che esplicitamente vengono utilizzate per lo scopo designato che le accomuna. Il tracciamento sarà poi effettuato coniugando requisiti e macro-componenti ottenuti, combinati tra di essi.
Qui di seguito si riporta quindi la leggenda delle suddivisioni succitate, con degli ulteriori accorgimenti:

- I nomi delle classi non sono riportati con il path intero, ma viene omessa la nomenclatura `it.hourglass.myTalk`.
- Dal momento che alcune componenti assumono dei ruoli preponderanti e fortemente ricorrenti nell'utilizzo standard dell'applicativo (in quanto facenti parte di componenti grafici fondamentali o snodi logici incaricati dello smistamento di input a funzionalità multiple) verranno nella maggior parte dei casi omessi o riportati solo in posizioni in cui svolgano un ruolo realmente rilevante. Nella maggior parte del resto dei casi, è possibile assumere una loro partecipazione. Tali classi includono:
  - `client.MyTalk`
  - `client.appcontroller.AppController`
  - `client.view.BaseView / client.presenter.BaseViewPresenter`
  - `client.view.PopupInformationView`
  - `client.view.FooterView / client.presenter.FooterPresenter`
  - `client.presenter.WidgetPresenter`
- Le componenti del package `it.hourglass.myTalk.wrappers`, pur differenziandosi tra di loro, concorrono nell'adempimento di una medesima funzione in situazioni differenti. Per tanto, saranno trattate come un unico componente. Lo stesso discorso vale per le componenti del package `it.hourglass.myTalk.event`, aventi tutti la medesima struttura ma utilizzati con poche differenze in ambiti diversi, soddisfacendo funzioni analoghe tra loro.

Componente	Sigla	Descrizione	Classi incluse
Accesso Funzionalità	MENU	*Componenti concorrenti a permettere l'accesso alle funzionalità ad utenti non autenticati*	client.view.AnnoMenuView client.presenter.AnnoMenuPresenter
	MENS	*Componenti concorrenti a permettere l'accesso alle funzionalità ad utenti autenticati*	client.view.SignedMenuView client.presenter.SignedMenuPresenter
Autenticazione	LOGV	*Visualizzazione della richiesta informazioni necessari all'inserimento di dati per l'autenticazione*	client.view.LoginView
	LOGP	*Logica di controllo e utilizzo dei dati inseriti dall'utente nelle operazioni di autenticazione*	client.presenter.UserListPresenter client.presenter.LoginPresenter server.model.businesslogic.Security client.shared.SignIn client.shared.Friendships client.shared.Message client.shared.Profile client.shared.SignIn client.shared.User
Profilo	PFVI	*Componenti concorrenti alla visualizzazione del profilo e dei dati personali*	client.view.EditProfileView client.view.profilemanagement.[...] [...].AvatarProfileManagementView [...].ExtendedDataProfileManagementView [...].PersonalDataProfileView client.shared.Profile client.shared.User
	PFMG	*Logica di controllo della modifica dei dati personali*	client.presenter.EditProfilePresenter client.presenter.profilemanagement[...] [...].AvatarProfileManagementPresenter [...].ExtendedDataProfileManagementPresenter [...].PersonalDataProfileManagementPresenter client.shared.Profile client.shared.User

Componente	Sigla	Descrizione	Classi incluse
	PFCN	*Componenti che concorrono al permettere la visualizzazione del profilo di un contatto*	client.presenter.ContactProfilePresenter client.view.contactprofile[...] [...].AvatarContactProfileView [...].ExtendedDataContactProfileView [...].PersonalDataContactProfileView client.shared.User
Contatti	FRRQ	*Componenti concorrenti alla gestione di richieste di amicizia*	client.presenter.FriendContactPresenter client.presenter.FriendMessagePresenter client.shared.Friendships client.shared.Message client.shared.Profile client.shared.User
	FRVI	*Visualizzazione delle amicizie instaurate e dei contatti relativi*	client.view.UserListView client.view.ContactListView client.shared.Friendships client.shared.Profile client.shared.User
	FRMN	*Componenti che concorrono a rendere possibile l'amministrazione e la modifica delle relazioni di amicizia*	client.presenter.ContactListPresenter client.presenter.ContactManagementPresenter client.shared.Friendships client.shared.Message client.shared.Profile client.shared.User
Registrazione	SIGV	*Componenti grafici legati alle operazioni di registrazione*	client.view.RegistrationView
	SIGP	Logica di controllo ed elaborazione dei dati inseriti in fase di registrazione	client.presenter.RegistrationPresenter client.presenter.ValuesCheck client.shared.User
Messaggistica	MSSD	*Componenti necessari all'invio di messaggi*	client.presenter.PopupMessagePresenter client.view.PopupMessageView client.shared.Message client.shared.Profile

Componente	Sigla	Descrizione	Classi incluse
	MSMG	*Componenti che rendono possibile la consultazione e l'amministrazione di messaggi*	client.presenter.MessagePresenter client.presenter.TextMessagePresenter client.view.MessageView client.shared.Message client.shared.Profile
Recupero Password	PWRE	*Componenti che concorrono specificatamente all'esecuzione della procedura necessaria al recupero password*	client.presenter.PasswordRecoverPresenter client.view.PasswordRecoverView client.presenter.ValuesCheck server.model.businesslogic.Security client.shared.User
Persistenza dati	DBPR	*Componenti che concorrono specificatamente alla memorizzazione e al prelievo di dati all'interno del database.*	server.model.dao.DAO server.model.dao.DAOImpl server.model.dao.HibernateUtil
Metodi Remoti	RPCC	*Componenti che concorrono specificatamente all'inoltro e adempimento delle richieste di chiamate asincrone remote.*	client.rpcservice.RPCService client.rpcservice.RPCServiceAsync server.model.businesslogic.RPCServiceImpl
Chiamata	CALLI	*Componenti che rendono possibile l'inizializzazione di chiamate*	client.communication.Call client.presenter.CallPopupPresenter client.view.CallPopup client.view.CallView
	CALLM	*Componenti che offrono la possibilità di monitorare e gestire il flusso della chiamata*	client.presenter.CallPresenter client.view.CallView
Comunicazioni WS	WSCM	*Componenti che concorrono specificatamente allo scambio di informazioni tra Client e Server WS*	client.communication.WSConnection client.communication. WSMessageBuilder server.wsserver.elaborator.Dialer server.wsserver.elaborator.Switch server.wsserver.interf.ParserIn server.wsserver.interf.ParserOut
Tools	CRYP	*Strumenti di criptazione delle informazioni*	server.model.businesslogic.Security

Componente	Sigla	Descrizione	Classi incluse
	EMSD	*Strumenti designati all'invio di e-mail all'utente*	`server.model.businesslogic.EmailSender`

8.1 Tracciamento componenti - requisiti

Seguono ora le coniugazioni concettuali e realizzative trovate tra i macro-componenti poco sopra delineati e i requisiti.

Codice Componente	Codice Requisito
PFVI	RUFUO 4, RUFUO 4.1, RUFUD 4.5, RUFUF 4.6, RUFUO 4.2, RUFUD 4.3, RGQD 20, RGQD 17, RUFUD 1.2.2, RUFUD 5.3.2, RUFUO 4.9, RUFUF 4.6.1, RUFUF 4.6.2, RUFUF 4.6.3, RUFUF 4.6.4, RUFUF 4.6.5, RUFUF 4.6.6, RUFUD 5.3.6.
LOGP	RUFUO 3.2, RUFUO 3, RUFUO 3.1, RGQD 20, RUFUO 3.4, RUFUD 5.3.2.
LOGV	RUFUO 3.2, RUFUO 3, RUFUO 3.1, RGQD 20, RUFUO 3.4, RUFUD 5.3.2.
SIGV	RUFUO 2, RUFUO 2.1, RGQD 20, RGQD 17, RUFUO 2.1.1, RUFUO 2.1.1.1, RUFUO 2.1.1.2, RUFUO 2.1.3, RUFUO 2.1.1.3, RUFUO 2.1.2, RUFUO 2.1.2.1, RUFUO 2.1.2.2, RUFUO 2.1.2.3, RUFUO 2.1.4, RUFUO 2.1.4.1, RUFUO 2.1.4.2, RUFUO 2.1.4.3, RUFUO 2.1.5, RUFUO 2.1.5.1, RUFUO 2.1.5.2, RUFUO 2.1.5.3, RUFUO 2.1.6, RUFUO 2.1.7.
CALLI	RUFUO 1, RUFUO 1.1, RUFUD 1.1.1, RUFUD 1.1.2, RUFUD 1.2, RUFUO 1.4, RUFUO 1.4.1, RUFUO 1.4.2, RUFUF 5.3, RGVO 16, RGQD 17, RUFUO 1.4.3.
CALLM	RUFUO 1.5, RGQD 20, RGVO 16, RGQD 17, RUFUO 1.5.1, RUFUD 1.5.2, RUFUO 1.5.3.
WSCM	RUFUO 1, RUFUO 1.1, RUFUO 2, RUFUD 4.3, RUFUD 1.1.1, RUFUD 1.1.2, RUFUD 1.2, RUFUO 1.4, RUFUO 1.4.1, RUFUO 1.4.2, RUFUD 5, RUFUD 5.1, RUFUF 5.2, RGVO 7, RGVO 9, RGVO 9.1, RGQO 21, RUFUD 1.2.1, RUFUO 1.4.3, RUFUO 3.4, RUFUD 5.3.1, RUFUF 5.3.3, RUFUO 1.5.1, RUFUD 5.1.1, RUFUD 5.1.2, RUFUD 1.5.2, RUFUO 1.5.3.
PFMG	RUFUO 4, RUFUO 4.1, RUFUD 4.5, RUFUF 4.6, RUFUO 4.2, RUFUO 4.2.1, RUFUD 4.3, RGQD 20, RGQD 18, RUFUD 5.3.2, RUFUF 4.6.1, RUFUF 4.6.2, RUFUF 4.6.3, RUFUF 4.6.4, RUFUF 4.6.5, RUFUF 4.6.6.
PFCN	RGQD 20.
FRRQ	RUFUD 5.1, RGQD 20.
FRVI	RUFUD 5, RUFUF 5.2, RUFUF 5.3, RGQD 20, RGQD 17, RUFUD 1.2.1, RUFUD 5.3.1, RUFUF 5.3.3, RUFUD 5.1.1, RUFUD 5.1.2.
FRMN	RUFUD 5, RUFUD 5.1, RUFUF 5.2, RUFUD 1.2.1, RUFUD 5.1.1, RUFUD 5.1.2.

Codice Componente	Codice Requisito
MSSD	RGQD 20, RUFUD 5.3.1, RUFUF 5.3.3.
MSMG	RGQD 20, RUFUD 1.2.2.
PWRV	RUFUD 4.4, RGQD 20, RGQD 17, RUFUD 4.4.1.
PWRP	RUFUD 4.4, RGQD 20, RGQD 18, RUFUD 4.4.1, RUFUD 4.4.2.
DBPR	RUFUO 2, RUFUO 2.2.1, RUFUO 3, RUFUO 4, RUFUO 4.1, RUFUD 4.5, RUFUF 4.6, RUFUO 4.2, RUFUD 4.3, RUFUD 4.4, RUFUD 5, RUFUD 5.1, RUFUF 5.2, RGVO 9, RGQO 21, RUFUD 1.2.1, RUFUD 5.3.1, RUFUD 1.2.2, RUFUF 5.3.3, RUFUO 2.2, RUFUD 5.1.1, RUFUD 5.1.2, RUFUD 4.4.1, RUFUF 4.6.1, RUFUF 4.6.2, RUFUF 4.6.3, RUFUF 4.6.4, RUFUF 4.6.5, RUFUF 4.6.6, RUFUD 5.3.6, RUFUD 4.4.2.
RPCC	RUFUO 2, RUFUO 3, RUFUO 4, RUFUO 4.1, RUFUD 4.5, RUFUF 4.6, RUFUO 4.2, RUFUD 4.3, RUFUD 4.4, RUFUD 5, RUFUD 5.1, RUFUF 5.2, RGVO 9, RGQO 21, RUFUD 1.2.1, RUFUD 5.3.1, RUFUD 1.2.2, RUFUF 5.3.3, RUFUO 2.2, RUFUD 5.1.1, RUFUD 5.1.2, RUFUD 4.4.1, RUFUF 4.6.1, RUFUF 4.6.2, RUFUF 4.6.3, RUFUF 4.6.4, RUFUF 4.6.5, RUFUF 4.6.6, RUFUD 5.3.6, RUFUD 4.4.2.
MENU	RUFUO 1, RUFUO 1.1, RUFUO 3, RUFUD 1.2, RUFUD 4.4, RUFUF 5.2, RGQD 20, RGVO 16, RGQD 17, RUFUD 4.4.1.
MENS	RUFUD 1.2, RUFUD 5, RUFUD 5.1, RGQD 20, RGVO 16, RGQD 17, RUFUD 1.2.1, RUFUD 1.2.2, RUFUO 4.9, RUFUF 4.6.1, RUFUF 4.6.2, RUFUF 4.6.3, RUFUF 4.6.4, RUFUF 4.6.5, RUFUF 4.6.6, RUFUD 5.3.6.
EMSD	RUFUO 2.2.1, RUFUO 4.2.1, RUFUO 2.2, RUFUD 4.4.1, RUFUD 4.4.2.
CRYP	RGQD 18, RUFUD 4.4.1.
CALLI	RUFUO 1, RUFUO 1.1, RUFUD 1.1.1, RUFUD 1.1.2, RUFUD 1.2, RUFUO 1.4, RUFUO 1.4.1, RUFUO 1.4.2, RUFUF 5.3, RGVO 16, RGQD 17, RUFUO 1.4.3.
CALLM	RUFUO 1.5, RGQD 20, RGVO 16, RGQD 17, RUFUO 1.5.1, RUFUD 1.5.2, RUFUO 1.5.3.
WSCM	RUFUO 1, RUFUO 1.1, RUFUO 2, RUFUD 4.3, RUFUD 1.1.1, RUFUD 1.1.2, RUFUD 1.2, RUFUO 1.4, RUFUO 1.4.1, RUFUO 1.4.2, RUFUD 5, RUFUD 5.1, RUFUF 5.2, RGVO 7, RGVO 9, RGVO 9.1, RGQO 21, RUFUD 1.2.1, RUFUO 1.4.3, RUFUO 3.4, RUFUD 5.3.1, RUFUF 5.3.3, RUFUO 1.5.1, RUFUD 5.1.1, RUFUD 5.1.2, RUFUD 1.5.2, RUFUO 1.5.3.

Codice Componente	Codice Requisito
SIGP	RUFUO 2, RGQD 20, RGQD 18, RUFUO 2.1.1, RUFUO 2.1.1.1, RUFUO 2.1.1.2, RUFUO 2.1.3, RUFUO 2.1.1.3, RUFUO 2.1.2, RUFUO 2.1.2.1, RUFUO 2.1.2.2, RUFUO 2.1.2.3, RUFUO 2.2, RUFUO 2.1.4, RUFUO 2.1.4.1, RUFUO 2.1.4.2, RUFUO 2.1.4.3, RUFUO 2.1.5, RUFUO 2.1.5.1, RUFUO 2.1.5.2, RUFUO 2.1.5.3.

8.2 Tracciamento requisiti - Componenti

Codice Requisito	Componenti
RUFUO 1	WSCM, MENU, CALLI.
RUFUO 1.1	WSCM, MENU, CALLI.
RUFUD 1.1.1	CALLI, WSCM.
RUFUD 1.1.2	CALLI, WSCM.
RUFUD 1.2	MENU, MENS, CALLI, WSCM.
RUFUD 1.2.1	MENS, FRVI, FRMN, RPCC, DBPR, WSCM.
RUFUD 1.2.2	MENS, PFVI, MSMG, RPCC, DBPR.
RUFUD 1.3	Requisito attualmente non realizzato.
RUFUD 1.3.1	Requisito attualmente non realizzato.
RUFUO 1.4	CALLI, WSCM.
RUFUO 1.4.1	CALLI, WSCM.
RUFUO 1.4.2	CALLI, WSCM.
RUFUO 1.4.3	CALLI, WSCM.
RUFUO 1.5	CALLM.
RUFUO 1.5.1	CALLMG, CALLC, WSCM.
RUFUD 1.5.2	CALLM, WSCM.
RUFUO 1.5.3	CALLM, WSCM.
RGVO 10	Tutte le componenti.
RGVO 10.1	Tutte le componenti.
RGVF 11	Tutte le componenti.
RGVF 12	Non tracciabile.
RGVF 13	Non tracciabile.
RGVF 14	Tutte le componenti.
RGVF 15	Tutte le componenti.
RGVO 16	MENS, MENU, CALLI, CALLM, CALLC.
RGQD 17	MENU, MENS, PFVI, FRVI, SIGV, PWRV, CALLI, CALLM.
RGQD 18	PFMG, PWRP, SIGP, CRYP.
RGQO 19	Non tracciabile.
RUFUO 2	WSCM, SIGV, SIGP, RPCC, DBPR.
RUFUO 2.1	SIGV.
RUFUO 2.1.1	SIGV, SIGP.
RUFUO 2.1.1.1	SIGV, SIGP.
RUFUO 2.1.1.2	SIGV, SIGP.
RUFUO 2.1.1.3	SIGV, SIGP.
RUFUO 2.1.2	SIGV, SIGP.
RUFUO 2.1.2.1	SIGV, SIGP.
RUFUO 2.1.2.2	SIGV, SIGP.
RUFUO 2.1.2.3	SIGV, SIGP.
RUFUO 2.1.3	SIGV, SIGP.

Codice Requisito	Componenti
RUFUO 2.1.4	SIGV, SIGP.
RUFUO 2.1.4.1	SIGV, SIGP.
RUFUO 2.1.4.2	SIGV, SIGP.
RUFUO 2.1.4.3	SIGV, SIGP.
RUFUO 2.1.5	SIGV, SIGP.
RUFUO 2.1.5.1	SIGV, SIGP.
RUFUO 2.1.5.2	SIGV, SIGP.
RUFUO 2.1.5.3	SIGV, SIGP.
RUFUO 2.1.6	SIGV.
RUFUO 2.1.7	SIGV.
RUFUO 2.2	SIGP, EMSD, RPCC, DBPR.
RUFUO 2.2.1	EMSD, DBPR.
RGQD 20	MENU, MENS, LOGV, LOGP, PFVI, PFMG, PFCN, FRRQ, FRVI, FMN, MSSD, MSMG, SIGV, SIGP, PWRV, PWRP, CALLO, CALLM, CALLC.
RGQO 21	RPCC, DBPR, WSCM.
RGQO 22	Non tracciabile.
RGQO 23	Non tracciabile.
RGQO 24	Non tracciabile.
RGQO 25	Non tracciabile.
RAFUD 26	Requisito attualmente non realizzato.
RAFUD 26.1	Requisito attualmente non realizzato.
RAFUD 26.2	Requisito attualmente non realizzato.
RAFUD 26.3	Requisito attualmente non realizzato.
RAFUD 27	Requisito attualmente non realizzato.
RAFUD 27.1	Requisito attualmente non realizzato.
RAFUD 27.1.1	Requisito attualmente non realizzato.
RAFUD 27.1.2	Requisito attualmente non realizzato.
RAFUD 27.1.3	Requisito attualmente non realizzato.
RAFUD 27.1.3.1	Requisito attualmente non realizzato.
RAFUD 27.1.4	Requisito attualmente non realizzato.
RAFUD 27.1.5	Requisito attualmente non realizzato.
RUFUO 3	MENU, LOGV, LOGP, RPCC, DBPR.
RUFUO 3.1	LOGV, LOGP.
RUFUO 3.2	LOGV, LOGP.
RUFUF 3.3	Requisito attualmente non realizzato.
RUFUO 3.4	LOGV, LOGP, WSCM.
RUFUO 4	PFVI, PFMG, RPCC, DBPR.
RUFUO 4.1	PFVI, PFMG, RPCC, DBPR.
RUFUO 4.2	PFVI, PFMG, RPCC, DBPR.
RUFUO 4.2.1	PFMG, EMSD.
RUFUD 4.3	PFVI, PFMG, RPCC, DBPR, WSCM.

Codice Requisito	Componenti
RUFUD 4.4	MENU, PWRV, PWRP, RPCC, DBPR.
RUFUD 4.4.1	MENU, PWRV, PWRP, RPCC, DBPR, EMSD, CRYP.
RUFUD 4.4.2	PWRP, RPCC, DBPR, EMSD.
RUFUD 4.5	PFVI, PFMG, RPCC, DBPR.
RUFUF 4.6	PFVI, PFMG, RPCC, DBPR.
RUFUF 4.6.1	MENS, PFVI, PFMG, RPCC, DBPR .
RUFUF 4.6.2	MENS, PFVI, PFMG, RPCC, DBPR .
RUFUF 4.6.3	MENS, PFVI, PFMG, RPCC, DBPR .
RUFUF 4.6.4	MENS, PFVI, PFMG, RPCC, DBPR.
RUFUF 4.6.5	MENS, PFVI, PFMG, RPCC, DBPR .
RUFUF 4.6.6	MENS, PFVI, PFMG, RPCC, DBPR .
RUFUD 4.7	Requisito attualmente non realizzato.
RUFUD 4.8	Requisito attualmente non realizzato.
RUFUO 4.9	MENS, PFVI.
RUFUD 5	MENS, FRVI, FRMN, RPCC, DBPR, WSCM.
RUFUD 5.1	MENS, FRRQ, FRMN, RPCC, DBPR, WSCM.
RUFUD 5.1.1	FRVI, FRMN, WSCM, RPCC, DBPR.
RUFUD 5.1.2	FRVI, FRMN, WSCM, RPCC, DBPR.
RUFUF 5.2	MENU, FRVI, FRMN, RPCC, DBPR, WSCM.
RUFUF 5.3	FRVI, CALLI.
RUFUD 5.3.1	FRVI, MSSD, WSCM, RPCC, DBPR .
RUFUD 5.3.2	LOGV, LOGP, PFVI, PFMG.
RUFUF 5.3.3	FRVI, MSSD, WSCM, RPCC, DBPR.
RUFUF 5.3.4	Requisito attualmente non realizzato.
RUFUD 5.3.5	Requisito attualmente non realizzato.
RUFUD 5.3.6	MENS, PFVI, PFMN, RPCC, DBPR.
RUFUD 5.4	Requisito attualmente non realizzato.
RUFUF 5.4.1	Requisito attualmente non realizzato.
RUFUD 5.4.2	Requisito attualmente non realizzato.
RUFUF 5.5	Requisito attualmente non realizzato.
RUFUF 5.5.1	Requisito attualmente non realizzato.
RUFUF 5.5.2	Requisito attualmente non realizzato.
RUFUF 5.5.3	Requisito attualmente non realizzato.
RGVO 6	Non tracciabile.
RGVO 7	WSCM.
RGVO 8	Tutte le componenti.
RGVO 9	WSCM, RPCC, DBPR.
RGVO 9.1	WSCM.